UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA IN SISTEMI INFORMATIVI

UNA DATA WAREHOUSE DI SUPPORTO ALLA TEXT CATEGORIZATION

Relatore:
Ch.mo prof. Antonio d'Acierno

Candidato:
Oscar Talamo
(matr: 041/1604)

ANNO ACCADEMICO 2003/2004

A mia nonna

INDICE

INDICE 3

INDICE DELLE FIGURE 8

CAPITOLO 1: INTRODUZIONE 9

1.1 Il Problema Affrontato 10

1.2 Siti di Riferimento 11

1.3 Stato dell'Arte 13

1.4 Architettura di Massima del Sistema 17

1.4.1 Linee Guida 18

1.4.2 Vincoli del Progetto 18

1.4.3 Tecnologie Adoperate 19

1.5 Passi del Processo di Sviluppo 20

1.6 Documentazione della Tesi 21

CAPITOLO 2: LA COLLECTION REUTERS-21578 22

2.1 Il Benchmark 23

2.2 Stato Iniziale dei Dati 23

2.3 Schema Logico "Ipotetico" 25

2.4 Entità Aggiuntive 26

2.4.1 Topic & Categorie 26

2.4.2 Split 27

2.4.3 Data Set 30

2.5 Schema Concettuale 30

CAPITOLO 3: LA COLLECTION 20 NEWSGROUPS 32

3.1 Il Benchmark 33

3.2 Stato Iniziale dei Dati 33

3.3 Schema Logico "Ipotetico" 35

3.4 Entità Aggiuntive 37

3.5 Ristrutturazione delle Entità 37
3.5.1 Data Set 38
3.5.2 Split 38
3.5.3 Categoria 39
3.5.4 Usenet articles 40
3.6 Schema Concettuale 41

CAPITOLO 4: LA COLLECTION OHSUMED 42
4.1 Il Benchmark 43
4.2 Stato Iniziale dei Dati 43
4.3 Schema Logico "Ipotetico" 46
4.4 Ristrutturazione delle Entità 47
4.4.1 Documento 47
4.4.2 Split 48
4.4.3 Topic 49
4.4.4 Data Set 49
4.5 Schema Concettuale 51

CAPITOLO 5: GENERALITA' DEL SISTEMA 52
5.1 Software Requirements Specifications 53
5.2 Funzionalità del prodotto 54
5.2.1 Validazione utente 55
5.2.2 Funzionalità riguardanti i dati delle Collection 55
5.2.3 Funzionalità riguardanti i dati degli Esperimenti 60
5.2.4 QUERY d'interesse 63
5.3 Use-Case Model 63
5.3.1 Actors 63
5.3.2 Permessi 64
5.3.3 Generic Abstract 65
5.4 L'unità organizzativa Collection* 69
5.4.1 Caso completo: Associazioni Documenti-Collection 69
5.4.2 Casi relativi alle Associazioni definite per mezzo di varianti 72
5.4.2.1 Associazioni Categorie-Documenti 72
5.4.2.2 Associazioni Categorie-Topic 72
5.4.2.3 Associazioni Documenti-Split 72

5.4.2.4 Associazioni Query-Documenti 73
5.4.2.5 Associazioni Query-Topic 73
5.4.2.6 Associazioni Split-DataSet 74
5.4.2.7 Associazioni Topic-DataSet 74
5.4.2.8 Adaptive Filtering 74
5.4.3 Casi relativi alle Entità definite per mezzo di varianti 75
5.4.3.1 Documenti 75
5.4.3.2 Query 78
5.4.3.3 Categorie 79
5.4.3.4 Split 82
5.4.3.5 Topic 84
5.4.3.6 DataSet 86
5.4.3.7 Collection 88
5.4.3.8 File 91
5.5 L'unità organizzativa Esperimenti* **91**
5.5.1 Ricercatori 92
5.5.2 Effectiveness Misure 93
5.5.3 Esperimenti 94
5.5.3.1 Esperimenti<>EM 97
5.5.3.2 Esperimenti<>Ricercatori 98
5.5.3.3 FV 98
5.5.3.4 Risultati CPC 99
5.5.3.5 Risultati DPC 99
CAPITOLO 6: IL DATABASE DI INTEGRAZIONE 100
6.1 Organizzazione della documentazione 101
6.2 Schema Concettuale dei Benchmarks 101
6.2.1 Domini Utilizzati 103
6.2.2 Uso del Prototipo per il dimensionamento 105
6.2.3 Osservazioni e Vincoli aggiuntivi 106
6.2.3.1 Collection & File 106
6.2.3.2 Data Set 106
6.2.3.3 Topic 107
6.2.3.4 Categoria 107
6.2.3.5 Documento 107

6.3 Schema Concettuale degli Esperimenti 108
6.3.1 Ulteriori Domini Utilizzati 109
6.3.2 Osservazioni e Vincoli aggiuntivi 109
6.3.3 Gestione dei Permessi 110
6.4 Schema Logico dei Benchmarks 111
6.4.1 Domini Aggiuntivi 112
6.4.2 Vincoli aggiuntivi 112
6.5 Schema Logico degli Esperimenti 113
6.5.1 Vincoli aggiuntivi 114
6.6 Viste 114
6.6.1 Viste per i Permessi 115
6.6.2 Viste per le Entità 116
6.6.3 Viste per le Relazioni 122
6.6.4 Viste delle QUERY d'interesse 125
6.7 Restrizioni del codice SQL dovute ad ACCESS 126
6.7.1 Implementazione Domini "Type" 128
6.7.2 Vincoli Manuali Aggiuntivi 129
6.8 I file DBDSATC ed il loro utilizzo 131

CAPITOLO 7: IL MODULO DI IN/OUT 133

7.1 Architettura del Modulo 134
7.2 Core del Modulo 135
7.3 Modello Object-Oriented del database 136
7.4 Classe Permessi 141
7.4.1 I Membri 141
7.4.2 Le Direttive 145
7.4.3 Sequence Diagram di una Transazione 146
7.5 Classe Abstract 147
7.5.1 Il Settaggio 148
7.5.2 I Membri 150
7.5.3 Le Direttive 159
7.5.4 Sequence Diagram delle Funzionalità Principali 160
7.6 Classe Vincoli 164
7.7 Classe DBDSATC 167
7.7.1 Implementazione dei Benchmarks 168

7.7.2 Implementazione Domini “Type” 175
7.7.3 Implementazione degli Esperimenti 176
7.7.4 I Membri della Classe DBDSATC 179
7.7.5 Le Direttive. 180
7.8 Organizzazione in Libreria del Modulo 182

CAPITOLO 8: I MODULI DI FEEDING 183
8.1 ReutersIn 184
8.1.1 Descrizione 184
8.1.2 Utilizzo 194
8.2 NewsgroupsIn 196
8.2.1 Descrizione 197
8.2.2 Utilizzo 203
8.3 OhsumedIn 205
8.3.1 Descrizione 206
8.3.2 Utilizzo 215
8.4 Test Dimensioni 216
8.5 Analisi del database 219

CAPITOLO 9: ESEMPIO DI INTERFACCIA GRAFICA 222
9.1 Librerie Adoperate 223
9.2 Menu e sue Funzionalità 225
9.3 ToolBar e sue Funzionalità 230
9.4 Descrizione delle Viste 233
9.5 Funzionalità in Modalità Report 236

CAPITOLO 10: CONCLUSIONI 239
10.1 Limitazioni del Software 240
10.2 Possibili Sviluppi Successivi 241

BIBLIOGRAFIA 243

INDICE DELLE FIGURE

Business Process Model 15
Component Diagram 17
Schema Logico del Reuters-21578 25
Schema Concettuale del Reurers-21578 26
Schema Concettuale Ristrutturato del Reurers-21578 30
Schema Logico del 20 Newsgroups 35
Schema Concettuale del 20 Newsgroups 36
Schema Concettuale Ristrutturato del 20 Newsgroups 41
Schema Logico dell'OHSUMED 46
Schema Concettuale dell'OHSUMED 47
Schema Concettuale Ristrutturato dell'OHSUMED 51
Organizzazione SRS 53
Organizzazione Use-Case Model 63
Schema Concettuale dei Benchmarks 102
Schema Concettuale degli Esperimenti 108
Schema Logico dei Benchmarks 111
Schema Logico degli Esperimenti 113
Diagramma Principale del Modulo di In/Out 134
Core del Modulo di In/Out 135
Modello Object-Oriented del database (Benchmarks) 137
Modello Object-Oriented del database (Esperimenti) 138
Classe Permessi 141
Sequence Diagram di una Transazione 146
Classe Abstract 147
Sequence Diagram di un Inserzione 160
Sequence Diagram di una Cancellazione 161
Sequence Diagram di una Ricerca 162
Sequence Diagram di una Modifica 163
Classe Vincoli 164
Modello Object-Oriented (1° Parte) 168
Modello Object-Oriented (2° Parte) 175
Modello Object-Oriented (3° Parte) 176
Classe DBDSATC 179
Diagramma di Massima dell'HIC 224

CAPITOLO 1:

INTRODUZIONE

1.1 Il Problema Affrontato
1.2 Siti di Riferimento
1.3 Stato dell'Arte
1.4 Architettura di Massima del Sistema
1.5 Passi del Processo di Sviluppo
1.6 Documentazione della Tesi

In questo capitolo si descrive il lavoro che, allo stato attuale, i ricercatori devono compiere per effettuare le loro analisi sugli algoritmi di Text Categorization. Una volta individuato il problema, si procede prima alla descrizione del Sistema proposto e poi a quella del processo di sviluppo adoperato per la sua realizzazione.

1.1 Il Problema Affrontato

La diffusione di sistemi informativi locali e remoti ha causato negli anni una sempre maggiore tendenza delle aziende ad immagazzinare i propri dati in formato digitale, preferendolo a quello cartaceo o in pellicola o altro. Mettere a disposizione degli utenti grandi quantità di dati, senza permettere di individuare le sole informazioni utili, non avrebbe avuto alcun senso; così è nata l'esigenza d'organizzare e classificare le diverse tipologie di dati. Per facilitare e velocizzare tale processo si sono sviluppati sistemi in grado di eseguire classificazioni automatiche.

Tra le diverse tipologie di classificatori che sono stati realizzati, quelli riguardanti la **Text Categorization** (**TC**) hanno richiamato negli ultimi 10 anno un particolare interesse da parte dei ricercatori. La **TC** è applicata in molte situazioni che vanno dall'indicizzazione di documenti basata su dizionari controllati, al filtraggio di documenti, alla classificazione gerarchica di documenti Web e, più in generale, in tutti quei compiti di gestione e organizzazione di documenti di testo.

Vi sono due differenti modi per adoperare i classificatori di testo: il **DPC** *(document-pivoted categorization)*, che a partire da un documento ricerca le categorie ad esso attinenti, ed il **CPC** (category-pivoted categorization), che a partire da una categoria ricerca tutti i documenti ad essa attinenti.

Inizialmente la Text Categorization (TC) fu applicata tramite un approccio di tipo *Knowledge Engineering* (**KE**), che si basava sull'identificazione, da parte d'esperti, di regole in grado di decidere l'appartenenza di un documento ad una particolare categoria. Tale paradigma col tempo fu rimpiazzato sempre più da quello del *Machine Learning* (**ML**), che superava il limite di dover formalizzare, in maniera precisa, regole definite, non sempre con chiarezza, dagli esperti del dominio applicativo d'interesse.

In questo nuovo paradigma del Machine Learning un processo induttivo costruisce il classificatore automatico di testi "imparando", da un insieme di documenti pre-classificati, le caratteristiche rappresentative delle categorie d'interesse.

L'insieme dei documenti pre-classificati prende il nome di **Data Set** ed è suddiviso in due sottoinsiemi:

- **TV:** il *training (and validation) set*, di cui una parte è adoperata per addestrare il classificatore (**Tr**), ed un'altra è utilizzata nella fase di validazione e di settaggio dei suoi parametri.

- **Te**: il *test set*, adoperato per testare il classificatore e ricavare un insieme di misure (*Effectiveness Misures*) in grado di valutarne l'efficienza.

Per mezzo delle effectiveness misures si ha dunque la possibilità di valutare il corretto funzionamento e la precisione di un classificatore. Adoperare tali misure, per confrontare i risultati dei numerosi algoritmi d'addestramento e classificazione che sono stati realizzati, non avrebbe alcun senso se i Data Set di partenza non fossero gli stessi. Per risolvere tale inconveniente è stato necessario creare dei *Benchmarks* che permettessero di paragonare tra loro i risultati ottenuti dai diversi algoritmi.

L'obiettivo di questa tesi è offrire supporti che siano in grado di velocizzare la valutazione dei diversi algoritmi d'addestramento e classificazione.
Un primo problema è stato quello di creare un database avente una struttura in grado di memorizzare i benchmarks disponibili negli ambienti di ricerca; compito reso difficile non solo dalla differenza nelle informazioni messe a disposizione riguardo a documenti e categorie, ma soprattutto dalla diversa organizzazione dei dati dipendente dal tipo d'applicazioni cui si rivolge il particolare benchmark.
In seguito alla creazione del database si è presentato il problema di aggiungere al sistema strutture in grado di memorizzare i dati dei diversi "*esperimenti*", dove con tale termine s'intende l'insieme d'informazioni relative alla valutazione di particolari algoritmi.
In fine si sono realizzati diversi moduli tra cui: una *libreria d'In/Out*, *moduli di feeding* e un *HIC* (Human Interaction Component).

1.2 Siti di Riferimento

Nel presente paragrafo si riportano i siti da cui si sono tratte le principali informazioni riguardanti il problema considerato:

- http://faure.iei.pi.cnr.it/~fabrizio/

  Sito del professor Fabrizio Sebastiani, considerato come punto di partenza di questa tesi. Un articolo di riferimento, che ben si è prestato ad uno studio preliminare sul processo di sviluppo degli applicativi di TC, è "*Machine Learning in Automated Text Categorization*", ACM Computing Surveys, Vol 34, No 1, Marzo 2002, pp 1-47.

- http://www.di.uniba.it/~nnets/

  Sito della professoressa Anna Maria Fanelli dal quale si sono ricavate delle informazioni di supporto all'articolo di G.M. Di Nunzio e A. Micarelli,

"Categorizzazione automatica di testi utilizzando Reti RBF Modificate" (Dipartimento d'Informatica e Automazione, Laboratorio d'Intelligenza Artificiale, Università degli Studi "Roma tre", Roma, Italia). L'articolo menzionato fa riferimento ad un innovativo esempio di TC.

- http://kdd.ics.uci.edu/

  Sito dell'**UCI Knowledge Discovery in Databases Archive.** Il sito contiene data set di supporto alle più svariate tipologie d'applicazioni nell'ambito dell'approccio Machine Learning. L'utilizzo dei benchmarks offerti è gratuito, ed è concesso a patto di inserire nella documentazione degli esperimenti effettuati un riferimento all'origine del data set adoperato. Da tale sito è stato possibile ricavare i benchmark ***"20 Newsgroups Data"*** e ***"Reuters-21578 Text Categorization Collection"*** (Un terzo benchmark per i TC, *"NSF Research Awards Abstracts 1990-2003"* non è preso in considerazione e potrebbe essere adoperato in seconda battuta per verificare la flessibilità del database realizzato per l'applicativo).

- http://www.daviddlewis.com/

  Sito di David D. Lewis, realizzatore della ***"Reuters-21578 Text Categorization Collection"***.

- http://telemat.die.unifi.it/book/Internet/Sgml/indsgml.htm

  Un utile manuale sullo *Standard Generalized Markup Language* (**SGML**), un metalinguaggio adoperato nel benchmark *Reuters* per descriverne la formattazione dei suoi documenti.

- http://www-2.cs.cmu.edu/~TextLearning/

  Sito del **CMU Text Learning Group** il cui obiettivo è quello di realizzare nuove Machine Learning per la **TC**. A capo del gruppo di lavoro c'è il professor Tom Mitchell, realizzatore del benchmark *"20 Newsgroups Data"*.

- http://www.google.it e http://www.yahoo.com/

  Motori di ricerca Google e Yahoo, dai quali si sono prelevate informazioni relative ai NewsGroups utilizzati come categorie nel benchmark *"20 Newsgroups Data"*.

- http://trec.nist.gov/

  Sito della **Text Retrieval Conference (TREC),** il cui proposito è di fornire le infrastrutture necessarie per una valutazione, su larga scala, delle metodologie relative all'*information retrieval* (l'**IR** si riferisce alla gestione dei documenti in base ai loro contenuti).

Al suo interno è stato possibile trovare il materiale relativo al benchmark *OHSUMED* che in tale sede viene adoperato dalla *TREC-9 Filtering Track* (un report riguardo ai risultati ottenuti da questo gruppo di lavoro è presente nel file prelevabile all'URL http://trec.nist.gov/pubs/trec9/papers/t9irep.pdf).

- http://www.ohsu.edu/dmice/

  Sito dell' **Oregon Health & Science University Department of Medical Informatics & Clinical Epidemiology**, tramite il quale è possibile accedere ad un indirizzo ftp (ftp://medir.ohsu.edu/pub/ohsumed/) dove l'OHSU, per assistere le ricerche nel campo dell'**IR,** mette a disposizione i dati del benchmark *OHSUMED*.

- http://www.nlm.nih.gov/mesh/

  Sito della **Medical Subject Headings (MeSH)**, da cui è possibile ricavare informazioni relative al vocabolario *MeSH* adoperato nel benchmark OHSUMED.

1.3 Stato dell'Arte

Il primo passo, necessario per descrivere lo stato dell'arte, è stato quello di individuare il processo interno seguito dai *ricercatori* per effettuare sperimentazioni sui nuovi algoritmi d'addestramento e classificazione.
Un ricercatore tipicamente realizza degli algoritmi focalizzando la propria attenzione su un particolare tipo d'applicazione, poi, secondo quest'ultimo, decide quale data set adoperare (si osservi che un benchmark non contiene soltanto un data set ma, spesso, è organizzato in modo da poterne definire più di uno combinando in modo opportuno le entità al suo interno).
Lo stato dell'arte attuale, che sarà descritto di seguito tramite un *Business Process Model*, mostra, in linee generali, come molti dei processi eseguiti sono inutilmente ripetuti per i diversi esperimenti e come il più delle volte è necessario realizzare del codice per estrarre le informazioni utili.
Il modello è realizzato tramite l'utilizzo e la definizione d'*Unità Organizzative*, di *Risorse* e di *Processi* (questi ultimi sono adoperati dai ricercatori per realizzare le fasi necessarie allo studio degli algoritmi).

Organization Units:

Con il termine *Ricercatore Generico* s'intende un individuo (o un insieme di individui) dedito allo sviluppo di un sistema per la TC.

Resources:

Le risorse rappresentano insiemi di dati relativi a particolari fonti o archivi. Tra le diverse risorse si è pensato di inserire "***Dati Esperimento***", che permette di memorizzare l'ipotetico insieme di dati creato durante la valutazione degli algoritmi. *"Dati Esperimento"* tipicamente consiste in:

- Un insieme di vettori d'informazioni, detti *Features Vector* (**FV**), contenenti ognuno un predefinito numero di metadati relativi ad un documento ed utilizzati dagli algoritmi d'addestramento e classificazione. Una volta ricavati tali vettori il ricercatore non avrà più bisogno dei documenti di partenza legati univocamente ai primi.
- L'insieme dei risultati ottenuti dal classificatore una volta effettuato l'addestramento, utile per ricavare le *effectiveness misures*.

Le altre risorse inserite rappresentano invece uno dei possibili benchmark cui ha accesso il ricercatore. Nel modello si sono riportate, in particolare, tre collezioni selezionate tra i benchmark presentati nell'articolo "*Machine Learning in Automated Text Categorization*" di Fabrizio Sebastiani. La scelta è avvenuta in base alla loro diffusione ed alla forte differenza nelle tipologie di documenti e categorie contenute. Di seguito si riportano alcune informazioni riguardanti le fonti scelte ed i file nelle quali sono organizzate:

- **Reuters collection**: è adatta alla costruzione di classificatori e ha come obiettivo fornire dati per la document-pivoted categorization (**DPC**).

  Tale collection è disponibile presso il sito della *UCI KDD Archive* all'URL http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html, i due file riportati contengono uno informazioni sul data set e sulla sua organizzazione (Readme.txt) e l'altro (reuters21578.tar.gz) sia il data set formattato in SGML che alcuni file elencanti le categorie adoperate nella collezione.
- **20 Newsgroups collection**: come la precedente fornisce dati per la document-pivoted categorization (DPC) ed è adatta alla costruzione di classificatori, ma presenta informazioni ben diverse rifacendosi a categorie gerarchiche legate ai newsgroups ed a documenti la cui formattazione è prossima all'e-mail.

  Anche questa collection è disponibile presso il sito della *UCI KDD Archive* all'URL http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html, ma i due file riportati non sono altro che due insieme di documenti classificati secondo 20

categorie di newsgroups (20_newsgroups.tar.gz), dove il secondo (mini_newsgroups.tar.gz) indica un sottoinsieme del primo.

Per avere una divisione del data set in *training set* e *test set* si è adoperato il file *"20News-bydate.tar.gz"*, messo a disposizione dei ricercatori interessati presso l'URL http://www.ai.mit.edu/~jrennie/20Newsgroups/20news-bydate.tar.gz.

- **OHSUMED collection**: quest'ultima collection a differenza delle precedenti fornisce dati per la category-pivoted categorization (**CPC**) ed è adoperata per il Text Filtering. Tale collection è disponibile presso il sito della *TREC* (Text Retrieval Conference) all'URL http://trec.nist.gov/data.html. I due file riportati contengono: uno informazioni sul data set e sulla sua organizzazione (README), l'altro (filtering.tar.gz) il data set. Alcune informazioni aggiuntive sono state prelevate dai file messi a disposizione *dalla Oregon Health and Science University* (**OHSU**) School of Medicine presso il sito ftp://medir.ohsu.edu/pub/ohsumed/.

Di seguito si riporta il Business Process Model implementato nel file *BusinessProcessModel_DBDSATC.bpm* inserito nel workspace DataSetDB.sws.

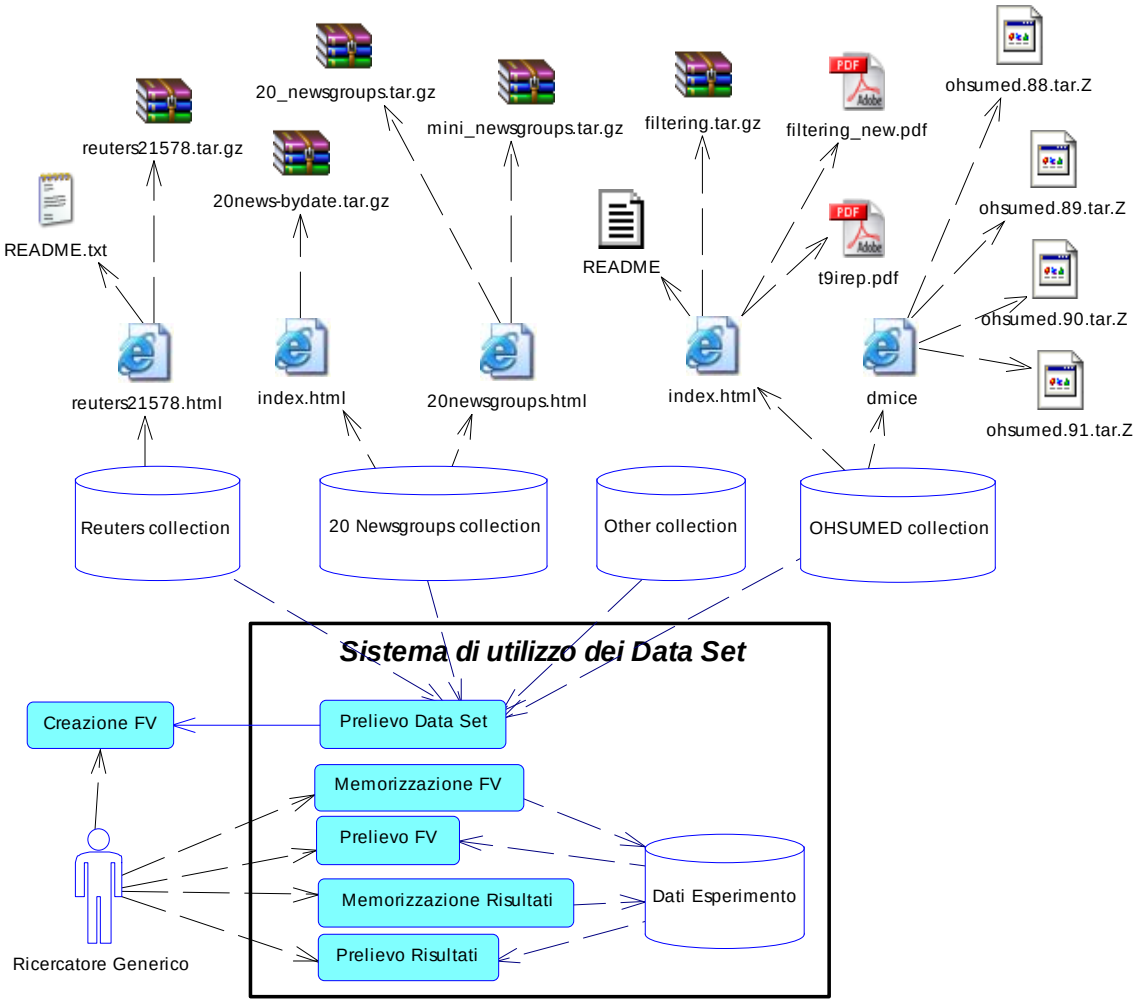


Figura 1: Business Process Model

Processes:

Il primo processo necessario ad un ricercatore è quello che permette di prelevare i dati dal benchmark di interesse. Il Processo **Prelievo Data Set** è descritto da due scenari. Il primo di questi mostra la necessità di realizzare del codice in grado di prelevare i dati che saranno adoperati per calcolare i *FV*:

- *Scenario 1* (assenza modulo):
  1. Il ricercatore interagisce con le varie collection per scegliere quale sia la più opportuna al suo esperimento.
  2. Scelta la collection procede a creare un modulo in grado di prelevare rapidamente i dati dei documenti.
  3. Preleva i dati dei documenti.
- *Scenario 2* (presenza modulo):
  1. Preleva i dati dei documenti.

Il processo **Creazione FV** è considerato esterno al "*Sistema d'utilizzo dei Data Set*", giacché le *FV* che saranno adoperate per un esperimento varieranno secondo le particolari analisi preliminari legate all'approccio *ML*. Il processo può essere descritto secondo lo scenario seguente:

1. Include(**Preleva Data Set**)
2. Ogni documento è formattato secondo le features individuate.

A tale punto il ricercatore ha la necessità di memorizzare i *FV* ricavati per non dovere ogni volta procedere al loro calcolo. Il processo **Memorizzazione FV** si preoccupa proprio di effettuare tale operazione:

- *Scenario 1* (mancanza modulo):
  1. Il ricercatore procede a creare un modulo in grado di memorizzare le *FV*.
  2. I *FV* sono memorizzati nella risorsa *"**Dati Esperimento**"*.
- *Scenario 2* (presenza modulo):
  1. I *FV* sono memorizzati nella risorsa *"**Dati Esperimento**"*.

Naturalmente, una volta memorizzati su un supporto i *FV*, il ricercatore dovrà adoperare i dati calcolati in precedenza per effettuare l'eventuale addestramento della propria *ML* e successivamente riadoperare tali *FV* anche per i test di classificazione da effettuare sulla *ML* ricavata. Queste operazioni richiedono un accesso rapido alla risorsa *"**Dati Esperimento**"*. Il processo **Prelievo FV** si preoccupa proprio di quest'operazione:

- *Scenario 1* (mancanza modulo):
  1. Il Ricercatore procede a creare un Modulo in grado di prelevare le FV.
  2. I FV sono prelevati dai "***Dati Esperimento***" e resi disponibili al ricercatore.
- *Scenario 2* (presenza modulo):
  1. I FV sono prelevati dai "***Dati Esperimento***" e resi disponibili al ricercatore.

In fine, i processi **Memorizzazione Risultati** e **Prelievo Risultati** si preoccupano di memorizzare e prelevare dalla risorsa "***Dati Esperimento***" i risultati delle classificazioni effettuate durante i test. In questo caso, come per i processi precedenti, esistono due scenari, dove il primo è necessario per creare dei moduli in grado di interagire con il supporto di memorizzazione.

1.4 Architettura di Massima del Sistema

Al fine di velocizzare l'analisi degli algoritmi relativi alla *TC* sarà, per quanto visto nel paragrafo precedente, necessario implementare i processi interni al "*Sistema d'utilizzo dei Data Set*" in modo che siano automatici e facilmente utilizzabili dai ricercatori. Nel *Component Diagram*, presente nella figura di seguito ed implementato nel file *ObjDataModel_DBDSATC v2.2.oom* inserito nel workspace DataSetDB.sws, si mostrano i moduli necessari alla realizzazione del prodotto:

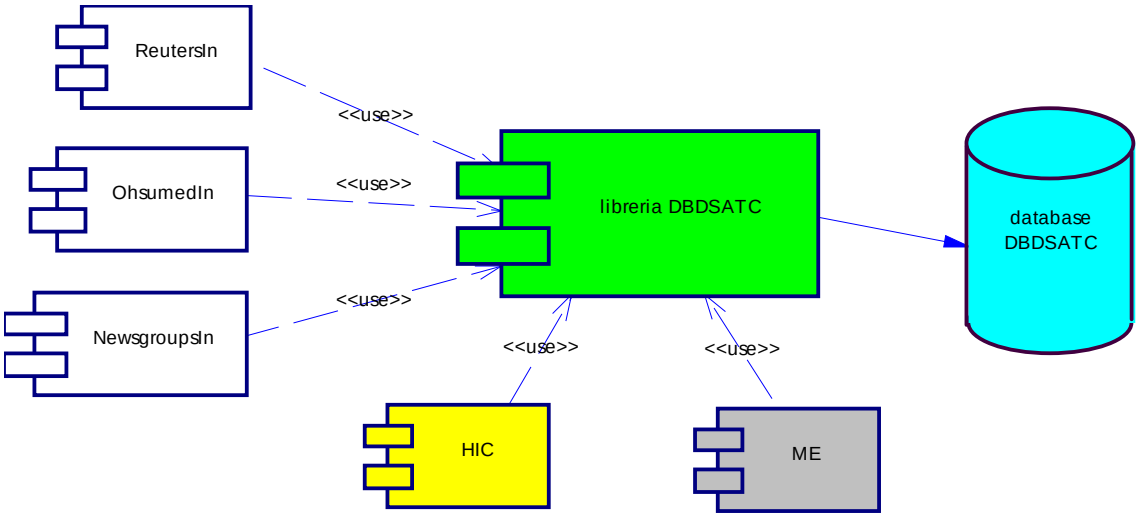


Figura 2: Component Diagram

Il termine **DBDSATC** è il nome del prodotto ed indica l'acronimo di DataBase Data Set Automated Text Categorization; con **ME** invece s'indicano i **Moduli Esterni** che saranno creati dai ricercatori. I tre sottoparagrafi che seguono riportano le principali linee guida, i vincoli e le tecnologie adoperate nella realizzazione del prodotto.

1.4.1 Linee Guida

1. **Interfacce del sistema**: il *modulo d'In/Out* (la *libreria DBDSATC*) realizzato per il prelievo e l'inserimento dei dati nel database deve avere interfacce realizzate in modo da permettere all'*HIC*, ai moduli di feeding, ad eventuali moduli di *TC* e committee, di interagire velocemente ed in maniera semplice col database.
2. **Caratteristiche d'utente**: il software è destinato a ricercatori dediti alla realizzazione di *TC*. Per l'utilizzo del *modulo d'In/Out* è richiesto lo studio della documentazione, fornita col prodotto, e la conoscenza dei linguaggi adoperati in modo da poter creare ulteriori moduli in grado di interagire con quello sviluppato. La gestione del database sarà affidata ad un *DBA* il quale oltre ad effettuare le operazioni di manutenzione dovrà all'occorrenza, se necessario, modificare la struttura del database tenendone conto in documenti di *release* che spieghino come e perché si siano effettuate delle modifiche al prodotto.
3. **Interfacce utente**: il *DBA* s'interfaccia al database tramite un'*HIC* avendo a sua disposizione un accesso indiscriminato ai dati, invece i ricercatori devono avere la possibilità di accedere tramite l'*HIC* solo ad ambienti informativi ed al menu che gli permette di creare dei report o di modificare gli esperimenti.
4. **Requisiti d'adattamento nella messa in funzione**: in futuro potrebbe essere utile creare dei moduli di feeding aggiuntivi per integrare nel database nuove collection. Tutto ciò richiede una documentazione accurata del *modulo d'In/Out*.

1.4.2 Vincoli del Progetto

1. **Vincolo strutturale**: uno dei vincoli più restrittivi è proprio la struttura del database che dovrà permettere non solo di inserire, senza alcuna perdita d'informazioni, i dati delle collection considerate, ma anche di immagazzinare dati relativi a future collection (ciò significa creare una struttura tanto flessibile da permettere l'inserimento di nuove relazioni nel database o facili modifiche di quelle esistenti).
2. **Conformità agli standards**: purtroppo per quanto riguarda le benchmark collection di data set non esiste uno standard di formattazione. Ciò lascia libertà d'azione ma anche la necessità, come già detto, di un'organizzazione dei dati generica, in modo da permettere il futuro inserimento di nuove collection.

3. **Affidabilità**: il sistema deve avere una buon'affidabilità poiché sarà il cuore di futuri esperimenti che saranno effettuati nell'ambito della *TC*.
4. **Sicurezza:** far sì che un ricercatore possa modificare solo i dati a lui necessari e solo quelli attinenti agli esperimenti cui appartiene.
5. **Mantenibilità:** la comprensibilità e la leggibilità sono fortemente richieste poiché su tale sistema dovranno andare a lavorare ricercatori che dovranno essere in grado di comprendere al meglio non solo le interfacce realizzate, ma anche il codice implementato in modo da poter agevolmente intervenire per future revisioni.

1.4.3 Tecnologie Adoperate

1. **Assunzioni e dipendenze:** il database dovrà essere posto su una macchina in grado di velocizzare i suoi processi, ciò perché i dati restituiti devono essere elaborati da moduli dediti all'addestramento di *ML* le cui funzioni già sono lente a prescindere dai tempi di latenza nel passaggio dei dati. A causa dell'ampia diffusione in ambiente universitario del pacchetto Office di Microsoft il database sarà realizzato utilizzando il **DBMS Access 2000**; ciò implica che si dovrà utilizzare un software in grado di supportare tale pacchetto applicativo.
2. **Portabilità**: in futuro potrebbe essere necessario adoperare il sistema su macchine *UNIX*, ciò richiede una certa attenzione nelle scelte da effettuare. Il progetto deve, dove è possibile, usare delle librerie C++ standard (almeno per quanto riguarda il *modulo d'In/Out*) per permettere la portabilità del codice su piattaforme *UNIX*. La gestione del database deve avvenire tramite l'utilizzo delle sole funzioni appartenenti agli *standard SQL*, ciò permetterà, nel caso del cambiamento del DBMS utilizzato, di recuperare il codice del *modulo d'In/Out*.
3. **Interfacce software**: sempre per agevolare la portabilità del prodotto si è deciso di utilizzare un'interfaccia *ODBC* per collegare il *modulo d'In/Out* al database. Ciò inoltre permetterà al sistema di funzionare anche su reti locali. La realizzazione del codice (effettuata in C++) avviene dunque tramite l'utilizzo delle *API ODBC*.
4. **CASE**: il processo di sviluppo sarà eseguito utilizzando *PowerDesigner 9.0* per l'analisi e la progettazione e l'ambiente di sviluppo *Microsoft Visual C++ 6.0* per quanto riguarda l'implementazione del codice.

1.5 Passi del Processo di Sviluppo

I capitoli della presente tesi seguono dove possibile gli step effettuati per lo sviluppo del prodotto. Si fa notare che i primi sei punti del seguente elenco sono stati effettuati per mezzo di un modello di sviluppo di tipo *Do-It-Twice,* necessario a causa dei numerosi vincoli, generati dal **DBMS Access 2000**, che hanno non poco condizionato le decisioni riguardo alla realizzazione del database e di conseguenza lo sviluppo del codice. La tesi non si preoccuperà di descrivere i prototipi realizzati, che però all'occorrenza saranno richiamati per giustificare le decisioni prese:

1) **Individuazione dei modelli concettuali dei benchmark di riferimento**: tale step si preoccupa di individuare per ogni benchmark selezionato un modello concettuale da utilizzare per realizzare un database che lo contenga. Tali modelli permetteranno di individuare le entità chiave e gli attributi d'ogni collection, oltre ad i vincoli indispensabili.
2) **Individuazione delle funzionalità richieste**: una volta realizzati i tre modelli si procede all' individuazione delle funzionalità comuni, di quelle necessarie e di quelle da aggiungere al modello concettuale del database per implementare i "***Dati Esperimento***" introdotti nel *paragrafo 1.3*. Tale passo è richiesto per comprendere cosa ci si aspetti dal *modulo d'In/Out*.
3) **Progettazione del database *DBDSATC***: in questo step ci si preoccupa di effettuare un merge dei modelli di riferimento e di implementare un modello E-R che sia in grado di inglobare nel sistema anche i "***Dati Esperimento***" cosi come formalizzati nello step 2. In tale fase s'individueranno anche i vincoli e le decisioni riguardo all'implementazione di questi ultimi. Il risultato della progettazione sarà l'insieme dei comandi relativi alle **DDL** da utilizzare per realizzare il database (come già detto tali comandi saranno solo quelli relativi allo *standard SQL*, in modo da poter cambiare a piacere il **DBMS** settato in *PowerDesigner* e realizzare tramite *ODBC* un nuovo database).
4) **Creazione del modulo d'In/Out (*libreria DBDSATC*)**: in tale fase si realizza l'*Object-Oriented Model* alla base del *modulo d'In/Out* ed il relativo codice. In questo step si procede anche ad un'accurata descrizione delle classi rese disponibili dalla libreria realizzata.
5) **Moduli di Feeding**: realizzazione dei moduli di feeding tramite l'utilizzo della libreria ricavata allo step 4.

6) **Settaggio dei domini del database**: tramite l'utilizzo del prototipo realizzato durante la prima fase del modello *Do-It-Twice* si procede al ridimensionamento dei domini utilizzati nel database. Nella tesi si riportano i risultati ottenuti adottando la libreria finale e non quella realizzata nel prototipo.
7) **HIC (*GestoreDSATC*)**: descrizione dell'interfaccia grafica realizzata tramite l'utilizzo della **libreria MFC**. L'obiettivo di questo prodotto è di creare un interfaccia che permetta di verificare le *funzionalità* del sistema implementate.

1.6 Documentazione della Tesi

La documentazione ed i file di riferimento sono riportati in un CD diviso in due directory "*Archivio File*" e "*Processo di Sviluppo DBDSATC*"; nella prima si trovano i diversi articoli adoperati per lo studio del dominio applicativo e i file attinenti alle collection di riferimento adoperate; nella seconda si trovano tre directory: una (*DB & Creazione*) contenete la versione conclusiva del DB; un'altra (*Progettazione DB*) contenente i file di sviluppo del software realizzati tramite il PowerDesegner 9.0 e organizzati nel workspace "DataSetDB.sws", ed un ultima (*GestioneDS*) contenete l'implementazione. La directory "*Progettazione DB*" contiene inoltre altre tre cartelle:

- *Codice*: contenente i file sorgenti (**.cpp**) e gli header (**.h**) che implementano le classi create in *PowerDesigner*.
- *DB Settato*: contenente il database realizzato, completo di tutti i vincoli e della struttura per gestire i permessi, ma privo di dati.
- *Prototipi*: contenente il codice dei prototipi realizzati in PowerDesigner.

Altre informazioni, riguardanti la documentazione realizzata durante il ciclo di sviluppo del sistema, saranno riportate all'occorrenza.

CAPITOLO 2:

LA COLLECTION REUTERS-21578

2.1 Il Benchmark
2.2 Stato Iniziale dei Dati
2.3 Schema Logico "Ipotetico"
2.4 Entità Aggiuntive
2.5 Schema Concettuale

Il capitolo è dedicato all'individuazione di uno schema concettuale in grado di descrivere al meglio la struttura della collezione analizzata. Il procedimento (che sarà ripetuto anche nello studio delle altre collezioni di riferimento considerate) effettua un reverse engineering partendo dallo schema fisico non di un database, ma dell'organizzazione dei dati in file (del resto anche se non si utilizza un DBMS si parla pur sempre di un "database" contenente informazioni).

2.1 Il Benchmark

I documenti contenuti nella collection sono coperti da copyright e messi a disposizione dalla *Reuters Ltd. and Carnegie Group*; il loro utilizzo è consentito solo per effettuare ricerche nell'ambito della *TC*. Il gruppo **Reuters** richiede a coloro che utilizzeranno tale data set di evidenziare l'origine dei dati utilizzati. Il benchmark contiene 21578 articoli della rivista (una versione precedente ne conteneva 22173) classificati secondo cinque vocabolari distinti.

Dal file "*readme.txt*", messo a disposizione insieme al benchmark, si possono estrarre molte informazioni relative ai diversi *split* dei dati disponibili ed all'individuazione di data set utilizzati in esperimenti precedenti, i cui risultati sono considerati come un buon metro per verificare l'efficacia di nuovi algoritmi per la realizzazione di classificatori del tipo **DPC**.

2.2 Stato Iniziale dei Dati

Il file compresso *reuters21578.tar.gz* contiene un insieme di file che permettono di ricavare i documenti e le categorie utilizzate.

Le informazioni relative ai documenti sono contenute in 22 file (i cui nomi vanno da "reut2-000.sgm" a "reut2-021.sgm"), ognuno contenente 1000 documenti (tranne l'ultimo) e formattato per mezzo dei tags **SGML** descritti nel file "lewis.dtd". Si riportano gli attributi individuati ed una loro breve descrizione:

- **REUTERS TOPICS**: tale attributo serve per conservare le informazioni riguardanti l'appartenenza alle classi del vocabolario TOPICS della collection Reuters-22173; i suoi valori possono essere *YES* (collegamento ad almeno un termine del vocabolario), *NO* (nessun legame), *BYPASS* (mancata verifica).
- **REUTERS LEWISSPLIT**: i possibili valori sono *TRAIN* (vi è un errore nella descrizione di tale valore che mentre nel file "*readme.txt*" è dichiarato come TRAINING, nei documenti è TRAIN), *TEST*, e *NOT-USED*. TRAIN indica che il documento è usato nel training set degli esperimenti riportati in LEWIS91d, LEWIS92b, LEWIS92e, e LEWIS94b. TEST indica che è usato nel test set degli stessi esperimenti, e NOT-USED significa che non è adoperato per gli esperimenti menzionati.

- **REUTERS CGISPLIT**: i valori possibili sono *TRAINING-SET e PUBLISHED-TESTSET* che indicano se un documento fa parte del training set o del test set degli esperimenti riportati in HAYES89 e HAYES90b.
- **OLDID**: il numero identificativo (ID) che il documento ha nella collezione Reuters-22173.
- **NEWID**: il numero identificativo (ID) che il documento ha nella distribuzione 1.0 della collezione Reuters-21578. Tale numero è assegnato ai documenti in ordine cronologico.
- **TOPICS**: include una lista delle categorie del vocabolario *TOPICS*.
- **EXCHANGES**: include una lista delle categorie del vocabolario *EXCHANGES*.
- **ORGS**: include una lista delle categorie del vocabolario *ORGS*.
- **PEOPLE**: include una lista delle categorie del vocabolario *PEOPLE*.
- **PLACES**: include una lista delle categorie del vocabolario *PLACES*.
- **DATE**: include la data ed il tempo del documento.
- **TEXT TYPE**: i valori possibili sono *NORM*, *BRIEF*, *UNPROC*, *UNKNOWN* e *BLAH*. NORM è il valore di default ed indica che il testo del documento ha una struttura normale. BRIEF indica che la storia è una breve nota di una, due linee. UNPROC si ha quando il formato della storia è particolare. I valori UNKNOWN e BLAH non sono adoperati ma sono menzionati nel file "lewis.dtd" (che, come detto, descrive la formattazione in **SGML** dei documenti).
- **LOCATION**: indica la locazione della storia ed è presente in un campo chiamato DATELINE, contenete anche l'informazione ridondante riguardo alla data.
- **AUTHOR**: indica l'autore del documento.
- **TITLE**: riporta il titolo del documento.
- **BODY**: contiene il testo del documento (nel caso di TEXT TYPE uguale a BRIEF, tale campo contiene il valore "Blah blah blah."):

Per quanto riguarda le categorie utilizzate per le classificazioni si riportano i cinque vocabolari adoperati ed i relativi file dove sono memorizzati i loro elementi:

EXCHANGES	*all-exchanges-strings.lc.txt*
ORGS	*all-orgs-strings.lc.txt*
PEOPLE	*all-people-strings.lc.txt*
PLACES	*all-places-strings.lc.txt*
TOPICS	*all-topics-strings.lc.txt*

La descrizione dei vocabolari non è presente in tali file (che sono semplici elenchi di termini), ma in *"cat-descriptions_120396.txt"* che, oltre alle informazioni riguardo il significato delle numerose sigle, permette di individuare un'organizzazione gerarchica delle categorie.

Per quanto riguarda le difficoltà incontrate nell'estrazione delle informazioni si rimanda al capitolo riguardante i moduli di feeding.

2.3 Schema Logico "Ipotetico"

Una volta individuata la struttura dei file della collezione si è realizzato uno schema logico "ipotetico"; questo passo è stato necessario per ricavare lo schema concettuale di un database in grado di contenere le informazioni del benchmark.
Gli schemi riportati nel capitolo sono stati implementati nei file *PhysicalDataModel_Reuters.pdm* e *ConceptualDataModel_Reuters.cdm* inseriti nel workspace "DataSetDB.sws"

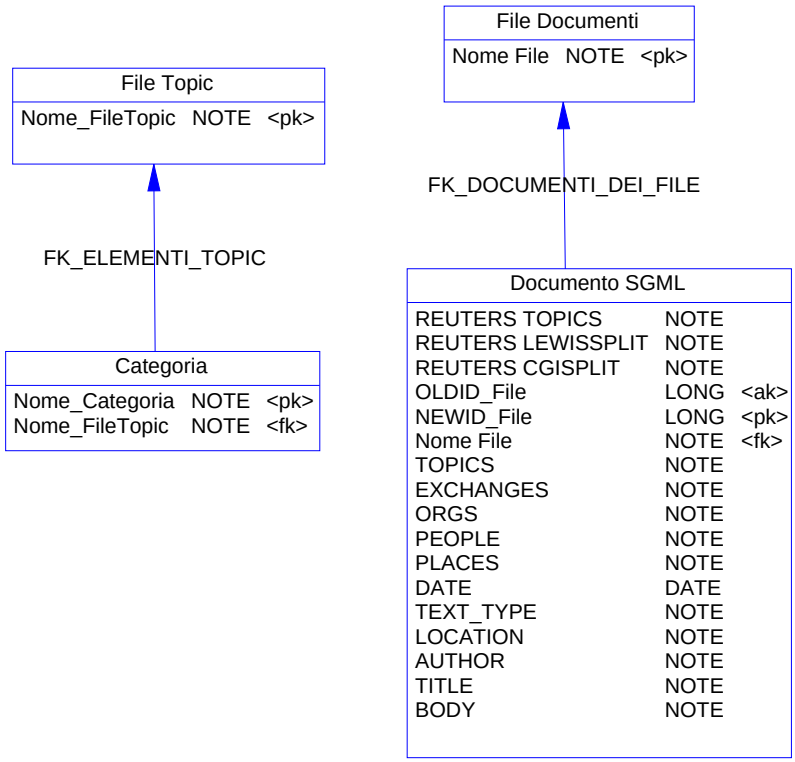


Figura 3: Schema Logico del Reuters-21578

Tale schema rappresenta quanto indicato nel paragrafo precedente senza tener conto delle informazioni aggiuntive presenti nel file "readme.txt"; prima di procedere allo studio delle nuove entità contenutevi, si è passati ad uno schema concettuale, considerato più idoneo alla descrizione di un modello E-R..

Si osservi che volutamente si è deciso di individuare, in entrambi gli schemi, solo la tipologia degli attributi e non la loro dimensione, ciò perché tale informazione non è riportata dai creatori della collezione. L'analisi delle dimensioni dei campi sarà effettuata in un secondo momento.

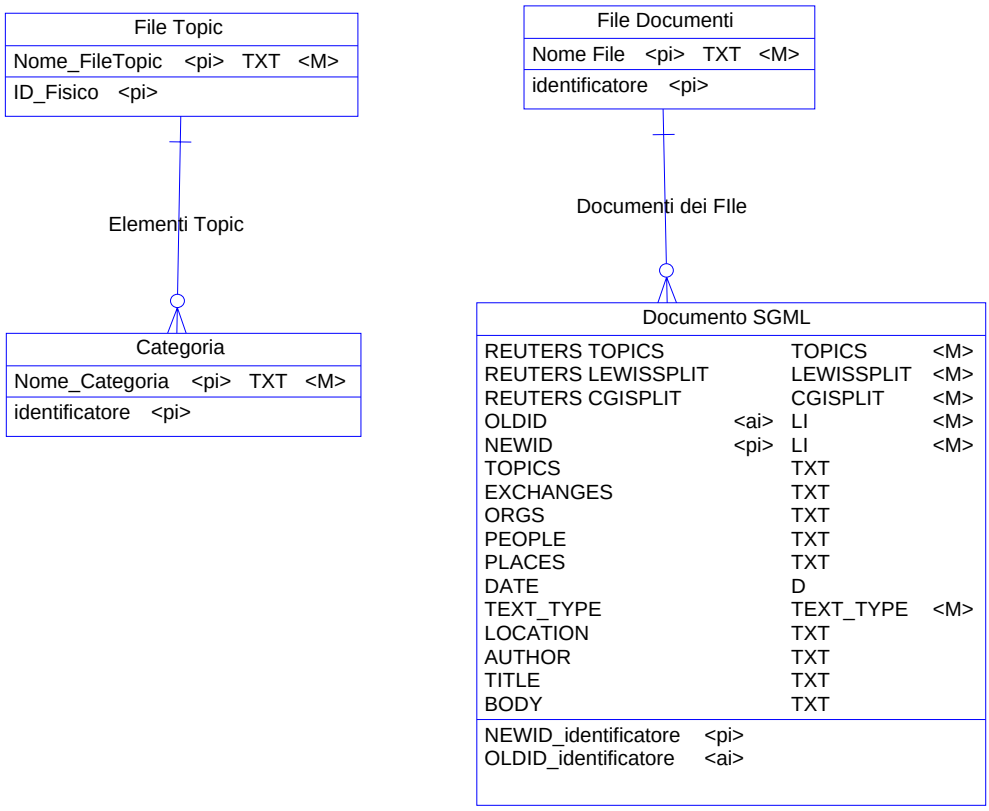


Figura 4: Schema Concettuale del Reurers-21578

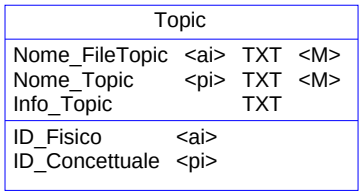
2.4 Entità Aggiuntive

Uno studio più accurato delle informazioni, rese disponibili nei file di descrizione del benchmak, ha permesso di identificare delle entità aggiuntive e un'organizzazione dei dati più complessa di quanto sia apparsa fin ora.

2.4.1 Topic & Categorie

Category Set	Number of Categories	Number of Categories w/ 1+ Occurrences	Number of Categories w/ 20+ Occurrences
EXCHANGES	39	32	7
ORGS	56	32	9
PEOPLE	267	114	15
PLACES	175	147	60
TOPICS	135	120	57

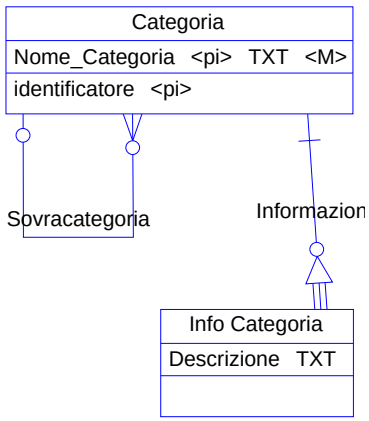
L'entità individuata dovrebbe essere come quella rappresentata nella figura.



Le informazioni aggiuntive che si trovano nel file "*cat-descriptions_120396.txt*" indicano che in realtà vi sono solo due diversi insiemi di categorie, uno che contiene tutti i vocabolari e l'altro che utilizza solo *TOPICS*:

Nome Topic	Descrizione Topic
Economics (HAYES90b)	Il totale di 674 categorie menzionate nel HAYES90b, è la somma delle 135 del HAYES89 più le altre 539 indicate in precedenza
Topics (HAYES89)	Una lista di 135 categorie in ambito economico che sono state usate nel HAYES89

Inoltre l'organizzazione delle categorie, presentata nel file indicato, permette di individuare una struttura gerarchica di queste ultime (da qui l'aggiunta della relazione Sovracategoria). In seguito a quanto riportato i 5 vocabolari indicati in precedenza non devono essere considerati dei **Topic**, ma sovracategorie principali di strutture ad albero. Tra le informazioni riportate inoltre è possibile ricavare per alcuni vocaboli (di cui molti sono acronimi) delle spiegazioni riguardo al loro significato. Quanto detto consiglia di aggiungere un'entità in grado di conservare le informazioni della categoria..



2.4.2 Split

I documenti del benchmark sono divisibili in diversi **Split** tramite l'utilizzo dei tre attributi *REUTERS TOPICS*, *REUTERS LEWISSPLIT* e *REUTERS CGISPLIT*. Un primo insieme di **Split** si riferisce proprio al significato dei termini TOPICS, LEISSPLIT e CGISPLIT e deve poter essere implementato nello schema affinché non si abbiano perdite d'informazioni riguardanti la collezione:

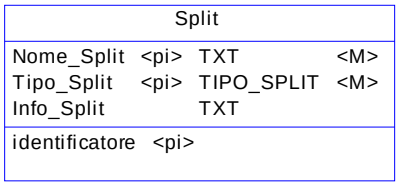
Nome Split	Valore Attributo	Descrizione Split
TOPICS assenti	REUTERS TOPICS = NO	Indica l'assenza di elementi della classe TOPICS nel Reuters-22173
TOPICS Bypass	REUTERS TOPICS = BYPASS	Indica la mancata determinazione del TOPICS nel Reuters-22173
TOPICS presenti	REUTERS TOPICS = YES	Indica la presenza di elementi della classe TOPICS nel Reuters-22173
LEWISSPLIT Tr	REUTERS LEWISSPLIT = TRAIN	Indica il Training Set adoperato da Lewis nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazioni avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b.
LEWISSPLIT Te	REUTERS LEWISSPLIT = TEST	Indica il Test Set adoperato da Lewis nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazioni avutesi per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b.
CGISPLIT Tr	REUTERS CGISPLIT = TRAINING-SET	Indica il Training Set adoperato da Hayes nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazione avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti HAYES89, HAYES90b.
CGISPLIT Te	REUTERS CGISPLIT = PUBLISHED-TESTSET	Indica il Test Set adoperato da Hayes nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazioni avutesi per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti HAYES89, HAYES90b.

Un secondo insieme si riferisce ad alcune combinazioni dei tre attributi e consente di individuare gli **Split** adoperati per i diversi **Data Set** contenuti nella collezione:

Nome Split	Valore Attributo	Descrizione Split
ModApte Tr	REUTERS LEWISSPLIT=TRAIN REUTERS TOPICS=YES	The Modified Apte (ModApte) Split Tr indica il Training Set adoperato da Apt adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti APTE94 e APTE94b con il Reuters-22173.
ModApte Te	REUTERS LEWISSPLIT=TEST REUTERS TOPICS= YES	The Modified Apte (ModApte) Split Te indica il Test Set adoperato da Apt adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti APTE94 e APTE94b con il Reuters-22173.
ModHayes Tr	REUTERS CGISPLIT = TRAINING-SET	The Modified Hayes (ModHayes) Split indica il Training Set adoperato da Hayes adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti HAYES89, HAYES90b con il Reuters-22173.

Nome Split	Valore Attributo	Descrizione Split
ModHayes Te	REUTERS CGISPLIT = PUBLISHED-TESTSET	The Modified Hayes (ModHayes) Split indica il Test Set adoperato da Hayes adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti HAYES89, HAYES90b con il Reuters-22173.
ModLewis Tr	REUTERS LEWISSPLIT = TRAIN TOPICS = YES or NO	The Modified Lewis (ModLewis) Split Tr indica il Training Set adoperato da Lewis adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b con il Reuters-22173.
ModLewis Te	REUTERS LEWISSPLIT = TEST TOPICS = YES or NO	The Modified Lewis (ModLewis) Split Te indica il Test Set adoperato da Lewis adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b con il Reuters-22173.

Da qui nasce l'esigenza di creare un'entità che sia in grado di memorizzare queste informazioni.



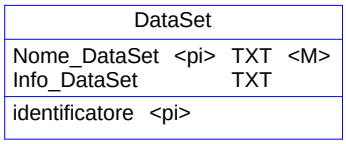
*Il dominio **TIPO_SPLIT*** indicato nella rappresentazione dell'entità deve permettere di implementare i diversi tipi di **Split** fino ad ora indicati. I valori attribuiti a tale dominio sono TRAIN, TEST, BYPASS, YES e NO.

Per quanto riguarda il primo insieme di **Split** individuato i valori dell'attributo *Tipo_Split* hanno richiesto l'utilizzo delle conversioni riportate di seguito:

- Per **REUTERS CGISPLIT** si ha che TRAINING-SET diviene *TRAIN* e PUBLISHED-TESTSET diviene *TEST*.
- Per **REUTERS LEWISSPLIT** si elimina NO-USED (dato ridondante).
- Per **REUTERS TOPICS** si conservano i valori originali (si forzatura il concetto di **Split** appartenenti ad un **Data Set** al fine di conservare le informazioni memorizzate nel campo; del resto la definizione di **Split** si riferisce all'individuazione di sottoinsiemi disgiunti di un particolare insieme di dati e non alla costruzione di un training e test set).

Invece per il secondo insieme di **Split** ricavati l'attributo *Tipo_Split* può avere il valore TRAIN o TEST, a seconda che ci si riferisca ad un training set o test set.

2.4.3 Data Set



Gli **Split** ricavati dalle combinazioni dei tre attributi REUTERS sono riportati al fine di definire i tre **Data Set** definiti nella collection:

- **The Modified Lewis** (ModLewis) che adopera gli split ModLewis
- **The Modified Apte** (ModApte) che adopera gli split ModApte
- **The Modified Hayes** (ModHayes) Split che adopera gli split ModHayes

Il termine *modified* sta ad indicare che i **Data Set** sono stati adattati all'attuale collezione partendo da quelli del benchmark Reuters-22173.

L'ultima nota di rilievo su quest'entità riguarda il **Topic** adoperato da ognuno dei **Data Set** riportati, che per i primi due è *"Topics (HAYES89)"* mentre per l'ultimo è *"Economics (HAYES90b)"*.

2.5 Schema Concettuale

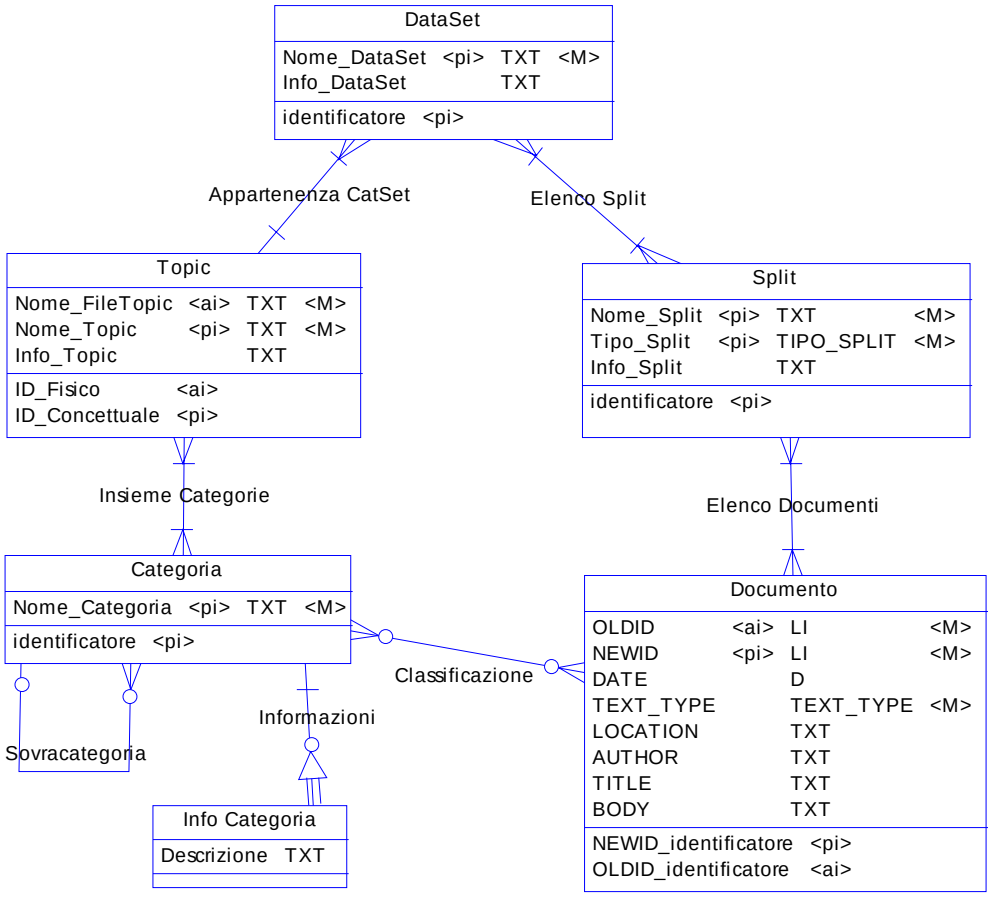


Figura 5: Schema Concettuale Ristrutturato del Reurers-21578

Nello schema si può osservare che l'entità *"Documento SGML"* dello schema logico è stata sostituita da quella **Documento**, che è stata alleggerita degli attributi REUTERS relativi agli **Split**. Tale decisione permette non solo di svincolare il documento dagli **Split** che lo adoperano, ma di creare un modello che sia in grado di definire nuovi **Split** senza perdere le informazioni fornite dalla collezione.

Si osservi che *Il dominio* ***TEXT_TYPE*** non fa altro che definire l'attributo già presentato nel *paragrafo 2.2*, i cui valori come già detto possono essere *NORM*, *BRIEF*, *UNPROC*, *UNKNOWN* e *BLAH.*

Infine i *campi info* si riferiscono ai dati prelevati dai file che descrizione il benchmark e dai siti internet attinenti alla collection.

CAPITOLO 3:

LA COLLECTION 20 NEWSGROUPS

3.1 Il Benchmark

3.2 Stato Iniziale dei Dati

3.3 Schema Logico "Ipotetico"

3.4 Entità Aggiuntive

3.5 Ristrutturazione delle Entità

3.6 Schema Concettuale

In questo capitolo, come nel precedente, si cercherà di ottenere uno schema concettuale in grado di rappresentare la seconda collezione di riferimento considerata. Le decisioni riguardo alla struttura del modello saranno prese svincolandosi dalla trattazione precedente in modo da poter avere, al momento della creazione del database finale, un'immagine dettagliata di diverse tipologie di benchmark così da essere in grado di realizzare un modello che consideri punti di partenza molto diversi tra loro.

3.1 Il Benchmark

Il materiale del benchmark è messo a disposizione per scopi educativi e ,come per il precedente, viene richiesto di aggiungere in ogni pubblicazione di esperimenti adoperanti tale materiale la sorgente dei Data Set adoperati.
l'*UCI KDD Archivi* attribuisce la creazione della collection a Tom Mitchell, ma nei file messi a disposizione si accenna a Ken Lang come creatore della collezione.
La 20 newsgroups collection contiene un popolare data set, adoperato nell'ambito della text categorization e del text clustering per realizzare classificatori del tipo **DPC**.
Il benchmark è presentato come un unico data set di 19997 messaggi, suddiviso in 20 differenti *Netnews* newsgroups, ognuno contenente circa 1000 messaggi. Ogni newsgroup è rappresentato da una diversa directory, mentre ogni messaggio si trova in un diverso file contenuto nella directory del newsgroup d'appartenenza.

3.2 Stato Iniziale dei Dati

Vi sono più versioni della collezione, ed ognuna contiene un certo numero di documenti organizzati in un unico data set o in test-set e training-set. Gli attributi dei documenti sono diversi (in numero) tra le varie versioni ritrovate.

Nome File	Descrizione del File
20_newsgroups.tar.gz	File contenente il Data Set originale
mini_newsgroups.tar.gz	Un tarred e gzipped subset del Newsgroup data che contiene 100 messaggi selezionati a caso per ogni newsgroup. Questo split è usualmente adoperato per gli addestramenti quando si adopera *Rainbow*, un programma adoperato dal **CMU Text Learning Group**
20news-bydate.tar.gz	In tale file i documenti sono ordinati per data e divisi in training (60% dei documenti) e test (40%) sets, non includono cross-posts (duplicates) e newsgroup-identifying headers (Xref, Newsgroups, Path, Follow-up-To, Date).

L'unica costante delle varie versioni è l'insieme dei venti newsgroups utilizzati come categorie ed individuabili dal nome delle directory contenenti i file dei messaggi. Altri dati relativi ai newsgroups si possono ricavare dallo studio delle strutture degli *Usenet*, le cui informazioni sono facilmente rintracciabili sui siti dei più noti motori di ricerca. Di seguito si riporta l'elenco delle categorie individuate:

alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc

Nel caso del *"20news-bydate.tar.gz"* l'organizzazione delle directory è diversa, infatti il file compresso contiene due directory una denominata 20news-bydate-train e l'altra 20news-bydate-test, ognuna contenente le venti directory rappresentanti i newsgroups riportati in precedenza.

La mancanza di documentazione riguardo alla collezione non permette di ricavare informazioni precise sulla struttura dei singoli messaggi; in seguito all'analisi di diversi documenti si è dedotto che la formattazione dei singoli attributi è data dal formalismo:

< NOME_ATTRIBUTO > < : > < spazio > < valore attributo >

e che ognuno degli attributi si trova su una singola riga. Le diverse righe degli attributi si susseguono in modo casuale, dopo l'ultimo si ha poi una riga, priva di caratteri, che avvisa l'inizio del testo del messaggio.

Non è stato possibile identificare tutti gli attributi con accuratezza, ma cercando di prelevare quelli più rilevanti e costanti si è pervenuti alla seguente lista:

- **Organization**: indica l'organizzazione cui fa riferimento il messaggio.
- **From**: indica l'autore del messaggio.
- **Data**: indica la data di creazione del messaggio.
- **Subject**: indica l'argomento del messaggio.
- **Testo**: si riferisce al testo del messaggio.
- **Lines**: indica il numero di linee del messaggio.
- **Keywords**: contiene una lista delle keyword associate al messaggio.

Per quanto riguarda le difficoltà incontrate nell'estrazione delle informazioni si rimanda al capitolo riguardante i moduli di feeding.

3.3 Schema Logico “Ipotetico”

Una volta individuata la struttura dei file della collezione si è realizzato uno schema logico “ipotetico“; questo passo è stato necessario per ricavare lo schema concettuale di un database in grado di contenere le informazioni del benchmark.

Lo schema viene arricchito con varie informazioni riguardanti le strutture fisiche delle directory presenti nei file compressi; inoltre, dal momento che alcune versioni della collezione individuano semplicemente un data set non diviso in split significativi (cioè training e test set), si è deciso di aggiungere alla struttura dell’entità **DirectorySplit** un attributo *Tipo Split* che permetta di indicare, tramite l’utilizzo di tre valori, se la directory contiene un *Data Set*, un *Training Set* o un *Test Set*.

Gli schemi riportati nel capitolo sono stati implementati nei file *PhysicalDataModel*_20_Newsgroups.*pdm* (contenente lo schema logico “ipotetico”) e *ConceptualDataModel*_20_Newsgroups.*cdm* (contenente lo schema concettuale) inseriti nel workspace “DataSetDB.sws”.

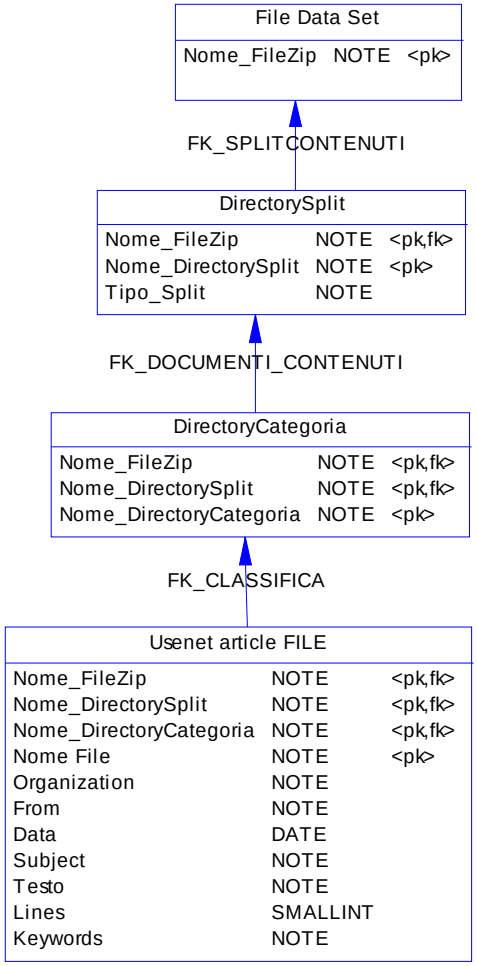


Figura 6: Schema Logico del 20 Newsgroups

La dipendenza dell'"**Usenet articles FILE**" dall'entità **DirectoryCategoria** si deve alla presenza, nelle diverse versioni della collezione, di file contenenti gli stessi messaggi ma con un numero di attributi diversi.

Dallo schema logico si è ricato un primo schema concettuale adoperato come punto di partenza per la costruzione del modello E-R finale del benchmark.

Per entrambi gli schemi si sono individuate, come per il modello del capitolo precedente, solo le tipologie degli attributi e non le loro dimensioni, la cui conoscenza non è stata fornita dagli ideatori della collezione.

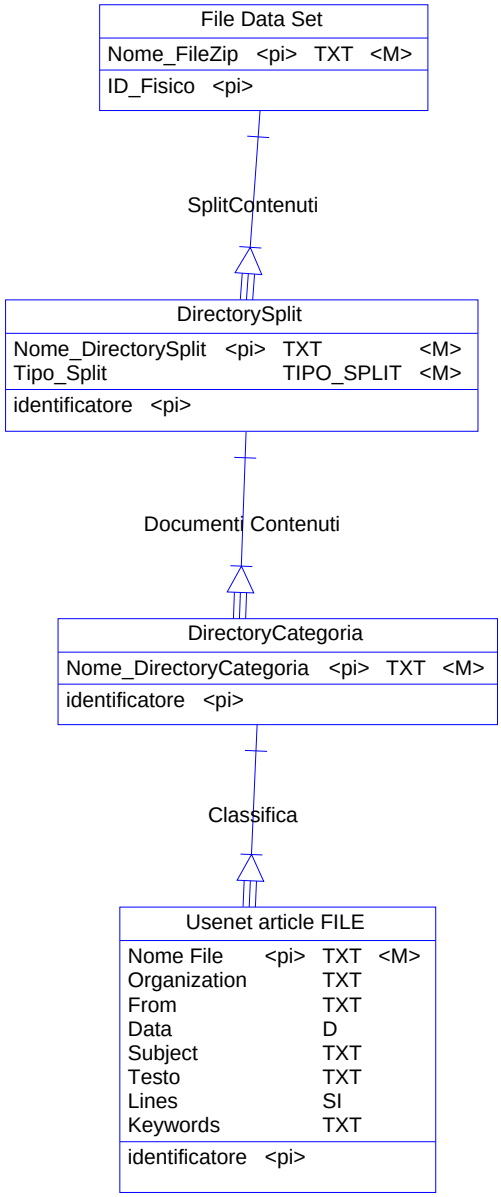


Figura 7: Schema Concettuale del 20 Newsgroups

Si osservi che nell'ultimo schema si è deciso di definire l'attributo *Tipo Split*, descritto in precedenza, tramite il *dominio* ***TIPO_SPLIT***.

3.4 Entità Aggiuntive

A differenza del benchmark Reuters-21578 tale collezione già dallo schema logico mostra tutte le sue entità principali. Le uniche due entità che vale la pena inserire si riferiscono all'organizzazione dei newsgroup.

- **Newsgroup Set**: l'entità serve ad identificare sottoinsiemi di newsgroup. Il suo inserimento è necessario a causa della presenza (non costante), all'interno dei documenti, di un attributo che identifica la lista dei newsgroup d'appartenenza. Poiché la lista contiene anche newsgroup non inclusi nell'insieme di quelli consentiti per il **Data Set** (che come già detto è composto sempre degli stessi 20), si usano due tuple dell'entità per distinguere tale insieme da un secondo contenente tutti i newsgroups esistenti.

  L'entità è paragonabile a quella **Topic** della collezione Reuters-21578, in tale caso però si dovrebbe considerare solo la tupla che si riferisce ai venti newsgroup usati nei **Data Set**, giacché solo questi sono riconosciuti come classi dei documenti presentati.

Newsgroups Set
Nome_Set <pi> TXT <M>
identificatore <pi>

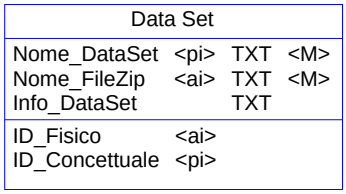
- **Info Categoria**: entità che si riferisce alle informazioni, sui singoli newsgroup, ricavabili dallo studio delle strutture degli *Usenet*.

Info Categoria
Info_Categoria TXT <M>

3.5 Ristrutturazione delle Entità

Il primo schema ricavato è molto ridondante e poco adatto a memorizzare le informazioni della collection giacché è fortemente vincolato dall'organizzazione in directory e file dei dati del benchmark. Nei sottoparagrafi che seguono si effettua una ristrutturazione delle entità definendo dove è possibile le tuple individuate tramite lo studio delle poche informazioni rese a disposizione nelle pagine web di presentazione delle diverse versioni della collezione.

3.5.1 Data Set



L'entità è arricchita con attributi che permettono di dare una più chiara interpretazione alle sue tuple. Si aggiunge oltre all'attributo *Nome_DataSet* anche un attributo *Info_DataSet* in grado di descrivere in grandi linee la tupla.

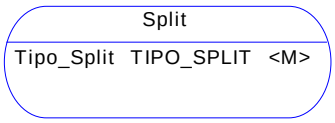
Nome Data Set	Nome File Zip	Descrizione Data Set
20 Newsgroups by date	20_newsgroups.tar.gz	I documenti sono ordinati per data e divisi in un training(60%) e test(40%) sets, non includono cross-posts (duplicates)
All 20 Newsgroups	mini_newsgroups.tar.gz	Contiene tutti i documenti della collection 20 Newsgroups associati a tutte le categorie.
Mini 20 Newsgroups	20news-bydate.tar.gz	Contiene un sottoinsieme di documenti della collection 20 Newsgroups associati a tutte le categorie.

Tutti i **Data Set** adoperano come insieme delle classi il **Newsgroup Set** contenente i venti newsgroup adoperati come categorie della collezione.

3.5.2 Split

Utilizzando come documenti di riferimento quelli presenti nel benchmark originale, che è l'unico i cui file "messaggi" contenengono al loro interno tutti gli attributi, si è deciso di convertire l'entità **DirectorySplit** in un associazione molti a molti, tra **Data Set** e **Usenet Articles**, contenente come unico attributo *Tipo_Split*.

Un **vincolo aggiuntivo** su tale campo è che se il **Data Set** non è diviso in training-set e test-set il suo valore deve essere "*Data Set*", altrimenti può essere scelto solo tra "*Test Set*" e "*Training Set*".



Le definizioni degli **Split** sono molto legate a quelle dei **Data Set**, un'immediata verifica di quanto detto si può avere osservando l'insieme degli **Split** individuati e riportato di seguito:

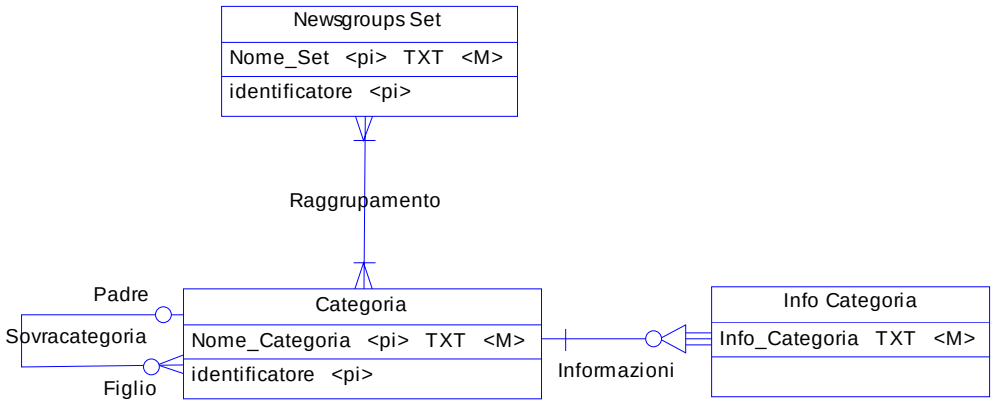
Nome Split	Tipo Split	Descrizione Split
20 Newsgroups by date Te	TEST SET	Il Test Set contiene il 40% dei documenti della collection e rimuove alcuni duplicati e campi.
20 Newsgroups by date Tr	TRAINING SET	Il Training Set contiene il 60% dei documenti della collection e rimuove alcuni duplicati e campi.

All 20 Newsgroups	DATA SET	Contiene tutti i documenti della collection 20 Newsgroups
Mini 20 Newsgroups	DATA SET	Un tarred e gzipped subset del Newsgroup data che contiene 100 messaggi selezionati a caso per ogni newsgroup. Questo split è usualmente adoperato per gli addestramenti.

La presenza nella collezione Reuters-21578 dell'**entità Split** probabilmente richiederà l'utilizzo di queste informazioni nella realizzazione del database finale:

3.5.3 Categoria

I newsgroups sono organizzati in strutture gerarchiche che richiedono delle dipendenze ricorsive (**Sovracategoria**) sull'entità **Categoria**. Per quanto riguarda le entità aggiuntive **Info Categoria** è in relazione dipendente con la **Categoria** di appartenenza e **Newsgroup Set** è in relazione molti a molti con **Categoria**.



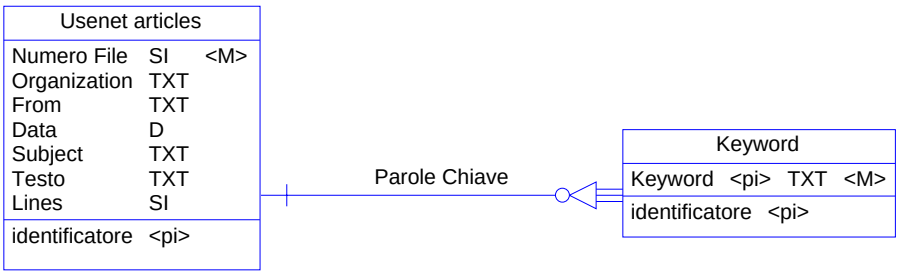
Per implementare l'organizzazione gerarchica delle categorie si procede all'estrazione delle sovracategorie individuate nei terminidei venti newsgroup della collezione; il campo descrizione della tabella presentata di seguito si riferisce ad informazioni ricavate dallo studio della struttura delle *Usenet*:

Nome Categoria	Sovracategoria	Descrizione
alt		Elenco degli argomenti disponibili
alt.atheism	alt	Ateismo
comp		Hardware, software, info per i consumatori
comp.graphics	comp	Graphics
comp.os	comp	Sistemi operativi
comp.os.ms-windows	comp.os	Informazioni generiche su Microsoft Windows
comp.sys	comp	Sistemi hardware
comp.sys.ibm	comp.sys	Sistemi IBM
comp.sys.ibm.pc	comp.sys.ibm	Personal Computer IBM
comp.sys.ibm.pc.hardware	comp.sys.ibm.pc	Hardware dei Personal Computer IBM
comp.sys.mac	comp.sys	Sistemi Macintosh
comp.sys.mac.hardware	comp.sys.mac	Hardware dei Macintosh
comp.windows	comp	Software riguardante i S.O. a finestre

comp.windows.x	comp.windows	X-Windows per LINUX
misc		Lavoro, salute e molto altro ancora
misc.forsale	misc	Offerte di vendita
rec		Giochi, hobby, sport
rec.autos	rec	Automobili
rec.motorcycles	rec	Motociclette
rec.sport	rec	Sport
rec.sport.baseball	rec.sport	Baseball
rec.sport.hockey	rec.sport	Hockey
sci		Scienze applicate, scienze sociali
sci.crypt	sci	Crittografia
sci.electronics	sci	Elettronica
sci.med	sci	Medicina
sci.space	sci	Spazio
soc		Società, cultura
soc.religion	soc	Religione
soc.religion.christian	soc.religion	Cristianesimo
talk		Attualità e dibattiti
talk.politics	talk	Politica
talk.politics.guns	talk.politics	Armi
talk.politics.mideast	talk.politics	Medioriente
talk.politics.misc	talk.politics	Politica generica
talk.religion	talk	Religione
talk.religion.misc	talk.religion	Religione generica

Le categorie presentate in tabella si riferiscono al **Newsgroup Set** della collezione, una seconda istanza come già detto nel *paragrafo 3.4* conterrà oltre a queste categorie anche tutte le altre a cui si fa riferimento nell'attributo *newsgroup* dei messaggi.

3.5.4 Usenet articles



All'articolo si è aggiunto l'attributo *Numero File* che inizialmente sembrava poter essere un identificatore primario. Tale convinzione è venuta meno in seguito ad alcune analisi effettuate per mezzo dei prototipi che hanno permesso di individuare in diverse directory (categorie) la presenza di file con lo stesso nome numerico ma contenenti informazioni riguardanti messaggi differenti (tale avvenimento è raro, ma presente).
Per quanto riguarda l'attributo *Keywords* si è preferito dissociarlo dall'entità, non solo perché multivalore, ma anche per la bassa ricorrenza all'interno dell'entità.

3.6 Schema Concettuale

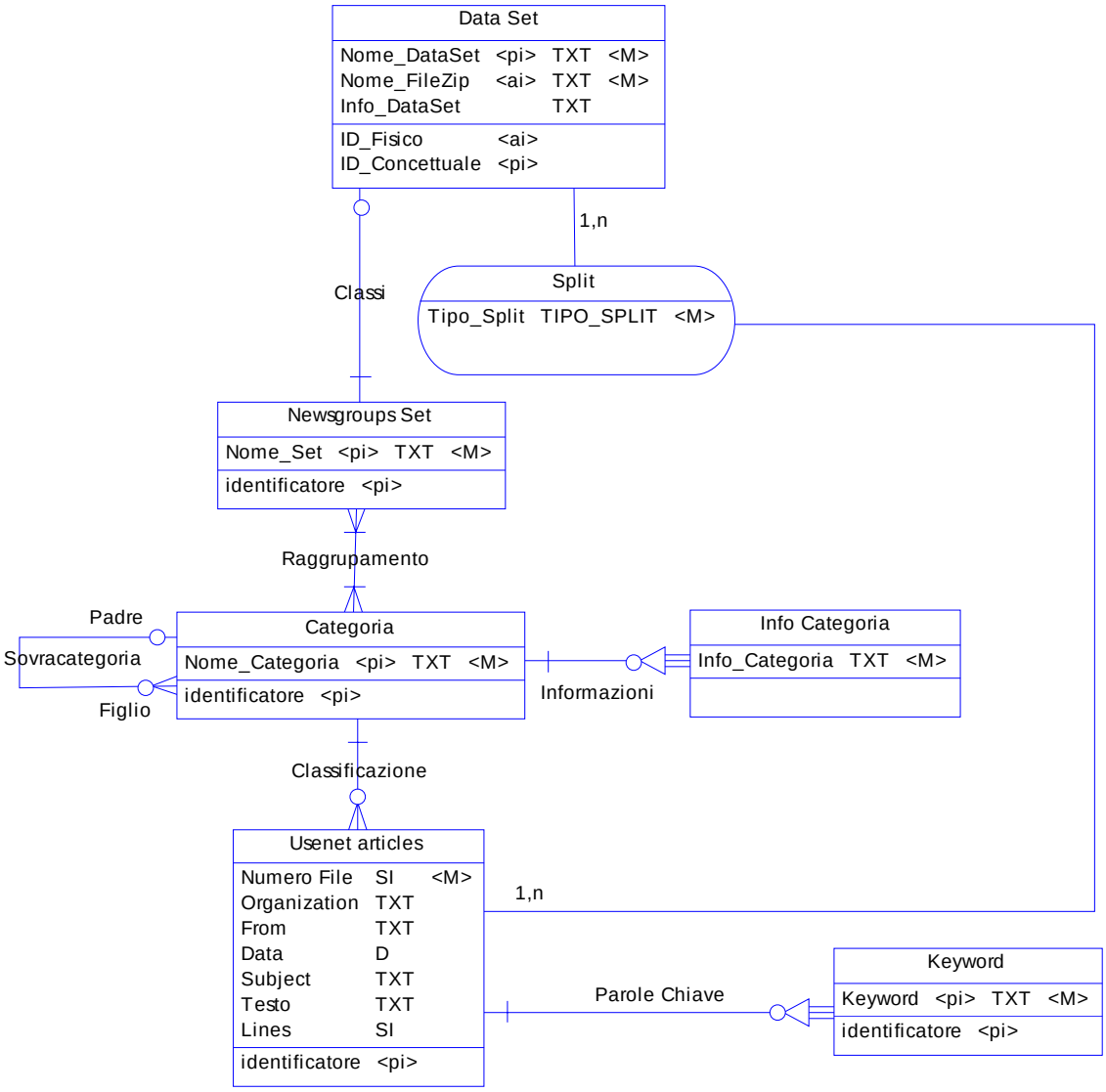


Figura 8: Schema Concettuale Ristrutturato del 20 Newsgroups

Lo schema riunisce quanto detto nel capitolo. Rispetto allo schema logico scompaiono le informazioni sull'organizzazione in file del benchmark, con l'esclusione in **Data Set** di *Nome_FileZip* che contiene il nome del file contenente la collezione.

CAPITOLO 4:

LA COLLECTION OHSUMED

4.1 Il Benchmark

4.2 Stato Iniziale dei Dati

4.3 Schema Logico "Ipotetico"

4.4 Ristrutturazione delle Entità

4.5 Schema Concettuale

Il capitolo è dedicato allo studio della terza collezione di riferimento considerata. Come nei capitoli precedenti, si cercherà di ottenere uno schema concettuale svincolato dagli altri ed in grado di rappresentare la collezione.

4.1 Il Benchmark

L'OHSUMED test collection è un set di 348.566 documenti tratti dal MEDLINE (utilizzando dei motori di ricerca è possibile rintracciare numerosi siti di riferimento; alcuni validi esempi possono essere visionati presso gli indirizzi http://medline.cos.com/ e http://www4.infotrieve.com/newmedline/search.asp), un database di informazioni mediche disponibile on-line, consistenti nel titolo e/o in un estratto di articoli tratti da 270 giornali medici nell'arco dei cinque anni che coprono il periodo tra il 1987-1991.
I campi resi disponibili oltre il titolo e l'estratto sono: MeSH indexing terms, author, source, and publication type. The National Library of Medicine ha permesso l'utilizzo dei riferimenti MEDLINE come data set a patto di soddisfatte le seguenti condizioni:

1. I dati devono essere adoperati solo per la realizzazione d'esperimenti nell'ambito della TC, e non per lo studio dei dati medici riportati.
2. Nell'utilizzo dei dati si deve specificare esplicitamente che i sono incompleti.

Il benchmark è adoperato per effettuare tre diversi tipi d'esperimenti su classificatori di tipo **CPC:** *adaptive filtering*, *batch filtering* e *routing*. Nell'adaptive filtering il sistema inizia con la sola definizione di un topic e di un piccolo insieme di esempi positivi e migliora il proprio profilo tramite feedback on-line. Il batch filtering ed il routing a contrario sono tra i più tradizionali metodi relativi al machine learning e adoperano un largo insieme di esempi di documenti già valutati.

4.2 Stato Iniziale dei Dati

I dati prelevati dal file *filtering.tar.gz* presente nel sito della ***TREC*** sono suddivisi in più file, formattati secondo la particolare entità o relazione del benchmark che contengono. Le informazioni riguardo alle formattazioni dei dati sono riportate dettagliatamente nel file *README*, messo a disposizione sia esternamente sia internamente al file *filtering.tar.gz*.
Le tuple degli *articoli* MEDLINE sono contenute nei file "*ohsumed.**", che permettono di ricavare due **Split**: un training set contenuto nel file *ohsumed.87* e un test set nel file *ohsumed.88-91*. Di seguito si riportano gli attributi messi a disposizione per gli *articoli*:

- **Sequential identifier**: identificativo utilizzato, tipicamente, per individuare la sequenza dei documenti da processare per effettuare l'addestramenti dei sistemi di prova.
- **MEDLINE identifier**: è l'identificativo degli articoli della *MEDLINE*. Viene utilizzato nei file contenenti i *relevance judgments* (cioè le associazioni articoli-query, che verranno descritte alla fine del paragrafo).
- **MeSH terms**: lista di termini appartenenti al vocabolario *MeSH* assegnati agli articoli da esperti del dominio applicativo.
- **Title**: titolo dell'articolo.
- **Source**: fonte di appartenenza dell'articolo.
- **Abstract**: estratto dell'articolo.
- **Publication types**: lista dei tipi di pubblicazioni a cui appartiene l'articolo.
- **Author**: lista degli autori dell'articolo.

Per quanto detto sembrerebbe che il benchmark contenga un unico **Data Set** diviso in due **Split**, ma ciò è incorretto dal momento che la particolarità della collezione non consiste nei documenti in se, ma nell'insieme dei **Topic** che mette a disposizione. Esiste un file per ogni **Topic** della collezione:

File
query.mesh.test.1-119
query.ohsu.test.1-43
query.mesh.1-4904
sample.map
Query.ohsu.1-63

I file "*query.**" contengono una lista di tuple chiamate **Query** la cui struttura è definita tramite gli attributi riportati di seguito:

- **Number**: una sigla associata alla query.
- **Title**: titolo della query che per i file OHSU rappresenta una breve descrizione del paziente, mentre per quelli MeSH il nome del concetto rappresentato nella descrizione.
- **Description**: indica per i file OHSU l'informazione richiesta, mentre per quelli MeSH descrive il concetto indicato.

Per quanto riguarda il file *"sample.map"* questi non fa altro che individuare un sottoinsieme di **Query** associando il **number** indicato nel *TREC-9 MeSH topics* (il file query.mesh.1-4904) ad un nuovo **number**.

Infine è possibile individuare i diversi **Data Set** tramite l'utilizzo di una serie di file *"qrels.*"* contenenti le associazioni, dette **Relevance Judgments,** che permettono di individuare gli *articoli* rilevanti per una **Query** (ogni file si riferisce alle **Query** di un particolare **Topic** ed agli *articoli* di un particolare **Split**). Ogni riga contiene due o tre colonne (la terza presente solo nel caso di *Topic OHSU* ed utilizzata per rappresentare l'affidabilità del giudizio riportato) che adoperano il formalismo descritto di seguito:

1. <Topic-**Number**> \t <Article- **MEDLINE identifier** > \t <*Relevant*> \n
   Utilizzato per gli OHSUMED relevance judgments (files: qrels.ohsu.*). Il campo *Relevant* può avere valore 1 (per indicare *possibly relevant*) o 2 (per indicare *definitely relevant*).

2. <Topic-**Number**> \t <Article- **MEDLINE identifier** > \n
   Utilizzato per i MeSH relevance judgments (files: qrels.mesh.*). In tale caso l'articolo, individuato dal **MEDLINE identifier,** è considerato "relevant" rispetto alla Query relativa al **Number** del *MeSH Topic* se il **Title** della Query è presente nella lista dei *MeSH terms* dell'articolo.

Nella collezione si indicano i file *"qrels.*"* da adoperare per ottenere dei **Data Set** che siano in grado di testare algoritmi per il batch filtering, l' adaptive filtering o la realizzazione di routing system (che adoperano gli stessi dati dei batch filtering). Inoltre sono riportati anche dei **Data Set** adoperabili per effettuare semplici pre-test sul proprio sistema.

Nome Data Set	***Relevance Judgments*** **(file qrels)**	***Split*** **(file ohsumed)**	***Topic*** **(file query)**
OHSU adaptive filtering	qrels.ohsu.adapt.87	ohsumed.87 (Tr)	query.ohsu.1-63
	qrels.ohsu.88-91	ohsumed.88-91 (Te)	query.ohsu.1-63
OHSU batch&routing filtering	qrels.ohsu.batch.87	ohsumed.87 (Tr)	query.ohsu.1-63
	qrels.ohsu.88-91	ohsumed.88-91 (Te)	query.ohsu.1-63
OHSU pre-test	qrels.ohsu.test.87	ohsumed.87 (Tr)	query.ohsu.test.1-43
MeSH adaptive filtering	qrels.mesh.adapt.87	ohsumed.87 (Tr)	query.mesh.1-4904
	qrels.mesh.88-91	ohsumed.88-91 (Te)	query.mesh.1-4904
MeSH batch&routing filtering	qrels.mesh.batch.87	ohsumed.87 (Tr)	query.mesh.1-4904
	qrels.mesh.88-91	ohsumed.88-91 (Te)	query.mesh.1-4904
MeSH pre-test	qrels.mesh.test.87	ohsumed.87 (Tr)	query.mesh.test.1-119

Per quanto riguarda le difficoltà incontrate nell'estrazione delle informazioni si rimanda al capitolo riguardante i moduli di feeding.

4.3 Schema Logico “Ipotetico”

Per ricavare lo schema logico del benchmark si sono aggiunte alle numerose informazioni prelevate dal file *README* le definizioni d’alcuni domini e di un ulteriore attributo per l’entità **Documento formattato** (relativa agli *articoli*):

- *Attributo* **Data**: data di stesura dell’articolo (informazione ricavata dal sito della ***OHSU*** tramite l’utilizzo dei file che vanno da *ohsumed.88* a *ohsumed.91*).
- *Dominio* **RELEVANT**: permette di definire se il giudizio relativo ad un’associazione articolo-Query è *possibly relevant* o *definitely relevant*.
- *Dominio* **Tipo_Split**: permette di distinguere se lo Split è un *Training Set* o un *Test Set*.
- *Dominio* **Tipo_DataSet**: permette di individuare il tipo di **Data Set** indicato: *Pre-Test*, *Routing & Batch* o *Adaptive*.

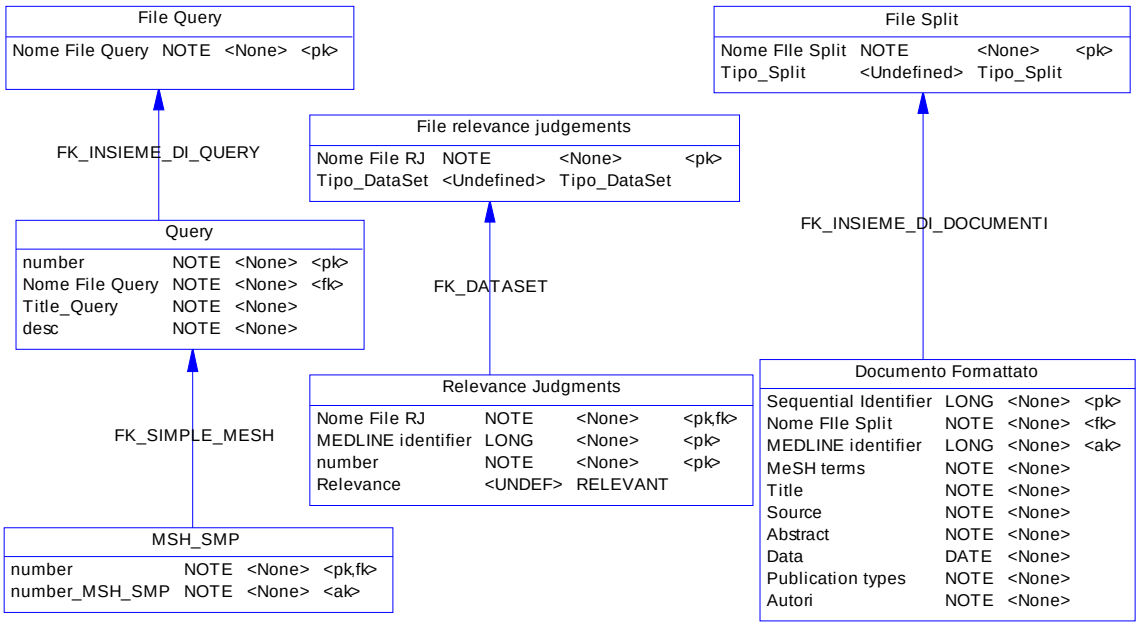


Figura 9: Schema Logico dell’OHSUMED

Da tale schema infine è possibile ricavare un primo schema concettuale che sarà il punto di partenza per la costruzione dello schema finale del benchmark.

Cosi come il modello dei capitoli precedenti si è deciso di individuare, in entrambi gli schemi, solo la tipologia degli attributi e non la loro dimensione, informazione non resa disponibile dagli ideatori della collezione.

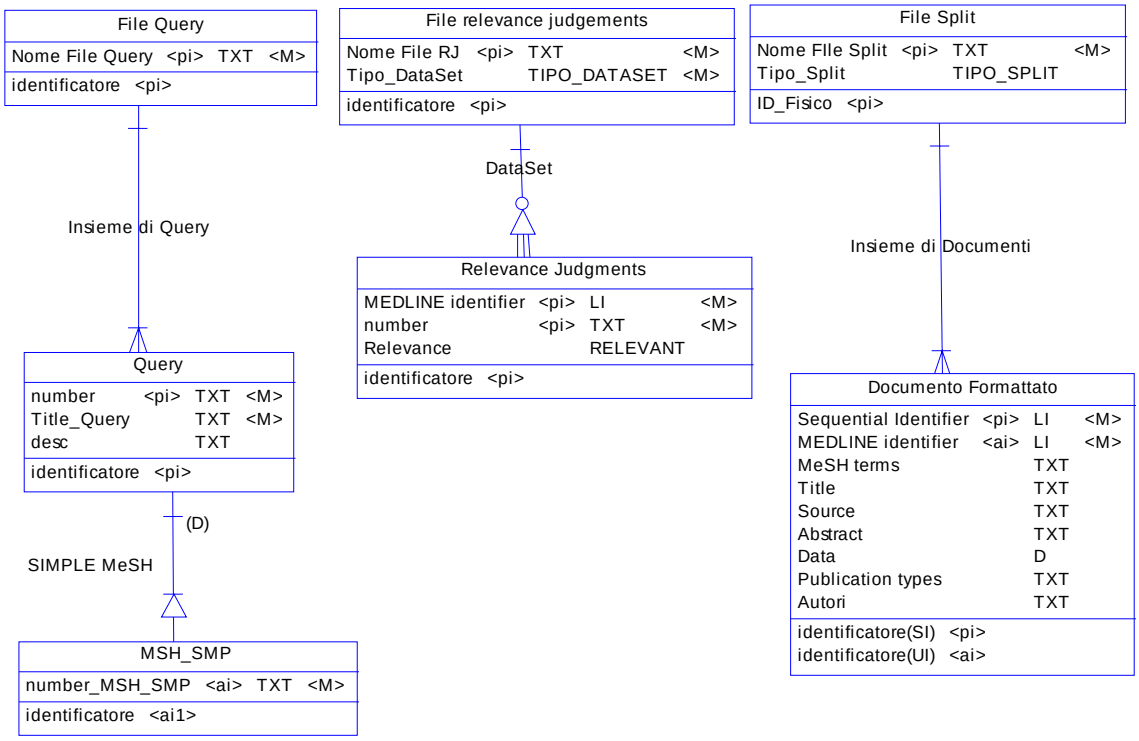


Figura 10: Schema Concettuale dell'OHSUMED

Gli schemi riportati nel capitolo sono stati implementati nei file *PhysicalDataModel*_OHSUMED.*pdm* e *ConceptualDataModel*_OHSUMED.*cdm* inseriti nel workspace "DataSetDB.sws".

4.4 Ristrutturazione delle Entità

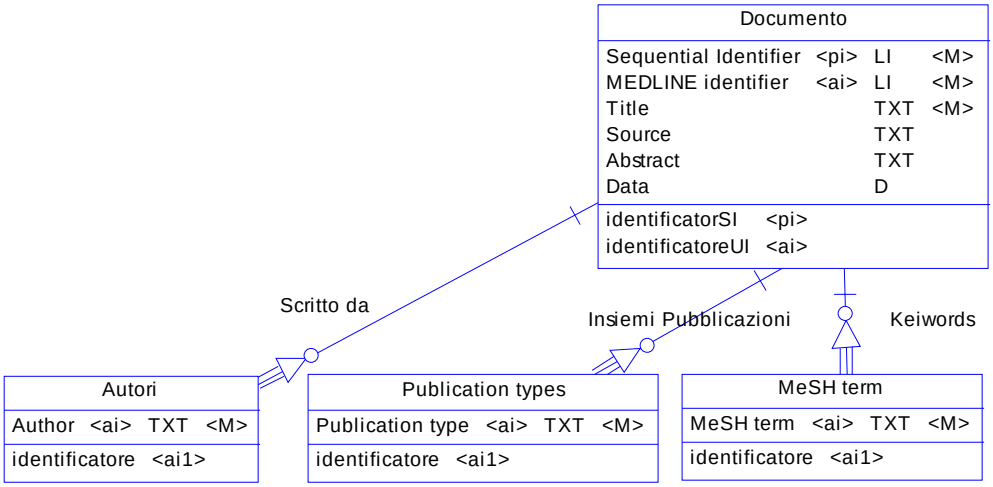
Lo schema concettuale ricavato nel paragrafo precedente anche se permette di realizzare un database che memorizzi le tabelle dei diversi file non è in grado di rappresentare correttamente tutte le entità introdotte nel *paragrafo 4.2*. Nei sottoparagrafi che seguono si effettua una ristrutturazione delle entità, definiscono, dove è possibile, anche le tuple individuate tramite lo studio del file *README*.

4.4.1 Documento

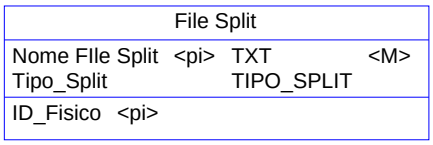
La prima entità da ristrutturare è **Documento Formattato** che, al suo interno, contiene tre attributi multi valore rappresentati, nella collezione, come un'unica riga con i singoli valori divisi da ";".

La scelta di creare tre entità distinte permette di rappresentare al meglio il modello E-R, ed al livello pratico di velocizzare le ricerche riguardanti tali attributi a scapito di un rallentamento nel prelievo dei dati appartenenti ai documenti. Si fa osservare inoltre che

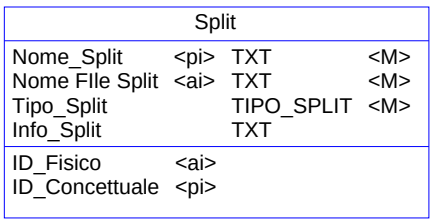
l'attributo *MeSH term*, nel caso si desideri studiare dei classificatori di tipo **DPC**, potrebbe anche essere adoperato come un'entità *categoria*.



4.4.2 Split



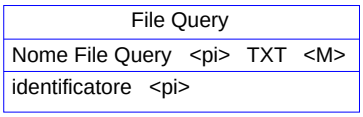
L'entità come definita in precedenza non permette di conservare le informazioni aggiuntive ricavate ed è dunque necessario ristrutturarla aggiungendo i due attributi riportati nella seguente rappresentazione grafica:



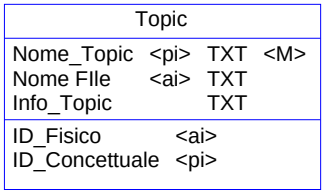
Tale modifica permette di descrivere le due tuple individuate nel *paragrafo 4.2* come indicato nella seguente tabella:

Nome File Split	Nome Split	Tipo_Split	Info Split
ohsumed.87	TREC-9 Training Set	Training Set	Indica il Training Set adoperato dalla TREC-9 per il batch ed il routing filtering, l'adaptive filtering e per i pre-test. Il Training Set si rifà ai riferimenti della MEDLINE del 1987. Si noti che il Training Set è anche adoperato per i pre-test
ohsumed.88-91	TREC-9 Test Set	Test Set	Il Test Set si rifà ai riferimenti della MEDLINE tra il 1988-1991

4.4.3 Topic



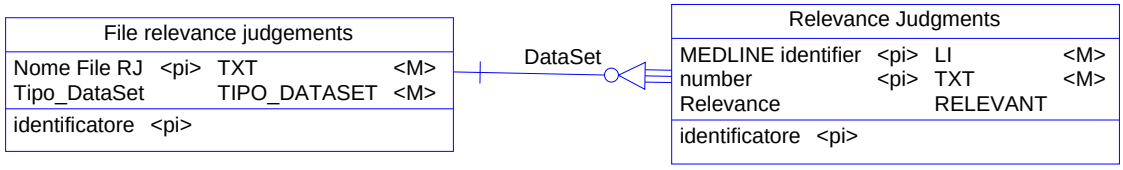
Tal entità, anche se rappresenta ottimamente la struttura del benchmark, di certo non evidenzia il suo legame con i **Topic**, per questo motivo è stato necessario sostituirla con l'entità riportata di seguito:



Grazie all'introduzione di quest'entità è ora possibile introdurre le tuple riportate nella seguente tabella:

File	Nome Topic	Descrizione Topic
query.mesh.test.1-119	MeSH pre-test	set of 119 MeSH test topics
query.ohsu.test.1-43	OHSU pre-test	Set of 43 OHSU test topics
query.mesh.1-4904	TREC-9 MeSH	set of 4904 TREC-9 MeSH topics
Sample.map	Subset TREC-9 MeSH	subset of the TREC-9 MeSH topics
query.ohsu.1-63	TREC-9 OHSU	set of 63 TREC-9 OHSU topics

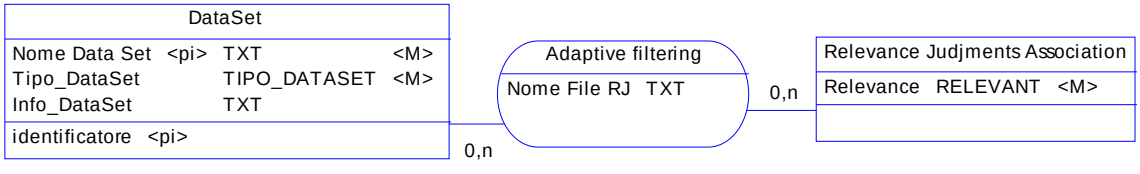
4.4.4 Data Set



Dalla definizione del dominio *TIPO_DATASET* (*paragrafo 4.3*) sembrerebbe che un file *"qrels.*"* possa rappresentare da solo un **DataSet** di tipo *Pre-Test*, *Routing & Batch* o *Adaptive*, ma questa sarebbe un'interpretazione sbagliata.

Solo i **Data Set** di tipo *Pre Test* sono composti da un unico file *"qrels.*"*, mentre gli altri ne richiedono due, uno che associ le **Query** del **Topic** di riferimento ai **Documenti** del training set e l'altro a quelli del test set.

Per risolvere tale inconveniente si deve effettuare una ristrutturare, che permetta di individuare il file d'appartenenza di un **Relevance Judgments** di un **DataSet**.



L'attributo **Nome file RJ** è dissociato dall'entità **DataSet,** perché non univoco; inoltre l'entità **Relevance Judgments** diventa un'associazione in grado di legare tra loro (come sarà più chiaro nello schema finale) **Query Documenti** e **DataSet**. Quest'ultima ristrutturazione è necessaria dato che nel benchmark non sempre basta dichiarare un **Topic** ed uno **Split** per individuare correttamente un **DataSet**, si pensi al caso degli *adaptive filtering* dove si adopera per l'addestramento dei sistemi solo un sottoinsieme delle possibili associazioni tra le **Query** del **Topic** selezionato e i **Documenti** appartenenti allo **Split** del training set.

Già nel *paragrafo 4.2* si sono introdotte le combinazione adoperate per definire i diversi **DataSet** del benchmark, in tale sede invece si riportano le tuple dell'entità **DataSet** ora introdotta:

Nome Data Set	Tipo_DataSet	Info_DataSet
MeSH adaptive filtering	Adaptive	
MeSH batch&routing filtering	Routing & Batch	
MeSH pre-test	Pre-Test	Non viene adoperato per ottimizzare i parametri del sistema, ma giusto per generare dei test per assicurarsi che il sistema funzioni correttamente.
OHSU adaptive filtering	Adaptive	I MeSH term sono adoperati come elenco di termini Thesaurus dei documenti
OHSU batch&routing filtering	Routing & Batch	I MeSH term sono adoperati come elenco di termini Thesaurus dei documenti
OHSU pre-test	Pre-Test	Non viene adoperato per ottimizzare i parametri del sistema, ma giusto per generare dei test per assicurarsi che il sistema funzioni correttamente. I MeSH term sono adoperati come elenco di termini Thesaurus dei documenti

4.5 Schema Concettuale

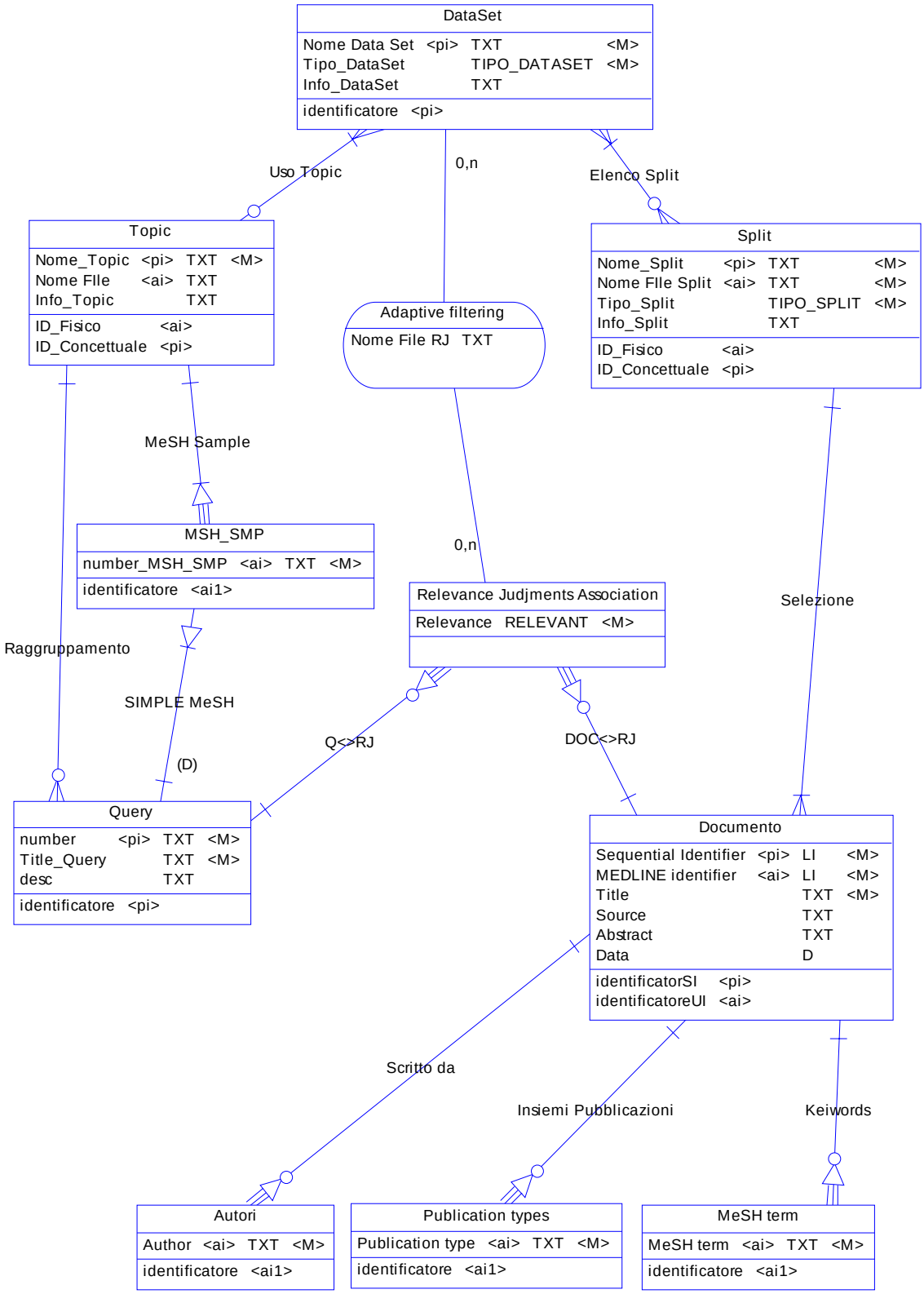


Figura 11: Schema Concettuale Ristrutturato dell'OHSUMED

CAPITOLO 5:

GENERALITA' DEL SISTEMA

5.1 Software Requirements Specifications

5.2 Funzionalità del prodotto

5.3 Use-Case Model

5.4 L'unità organizzativa Collection*

5.5 L'unità organizzativa Esperimenti*

In questo capitolo si procede alla realizzazione del secondo step indicato nel paragrafo 1.5. L'individuazione delle generalità del sistema è diretta conseguenza di tutti i modelli fin ora ricavati ed è stata possibile anche grazie alla stesura di un documento di "specifica dei requisiti software". I risultati di questo step saranno il punto di partenza per i successivi due capitoli.

5.1 Software Requirements Specifications

Una volta individuato il dominio applicativo e l'insieme dei possibili modelli E-R, che il sistema deve essere in grado di gestire, si è realizzato il documento di specifica dei requisiti software (contenuto nel file *"Software Requirements Specifications v3.0.doc")* che adopera come supporto grafico il file "*SRS DBDSATC v2.0.oom*" inserito nel workspace *DataSetDB.sws* (si veda il *paragrafo 1.6*).

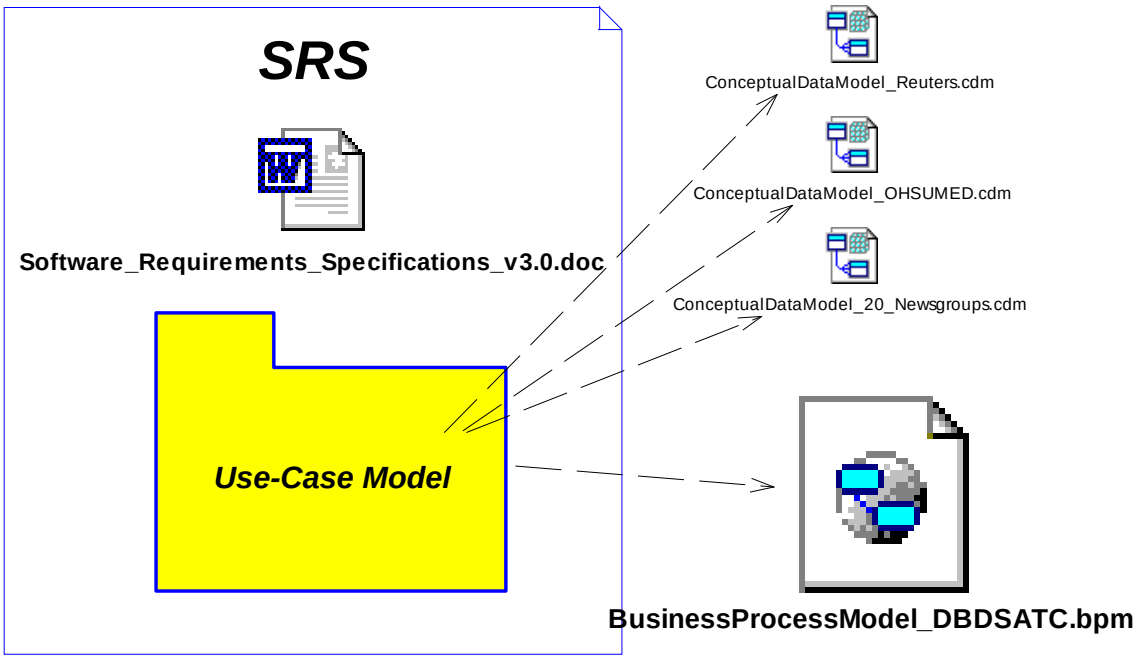


Figura 12: Organizzazione SRS

Molti dei punti salienti del documento SRS sono già stati descritti nel *paragrafo 1.4*, in questo capitolo invece si desidera individuare dettagliatamente le funzionalità del prodotto insieme ai suoi requisiti funzionali.

Innanzi tutto definiamo gli obiettivi relativi alla realizzazione del prodotto software che dovrà consentire, come già detto, un'efficiente gestione dei data set e un modello dei dati in grado di memorizzare e trattare collection già presenti negli ambienti di ricerca. Le finalità di tale sistema saranno:

1) **Creazione di una struttura del database generica** che permetta di gestire nuove collection e gli esperimenti effettuati. dai ricercatori.

2) **Modulo d'In/Out**:
   a) Creazione di funzionalità d'inserimento dati che saranno adoperate da:
      i) I moduli di feeding per inserire nel database i benchmark:
         (1) Reuters-21578 collection

(2) 20 Newsgroups collection
(3) OHSUMED collection

ii) I ME per salvare nel database i FV creati per i diversi esperimenti.
iii) I ME per salvare nel database le valutazioni effettuate per i diversi esperimenti.
iv) L'HIC per memorizzare le informazioni

b) Creazione di funzionalità di prelievo dati che saranno adoperate da:
   i) I ME per prelevare i dati del data set necessario.
   ii) L'HIC per visualizzare le informazioni
c) Creazione di funzionalità per modificare i dati che saranno adoperate da:
   i) I ME per modificare i dati del data set necessario.
   ii) L'HIC per modificare le informazioni.

3) **Creazione di un HIC** che permetta un'accurata gestione del database da parte del DBA e dei ricercatori.
4) Realizzazione di Report riguardanti statistiche sulle collection.
5) Realizzazione di Report riguardanti statistiche sugli esperimenti.
6) Realizzazioni di Report in grado di confrontare i risultati tra più esperimenti.
7) (Facoltativo) Eventuali ME di TC.
8) (Facoltativo) Eventuale ME per realizzare committee.

5.2 Funzionalità del prodotto

Il paragrafo presenta un elenco delle entità individuate e delle funzionalità che si è deciso di implementare su di esse. Nelle tre versioni del SRS che si sono realizzate non è stato necessario apportare modifiche se non per quanto riguarda l'introduzione del concetto dell'*Adaptive Filtering* che in un primo momento è stato totalmente trascurato considerando il suo training set ricavabile da un semplice sottoinsieme degli split di partenza. La precedente considerazione risultava errata giacché l'insieme dei *Relevance Judgments* che si potevano presentare conteneva solo un sottoinsieme delle possibili tuple legate ad un documento. Quindi si è concluso che sarebbe stato possibile individuare il training set di un *Adaptive Filtering* solo tramite una relazione diretta tra *Data Set* e *Relevance Judgments* (cosi come è stato presentato nel *paragrafo 4.4.4*).

5.2.1 Validazione utente

Permette di riconoscere se si deve abilitare la modalità *DBA* o *Ricercatore*; la seconda da la possibilità di modificare le informazioni che non causano perdita di dati per altri ricercatori i cui esperimenti sono estranei all'utente corrente.

5.2.2 Funzionalità riguardanti i dati delle Collection

Le funzionalità sono state ricavate dallo studio del prototipo realizzato per lo schema concettuale del database. Le entità riportate sono tutte quelle individuate nei capitoli relativi allo studio dei benchmark più una che si riferisce proprio a questi ultimi in modo da poter individuare le informazioni della collezione di appartenenze di un *Data Set*. L'elenco riportato è stato un buon punto di riferimento specialmente per quanto riguarda i filtraggi che richiedono dati esterni alle entità considerate. Per quanto riguarda invece gli attributi si è rimandata la loro definizione ai casi d'uso relativi alle apposite entità.

- ***Documenti***
  - *Inserzione* informazioni sul documento.
  - *Ricerca* dei documenti:
    - Ricerca per uno dei suoi attributi
    - Ricerca dei Documenti di un: Split, Data Set (tramite Split), Collection, Categoria, Query, Topic(tramite Categoria e Query), Esperimento.
  - Ricerca dato un documento delle associazioni a: Categorie, Query, Split, Collection, FV, Risultati CPC, Risultati DPC.
  - *Modifica* delle informazioni (a scapito della non congruenza con gli FV inseriti in precedenza) di un insieme di Documenti definito come in Ricerca.
  - Aggiunta/eliminazione di un'associazione ad una Categoria.
  - Aggiunta/modifica Rilevanza/eliminazione di un'associazione ad una Query.
  - Aggiunta/eliminazione di un'associazione ad uno Split.
  - Aggiunta/modifica degli ID_Originali/eliminazione di un'associazione ad una Collection.
  - *Cancellazione* di un insieme di documenti definito come in Ricerca:
    - Cancellazione associazioni ad eventuali Split.

- Cancellazione associazioni ad eventuali categorie e/o query.
- Cancellazione associazioni a Collection
- Cancellazione degli FV dei Documenti da cancellare.
- Cancellazione dei Risultati_CPC dei Documenti da cancellare.
- Cancellazione dei Risultati_DPC dei Documenti da cancellare.
- Cancellazione informazioni documento.

▪ *Visualizzazione e Report* dei documenti.

➢ ***Categorie***

▪ *Inserzione* informazioni sulla categoria
▪ Inserzione Sovracategoria/Sottocategoria.
▪ *Ricerca* delle categorie:
  - Ricerca per uno dei suoi attributi
  - Ricerca delle Categorie di un: Topic, Data Set (tramite Topic), Split (tramite Data Set), Collection (tramite Topic), Documento, Esperimento, Query (tramite Documento).

▪ Ricerca SottoCategorie di una Categoria.
▪ Ricerca SovraCategorie di una Categoria.
▪ Ricerca data una Categoria delle associazioni a: Topic, Documenti, Risultati DPC.
▪ *Modifica* delle informazioni (o della Sovracategoria/Sottocategoria) di un insieme di categorie definito come in Ricerca.
▪ Aggiunta/eliminazione d'associazioni ai Documenti.
▪ Aggiunta/eliminazione di un Topic d'appartenenza.
▪ *Cancellazione* di un insieme di Categoria definito come in Ricerca.
  - Cancellazione delle associazioni a Topic.
  - Cancellazione delle associazioni a Documenti.
  - Eventuale correzione delle sottocategorie:
    ♦ Diventano figlie del padre della categoria eliminata.
    ♦ Diventano orfane.
  - Cancellazione dei Risultati_CPC delle Categorie da cancellare.
  - Cancellazione delle informazioni della categoria.

▪ *Visualizzazione e Report* delle categorie.

- ***Query***
  - *Inserzione* informazioni sulla Query.
  - *Ricerca* delle Query:
    - Ricerca per uno dei suoi attributi
    - Ricerca delle Query di un: Topic, Data Set (tramite Topic), Split (tramite Data Set), Collection (tramite Topic), Documento, Esperimento, Categoria (tramite Documento).
  - Ricerca data una Query delle associazioni a: Topic, Documenti, Risultati CPC.
  - *Modifica* delle informazioni di una Query.
  - Aggiunta/modifica della Rilevanza/eliminazione d'associazioni ai Documenti.
  - Aggiunta/Modifica della Sigla/eliminazione di un Topic d'appartenenza.
  - *Cancellazione* di un insieme di Query definito come in Ricerca.
    - Cancellazione delle associazioni a Topic.
    - Cancellazione delle associazioni a Documenti.
    - Cancellazione dei Risultati_DPC delle Query da cancellare.
    - Cancellazione delle informazioni della Query.
  - *Visualizzazione e Report* delle Query.

- ***Split***
  - *Inserimento* delle specifiche del nuovo Split
  - *Ricerca* degli Split.
    - Ricerca per uno dei suoi attributi (ogni Split appartiene a più ad una collection)
    - Ricerca gli Spit di: Data Set, Documento, Esperimento, Topic (Tramite Data Set).
  - Ricerca dato uno Split delle associazioni a: Data Set, Documenti.
  - *Modifica* di alcune specifiche di uno Split.
  - Aggiunta/eliminazione di un documento dallo Split.
  - Aggiunta/eliminazione di uno Split da un Data Set.
  - *Cancellazione* di uno Split:
    - Cancellazione di tutte le associazioni create tra Split e documenti.
    - Cancellazione di tutte le associazioni create tra Split e Data Set.

- Eventuale cancellazione di tutti i documenti dello Split.
- Cancellazione delle specifiche dello Split.

▪ *Visualizzazione e Report* degli Split.

➢ ***Topic***

▪ *Creazione* di nuovi Topic:
- Inserimento delle specifiche del nuovo Topic.
- Definizione del tipo di Topic (Categoria o Query).

▪ *Ricerca* dei Topic
- Ricerca per uno dei suoi attributi (ogni Topic appartiene a più ad una Collection)
- Ricerca i Topic di: Data Set, Esperimento, Query o Categoria a seconda dei casi, Documento, Split (Tramite Data Set).
- Ricerca dato un Topic delle associazioni a: Data Set, Query o Categorie a seconda dei casi

▪ *Modifica* di alcune specifiche (tranne il tipo di Topic) di un Topic.
▪ Aggiunta/eliminazione di una Categoria dal Topic.
▪ Aggiunta/modifica della Sigla/eliminazione di una Query dal Topic.
▪ Aggiunta/eliminazione di un Topic da un Data Set.
▪ *Cancellazione* di un Topic:
- Cancellazione di tutte le associazioni create tra Topic e Categorie (o Query secondo i casi).
- Cancellazione associazioni con i Data Set.
- Eventuale cancellazione di tutte le Categorie (o Query) associate al Topic.
- Cancellazione delle specifiche del Topic.

▪ *Visualizzazione e Report* dei Topic.

➢ ***Data Set***

▪ Inserimento *delle specifiche del nuovo Data Set.*
▪ *Ricerca* dei Data Set:
- Ricerca per uno dei suoi attributi
- Ricerca i Data Set di: Collection (tramite Split e Topic), Esperimento, Split, Topic, Ricercatori (tramite Esperimenti).

- Ricerca dato un Data Set delle associazioni a: Topic, Split ed a "Adaptive Filtering"..
- *Modifica* di alcune specifiche di un Data Set.
- Associazione/cancellazione di uno Split dal Data Set.
- Associazione/cancellazione di un Topic dal Data Set.
- *Cancellazione* di un Data Set:
  - Cancellazione delle specifiche del Data Set definito come in Ricerca.
  - Cancellazione delle associazioni degli Split al Data Set.
  - Cancellazione delle associazioni dei Topic al Data Set.
  - Cancellazione degli Esperimenti associati al Data Set.
  - Eventuale cancellazione di tutto ciò che fosse collegato al Data Set.
- *Visualizzazione e Report* dei Data Set.

➢ ***Collection***

- *Inserimento* delle specifiche della collection.
- *Ricerca* delle collection:
  - Ricerca per uno dei suoi attributi
  - Ricerca delle Collection di un: Data Set (Tramite Split e Topic), Split, Topic, Categoria (tramite Topic), Query (tramite Topic), Documento, Esperimento (tramite Data Set).
- Ricerca data una Collection delle associazioni a: Documenti
- *Modifica* delle specifiche.
- *Aggiunta/modifica/eliminazione di un File della Collection*.
- Aggiunta/modifica degli ID/eliminazione di un documento da una Collection.
- *Cancellazione* di una collection:
  - *Cancellazione dei File associati alla Collection*
  - Cancellazione delle associazioni dei documenti alla collection.
  - Cancellazione delle associazioni degli Split alla collection.
  - Cancellazione delle associazioni dei Topic alla collection.
  - Eventuale cancellazione indiscriminata di tutto ciò che dipendeva dalla collection.
  - Cancellazione delle informazioni di una collection.
- *Visualizzazione e Report* delle collection.

5.2.3 Funzionalità riguardanti i dati degli Esperimenti

Così come la realizzazione degli schemi concettuali dei benchmark ci ha permesso di individuare le entità riguardanti le collezioni, il *Business Process Model* introdotto nel *paragrafo 1.3* ci permette di individuare non solo le entità, ma anche le funzionalità riguardanti i "***Dati Esperimento***" già, in grandi linee, descritte nei processi Memorizzazione FV, Prelievo FV, Memorizzazione Risultati e Prelievo Risultati.
Innanzi tutto sarà necessario l'utilizzo di funzionalità che si riferiscano ai Ricercatori:

- ***Ricercatori***
  - *Inserimento* delle informazioni di un ricercatore.
  - *Ricerca* dei Ricercatori:
    - Ricerca per uno dei suoi attributi
    - Ricerca dei Ricercatori che usano un: Esperimento, Topic (trami Data Set), Split (trami Data Set), Data Set (Tramite Esperimento), Collection (tramite Topic e Split).
  - Ricerca dato un Ricercatore delle associazioni a: Esperimenti.
  - *Modifica* delle informazioni di un ricercatore.
  - Aggiunta/eliminazione di un Ricercatore da un Esperimento.
  - *Cancellazione* delle informazioni di un ricercatore
  - *Visualizzazione e Report* dei ricercatori.

Poi serviranno delle funzionalità che permettano di gestire gli Esperimenti, le misurazioni su di essi (a tale scopo saranno necessarie delle funzionalità che permettano di memorizzare delle misurazioni non standard dell'efficienza dell'esperimento) ed i dati utilizzati dai processi Memorizzazione FV, Prelievo FV, Memorizzazione Risultati e Prelievo Risultati.

- ***Effectiveness Misure***
  (sono misurazioni non standard dell'efficienza dell'esperimento)
  - *Inserimento* di un'Effectiveness Misure..
  - *Ricerca* di un'Effectiveness Misure:
    - Ricerca per uno dei suoi attributi.
    - Ricerca delle Effectiveness Misure usate da un: Esperimento.
  - Ricerca data una Effectiveness Misure dei suoi valori per i vari: Esperimenti.

- *Modifica* di un'Effectiveness Misure.
- Aggiunta/Modifica dei valori/eliminazione dell'associazione ad un esperimento.
- *Cancellazione* di un'Effectiveness Misure e di tutte le sue occorrenze negli esperimenti.
- *Visualizzazione e Report* delle Effectiveness Misure.

- ***Features Vector***
  - *Inserzione* dell'apposito FV per un documento di un Esperimento.
  - *Ricerca* dei FV:
    - Ricerca per uno dei suoi attributi.
    - Ricerca delle FV di un: Data Set (tramite Esperimento).
  - *Modifica* di una FV di un documento associato ad un esperimento.
  - *Cancellazione* di un insieme di FV definito come in Ricerca.
    - Possibilità di cancellare anche una sola Feature di un insieme di FV.
    - Possibilità di cancellare tutte le Feature di un esperimento
    - Ecc…
  - *Visualizzazione e Report* delle FV.

- ***Risultati CPC***

  (è l'insieme dei *Relevant Judgments* che si sono ottenuti dal test di un classificatore di tipo CPC).
  - *Inserzione* dell'apposito Risultato CPC di una coppia Documento Query di un esperimento.
  - *Ricerca* dei Risultati CPC:
    - Ricerca per uno dei suoi attributi.
    - Ricerca dei Risultati CPC di un: Data Set (tramite Esperimento).
  - *Modifica* di un Risultato CPC di una coppia Documento Query di un esperimento.
  - *Cancellazione* di un insieme di Risultati CPC definito come in Ricerca.
  - *Visualizzazione e Report* dei Risultati CPC.

- ***Risultati DPC***

  (è l'insieme delle *Classificazioni* che si sono ottenute dal test di un classificatore di tipo DPC).

- *Inserzione* dell'apposito Risultato DPC di una coppia Documento Categoria di un esperimento.
- *Ricerca* dei Risultati DPC:
  - Ricerca per uno dei suoi attributi.
  - Ricerca dei Risultati DPC di un: Data Set (tramite Esperimento).
- *Modifica* di un Risultato DPC di una coppia Documento Categoria di un esperimento.
- *Cancellazione* di un insieme di Risultati DPC definito come in Ricerca.
- *Visualizzazione e Report* dei Risultati DPC.

➢ ***Esperimenti***

- *Inserimento* delle specifiche dell'esperimento tra gli attributi è possibile.
  - Selezionare un Data Set.
  - Selezionare Tipo di esperimento (CPC o DPC).
- *Ricerca* degli Esperimenti:
  - Ricerca per uno dei suoi attributi.
  - Ricerca degli Esperimenti di un: Ricercatore, Topic (tramite Data Set), Split (tramite Data Set), Collection (tramite Split e Topic), Effectiveness Misure.
- Ricerca dato un Esperimento delle associazioni a: Effectiveness Misure e Ricercatori.
- *Modifica* delle specifiche dell'esperimento.
- Aggiunta/modifica del valore/eliminazione di una Effectiveness Misure.
- Aggiunta/eliminazione di un ricercatore.
- Gestione FV riferite ad un esperimento (vedi Features Vector).
- Gestione Risultati CPC riferiti ad un esperimento (vedi Risultati CPC).
- Gestione Risultati DPC riferiti ad un esperimento (vedi Risultati DPC).
- *Cancellazione* di un esperimento:
  - Cancellazione associazioni dei ricercatori
  - Cancellazione delle associazioni alle Effectiveness Misure aggiuntive.
  - Cancellazione dei FV associati ai documenti dell'esperimento.
  - Cancellazione delle specifiche dell'esperimento.
- *Visualizzazione e Report* degli esperimenti (in grado di confrontare i risultati tra più esperimenti).

5.2.4 QUERY d'interesse

Per non creare confusione con l'entità Query si adopererà il termine QUERY quando ci si riferisce ai comandi di ricerca sul database..

- *Associazioni Query-Documenti* di un particolare: Esperimento (tramite Data Set), Data Set (tramite Topic&Split con possibilità di scelta del Tipo di Split), Data Set (tramite Adaptive Filtering), Topic, Split.

- *Associazioni Categorie-Documenti* di un particolare: Esperimento (tramite Data Set), Data Set (tramite Topic&Split con possibilità di scelta del Tipo di Split), Topic, Split.

5.3 Use-Case Model

Il cuore dell'Use-Case Model è il pacchetto "Modulo In/Out", contenente tutti i casi d'uso necessari per descrivere le funzionalità della *libreria DBDSATC*.

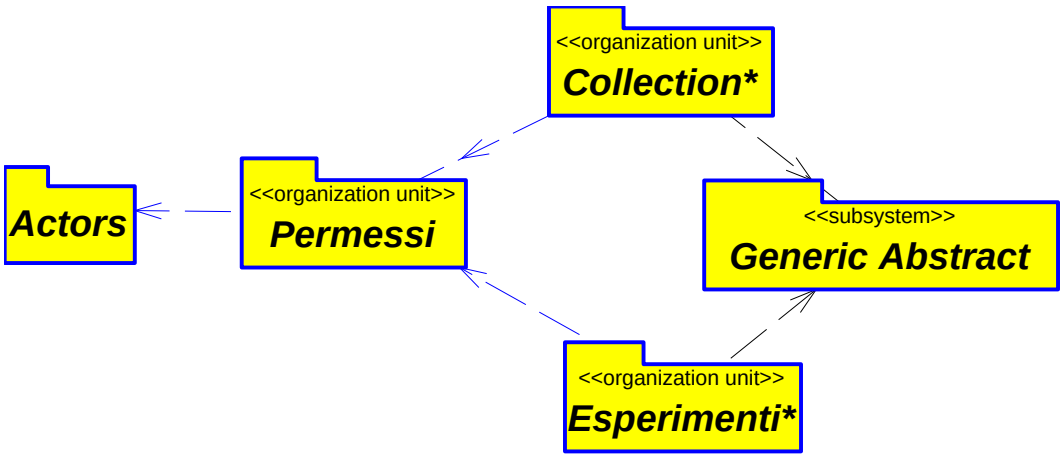


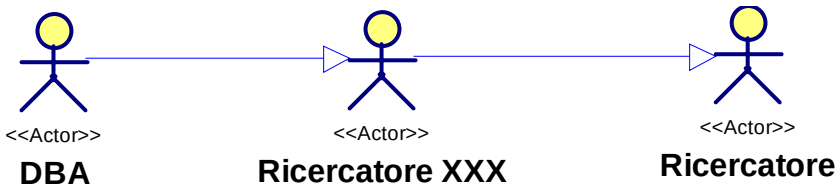
Figura 13: Organizzazione Use-Case Model

Si osservi che le unità organizzative **Collection*** ed **Esperimenti*** contengono informazioni molto ripetitive e dunque non saranno riportate nella loro completezza.

5.3.1 Actors

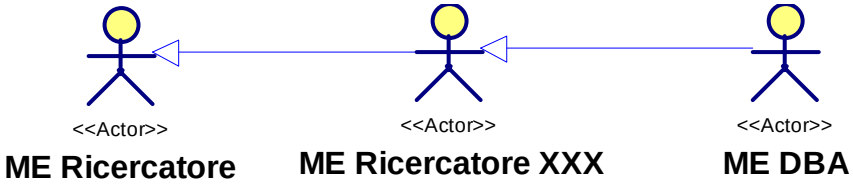
Gli attori, definiti nel pacchetto **Acotrs,** si riferiscono ai Moduli Esterni che utilizzano la *libreria DBDSATC*. La comprensione degli attori è, però, legata all'individuazione delle categorie d'utenti che adoperano il sistema. Il singolo ricercatore definito nel

Business Process Model non è sufficiente per i nostri scopi, poiché non assicurerebbe ne il requisito di sicurezza ne quello relativo all'interfaccia utente riportati nel *paragrafo 1.4*. Dunque invece di un singolo utente si terrà conto delle tre tipologie d'utenti riportati nel grafico:



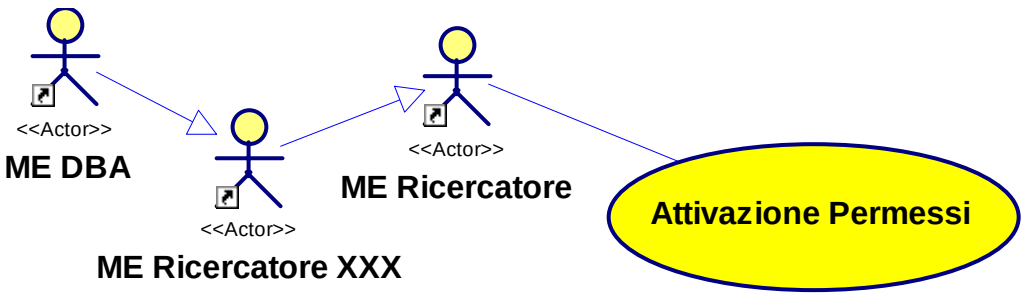
Il **Database Administrator (DBA)** è colui che si preoccuperà di gestire il DBDSATC. Si noti che molte funzioni possono essere effettuate solo dal *DBA*, perché la modifica d'alcuni dati (come ad esempio *Split* o *Data Set*....) potrebbero causare la perdita d'informazioni necessarie ad esperimenti in corso o già effettuati. Il **Ricercatore** è un individuo dedito allo sviluppo di un sistema di *TC*, tale attore ha accesso alle sole funzionalità generiche concesse agli utenti del sistema. Il **Ricercatore XXX** invece oltre a poter utilizzare tutte le funzionalità generiche di un *Ricercatore*, può modificare i dati relativi ad i suoi esperimenti.

Non dovrebbero, ora, esserci problemi nel comprendere le seguenti definizioni presenti nel pacchetto Actors:

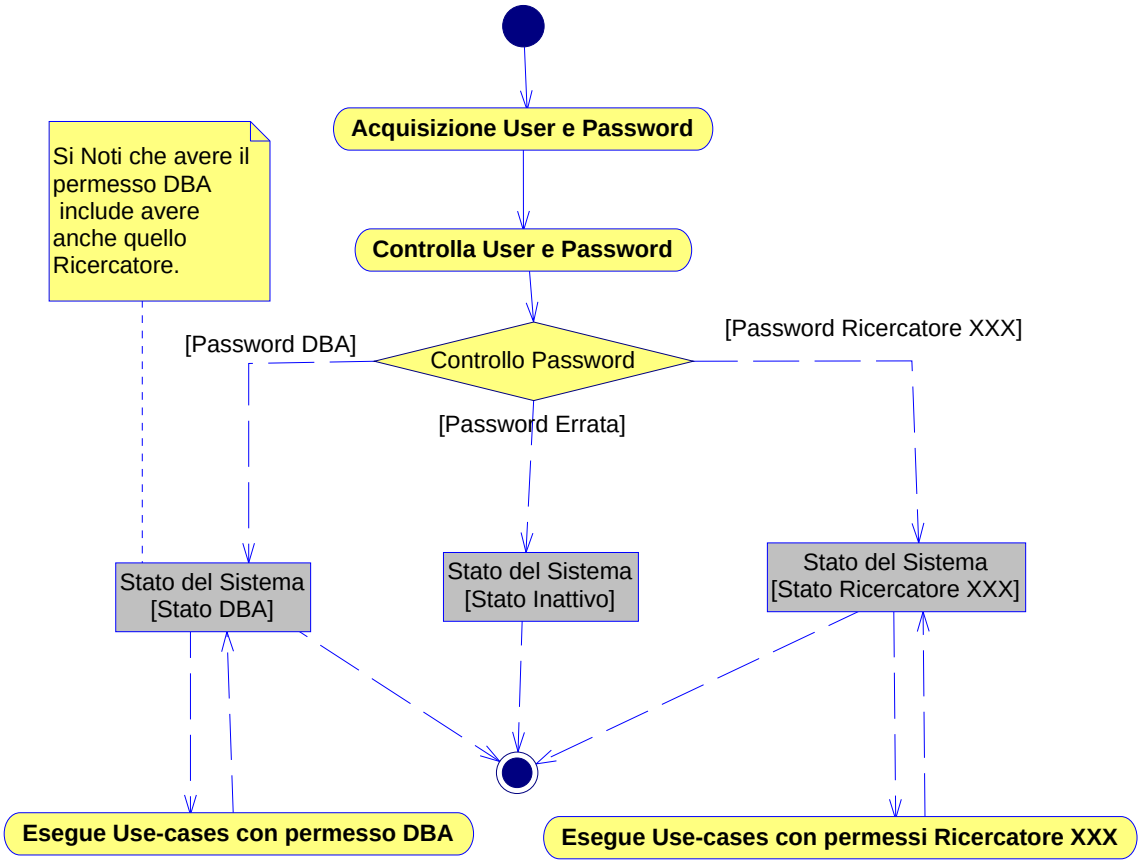


Questi attori individuano uno dei moduli che dovranno essere realizzati in un secondo momento aventi accesso, secondo i casi, a tutte le funzionalità del *Ricercatore* generico (**ME Ricercatore**), di un particolare *Ricercatore* (**ME Ricercatore XXX**) o del *DBA* (**ME DBA**). Un *ME* naturalmente potrà gestire gli accessi utilizzando le funzionalità offerte dal sistema e quindi essere adoperato da più di questi attori.

5.3.2 Permessi



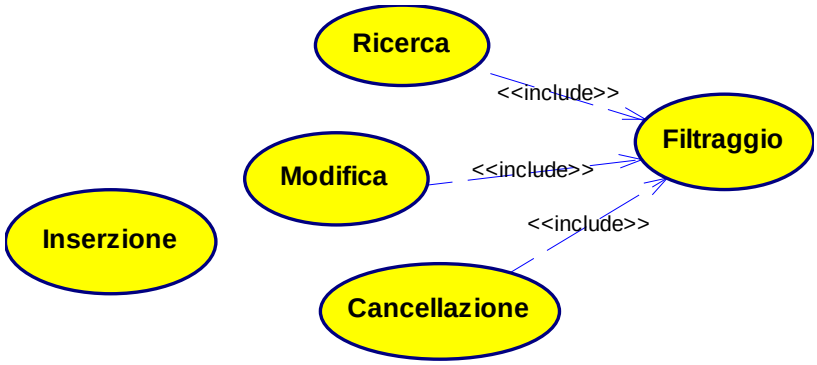
Il caso d'uso *Attivazione Permessi* è descritto tramite l'*Activity Diagram* riportato di seguito, al cui interno si possono individuare gli stati d'utilizzo del sistema:



Già dal diagramma è osservabile che non esiste uno stato in grado di individuare un **Ricercatore** generico**;** tale situazione si deve alla possibilità da parte di ogni utente di creare esperimenti e quindi di diventare un **Ricercatore XXX**, inoltre c'è da considerare che l'attore *Ricercatore* viene inserito solo per individuare quelle operazioni utilizzabili da tutti i *Ricercatori XXX*.

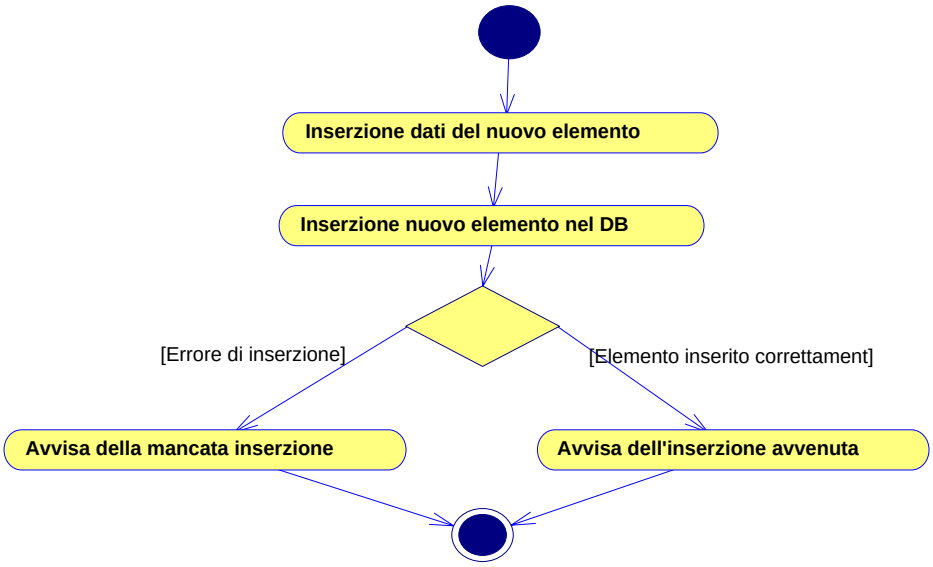
5.3.3 Generic Abstract

Gli use-cases presentati in questo sottosistema sono generici ed estesi all'occorrenza attraverso quelli delle unità organizzative **Collection*** ed **Esperimenti***. Tali casi d'uso sono quelli tipicamente adoperati sulle entità di un database e anche se potrebbe sembrare esagerato scendere tanto in dettaglio, non si deve dimenticare che l'obiettivo principale è quello di realizzare un *modulo di In/Out* le cui funzionalità sono di livello più basso rispetto a quelle che tipicamente ci si aspetta da una semplice *HIC*. I casi d'uso non fanno riferimento ad alcun attore poiché solo le loro estensioni sono indicative al fine del sistema.



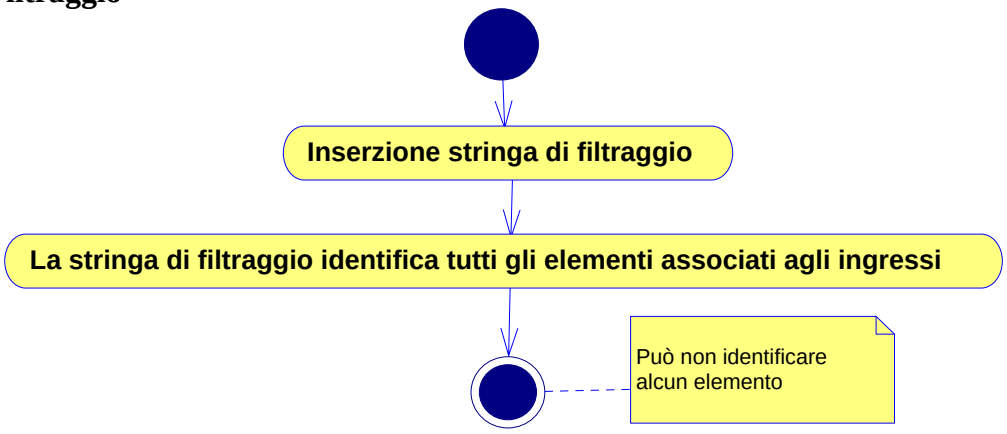
Procediamo con la descrizione degli Activity Diagrams d'ogni caso d'uso presentato nel sottosistema. Le varie attività introdotte, naturalmente, staranno solo ad indicare i vari passi che il caso d'uso dovrà effettuare e non come dovranno essere implementati.

1) Inserzione



Il diagramma è di facile comprensione, l'unica osservazione di rilievo è la definizione dell'attività "*Inserzione dati del nuovo elemento*" il cui insieme di dati sarà definito tramite l'attività "*Scelta Dati Attributi...*" presente in tutte le estesnsioni del caso d'uso corrente.

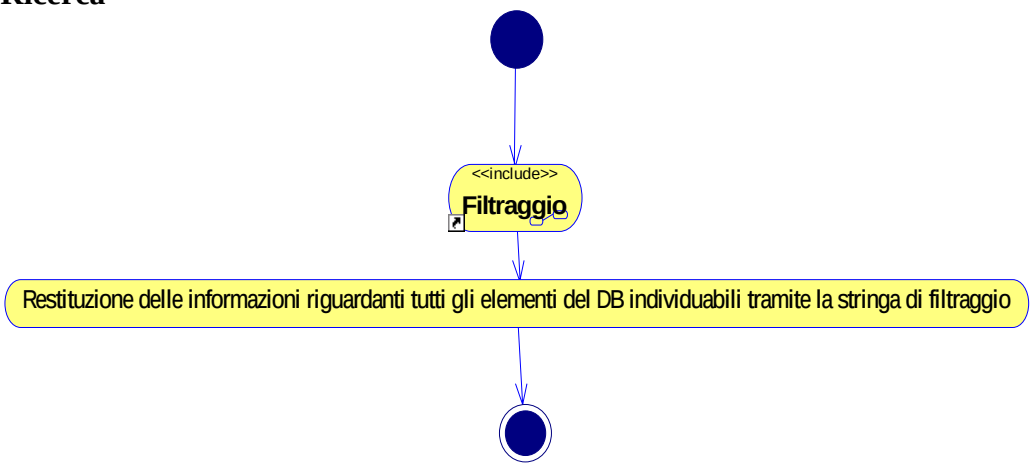
2) Filtraggio



Tale caso d'uso in realtà non ha senso da solo ed è introdotto perché incluso nei tre casi d'uso rimanenti.

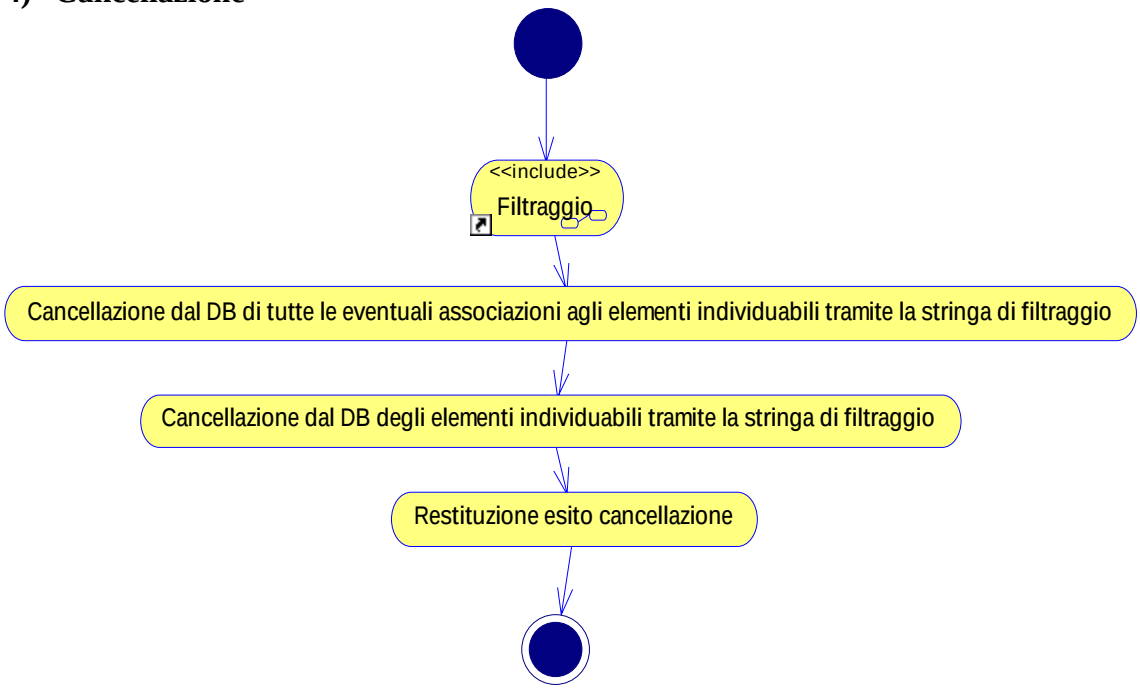
L'attività "*Inserzione stringa di filtraggio*" adopera un insieme di filtri che saranno definiti tramite l'attività "*Scelta Dati Filtri...*" presente nella particolare estensione dei casi d'uso Ricerca, Cancellazione e Modifica.

3) Ricerca



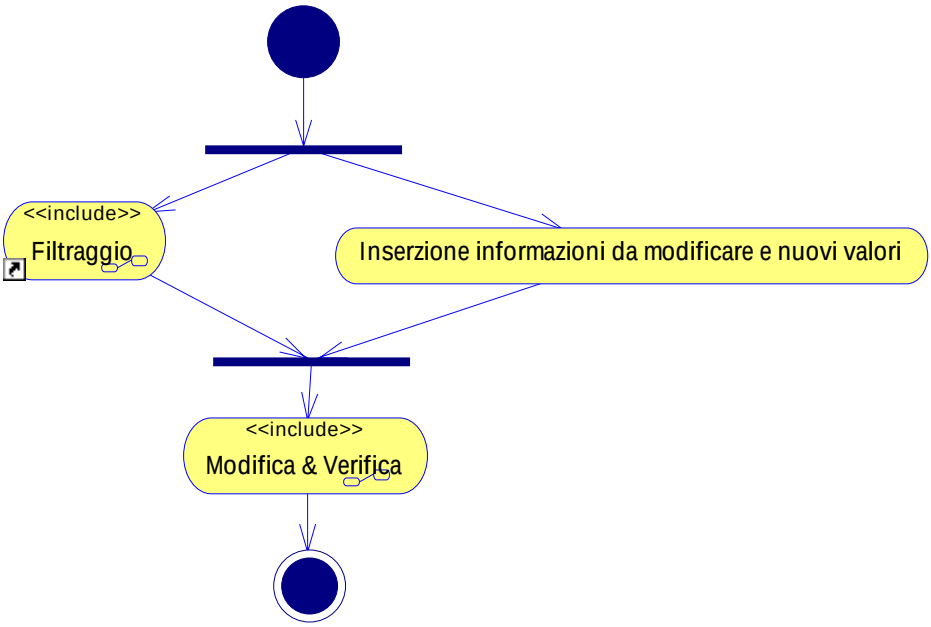
Le estensioni di questo caso d'uso oltre a possedere un'attività "*Scelta Dati Filtri...*" dovranno individuare anche gli attributi relativi ai particolari elementi ricercati, il cui insieme sarà implicitamente ricavato dall'attività "*Scelta Dati Attributi...*".

4) Cancellazione



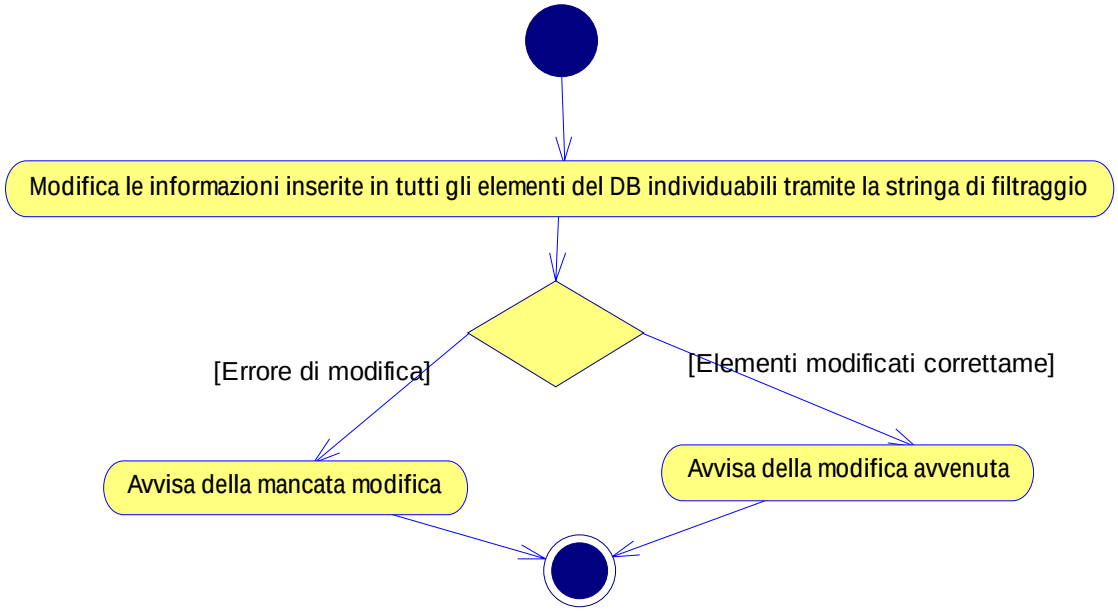
Il termine elementi presente in tale diagramma si riferisce alle tuple definite nelle estensioni del caso d'uso corrente.

5) Modifica



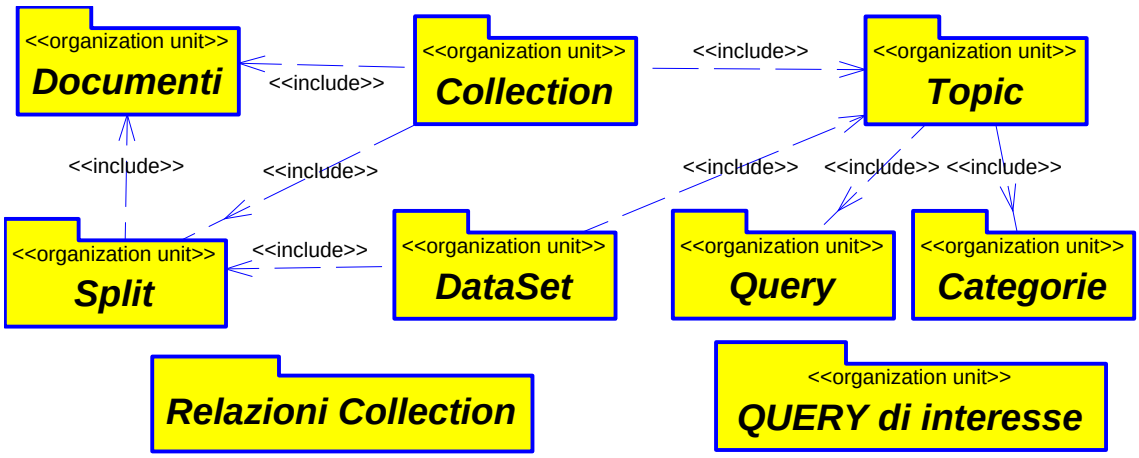
Le estensioni di questo caso d'uso dovranno fare riferimento sia all'attività "*Scelta Dati Filtri...*" che a quella "*Scelta Dati Attributi...*" per poter conoscere il tipo di dati e i filtri adoperati.

L'attività "*Modifica & Verifica*" è rappresentata come composta, siccome la descrizione delle estensioni di tale caso d'uso richiede spesso la ripetizione nei diagrammi di tale attività.



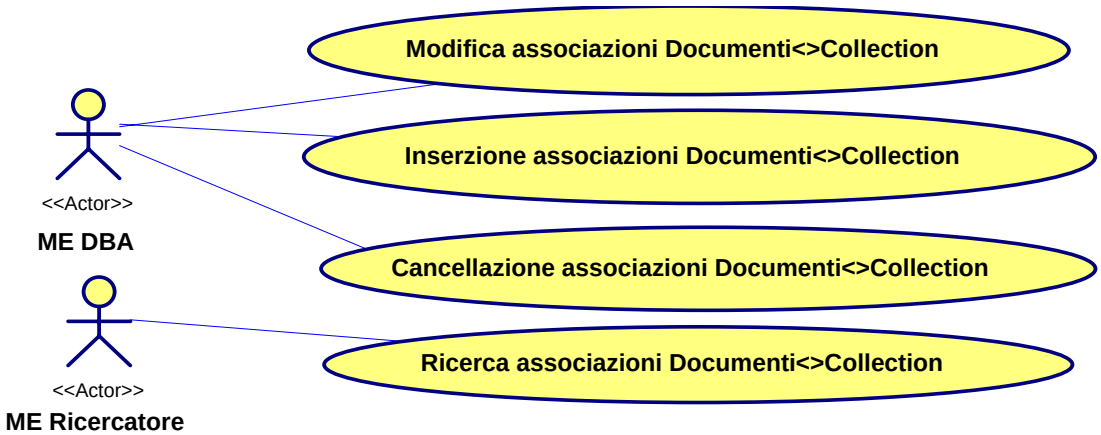
5.4 L'unità organizzativa Collection*

*Collection** contiene i casi d'uso che si riferiscono alla gestione delle entità presentate nel *paragrafo 5.2.2* e delle relazioni tra di esse; Inoltre contiene anche i casi d'uso relativi al *paragrafo5.2.4* dal momento che le funzionalità ivi descritte si riferiscono comunque a relazioni tra entità rientranti nell'unità organizzativa.



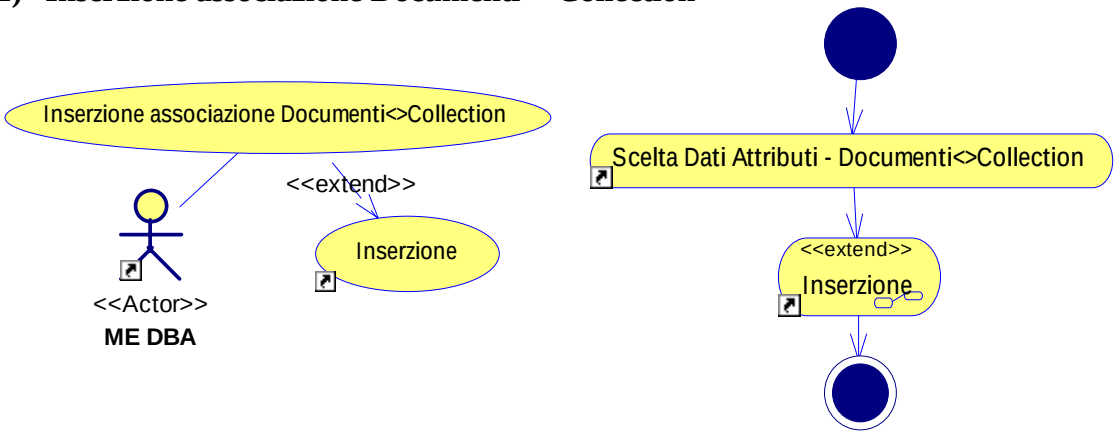
Come già detto i casi d'uso sono molto simili tra loro, quindi cominceremo col descrivere quelli relativi alle *Relazioni*, dove il primo sarà completo e gli altri mostreranno solo le varianti, per poi procedere alla descrizione di quelli relativi alle entità, sempre ristringendo le osservazioni sulle sole varianti.

5.4.1 Caso completo: Associazioni Documenti-Collection



I quattro casi d'uso che si riferiscono all' associazione non sono altro che estensioni dei casi d'uso rientranti nei "*Generic Abstract*". Le descrizioni riportate di seguito focalizzeranno l'attenzione sui punti che presenteranno le varianti per le successive descrizioni.

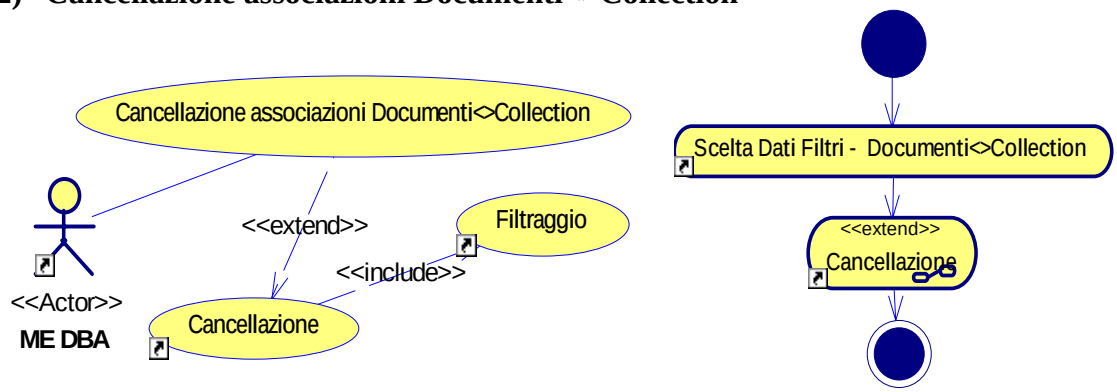
1) Inserzione associazione Documenti<>Collection



Dall'activity diagram riportato dovrebbe essere ora più chiaro quanto detto, nel *paragrafo 5.3.3*, nel riferirsi alle attività "*Scelta Dati Attributi...*". Quasi tutti i casi d'uso estesi dal presente si comporteranno in tale maniera, quindi sarà sufficiente definire per ogni estensione l'apposita attività di riferimento "*Scelta Dati Attributi...*".

Scelta Dati Attributi - Documenti<>Collection: individua come dati i valori in grado di identificare un Documento e una Collection di riferimento, senza però specificare quali siano tali dati (in tale fase non è ancora nota la precisa struttura delle entità ne tanto meno la loro chiave primaria). Ci si riferirà a tale tipologia di dati tramite l'utilizzo di parentesi quadre: [Documento], [Collection]. Inoltre si è pensato di aggiungere in tale associazione anche le informazioni relative agli identificativi di riferimento del documento rispetto al benchmark associato: *ID1_Original_Collection*, *ID2_Original_Collection*.: Questi due attributi avranno però significati diversi a seconda del benchmark dunque nasce l'esigenza di descrivere il loro significato da qualche parte (cosa che sarà effettuata nell'entità Collection).

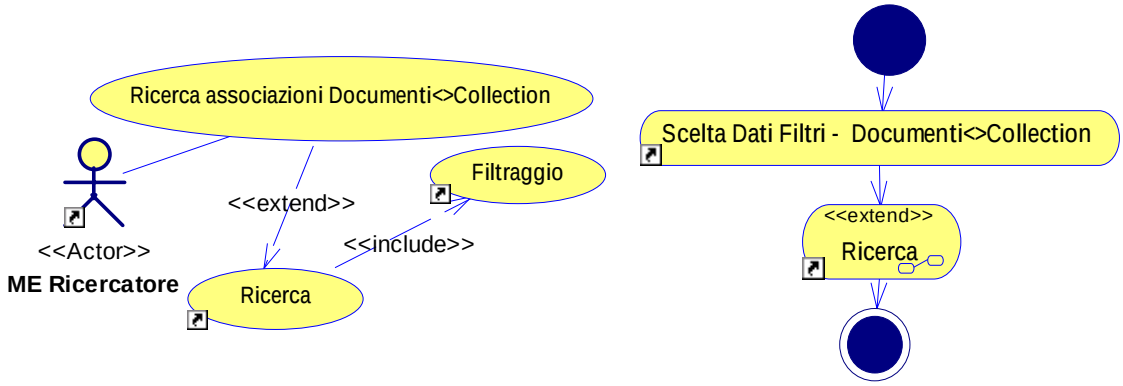
2) Cancellazione associazioni Documenti<>Collection



Tutti i casi d'uso estesi da cancellazione, così come per l'inserzione, si comporteranno come il presente quindi ci basterà definire per ognuno di essi l'attività di riferimento "*Scelta Dati Filtri...*".

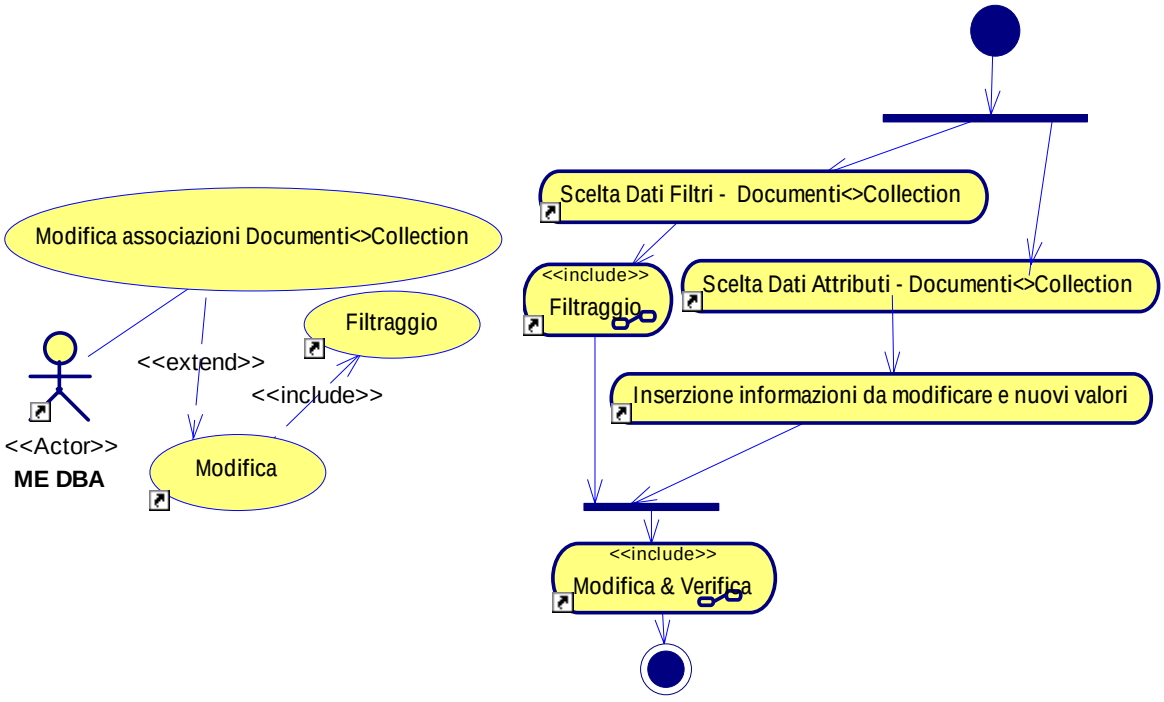
Scelta Dati Filtri – Documenti<>Collection: individua come *Filtri Interni* quelli che si riferiscono a: ID1_Original_Collection, ID2_Original_Collection, [Documento] e [Collection]; mentre non ha *Filtri Esterni* di interesse.

3) Ricerca associazioni Categorie<>Documenti



Tutte le informazioni necessarie per interpretare l'activity diagram sono già state introdotte in precedenza compresa l'attività "**Scelta Dati Filtri - Documenti<>Collection"**, che è uguale a quella definita per la cancellazione.

4) Modifica associazioni Documenti<>Collection

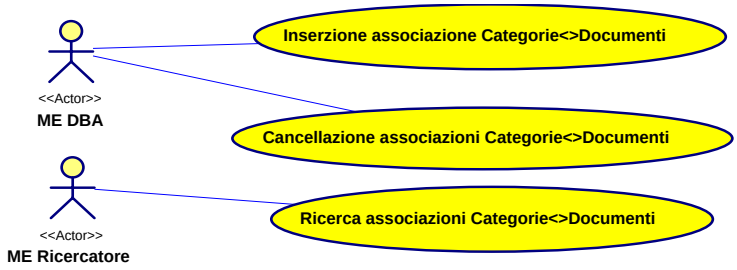


Anche in questo caso tutte le informazioni necessarie per interpretare l'activity diagram sono già state introdotte in precedenza comprese le attività "**Scelta Dati Filtri - Documenti<>Collection**" e "**Scelta Dati Attributi - Documenti<>Collection"**, che sono uguali a quelle definite per la cancellazione e per l'inserimento.

5.4.2 Casi relativi alle Associazioni definite per mezzo di varianti

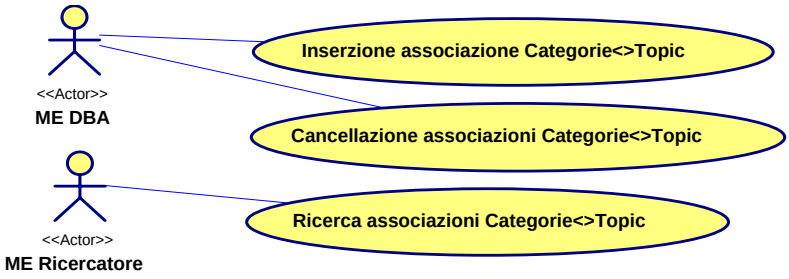
I casi d'uso relativi alle associazioni non presentano alcuna particolare difficoltà e gli unici fattori di interesse sono le definizioni delle attività "*Scelta Dati Attributi...*" e "*Scelta Dati Filtri...*". All'occorrenza si specificheranno alcuni vincoli aggiuntivi.

5.4.2.1 Associazioni Categorie-Documenti



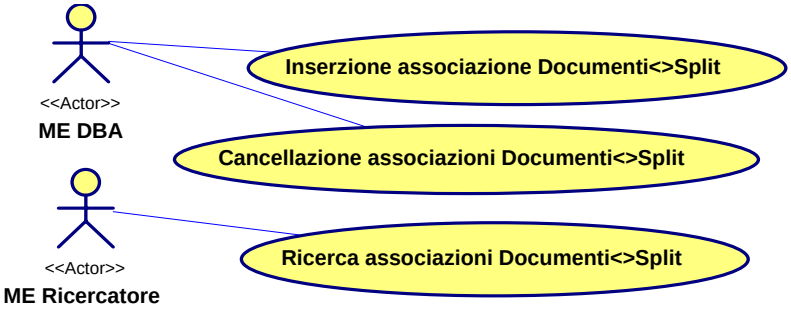
- **Scelta Dati Attributi - Categorie<>Documenti**: individua come dati i valori in grado di identificare una Categoria e un Documento: [Categoria], [Documento].
- **Scelta Dati Filtri - Categorie<>Documenti**: individua come *Filtri Interni* quelli che si riferiscono a: [Categoria] e [Documento]; mentre come *Filtri Esterni* quelli che si riferiscono a: [Esperimento], [Data Set], [Topic] e [Split].

5.4.2.2 Associazioni Categorie-Topic



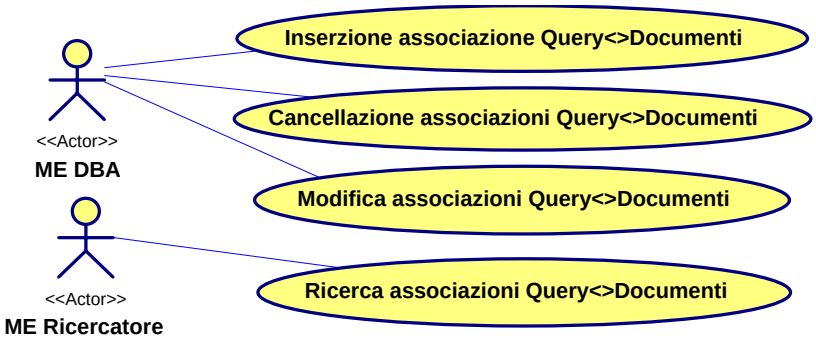
- **Scelta Dati Attributi - Categorie<>Topic**: individua come dati i valori in grado di identificare una Categoria e un Topic: [Categoria], [Topic].
- **Scelta Dati Filtri - Categorie<>Topic**: individua come *Filtri Interni* quelli che si riferiscono a: [Categoria] e [Topic]; mentre non ha *Filtri Esterni* d'interesse.

5.4.2.3 Associazioni Documenti-Split



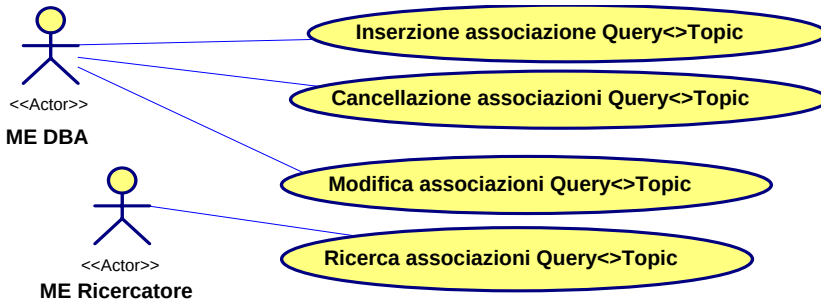
- **Scelta Dati Attributi - Documenti<>Splitc**: individua come dati i valori in grado di identificare un Documento e uno Split: [Documento], [Split].
- **Scelta Dati Filtri - Documenti <>Split**: individua come *Filtri Interni* quelli che si riferiscono a: [Documenti] e [Split]; mentre non ha *Filtri Esterni* d'interesse.

5.4.2.4 Associazioni Query-Documenti



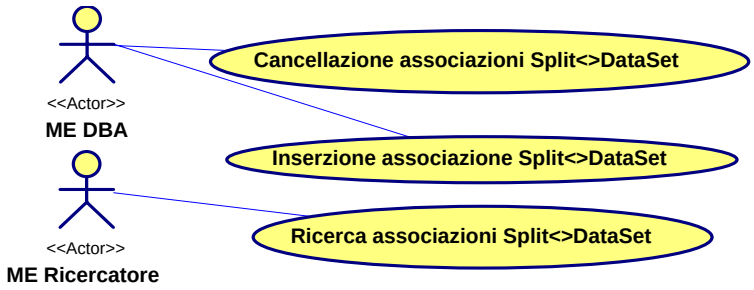
- **Scelta Dati Attributi - Query<>Documenti**: individua come dati i valori in grado di identificare una Query e un Documento: [Query], [Documento]; Inoltre si è pensato di aggiungere in tale associazione anche le informazioni relative all'attributo *Relevant* introdotto nello studio del benchmark OHSUMED
- **Scelta Dati Filtri - Query<>Documenti**: individua come *Filtri Interni* quelli che si riferiscono a: [Documento], [Query], Relevant; mentre come *Filtri Esterni* quelli che si riferiscono a [Esperimento], [Data Set], [Topic] e [Split].

5.4.2.5 Associazioni Query-Topic



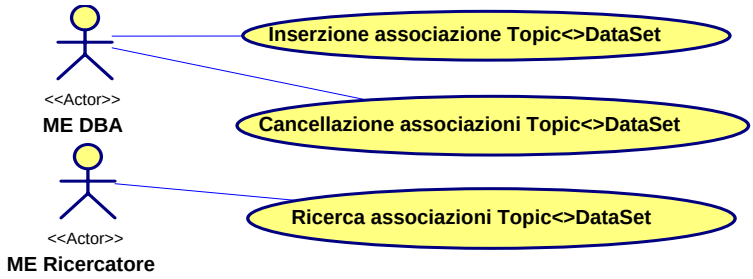
- **Scelta Dati Attributi - Query<>Topic**: individua come dati i valori in grado di identificare una Query e un Topic: [Query], [Topic]. Inoltre si è pensato di aggiungere in tale associazione anche le informazioni relative alla *Sigla* di riferimento della Query che, come visto nel benchmark OHSUMED, non è univoca e dipende dal Topic di appartenenza.
- **Scelta Dati Filtri - Query<>Topic**: individua come *Filtri Interni* quelli che si riferiscono a: [Query], [Topic] e Sigla; mentre non ha *Filtri Esterni* d'interesse.

5.4.2.6 Associazioni Split-DataSet



- **Scelta Dati Attributi - Split<>DataSet**: individua come dati i valori in grado di identificare uno Split e un DataSet: [Split], [DataSet].
- **Scelta Dati Filtri - Split<>DataSet**: individua come *Filtri Interni* quelli che si riferiscono a: [Split] e [DataSet]; mentre non ha *Filtri Esterni* di interesse.

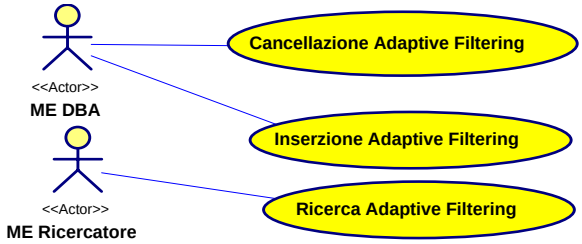
5.4.2.7 Associazioni Topic-DataSet



- **Scelta Dati Attributi - Topic<>DataSet**: individua come dati i valori in grado di identificare un Topic e un DataSet: [Topic], [DataSet].
- **Scelta Dati Filtri - Topic<>DataSet**: individua come *Filtri Interni* quelli che si riferiscono a: [Topic] e [DataSet]; mentre non ha *Filtri Esterni* di interesse.

5.4.2.8 Adaptive Filtering

Alcune osservazioni preliminari sono state effettuate all'inizio del *paragrafo 5.2*.



- **Scelta Dati Attributi - Adaptive Filtering**: individua come dati i valori in grado di identificare un Data Set, una Query ed un Documento: [DataSet], [Query] e [Documento].
- **Scelta Dati Filtri - Adaptive Filtering**: individua come *Filtri Interni* quelli che si riferiscono a: [DataSet], [Query] e [Documento]; mentre non ha *Filtri Esterni* d'interesse.

5.4.3 Casi relativi alle Entità definite per mezzo di varianti

I casi d'uso che saranno presentati di seguito si riferiscono ad ulteriori estensioni di quelli rientranti nei "*Generic Abstract*". Nei diagrammi locali delle diverse entità saranno inseriti tutti i casi d'uso attinenti, compresi quelli già introdotti in precedenza e quelli che saranno introdotti nell'unità organizzativa Esperimenti*.

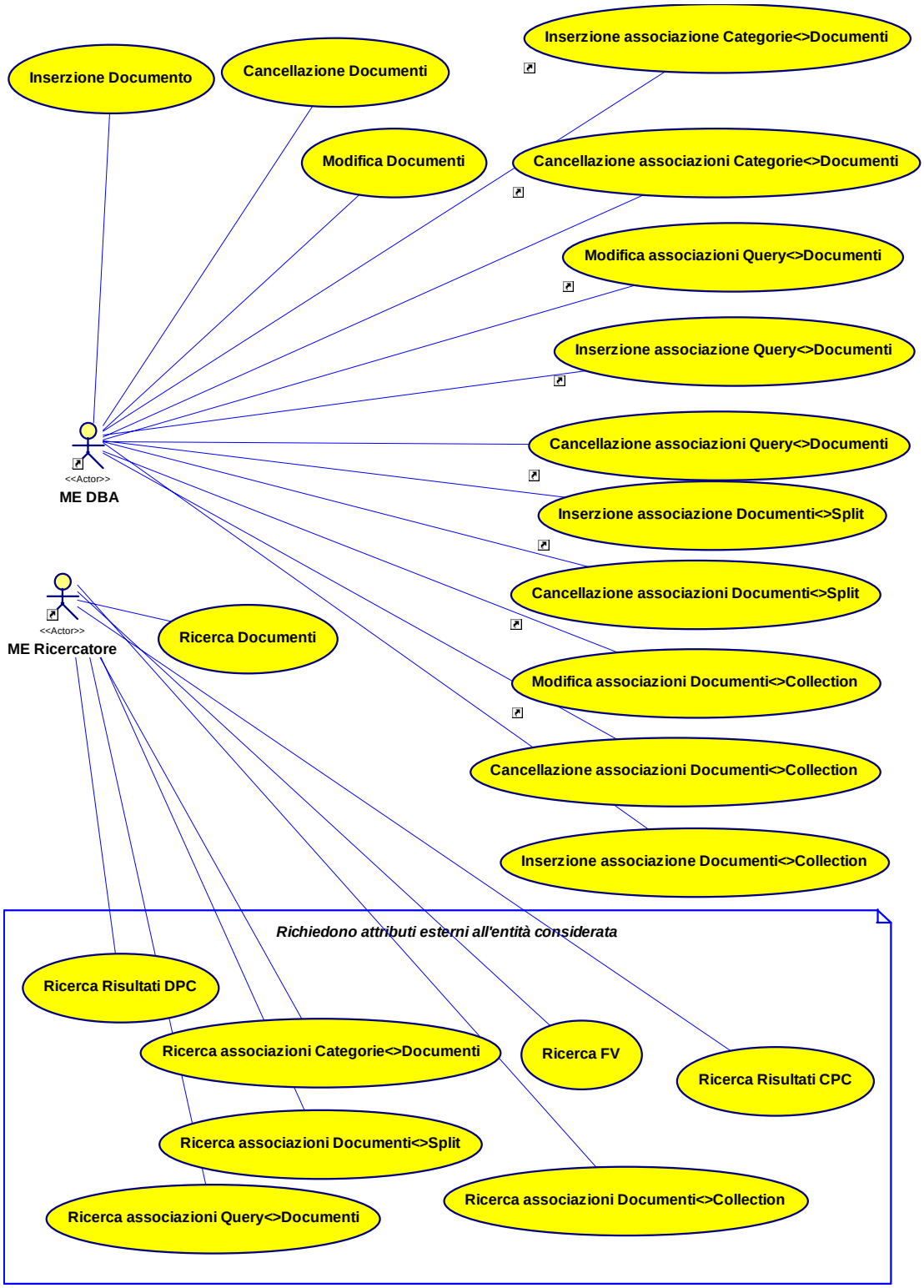
5.4.3.1 Documenti

Come prima cosa si definiscono le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi - Documenti**: individua come dati un insieme che permetta di considerare tutti gli attributi riscontrati nei benchmark studiati: *Tipo_Testo*, *Tipo_Pubblicazione*, *Linee*, *Provenienza*, *Autore_Documento*, *Data_Documento*, *Luogo*, *Lingua* (aggiunto poichè tale fattore influenza in maniera sostanziale gli eventuali *FV* che saranno ricercati sul documento), *Titolo_Documento*, *Testo*.
  Di seguito si descrivono i vari attributi e da chi sono richiesti:

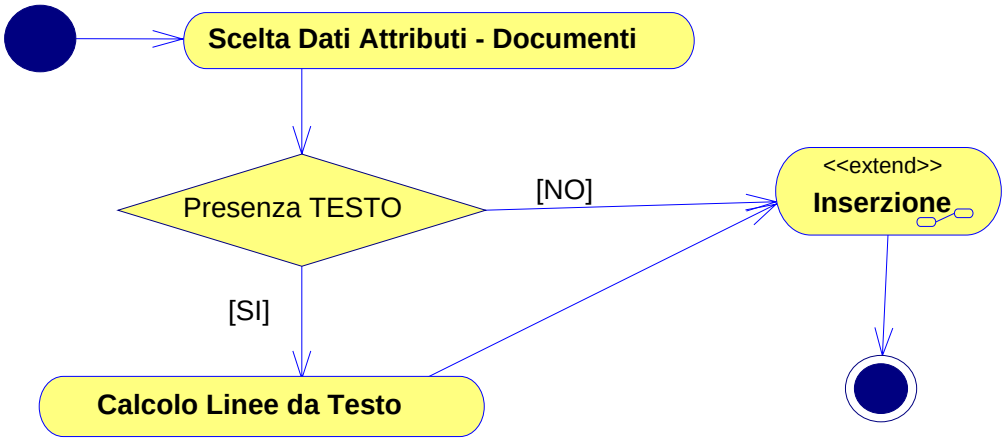
Attributo	Ohsumed	Reuters	20 Newsgroups
Tipo_Testo		campo TEXT_TYPE	
Tipo_Pubblicazione	campo Publication_Type	valore "Newswire"	Valore "E-mail"
Linee	da calcolare	da calcolare	campo Lines
Provenienza	campo Source	Valore "Reuters Ltd. and Carnegie Group"	campo Organization
Autore_Documento	campo Author	campo Author	campo From
Data_Documento	campo Data	campo DATE	campo Data
Luogo		LOCATION	
Lingua	Inglese	Inglese	Inglese
Titolo_Documento	campo Title	campo TITLE	campo Subject
Testo	campo Abstract	campo TEXT	Campo Testo

- **Scelta Dati Filtri - Documenti**: individua come *Filtri Interni* quelli che si riferiscono agli attributi: Tipo_Testo, Tipo_Pubblicazione, Linee, Provenienza, Autore_Documento, Data_Documento, Luogo, Lingua, Titolo_Documento e Testo; mentre come *Filtri Esterni* quelli che si riferiscono a [Split], [DataSet], [Collection], [Categoria], [Query], [Topic], [Esperimento].

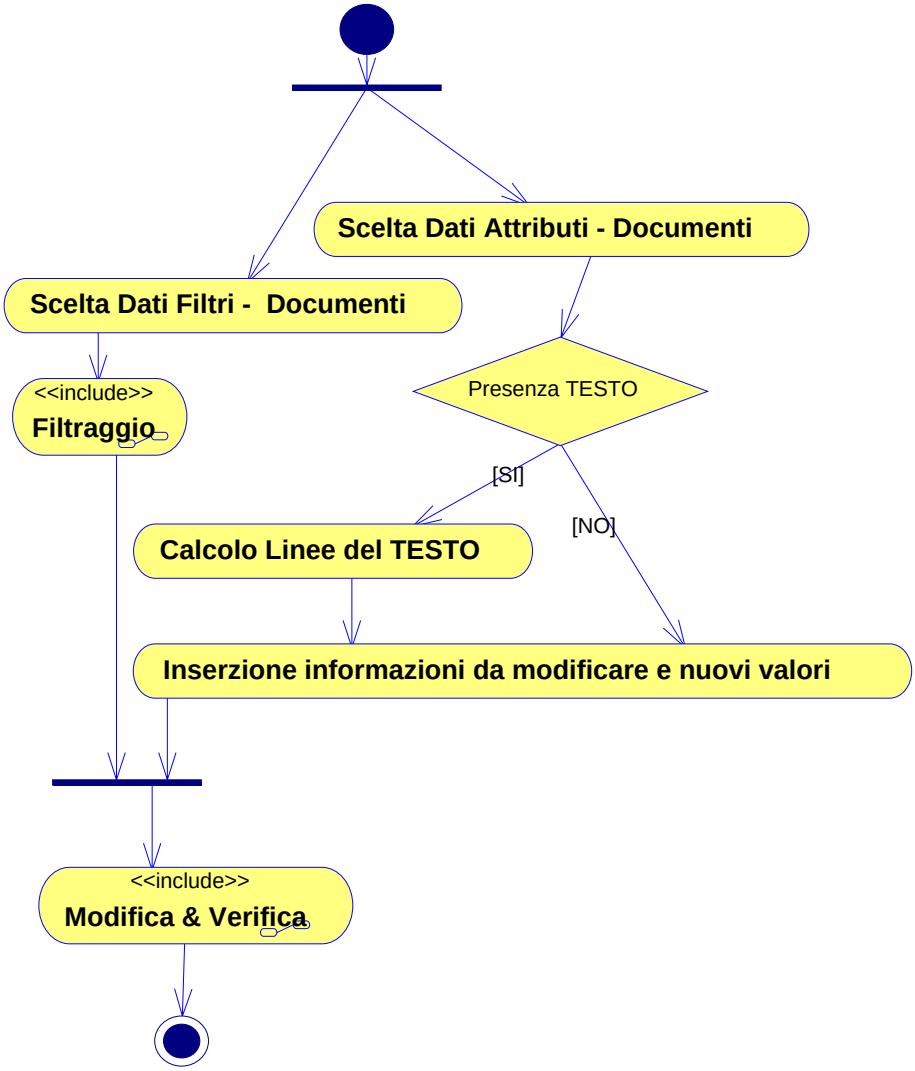


I casi d'uso che si riferiscono alla ricerca ed alla cancellazione seguono gli standard fin ora discussi, invece ci sono alcune modifiche per quanto riguarda l'inserzione e la modifica, poiché la presenza di un *Testo* richiede il calcolo (o ricalcalo) del numero di *Linee* cui è composto per poi poterlo conservare nell'archivio (tale informazione è ridondante ma si è deciso di conservarla poiché è esplicitamente utilizzata nel benchmark 20 Newsgroups).

1) Inserzione Documento



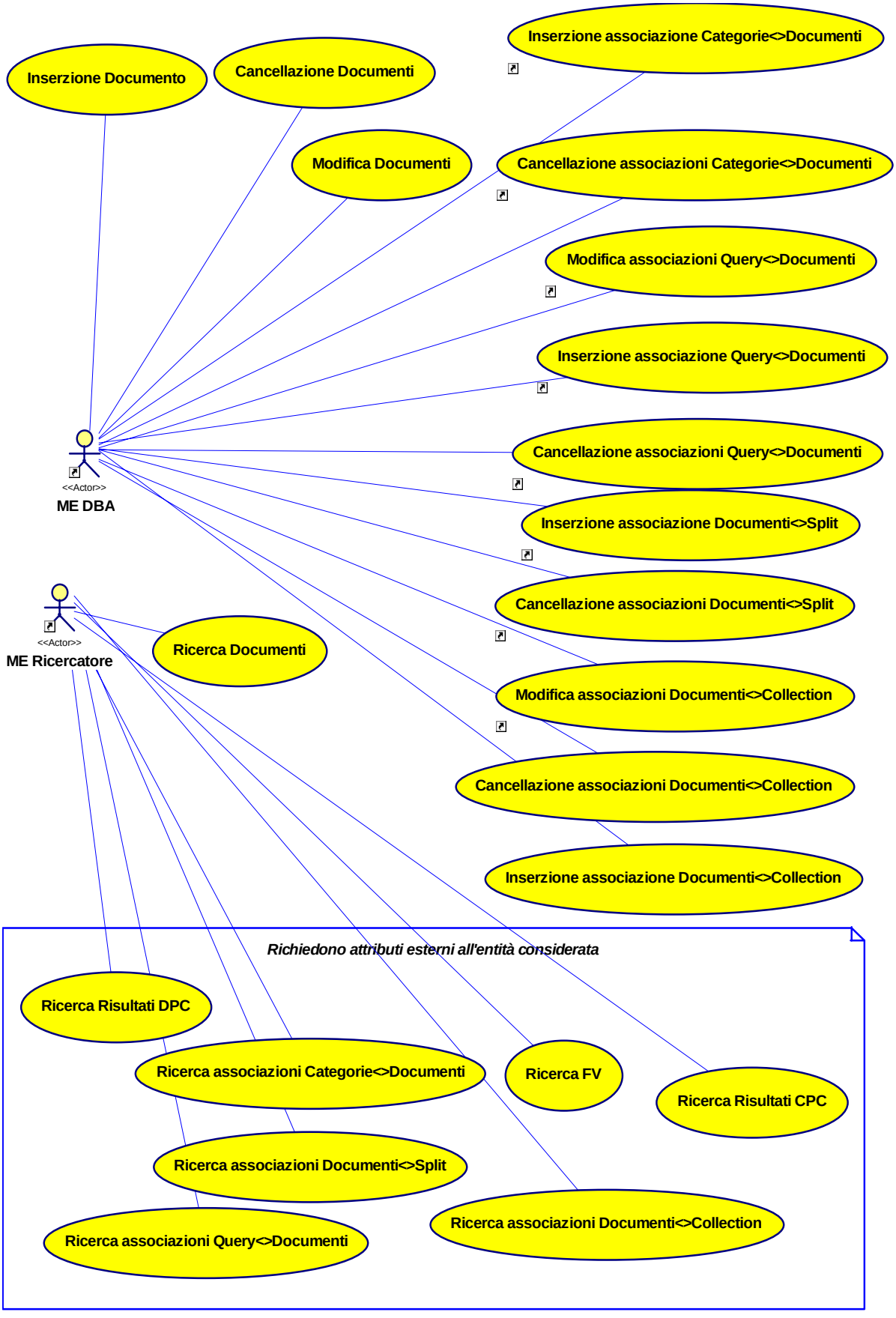
2) Modifica Documenti



3) Cancellazione Documento

Prerequisito del caso d'uso è che la cancellazione di un Documento non è possibile se appartiene ad uno Split.

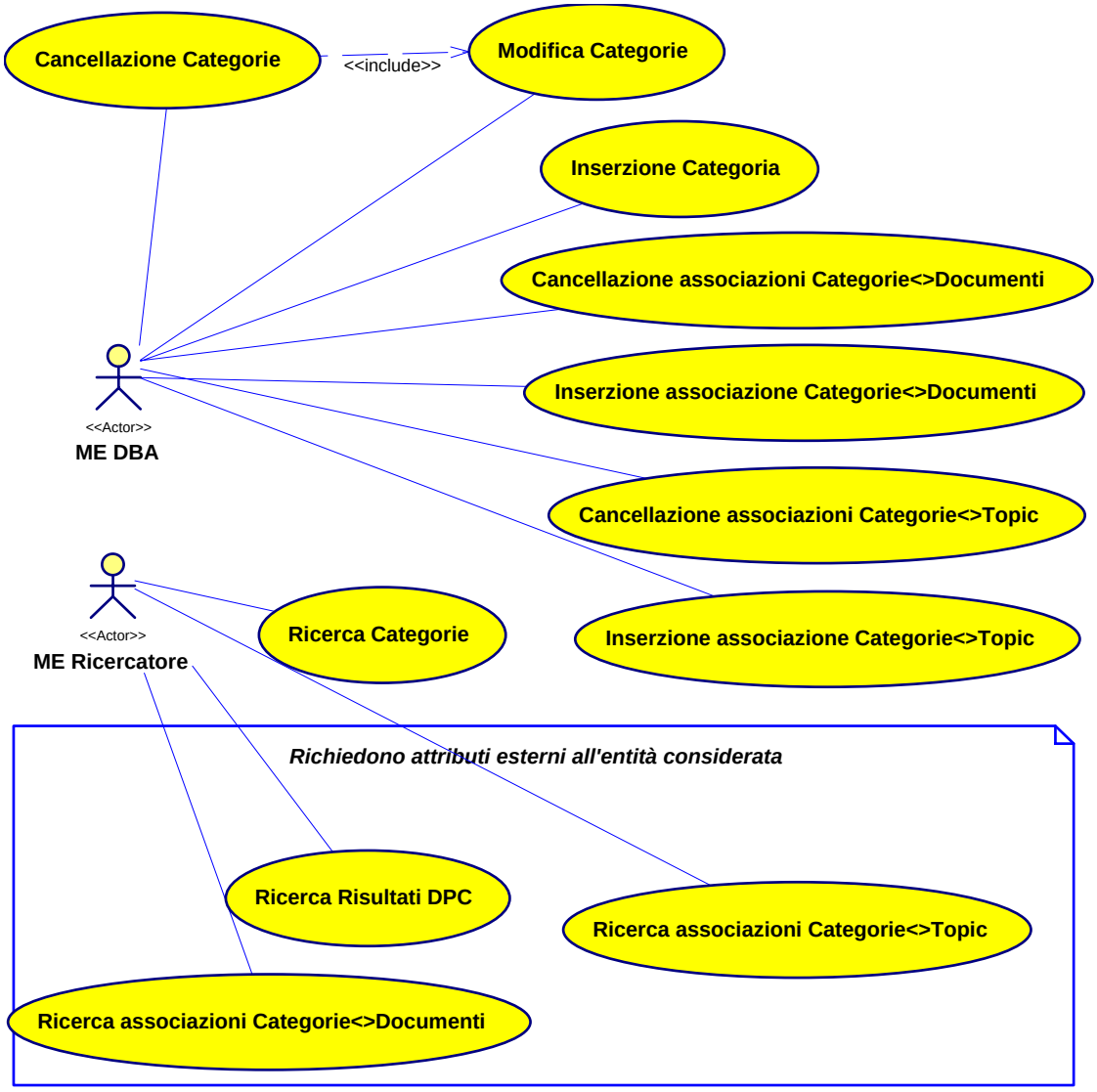
5.4.3.2 Query


Inserzione Documento
Cancellazione Documenti
Inserzione associazione Categorie<>Documenti
Modifica Documenti
Cancellazione associazioni Categorie<>Documenti
Modifica associazioni Query<>Documenti
Inserzione associazione Query<>Documenti
<<Actor>>
ME DBA
Cancellazione associazioni Query<>Documenti
Inserzione associazione Documenti<>Split
Cancellazione associazioni Documenti<>Split
<<Actor>>
ME Ricercatore
Ricerca Documenti
Modifica associazioni Documenti<>Collection
Cancellazione associazioni Documenti<>Collection
Inserzione associazione Documenti<>Collection
Richiedono attributi esterni all'entità considerata
Ricerca Risultati DPC
Ricerca associazioni Categorie<>Documenti
Ricerca FV
Ricerca Risultati CPC
Ricerca associazioni Documenti<>Split
Ricerca associazioni Documenti<>Collection
Ricerca associazioni Query<>Documenti

I casi d'uso qui presentati (*tranne* **Cancellazione Query** *il cui prerequisito per essere effettuato e la non appartenenza ad un Topic*) non hanno alcuna particolarità quindi per definire le caratteristiche proprie delle estensioni dei "*Generic Abstract*" sarà sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi - Query**: individua come dati un insieme che permetta di considerare gli attributi relativi alle Query definite nel benchmark OHSUMED: *Titolo_Query* e *Descrizione_Query*.
- **Scelta Dati Filtri - Query**: individua come *Filtri Interni* quelli che si riferiscono a: Titolo_Query e Descrizione_Query; mentre come *Filtri Esterni* quelli che si riferiscono a [Topic], [Split], [DataSet], [Collection], [Esperimento], [Documento], [Categoria].

5.4.3.3 Categorie

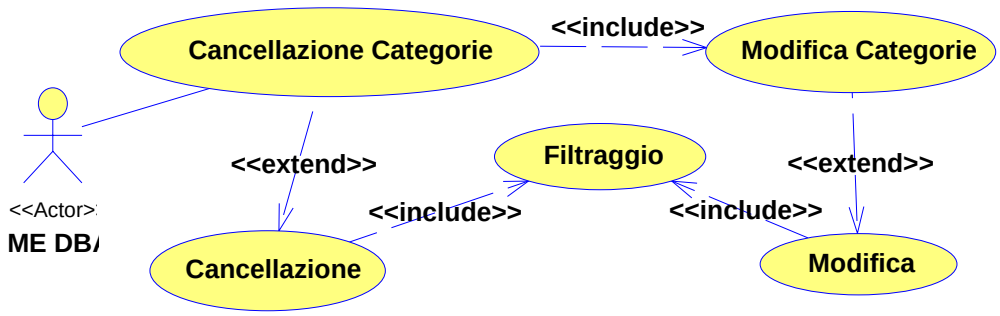


Definiamo le attività specifiche dei casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi - Categorie**: individua come dati un insieme che permetta di considerare gli attributi, identificati nei benchmark studiati, relativi alla struttura gerarchica dell'entità: *Nome_Categoria*, *Descrizione_Categoria*, [Padre], [Figlio].
- **Scelta Dati Filtri - Categorie**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Categoria, Descrizione_Categoria, [Padre], [Figlio]; mentre come *Filtri Esterni* quelli che si riferiscono a: [Topic], [Split], [Data Set], [Collection], [Documento], [Esperimento], [Query].

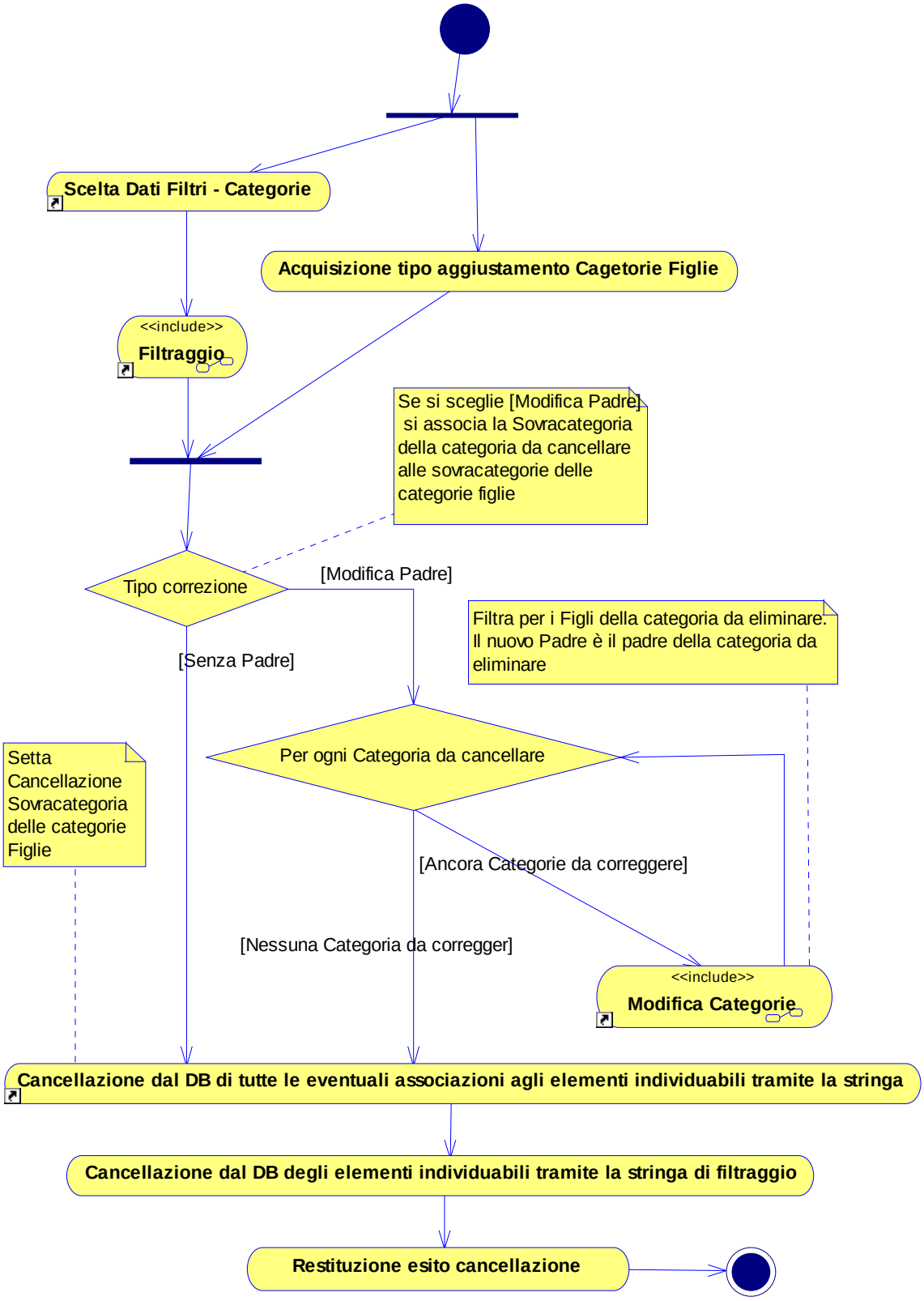
Tutti i casi d'uso seguono gli standard fin ora discussi tranne la cancellazione che richiede alcune modifiche, ciò a causa della possibilità di associare alle sottocategorie della categoria da eliminare la sovracategoria di quest'ultima, funzionalità presentata nel *paragrafo 5.2.2*.

- **Cancellazione Categorie**

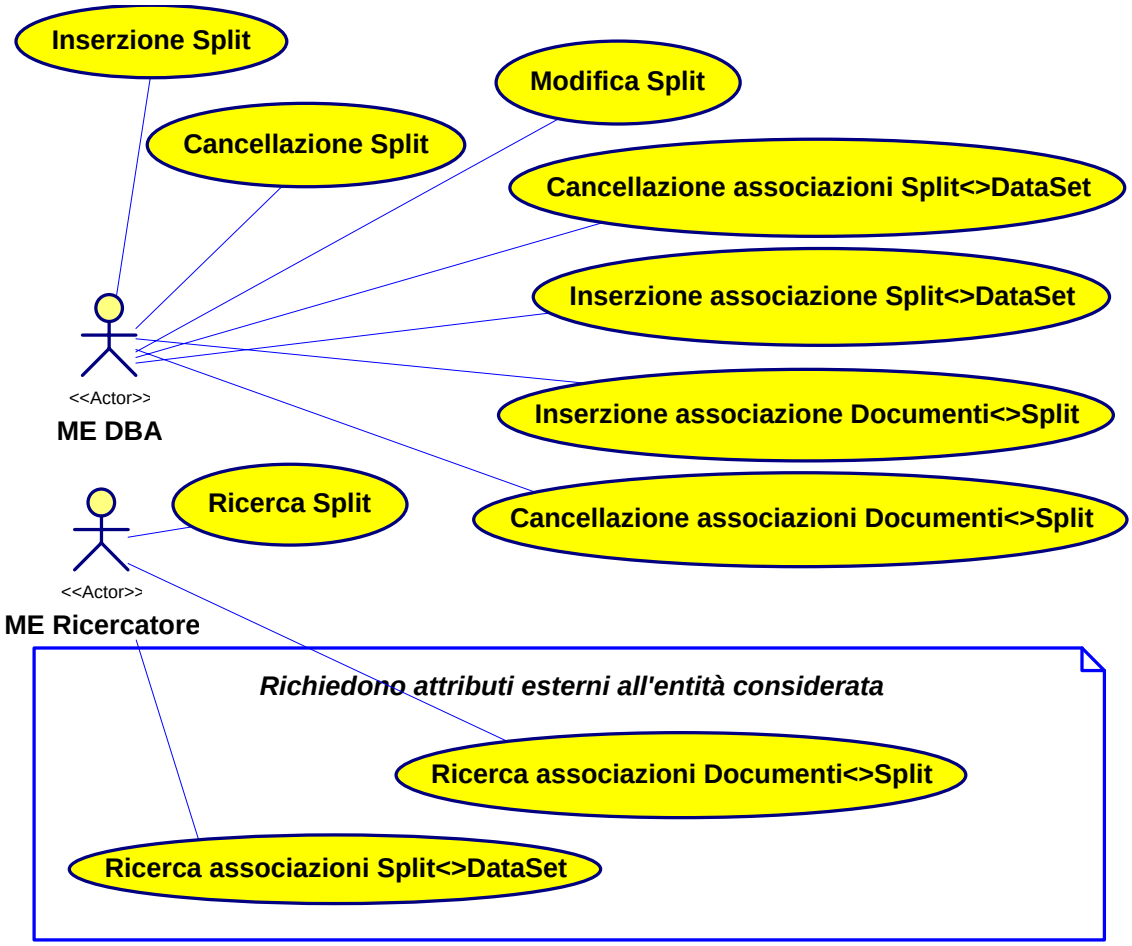


La cancellazione di una Categoria non è consentita se appartiene ad un Topic, inoltre tale caso d'uso è abbastanza complesso e oltre ad essere un estensione della cancellazione presente in "*Generic Abstract*" include al suo interno anche il caso d'uso **Modifica Categorie**.

Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso, al suo interno sono presenti anche alcune note che permettono di chiarire i passaggi più delicati.


Scelta Dati Filtri - Categorie
Acquisizione tipo aggiustamento Cagetorie Figlie
<<include>>
Filtraggio
Se si sceglie [Modifica Padre] si associa la Sovracategoria della categoria da cancellare alle sovracategorie delle categorie figlie
Tipo correzione
[Modifica Padre]
[Senza Padre]
Filtra per i Figli della categoria da eliminare. Il nuovo Padre è il padre della categoria da eliminare
Per ogni Categoria da cancellare
Setta Cancellazione Sovracategoria delle categorie Figlie
[Ancora Categorie da correggere]
[Nessuna Categoria da corregger]
<<include>>
Modifica Categorie
Cancellazione dal DB di tutte le eventuali associazioni agli elementi individuabili tramite la stringa
Cancellazione dal DB degli elementi individuabili tramite la stringa di filtraggio
Restituzione esito cancellazione

5.4.3.4 Split

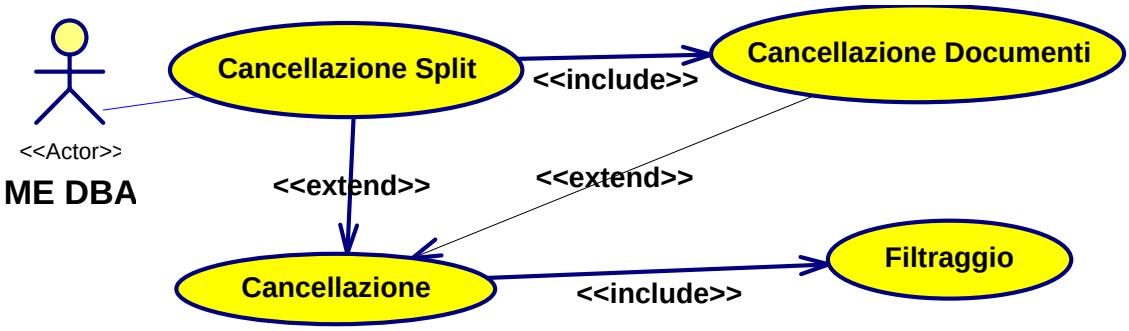


Definiamo le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi - Split**: individua come dati un insieme che permette di considerare gli attributi, identificati nei benchmark studiati, relativi agli Split: *Nome_Split*, *Tipo_Split* e *Descrizione_Split*.
- **Scelta Dati Filtri - Split**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Split, Tipo_Split e Descrizione_Split; mentre come *Filtri Esterni* quelli che si riferiscono a: [Collection], [DataSet], [Documento], [Esperimento], [Topic].

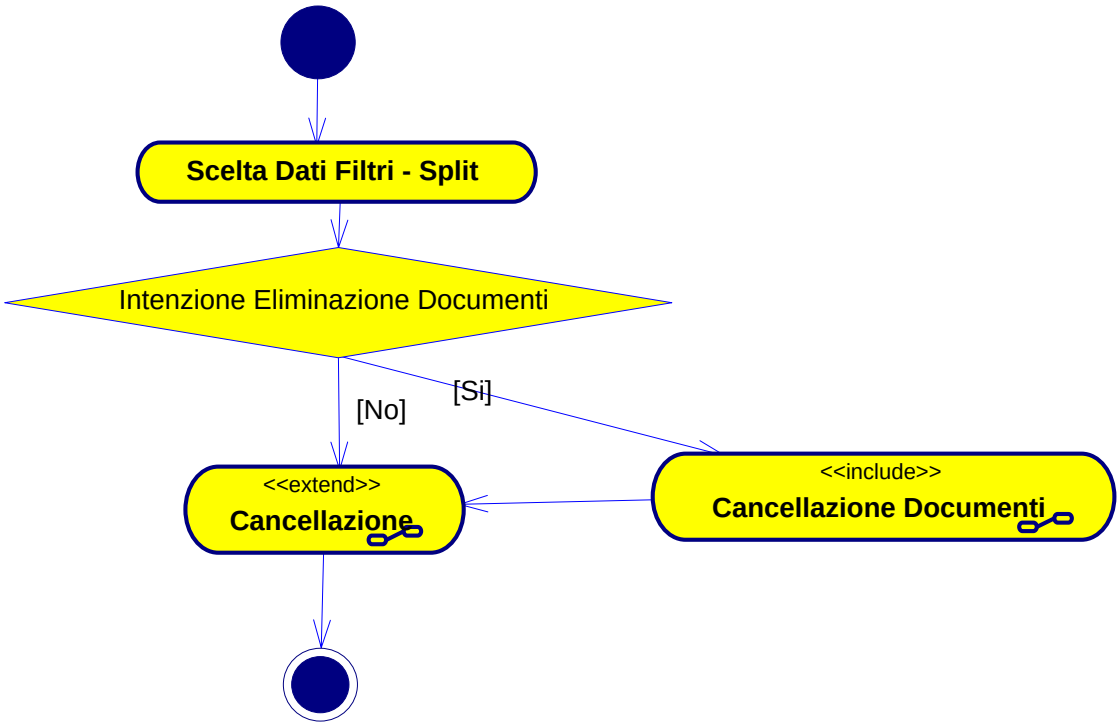
Tutti i casi d'uso seguono gli standard fin ora discussi tranne la cancellazione che richiede alcune modifiche, ciò a causa della possibilità di eliminare anche tutti i documenti degli Split considerati, funzionalità presentata nel *paragrafo 5.2.2*.

- **Cancellazione Split**

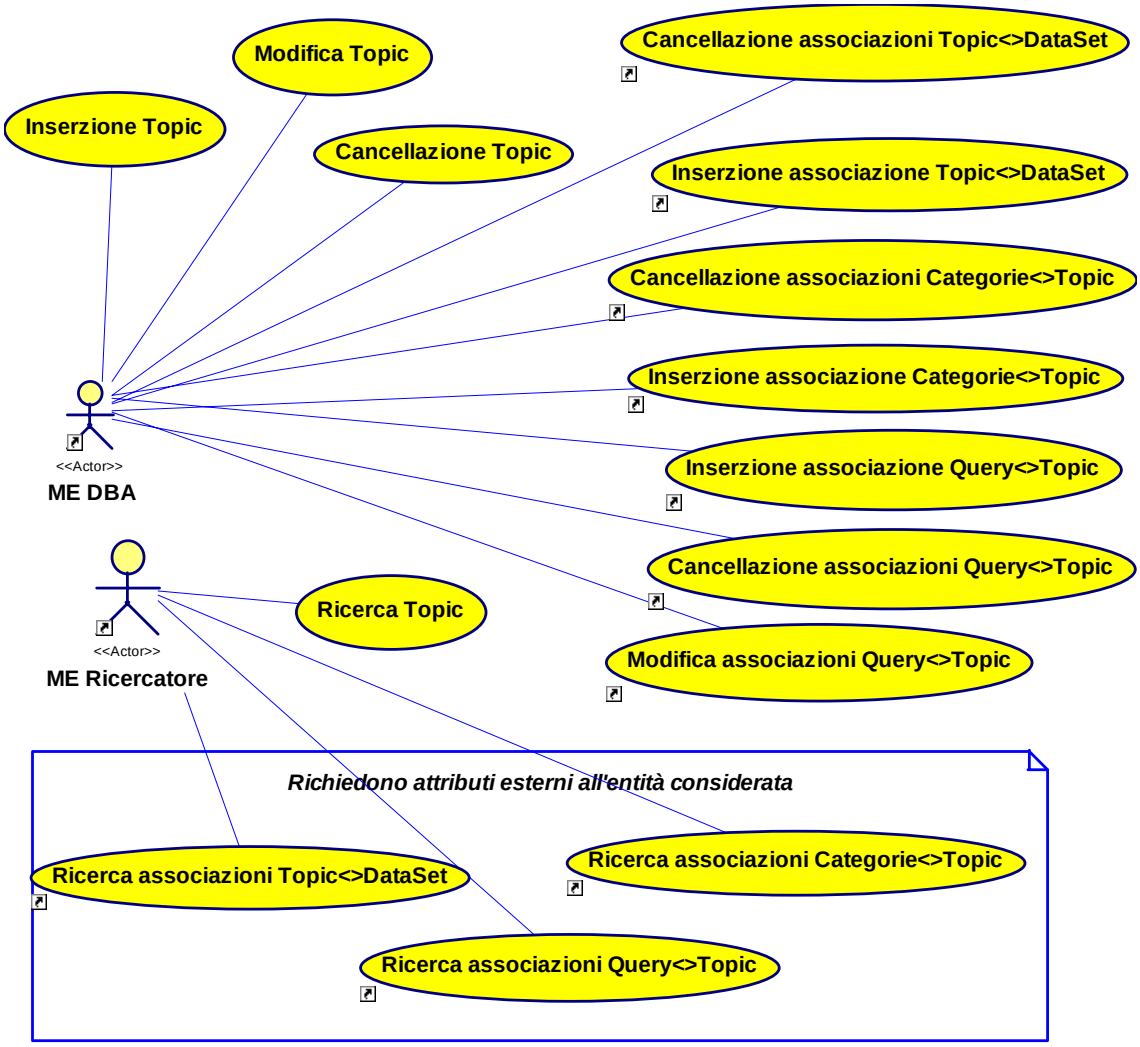


La cancellazione di uno Split non è possibile se appartiene ad un DataSet, inoltre, tale caso d'uso, oltre essere un'estensione della cancellazione presente in "*Generic Abstract*", include anche il caso d'uso **Cancellazione Documenti**.

Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso:



5.4.3.5 Topic



Definiamo le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi - Topic**: individua come dati un insieme che permetta di considerare gli attributi, identificati nei benchmark studiati, relativi ai Topic: *Nome_Topic*, *Tipo_Topic*, *Descrizione_Topic*.
- **Scelta Dati Filtri - Topic**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Topic, Tipo_Topic e Descrizione_Topic; mentre come *Filtri Esterni* quelli che si riferiscono a: [Collection], [DataSet], [Esperimento], [Query], [Categoria], [Documento], [Split].

I casi d'uso riportati di seguito richiedono alcune modifiche rispetto a quelli standard.

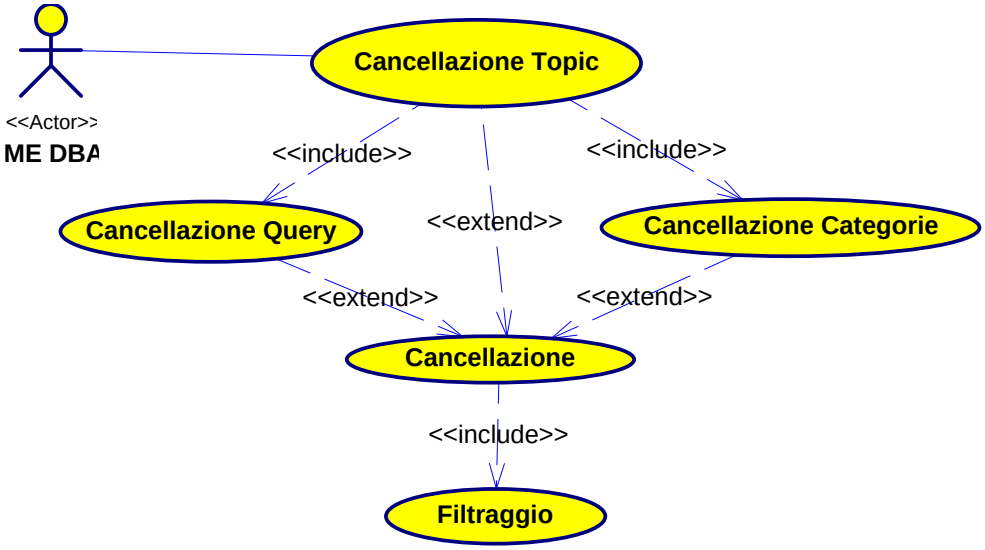
1) Modifica Topic

Questo caso d'uso è simile a quello standard. Utilizza le attività **Scelta Dati Filtri – Topic** e **Scelta Dati Attributi – Topic** con l'accortezza di non considerare tra gli

attributi modificabili il Tipo_Topic, dal momento che da esso dipendono le eventuali associazioni a Query o Categorie di un Topic.

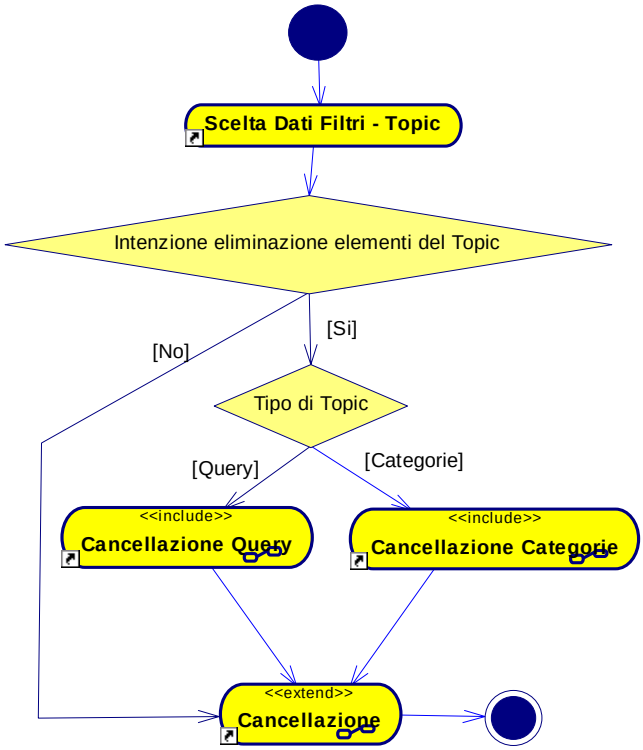
2) Cancellazione Topic

La cancellazione di un Topic non è possibile se appartiene ad un DataSet, inoltre richiede alcune modifiche a causa della possibilità di poter eliminare anche tutte le eventuali Categorie (o Query) del Topic, funzionalità presentata nel *paragrafo 5.2.2*.



Tale caso d'uso oltre ad essere un estensione della cancellazione presente in "*Generic Abstract*" include al suo interno anche i casi d'uso **Cancellazione Query** e **Cancellazione Categorie**.

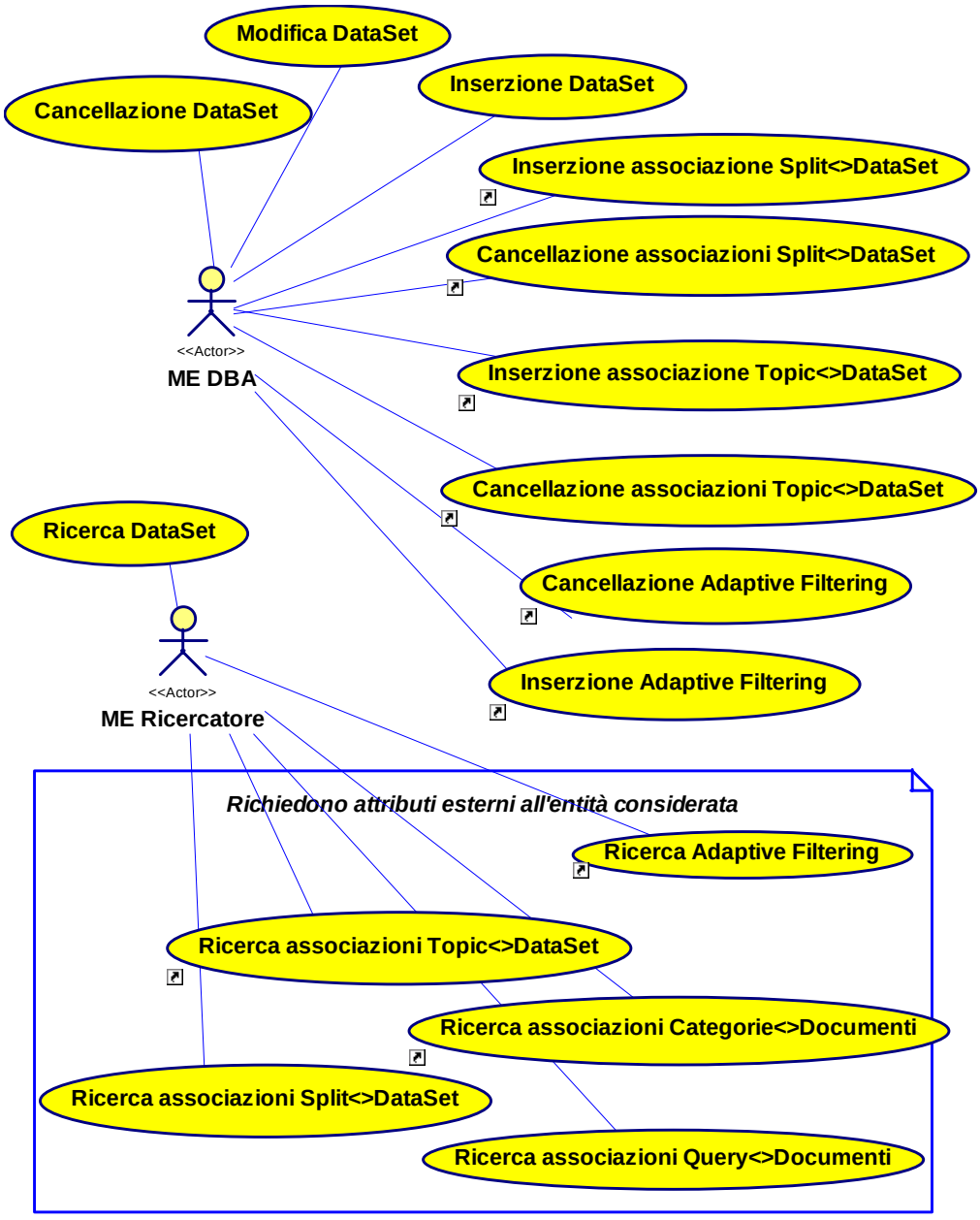
Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso:



5.4.3.6 DataSet

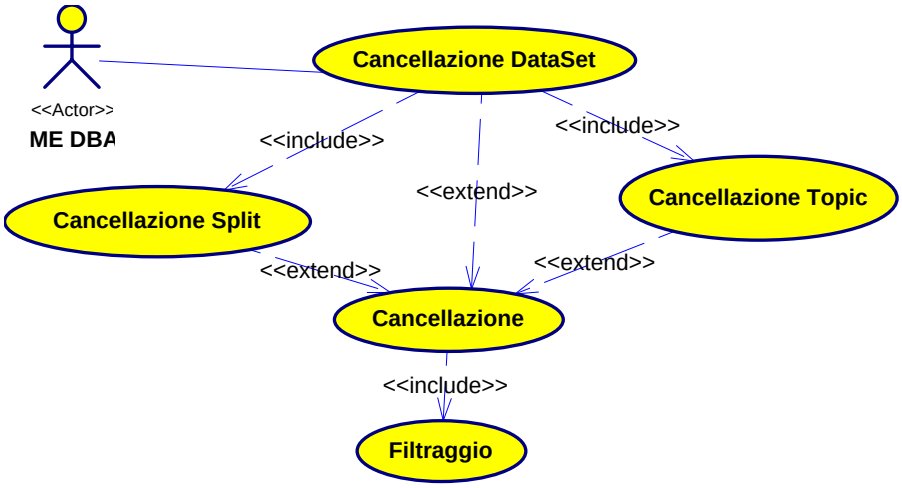
Definiamo le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi – DataSet**: individua come dati un insieme che permetta di considerare gli attributi, identificati nei benchmark studiati, relativi ai DataSet: *Nome_Data_Set*, *Tipo_DataSet*, *Descrizione_Data_Set*.
- **Scelta Dati Filtri - DataSet**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Data_Set, Tipo_DataSet, Descrizione_Data_Set; mentre come *Filtri Esterni* quelli che si riferiscono a: [Collection], [Esperimento], [Split], [Topic], [Ricercatori].



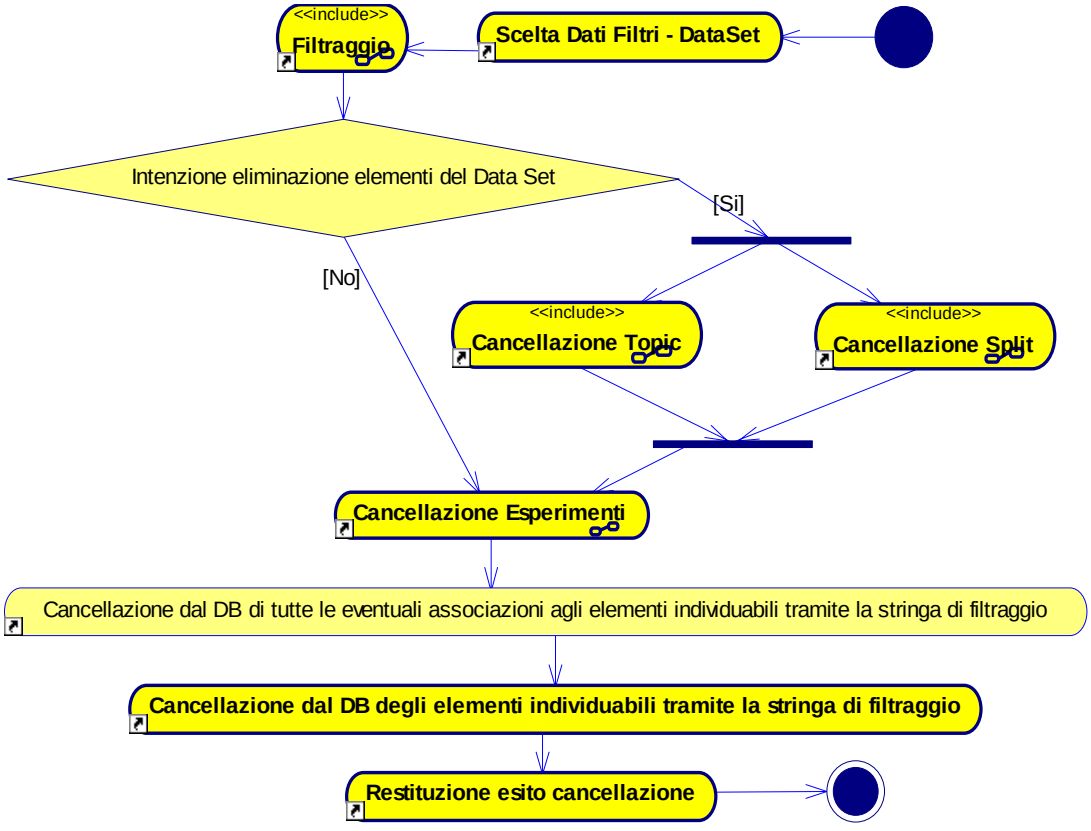
Tutti i casi d'uso seguono gli standard fin ora discussi tranne la cancellazione che richiede alcune modifiche, ciò a causa della possibilità di eliminare anche tutte le entità dipendenti dal DataSet, funzionalità presentata nel *paragrafo 5.2.2*.

- **Cancellazione DataSet**

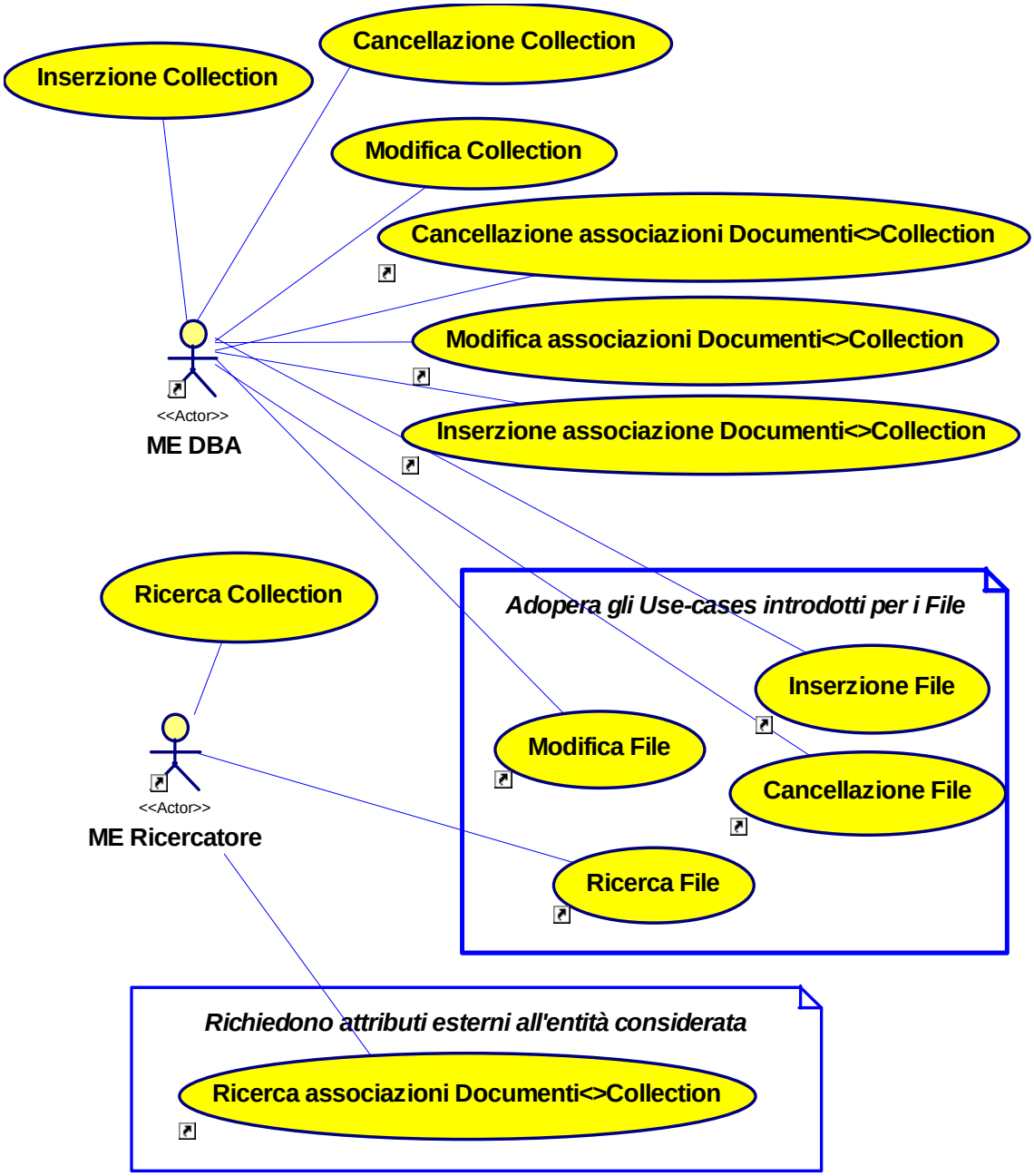


La cancellazione di un DataSet non è consentita se esistono Esperimenti che lo adoperano, inoltre tale caso d'uso oltre ad essere un estensione della cancellazione presente in "*Generic Abstract*" include al suo interno anche i casi d'uso **Cancellazione Split** e **Cancellazione Topic**.

Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso:



5.4.3.7 Collection



Definiamo le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

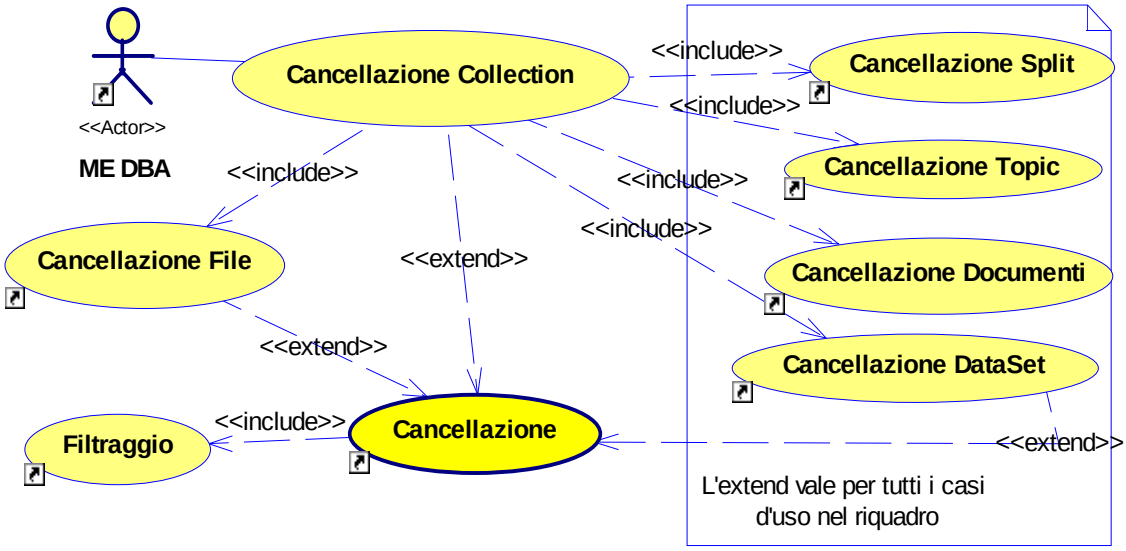
- **Scelta Dati Attributi – DataSet**: individua come dati un insieme che permette di considerare gli attributi, identificati nei benchmark studiati, relativi alle Collection: *Nome_Collection*, *Versione*, *Data_Collection*, *Autore_Collection*, *Sito*, *URL*, *Modulo di feeding*, *Note_Documenti*, *Note_Categorie*, *Descrizione_ID1*, *Descrizione_ID2*, *Descrizione_Collection*. Di seguito si riportano i dati relativi ai tre benchmark presentati nei capitoli precedenti:

Nome_Collection	Reuters-21578	OHSUMED	20 Newsgroups
Versione	Distribution 1.0		
Data_Collection	26-set-97	06-lug-94	09-set-99
Autore_Collection	David D. Lewis	William Hersh	Ken Lang; Tom Mitchell
Sito	UCI KDD Archive	Text REtrieval Conference	UCI KDD Archive
URL	http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html	http://trec.nist.gov/data.html	http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
Modulo_di_feeding	ReutersIn.exe	ohsumedIn	NewsgroupsIn
Note_Documenti	Newswire collection	Subset of the Medline document base	Messages posted to Usenet newsgroups
Note_Categorie	Categories related to economics	Categories are the postable terms of the MeSH thesaurus	Categories are the newsgroups
Descrizione_ID1	OLDID: The identification number (ID) the story had in the Reuters-22173 collection	Sequential identifier (important note: documents should be processed in this order).	Identifica il numero del File che contiene il documento, si noti che tale numero non è univoco poiché in realtà dipende dalla categoria a cui appartiene il Documento e che tale informazione viene inserita per completezza.
Descrizione _ID2	NEWID: The identification number (ID) the story has in the Reuters-21578, Distribution 1.0 collection. These IDs are assigned to the stories in chronological order	MEDLINE identifier (UI) (<DOCNO> used for relevance judgments).	Non adoperato
Descrizione_ _Collection		I campi dei documenti sono da interpretare come: PROVENIENZA: rappresenta il campo "Source" DATA_DOCUMENTO: è rappresentativo solo l'anno TESTO: rappresenta il campo "Abstract"	I campi dei documenti sono da interpretare come: PROVENIENZA: rappresenta il campo "Organization" AUTORE_DOCUMENTO: rappresenta il campo "From" TITOLO_DOCUMENTO: rappresenta il campo "Subject" LINEE: rappresenta il campo "Lines" (nel Data Set originale ci sono alcuni errori)

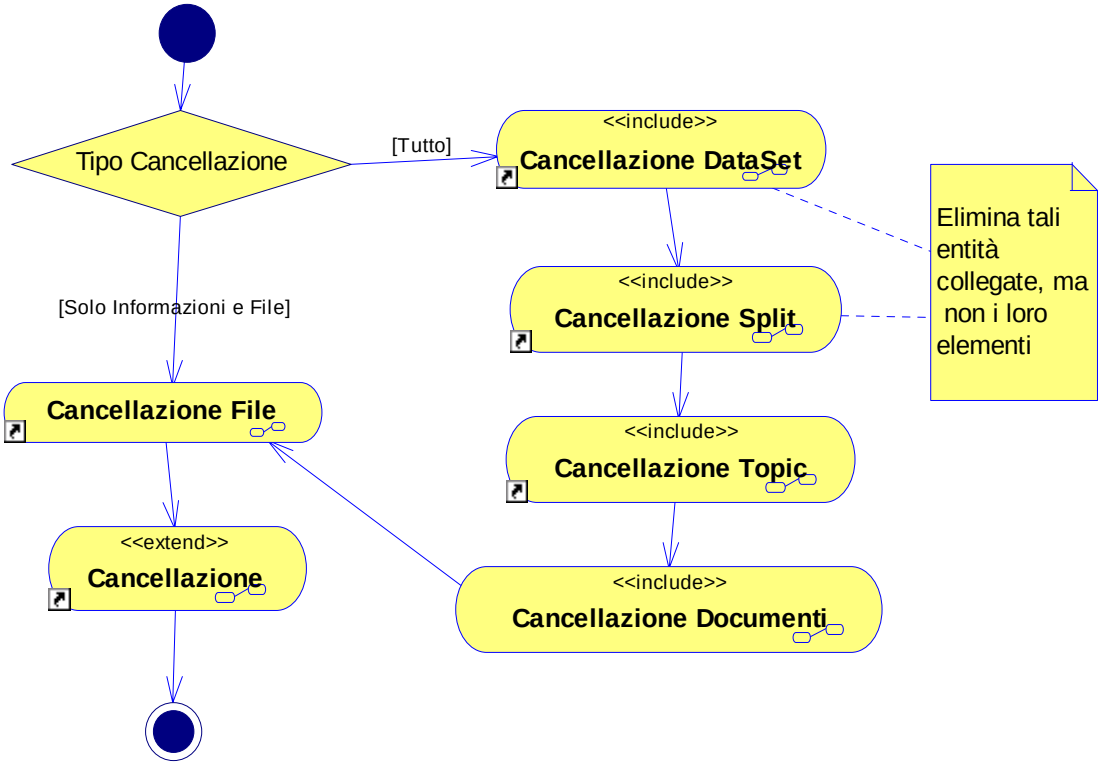
- **Scelta Dati Filtri - DataSet**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Collection, Versione, Data_Collection, Autore_Collection, Sito, URL, Modulo_di_feeding, Note_Documenti, Note_Categorie, Descrizione_ID1, Descrizione_ID2, Descrizione_Collection; mentre come *Filtri Esterni* quelli che si riferiscono a: [DataSet], [Split], [Topic], [Categoria], [Query], [Documento], [Esperimento].

Tutti i casi d'uso seguono gli standard fin ora discussi tranne la cancellazione che richiede alcune modifiche, ciò a causa della possibilità di eliminare anche tutte le entità dipendenti dalla Collection, funzionalità presentata nel *paragrafo 5.2.2*.

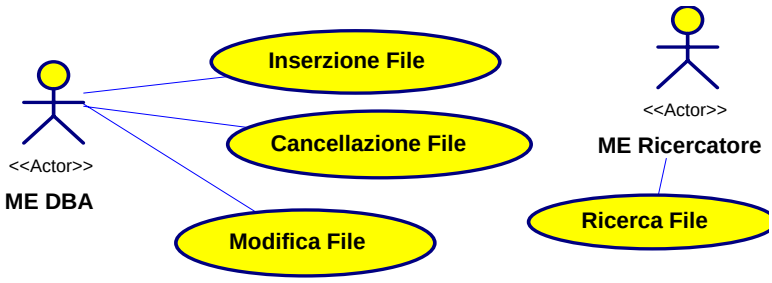
- **Cancellazione Collection**



Tale caso d'uso, oltre ad essere un'estensione della cancellazione presente in "*Generic Abstract*", include anche i casi d'uso **Cancellazione File** e **Cancellazione DataSet**.
Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso:



5.4.3.8 File



I quattro casi d'uso qui presentati non hanno alcuna particolarità, quindi, per definire le caratteristiche delle estensioni ricavate dai "*Generic Abstract*", è sufficiente descrivere le due attività specifiche:

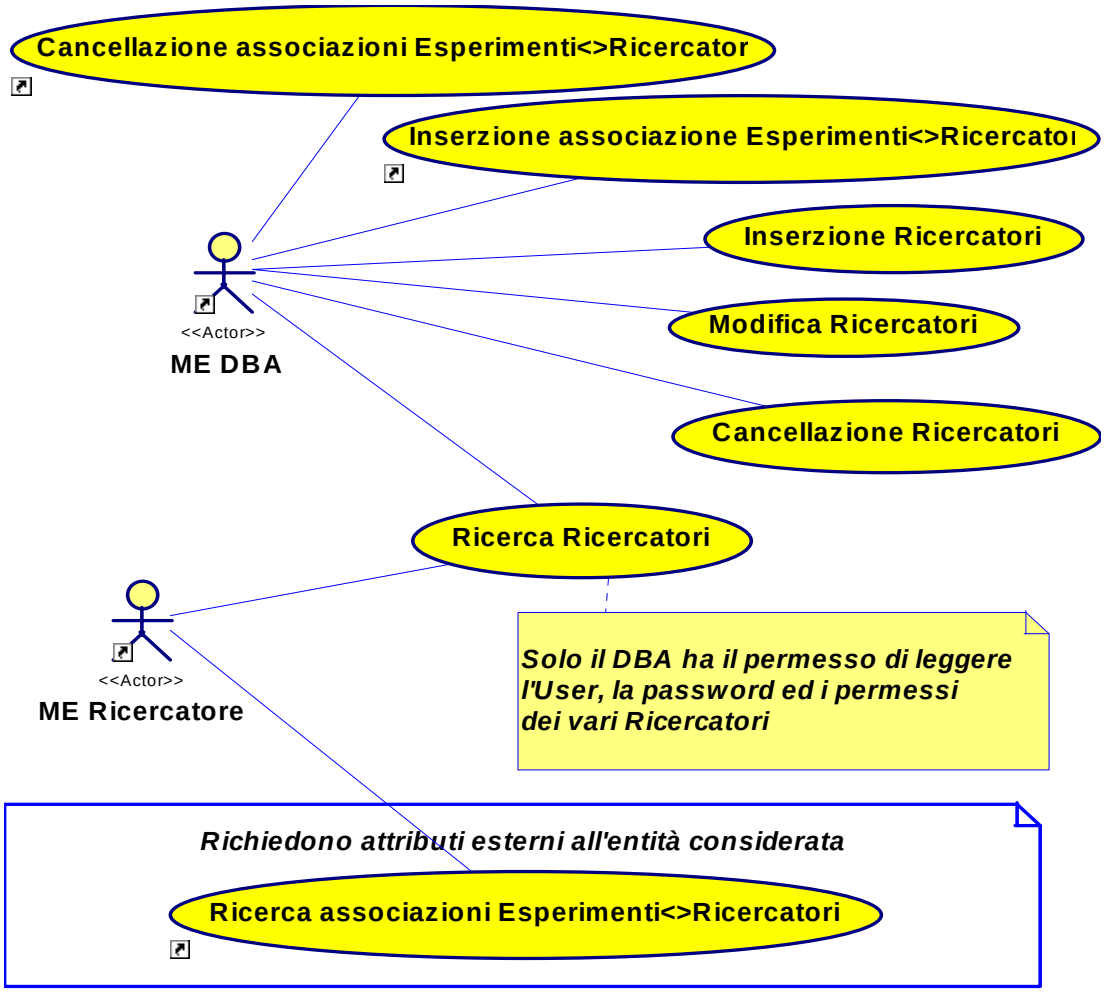
- **Scelta Dati Attributi - File**: individua come dati un insieme che permette di considerare tutti gli attributi, identificati nei benchmark, relativi ai file: *Nome_File*, *Descrizione_File*.
- **Scelta Dati Filtri - Query**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_File, Descrizione_File; mentre come unico *Filtri Esterni* quello che si riferisce a [Collection].

5.5 L'unità organizzativa Esperimenti*

Si fa osservare che nella descrizione del paragrafo precedente molte delle cancellazioni sono vincolate ad entità superiori e che, infine, la cancellazione di un DataSet è vincolata al suo eventuale utilizzo da parte di un Esperimento. Tale decisione è stata presa per preservare l'insieme di dati necessario ad un Esperimento in corso.
*Esperimenti** contiene al suo interno i casi d'uso che si riferiscono alla gestione delle entità presentate nel *paragrafo 5.2.3* e delle loro relazioni.

Incominceremo col descrivere i casi d'uso relativi alle informazioni dei *Ricercatori* e delle *Effectiveness Misures* per poi passare a quelli relativi agli *Esperimenti* al cui interno vi sono anche tutti quelli ricavabili dalla gestione dei "***Dati Esperimento***" introdotti nel *paragrafo 1.3*.
Si osservi che nei casi d'uso a seguire apparirà finalmente l'Attore **ME Ricercatore XXX**, il cui utilizzo sarà ricorrente poiché molte delle funzionalità richiederanno la verifica d'appartenenza dei dati al particolare ricercatore.

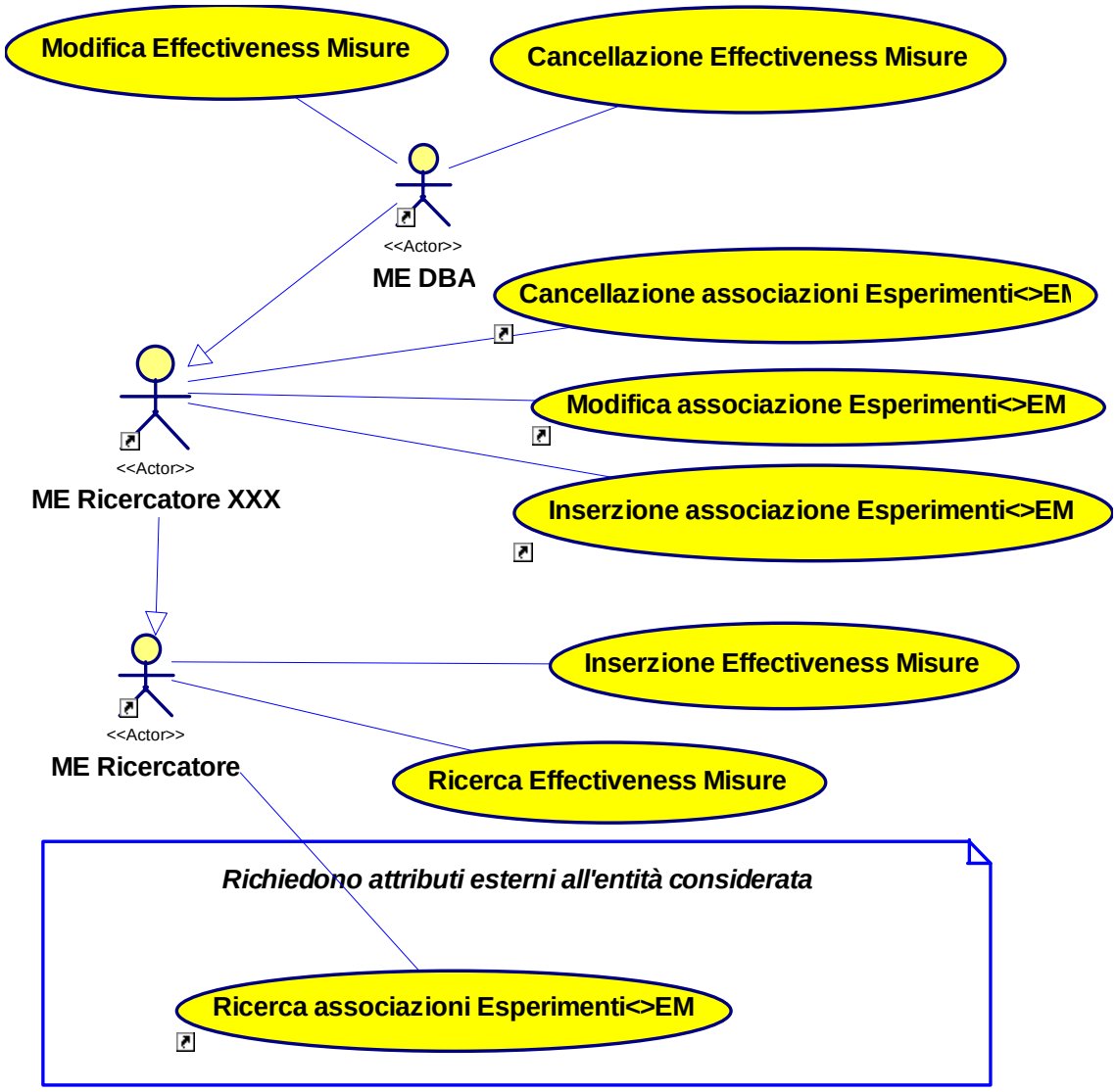
5.5.1 Ricercatori



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi – Ricercatori**: individua come dati un insieme di informazioni tipicamente associate ad un Ricercatore: *Nome*, *Cognome*, *Data_Nascita*, *Abitazione*, *Citta*, *Telefono_Casa*, *Telefono_Cellullare*, *Dipartimento*, *User*, *Password*, *Gruppo* (DBA o Ricercatore). **L'unica osservazione di rilievo è l'accesso ai dati User, Password e Gruppo, che è consentito solo ai DBA.**
- **Scelta Dati Filtri - Ricercatori**: individua come *Filtri Interni* quelli che si riferiscono a: Nome, Cognome, Data_Nascita, Abitazione, Citta, Telefono_Casa, Telefono_Cellullare, Dipartimento, User, Password, Gruppo; mentre come *Filtri Esterni* quelli che si riferiscono a: [Esperimento], [Topic], [Split], [DataSet], [Collection].

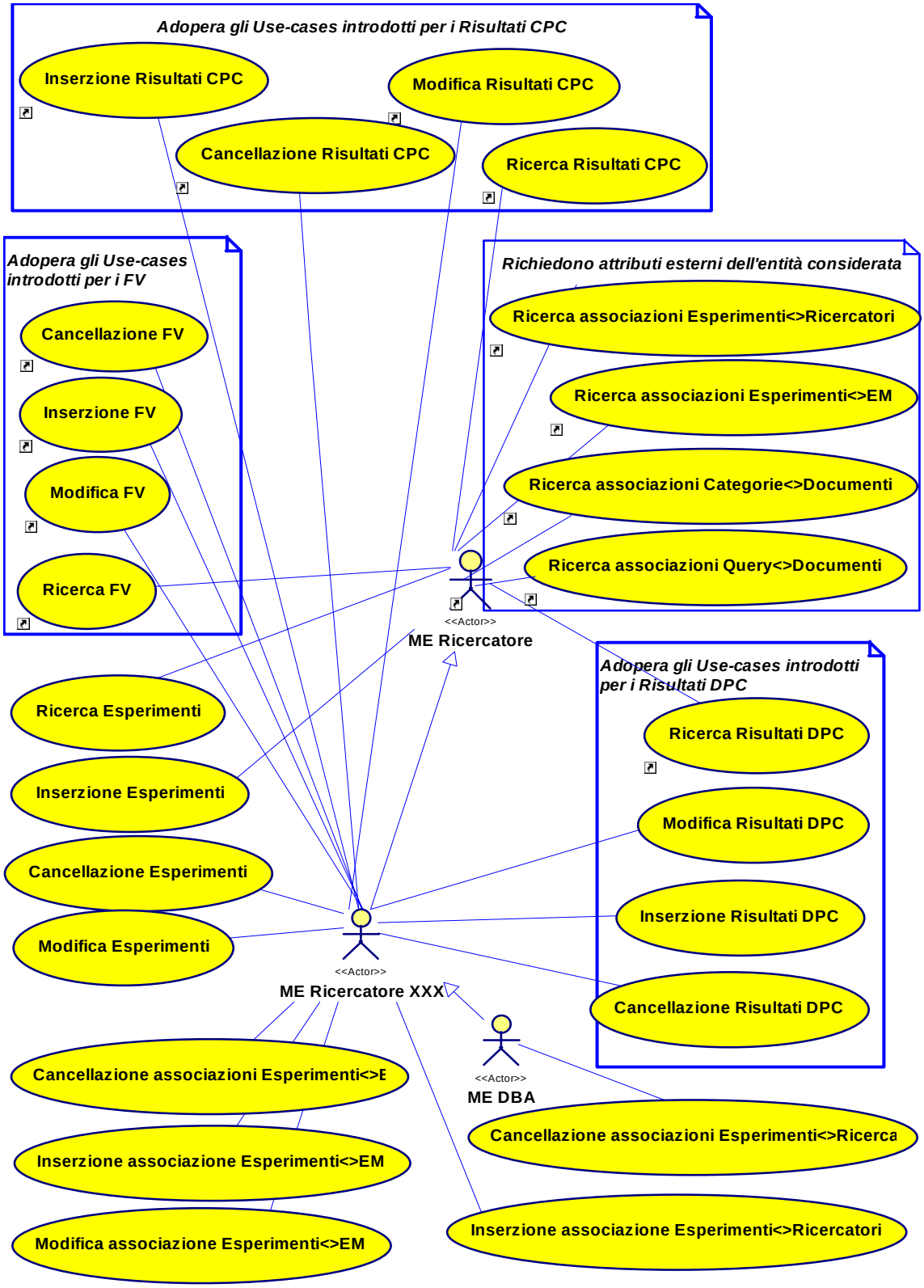
5.5.2 Effectiveness Misure



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi – Effectiveness Misure**: individua come dati un insieme di informazioni associate alla definizione di una Misura: *Nome_Misura*, *Descrizione*.
- **Scelta Dati Filtri - Effectiveness Misure**: individua come *Filtri Interni* quelli che si riferiscono a: Nome_Misura, Descrizione; mentre come *Filtro Esterno* solo l'[Esperimento].

5.5.3 Esperimenti



Come si può notare dal diagramma molti dei casi d'uso si riferiscono ad entità specifiche degli *Esperimenti* le cui funzionalità erano già state individuate nel *paragrafo 5.2.3*. Prima di analizzare tali casi d'uso descriviamo quelli che si riferiscono prettamente agli Esperimenti:

Definiamo le attività specifiche dei vari casi d'uso base in modo da poter individuare gli attributi dell'entità ed i filtraggi desiderati:

- **Scelta Dati Attributi – Esperimenti**: individua come dati un insieme che permette di considerare il tipo di esperimento e le Effectiveness Misure che tipicamente sono adoperate per la sua valutazione: *Nome Esperimento*, *Tipo Esperimento*, *Descrizione Esperimento*, *Microaveraging Precision*, *Microaveraging Recall*, *Macroaveraging Precision*, *Macroaveraging Recall*, *Accuracy*, *Error*, *Training Efficiency*, *Classification Efficiency*, *Utility*.

  Di seguito si riportano delle brevi descrizioni relative alle misure indicate:

  1) Per quanto riguarda le misure *Precision* e *Recall* si devono prima introdurre i concetti di:
  - **TP_i** individua il numero di documenti correttamente classificati come appartenenti alla categoria c_i.
  - **TN_i** individua il numero di documenti correttamente classificati come non appartenenti alla categoria c_i.
  - **FP_i** individua il numero di documenti erratamente classificati come appartenenti alla categoria c_i.
  - **FP_i** individua il numero di documenti erratamente classificati come non appartenenti alla categoria c_i.

  Poi è necessario definire la **precision** (permette di calcolare la probabilità secondo cui classificato un documento a caso d_x sotto la categoria c_i, tale decisione sia corretta): $\hat{\pi}_i = \frac{TP_i}{TP_i + FP_i}$

  e il **recall** (che è la probabilità secondo cui un documento a caso d_x venga classificato sotto la categoria c_i): $\hat{\rho}_i = \frac{TP_i}{TP_i + FN_i}$

  Date tali definizioni, e indicata con $|C|$ la cardinalità dell'insieme delle categorie, è ora possibile introdurre le misure:

  Microaveraging Precision: $\hat{\pi}^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)}$

  Microaveraging Recall: $\hat{\rho}^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$

Macroaveraging Precision: $\widehat{\pi}^{M} = \frac{\sum_{i=1}^{|C|} \widehat{\pi}_i}{|C|}$

Macroaveraging Recall: $\widehat{\rho}^{M} = \frac{\sum_{i=1}^{|C|} \widehat{\rho}_i}{|C|}$

2) *Accuracy*: $\widehat{A} = \frac{TP + TN}{TP + TN + FP + FN}$

3) *Error*: $\widehat{E} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \widehat{A}$

4) *Training Efficiency*: è il tempo medio impiegato per costruire un classificatore per una categoria c_i a partire da un Tr.

5) *Classification Efficiency*: è il tempo medio impiegato per classificare un nuovo documento rispetto ad una categoria c_i.

6) *Utilità*: si basa sull'utilizzo d'alcuni valori legati ai concetti economici di guadagno e perdita. I parametri da considerare sono:

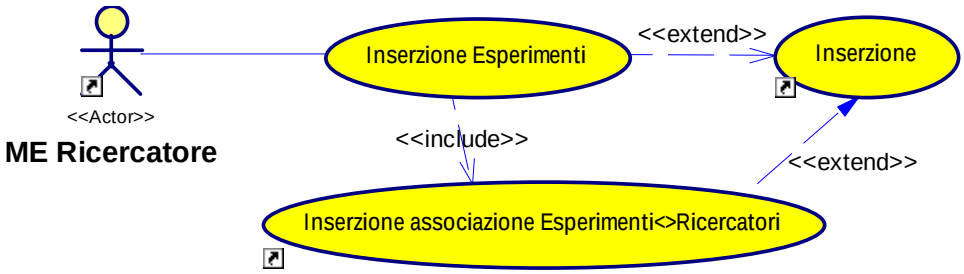
- μ_{TP}: legato al guadagno ricavato dai TP.
- μ_{FP}: legato al guadagno ricavato dai FP.
- μ_{FN}: legato al guadagno ricavato dai FN.
- μ_{TN}: legato al guadagno ricavato dai TN.

Dove naturalmente i valori di μ_{TP} e μ_{TN} sono maggiori di quelli di μ_{FP} e μ_{FN}. La definizione di tali grandezze ci avverte che tale misura non è univocamente determinata e che dipende dai valori considerati, dunque sarà necessario, all'occorrenza, riportare, in *Descrizione_Esperimento*, le decisioni prese riguardo al calcolo di tale misura).

- **Scelta Dati Filtri - Esperimenti**: individua come *Filtri Interni* quelli che si riferiscono agli attributi dell'entità: Nome_Esperimento, Tipo_Esperimento, Descrizione_Esperimento, Microaveraging Precision, Microaveraging Recall, Macroaveraging Precision, Macroaveraging Recall, Accuracy, Error, Training Efficiency, Classification Efficiency, Utility; mentre come *Filtri Esterni* quelli che si riferiscono a: [DataSet], [Ricercatore], [Topic], [Split], [Collection], [Effectiveness Misure].

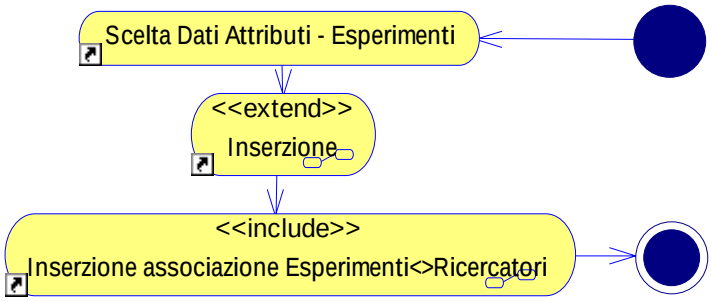
Tutti i casi d'uso seguono gli standard fin ora discussi tranne l'inserimento che aggiunge oltre all'Esperimento anche l'associazione tra l'Esperimento ed il Ricercatore che lo ha creato:

- **Inserzione Esperimento**

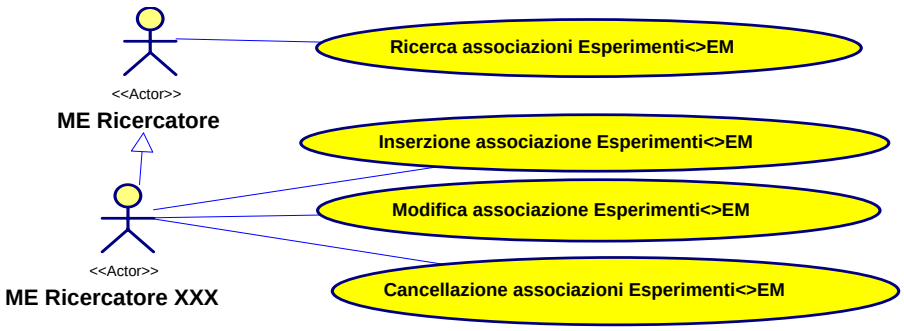


Tale caso d'uso oltre ad essere un estensione dell'inserzione presente in "*Generic Abstract*" include al suo interno anche il caso d'uso **Inserzione associazione Esperimenti<>Ricercatori**.

Di seguito si riporta l'activity diagram costruito per descrivere il caso d'uso:



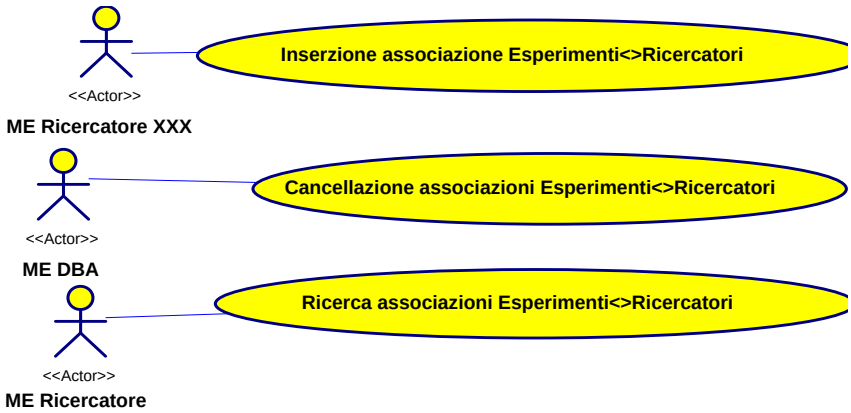
5.5.3.1 Esperimenti<>EM



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi - Esperimenti<>EM**: individua come dati i valori in grado di identificare un Esperimento e un'Effectiveness Misure: [Esperimento], [Effectiveness Misure], inoltre si considera il valore associato alla misura aggiunta *Valore_Misura*.
- **Scelta Dati Filtri - Categorie<>Documenti**: individua come *Filtri Interni* quelli che si riferiscono agli attributi: [Esperimento], [Effectiveness Misure], Valore_Misura; mentre non ci sono *Filtri Esterni* di rilievo.

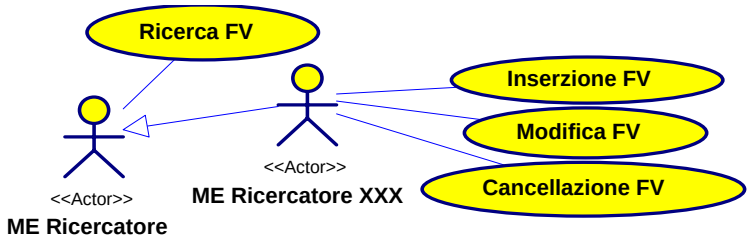
5.5.3.2 Esperimenti<>Ricercatori



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi - Esperimenti<>Ricercatori**: individua come dati i valori in grado di identificare un Esperimento e un Ricercatore: [Esperimento], [Ricercatore].
- **Scelta Dati Filtri - Categorie<>Ricercatori**: individua come *Filtri Interni* quelli che si riferiscono agli attributi: [Esperimento], [Ricercatore]; mentre non ci sono *Filtri Esterni* di rilievo.

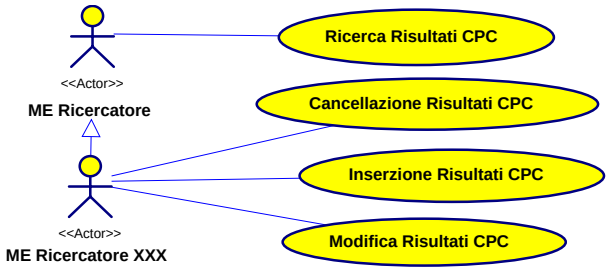
5.5.3.3 FV



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi - FV**: individua come dati i valori in grado di identificare un Esperimento e un Documento: [Esperimento], [Documento], inoltre si considerano due attributi in grado di identificare la particolare Feature considerata ed il suo valore: *Nome_Feature*, *Misura*.
- **Scelta Dati Filtri - FV**: individua come *Filtri Interni* quelli che si riferiscono agli attributi: [Esperimento], [Documento], Nome_Feature, Misura; mentre come *Filtro Esterno* solo [DataSet].

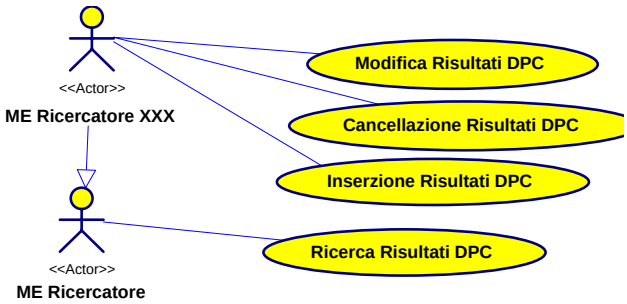
5.5.3.4 Risultati CPC



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi – Risultati CPC**: individua come dati i valori in grado di identificare un Esperimento, un Documento e una Query: [Esperimento], [Query], [Documento], inoltre si considera un attributo in grado di memorizzare un *Parametro_CPC* che possa comunicare informazioni aggiuntive sul filtraggio effettuato.
- **Scelta Dati Filtri - Risultati CPC**: individua come *Filtri Interni* quelli che si riferiscono a: [Esperimento], [Query], [Documento], Parametro_CPC; mentre come *Filtro Esterno* solo [DataSet].

5.5.3.5 Risultati DPC



I casi d'uso qui presentati non hanno alcuna particolarità quindi per definire le caratteristiche delle estensioni dei "*Generic Abstract*" è sufficiente descrivere le due attività specifiche:

- **Scelta Dati Attributi – Risultati DPC**: individua come dati i valori in grado di identificare un Esperimento, un Documento e una Categoria: [Esperimento], [Categoria], [Documento], inoltre si considera un attributo in grado di memorizzare un *Parametro_DPC* che possa comunicare informazioni aggiuntive sulla classificazione effettuata.
- **Scelta Dati Filtri - Risultati DPC**: individua come *Filtri Interni* quelli che si riferiscono a: [Esperimento], [Categoria], [Documento], Parametro_DPC; mentre come *Filtro Esterno* solo [DataSet].

CAPITOLO 6:

IL DATABASE DI INTEGRAZIONE

6.1 Organizzazione della documentazione

6.2 Schema Concettuale dei Benchmarks

6.3 Schema Concettuale degli Esperimenti

6.4 Schema Logico dei Benchmarks

6.5 Schema Logico degli Esperimenti

6.6 Viste

6.7 Restrizioni del codice SQL dovute ad ACCESS

6.8 I file DBDSATC ed il loro utilizzo

In questo capitolo si procede alla realizzazione del database che dovrà contenere tutte le informazioni necessarie ad implementare le funzionalità descritte nel capitolo precedente. Lo schema logico finale sarà realizzato cercando di svincolarsi il più possibile dal DBMS Access, e quando ciò sarà impossibile si utilizzeranno delle strutture alternative implementabili in ogni DBMS tramite comandi SQL standard.

6.1 Organizzazione della documentazione

Gli schemi riportati nel capitolo, riguardanti la progettazione concettuale del database, sono implementati nel file "*ConceptualDataModel_DBDSATC v2.0.cdm*", mentre quelli relativi alla progettazione logica in "*PhysicalDataModel_DBDSATC v2.0.pdm*". Tutta la documentazione di interesse è stata inserita, come nei casi precedenti, nel workspace *DataSetDB.sws* (si veda il *paragrafo 1.6*).
Per individuare con più chiarezza l'organizzazione d'entità e relazioni si sono realizzati per ogni passo di sviluppo del database due schemi, uno relativo ai dati che si riferiscono ai soli benchmark e l'altro a quelli adoperati per gli esperimenti (naturalmente in quest'ultimo appariranno anche le entità relazionate direttamente alle informazioni degli esperimenti).
Si osservi che, all'interno del database realizzato, tutti i nomi relativi alle tabelle, agli attributi, ed ai vincoli sono da intendersi in maiuscolo, mentre i caratteri '<' e '>' sono convertiti in '_'.

6.2 Schema Concettuale dei Benchmarks

Lo schema riportato è stato realizzato utilizzando tutte le entità degli schemi finali dei benchmark studiati. I campi riportanti informazioni relative ai file sono stati estratti dalle diverse entità per essere conservati in strutture dati contenenti le informazioni relative ai particolari benchmark; ciò permette di realizzare entità più snelle e con dati significativi per i ricercatori (le informazioni dell'organizzazione fisica dei benchmark possono essere mascherata dal momento che non sono necessarie ai fini dello sviluppo di esperimenti su algoritmi di addestramento e classificazione). Il modello E-R riportato di seguito fornisce una descrizione esplicita e di facile lettura. I sottoparagrafi realizzati hanno il compito di descrivere quanto non evidente dallo schema e di giustificare le decisioni prese nel realizzare la struttura riportata.

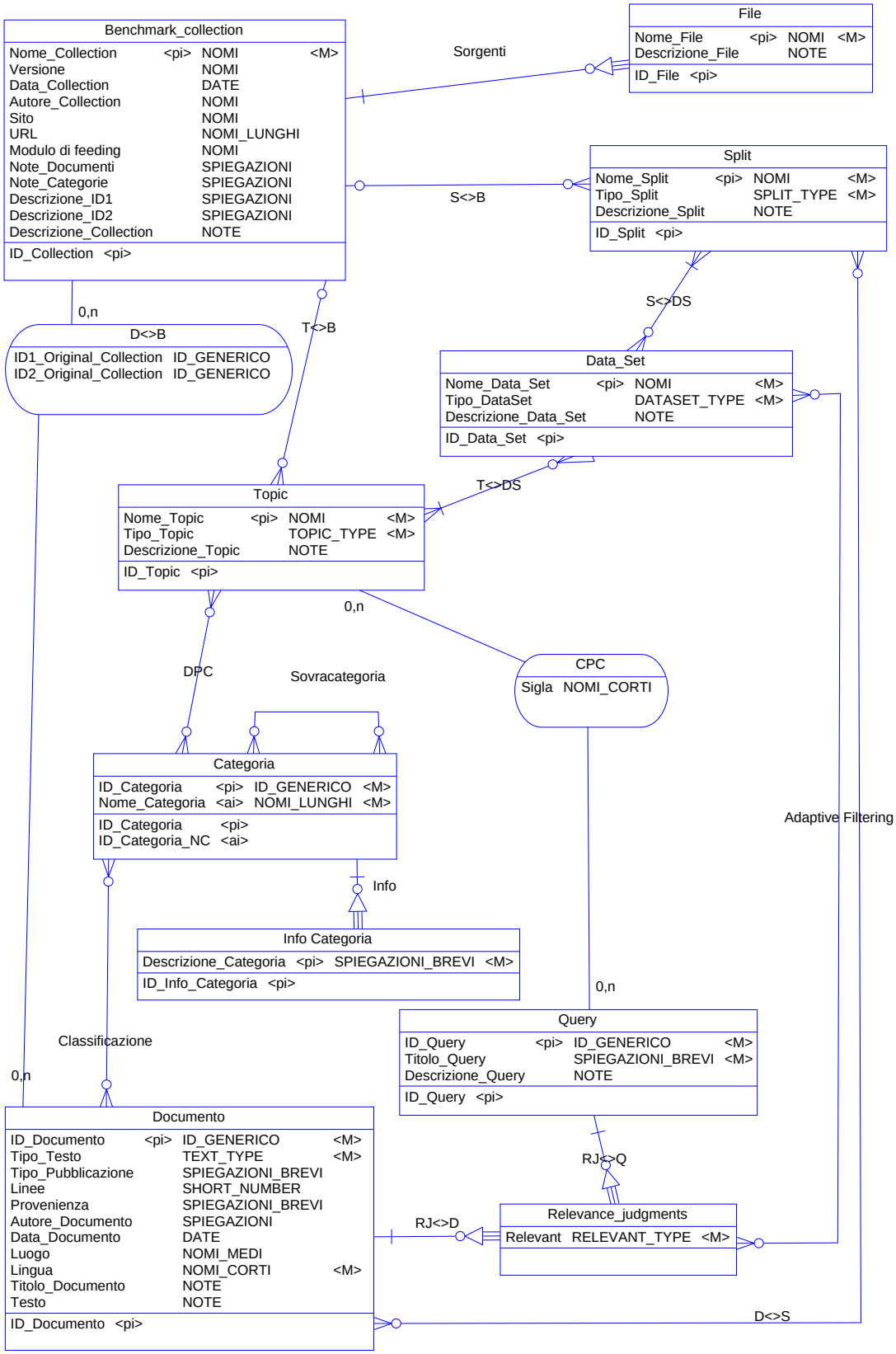


Figura 14: Schema Concettuale dei Benchmarks

6.2.1 Domini Utilizzati

I domini enumerativi utilizzano dati del tipo *Variable characters (15)*, tale decisione migliora la leggibilità delle informazioni a scapito dello spazio occupato, che per molti di tali domini, in ogni caso, è di poca rilevanza, ciò perché le entità che li adoperano hanno cardinalità molto basse:

DATASET_TYPE	Significato
DPC	Data Set adoperato per realizzare classificatori del tipo DPC
CPC	Data Set adoperato per realizzare classificatori del tipo CPC
ADAPTIVE	Il tipo Adaptive sta ad indicare un CPC che adopera per il training un sottoinsieme dei Relevance Judgments del Data Set (normalmente si considerano tutti quelli contenenti i Documenti dello Split di tipo tTraining-Test e le Query del Topic di tipo Query). Tale valore indica che l'insieme delle informazioni da utilizzare per il training è ricavabile tramite la relazione ***Adaptive Filtering***

SPLIT_TYPE	Significato
BYPASS	Tali valori sono inseriti per poter conservare le informazioni relative al vocabolario TOPICS del Reuters. Inoltre per mezzo di questi valori sarà possibile in futuro creare altri Split che permettano di distinguere sottoinsiemi di documenti in base alla presenza, assenza o non considerazione delle più svariate caratteristiche.
NO	
YES	
DATA SET	Permette di individuare uno Split che si riferisce alla totalità dei documenti di un Data Set, o che comunque non sia stato suddiviso in training o test set
TRAINING SET	Individua lo Split come insieme di documenti adoperati per un training set
TEST SET	Individua lo Split come insieme di documenti adoperati per un test set

TOPIC_TYPE	Significato
CATEGORIE	Indica che i Topic adoperati sono delle Categorie (tale valore è necessario per il corretto funzionamento del codice che sarà implementato).
QUERY	Indica che i Topic adoperati sono delle Query (tale valore è necessario per il corretto funzionamento del codice che sarà implementato).
KEYWORDS	Le Keywords sono viste come dei Topic, del resto sono molto prossime alle categorie.

TEXT_TYPE	Significato
BLAH	Tale dominio permette di implementare il campo TEXT TYPE, introdotto nel paragrafo 2.2 per definire alcune delle informazioni rese disponibili nei documenti Reuters. I significati dei valori sono NORM: normale BLAH: sciocchezze UNKNOWN: struttura sconosciuta UNPROC: struttura difficile da identificare BRIEF: breve
BRIEF	
NORM	
UNKNOWN	
UNPROC	

RELEVANT_TYPE	Significato
DEFINITELY	Indica che gli esperti del dominio hanno ritenuto sicuramente rilevante un particolare Documento rispetto ad una Query
POSSIBLY	Indica che gli esperti del dominio hanno ritenuto che un Documento sia con gran probabilità rilevante rispetto ad una Query
NOT RELEVANT	Il Documento non è rilevante rispetto la Query (tale valore non è adoperato mai, poiché è un'informazione ridondante dato che la mancanza della coppia Documento-Query nell'Entità **Relevant Judgments** già di per se ha questo significato)

Di seguito si riportano invece i domini i cui valori sono ben definiti:

Dominio	Tipo SQL	Vincoli
DATE	Date (contiene giorno mese e anno)	
ID_GENERICO	Long Integer	
SHORT_NUMBER	Short integer	≥0

Infine restano solo i Domini testuali che variano tra loro per il massimo numero di caratteri adoperabili:

Dominio	Tipo SQL
NOMI_CORTI	Variable characters (15)
NOMI	Variable characters (30)
NOMI_MEDI	Variable characters (60)
NOMI_LUNGHI	Variable characters (90)
SPIEGAZIONI_BREVI	Variable characters (160)
SPIEGAZIONI	Variable characters (255), dive 255 è il massimo numero di caratteri consentito in *Access 2000* per tale tipo di dati.
NOTE	Text

6.2.2 Uso del Prototipo per il dimensionamento

La mancanza d'informazioni riguardo alle dimensioni dei campi testuali, ha generato non poche difficoltà, specialmente a causa della gran mole di dati a disposizione. Un prototipo del sistema è stato realizzato adoperando per il database delle dimensioni indicative, e utilizzando un primo *modulo d'In/Out* poco ottimizzato. Adoperando il modulo realizzato si sono poi costruiti i diversi *moduli di feeding*, tramite i quali si sono potuti individuare non solo i punti deboli del database e del *modulo d'In/Out*, ma soprattutto le dimensioni dei vari attributi ed alcuni errori relativi alle formattazioni dei documenti.

Per effettuare uno studio sulle dimensioni dei domini utilizzati si è adoperato "*Test Dimensioni.exe*" il cui codice è presente nella directory "*Progettazione DB\Codice\Test*" o "*GestioneDS\Test Dimensioni"* (si veda il *paragrafo 1.6*). Tale codice adopera il database ed il *modulo d'In/Out* finali, una versione relativa ai prototipi è invece presente nella directory "*Progettazione DB\Prototipi\TestClassi*".

Il codice relativo a *Test Dimensioni* sarà descritto nel *paragrafo 8.4*, in tale sede si riportano solo i risultati ottenuti e memorizzati nel file "*Progettazione DB\Dimensioni da adoperare v2.0.txt*":

Dominio	Dimensioni Massime degli Attributi attinenti
Nomi_Corti [VARCHAR(15)]	DimMax_LINGUA: 7
Nomi [VARCHAR(30)]	DimMax_NOME_COLLECTION: 13 DimMax_VERSIONE: 16 DimMax_AUTORE_COLLECTION: 22 DimMax_SITO: 25 DimMax_NOME_FILE: 22 DimMax_MODULO_DI_FEEDING: 13 DimMax_NOME_SPLIT: 24 DimMax_NOME_TOPIC: 20 DimMax_NOME_DATA_SET: 28
Nomi_Medi [VARCHAR(60)]	DimMax_LUOGO: 45
Nomi_Lunghi [VARCHAR(90)]	DimMax_URL: 63 DimMax_NOME_CATEGORIA: 86
Spiegazioni_Brevi [VARCHAR(160)]	DimMax_PROVENIENZA: 116 DimMax_TITOLO_QUERY: 148 DimMax_TIPO_PUBBLICAZIONE 130 DimMax_DESCRIZIONE_CATEGORIA: 147
Spiegazioni [VARCHAR(255)]	DimMax_NOTE_DOCUMENTI: 36 DimMax_NOTE_CATEGORIE: 55 DimMax_DESCRIZIONE_ID1: 219 DimMax_DESCRIZIONE_ID2: 164 DimMax_AUTORE_DOCUMENTO: 223
Note [NOTE(oltre 255)]	DimMax_DESCRIZIONE_COLLECTION: 271 DimMax_DESCRIZIONE_FILE: 306 DimMax_TITOLO_DOCUMENTO: 402 DimMax_TESTO: 160499 DimMax_DESCRIZIONE_QUERY: 942 DimMax_DESCRIZIONE_SPLIT: 249 DimMax_DESCRIZIONE_TOPIC: 98 DimMax_DESCRIZIONE_DATA_SET: 2319

L’appartenenza di un attributo ad un particolare dominio è stata decisa in base alla cardinalità delle entità e della possibilità che si presentino in futuro casi in cui le dimensioni massime possano essere maggiori di quelle ricavate sperimentalmente.

6.2.3 Osservazioni e Vincoli aggiuntivi

6.2.3.1 Collection & File

L'entità contiene le informazioni relative ai benchmarks del database. Gli attributi utilizzati sono ricavati dalle informazioni presentate nei file "readme" delle collection.
Si noti che tal entità contiene l'attributo mulrivalore File, che permette di conservare le informazioni relative ai supporti da cui si sono prelevati i dati presenti nelle diverse collection. L'eliminazione di tale attributo multivalore è effettuata tramite l'usuale creazione di una nuova entità (File), il cui identificativo misto è definito per mezzo della ***Dependent Relationship Sorgenti***.
Gli ***attributi Descrizione_ID?*** permettono di spiegare il significato degli ID di un documento in relazione alla loro collection di origine (tali ID sono riportati nella relazione di afferenza D<>B).
L’**Associazione D<>B** permette di ricavare per mezzo di un’entità "Benchmark Collection" l'insieme di "Documenti" che si trovano al suo interno. L'utilizzo dell'attributo "ID1_Original_Collection" e "ID2_Original_Collection" si deve alla conservazione delle informazioni sulla numerazione dei documenti nella Collection d’origine. In realtà basterebbe riportare un solo ID poiché solo le Collection Reuters e OHSUMED ne utilizzano due, ed in entrambi i casi da uno dei due ID si può determinare univocamente l’altro. La decisione di riportare entrambi gli ID si deve alla necessità di evitare futuri utilizzi dei file dei benchmark per ricavare quello non incluso.
I valori posseduti dal MEDLINE identifier della collection OHSUMED richiedono che il dominio ID_Generico debba poter contenere almeno un numero pari a `92000000`.
Le **relazione T<>B** e **S<>B** associano, dove è possibile, il Topic o lo Split alla Collection da cui sono stati ricavati. Si noti che sia uno Split che un Topic possono non essere associati a nessuna Collection dal momento che un ricercatore è libero di definirne di nuovi e di associarvi un insieme di Documenti o Categorie/Query scelto tra quelli/e disponibili nel database.

6.2.3.2 Data Set

L’entità gestisce le informazioni dei Data Set, ma come sono trattati i legami con le altre entità del benchmark?

Si è lasciata completa libertà nella creazione delle relazioni annesse, anche se la norma richiederebbe associazioni con un Topic e due Split (uno per il training e l'altro per il test set) ed in alcuni casi l'inserzione di un'associazione con un Topic di tipo KEYWORDS. Sarà compito del realizzatore del Data Set fornire, nel campo addetto alla descrizione della tupla, le eventuali informazioni sull'utilizzo delle entità che si è deciso di collegarvi.
Definire dei vincoli riguardanti le possibili relazioni, avrebbe limitato il sistema, basti pensare all'attuale possibilità di realizzare un Data Set che non individui un training ed un test set ma un solo Split il cui valore dell'attributo Tipo_Split sia "*Data Set*", oppure alla possibilità di adoperare più Topic per uno stesso Data Set.

6.2.3.3 Topic

Per quanto detto nella descrizione del dominio TOPIC_TYPE è possibile inserire in tal entità anche la descrizione d'insiemi di keywords. I due insiemi identificati e non adoperati ufficialmente come classi sono: le "*Newsgroups Keywords*" per la collection 20 Newsgroups e i "*MeSH terms*" per quell'OHSUMED

6.2.3.4 Categoria

La ***dipendenza ricorsiva sovracategoria*** permette di rintracciare la "Categoria Padre" della tupla corrente e quindi di definire delle strutture gerarchiche. Si osservi che la "Classificazione" si riferisce sempre alla categoria più interna. Tale scelta si deve al fatto che in generale non si conoscono né il numero né il tipo di livelli delle gerarchie delle "Categorie" di un "Topic". Inoltre i Nodi di partenza non hanno alcun Padre.
L'attributo ***Info Categoria*** dell'entità Categoria è considerato a se stante non solo perché poche categorie hanno informazioni, ma anche perché diversi benchmark possono associare alla categoria un campo informazione con valore diverso. Inoltre in tale modo sarà possibile aggiungere ulteriori osservazioni su una particolare categoria riguardanti ad esempio feature particolarmente rilevanti.

6.2.3.5 Documento

Gli attributi Tipo_Pubblicazione e Autore_Documento sono stati implementati non considerando che per la collection OHSUMED siano multivalore. Tale scelta è stata presa per velocizzare il prelievo dei dati dei documenti, ciò a scapito delle ricerche riguardanti tali attributi. Del resto l'obiettivo del sistema è di velocizzare il più possibile il prelievo delle informazioni dei documenti da uno Split, in modo da limitare i tempi di latenza presenti nei software che realizzanno il calcolo delle feature.

6.3 Schema Concettuale degli Esperimenti

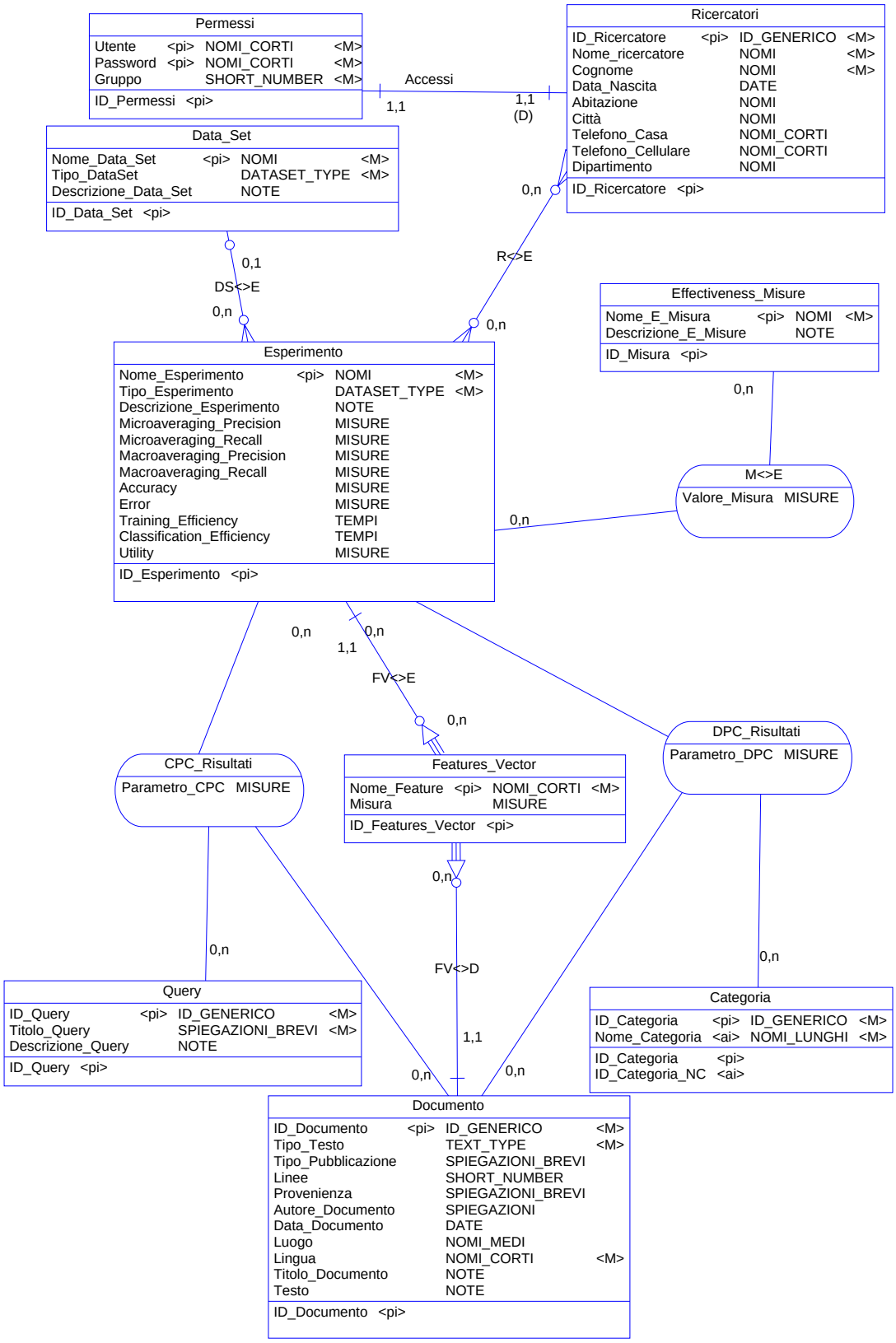


Figura 15: Schema Concettuale degli Esperimenti

6.3.1 Ulteriori Domini Utilizzati

Gli unici domini aggiunti nello schema sono quelli riportati di seguito

Dominio	Tipo SQL	Vincoli
MISURE	Short Float	≥0
TEMPI	Time (contiene giorno mese, anno, ora, minuti, secondi)	

L'assenza d'informazioni esistenti da inserire e la semplicità della struttura di tali entità, ha consentito una notevole libertà di scelta nel dimensionamento dei campi testuali, avvenuto semplicemente inquadrando i diversi attributi individuati in uno dei domini definiti nel paragrafo precedente. L'unica accortezza si è avuta per il campo *Nome_Esperimento* che, essendo l'unico attributo identificativo presente nell'entità Esperimenti, è stato subito individuato come probabile chiave primaria e dunque definito il più piccolo possibile per facilitare i numerosi join cui sarà sottoposto.

6.3.2 Osservazioni e Vincoli aggiuntivi

L'attributo ***"Tipo_Esperimento"***, dell'entità Esperimento, è un dato ridondante e non fa altro che riportare il tipo del Data Set utilizzato per l'esperimento.
L'attributo "***Parametro_?***" delle entità CPC_Risultati e DPC_Risultati permette di restituire il risultato del filtraggio o della classificazione anche nel caso non sia binario, o un qualsiasi altro dato numerico d'interesse.
L'entità **Effectiveness_Misure** permette di aggiungere ad un esperimento una misurazione alternativa della sua efficacia. La motivazione della sua presenza è che diversi gruppi di ricerca e diversi tipi di Learning Machine possono usare tecniche di misurazione molto diverse e non tutte prevedibili, quindi a priori non si è in grado di conoscere tutte le possibili misurazioni necessarie.
L'entità **Features_Vector** permette di memorizzare le *Features* associate ad ogni *Documento* adoperato da un particolare *Esperimento*. Come per le Effectiveness Misure, non è stato possibile definire un insieme di Features a priori, quindi l'entità ha come istanza un singolo elemento del vettore e non l'intero vettore. Le "*Dependent Relationship*" FV<>E e FV<>D definiscono le relazioni esistenti tra l'entità "Features Vector" e quella "Esperimento" e "Documento", inoltre permettono di rappresentare l'identificativo misto definito per i "Features Vector" e composto dall'identificativo dell'Esperimento, del Documento e della Feature.

6.3.3 Gestione dei Permessi

Gli attributi di un Ricercatore vengono divisi in due entità in modo da poter gestire il vincolo, descritto nel *paragrafo 5.5.1*, secondo cui alcuni di questi sono visibili a tutti i *Ricercatori* ed altri solo ai *DBA*.

Tal entità permette di identificare se un utente può avere accesso al database ed è implementata in modo da aumentare la portabilità, poiché l'utilizzo dei comandi SQL relativi al **CREATE USER** è troppo vincolato al particolare DBMS (e in ogni modo sembra non sia possibile utilizzare tali comandi riferendosi ad un DBMS Access 2000 gestito tramite ODBC).

Inoltre si ricorda che. in mancanza dei comandi SQL relativi al **GRANT** (anche questi non gestibili tramite ODBC quando il DBMS adoperato è Access 2000), sarà necessario gestire i permessi relativi a singole righe implementando tali vincoli via codice.

All'interno del database saranno implementati dunque solo 2 utenti, uno di Sistema (a cui faranno riferimento i *DBA*) ed uno di riferimento per i Ricercatori (a cui faranno riferimento i *Ricercatori XXX*). Il *modulo d'In/Out* si dovrà preoccupare di controllare se l'*User* e la *Password* inserite appartengono ad un istanza dell'entità Permessi e poi di aprire l'utente del database relativo al particolare *GRUPPO* (i cui valori possono essere 1 per indicare che l'utente è un **DBA** o 2 per indicare che è un **RicercatoreXXX)** relativo all'utente. Per quanto riguarda i vincoli di riga il *modulo* dovrà verificare tramite la ***relazione R<>E*** quali siano gli Esperimenti modificabili dall'utente attuale.

Una descrizione più accurata della gestione dei permessi sarà presentata dopo la definizione dello schema logico.

6.4 Schema Logico dei Benchmarks

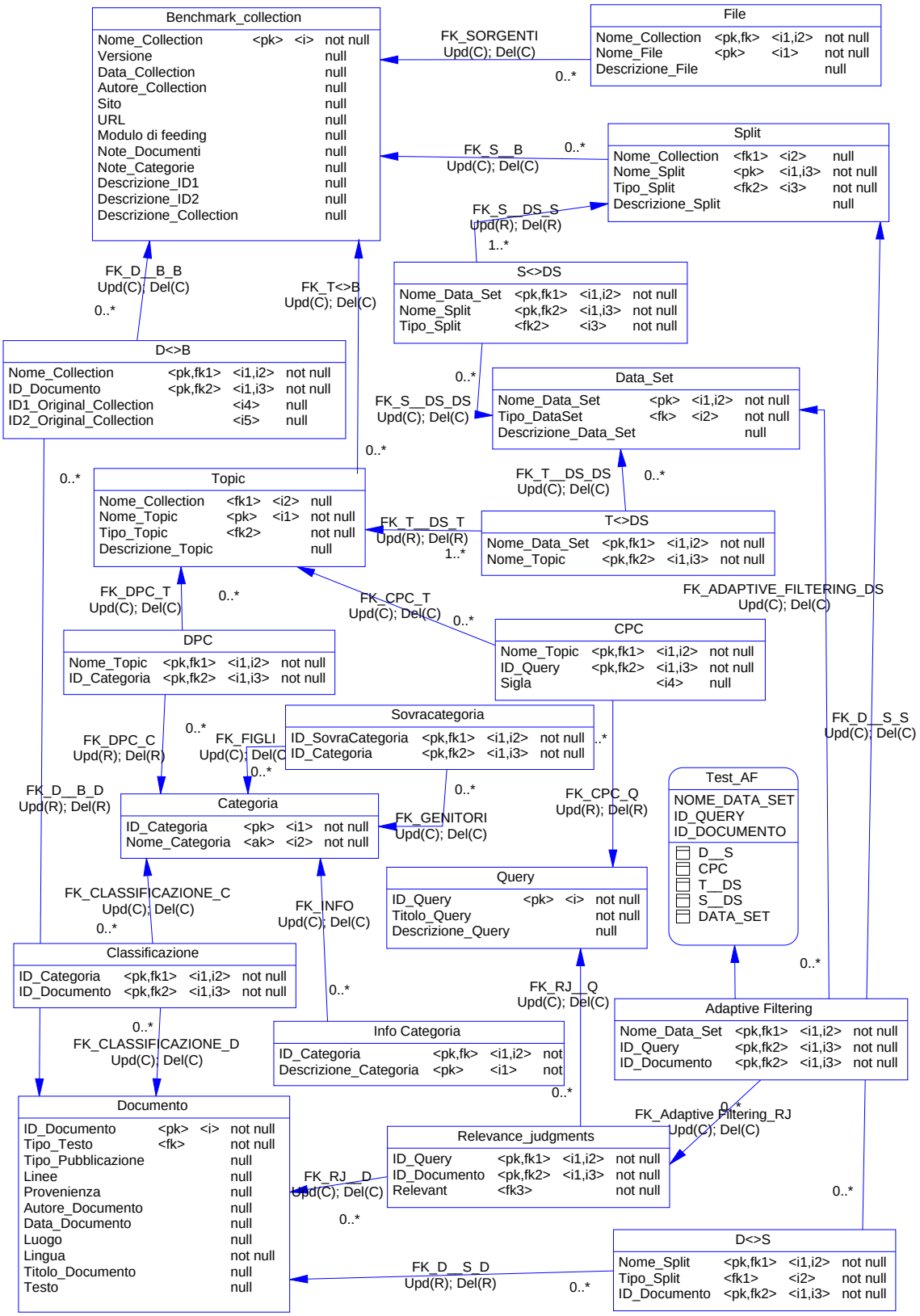


Figura 16: Schema Logico dei Benchmarks

6.4.1 Domini Aggiuntivi

ID_COUNTER è l'unico dominio aggiunto negli schemi logici, la gestione dell'identificativo in tale caso si lascia al DBMS. Si osservi che tale dominio sarà adoperato solo nelle entità e non nelle relazioni, dove naturalmente gli identificativi devono essere ID_GENERICI che tramite vincoli referenziali si riferiscono ai domini ID_COUNTER delle entità di riferimento.

Dominio	Tipo SQL	Vincoli
ID_COUNTER	COUNTER	Gestito dal DBMS

6.4.2 Vincoli aggiuntivi

Una singola riga della ***tabella Sovracategoria*** deve avere i due campi distinti, giacché non ha senso affermare che una Categoria è Sovracategoria di se stessa.

Nella ***tabella D<>S*** si aggiunge il campo *Tipo_Split* per semplificare le ricerche ed avere informazioni più dettagliate.

L'inserimento di un'istanza in ***Adaptive Filtering*** è valido solo se si rispetta la relazione con la vista ***Test_AF:***

```
select T__DS.NOME_DATA_SET, CPC.ID_QUERY, D__S.ID_DOCUMENTO
from D__S, CPC, T__DS, S__DS, DATA_SET
where T__DS.NOME_TOPIC = CPC.NOME_TOPIC
AND S__DS.NOME_SPLIT = D__S.NOME_SPLIT
AND S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET
AND T__DS.NOME_DATA_SET = DATA_SET.NOME_DATA_SET
AND DATA_SET.TIPO_DATASET = 'ADAPTIVE'
```

6.5 Schema Logico degli Esperimenti

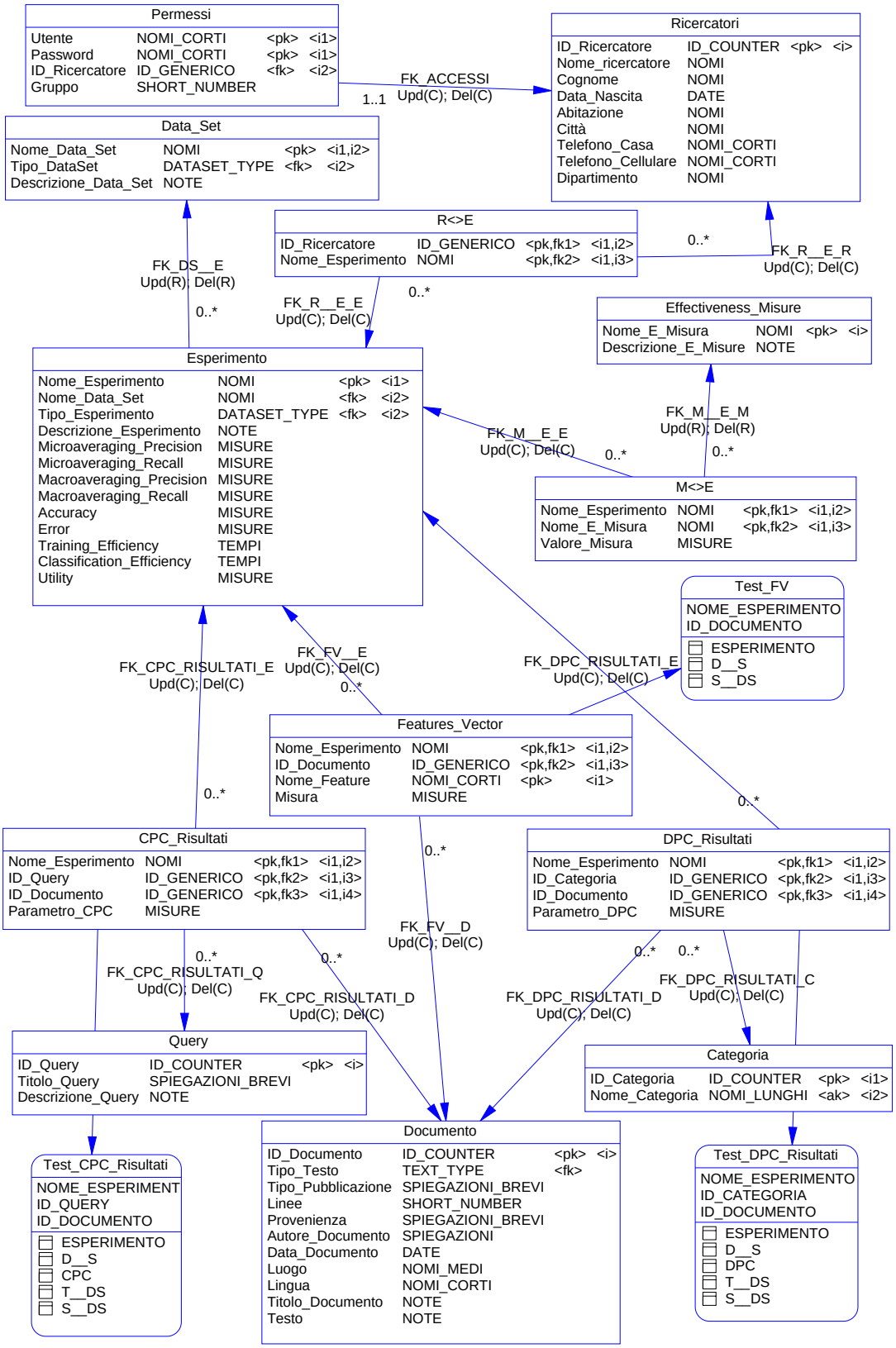


Figura 17: Schema Logico degli Esperimenti

6.5.1 Vincoli aggiuntivi

L'inserimento di un'istanza in ***Features_Vector*** è valido solo se si rispetta la relazione con la vista ***Test_FV:***

```
select ESPERIMENTO.NOME_ESPERIMENTO, D__S.ID_DOCUMENTO
from ESPERIMENTO, D__S, S__DS
where S__DS.NOME_SPLIT = D__S.NOME_SPLIT
  AND S__DS.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET
```

L'inserimento di un'istanza in ***CPC_Risultati*** è valido solo se si rispetta la relazione con la vista ***Test_CPC_Risultati:***

```
select ESPERIMENTO.NOME_ESPERIMENTO, CPC.ID_QUERY, D__S.ID_DOCUMENTO
from ESPERIMENTO, D__S, CPC, T__DS, S__DS
where T__DS.NOME_TOPIC = CPC.NOME_TOPIC
  AND S__DS.NOME_SPLIT = D__S.NOME_SPLIT
  AND S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET
  AND ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET
```

L'inserimento di un'istanza in ***DPC_Risultati*** è valido solo se si rispetta la relazione con la vista ***Test_DPC_Risultati:***

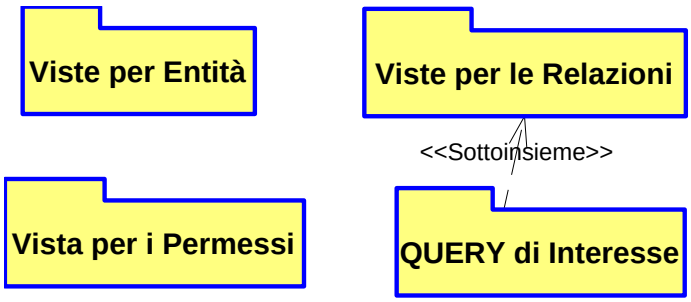
```
select ESPERIMENTO.NOME_ESPERIMENTO, DPC.ID_CATEGORIA, D__S.ID_DOCUMENTO
from ESPERIMENTO, D__S, DPC, T__DS, S__DS
where T__DS.NOME_TOPIC = DPC.NOME_TOPIC
  AND S__DS.NOME_SPLIT = D__S.NOME_SPLIT
  AND S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET
  AND ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET
```

6.6 Viste

Una volta creata la struttura del database si deve rendere visibile esternamente dando però la libertà di modificare in futuro le tabelle implementate. Tale risultato si ottiene creando, per tutte le tabelle realizzate, delle viste da far adoperare al *modulo d'In/Out*. L'utilizzo delle viste fa sì che il codice sia duraturo, poiché, anche se in futuro si dovessero modificare una o più tabelle, basterebbe modificare di conseguenza le viste che le adoperano in modo da ottenere in uscita lo stesso risultato che si aveva prima delle modifiche.

Oltre alle viste delle tabelle sarà necessario crearne altre per realizzare i filtraggi esterni esposti nel capitolo precedente. Di seguito si riportano gli insiemi organizzativi utilizzati per raggruppare le viste realizzate. L'insieme "*QUERY d'interesse*" è un

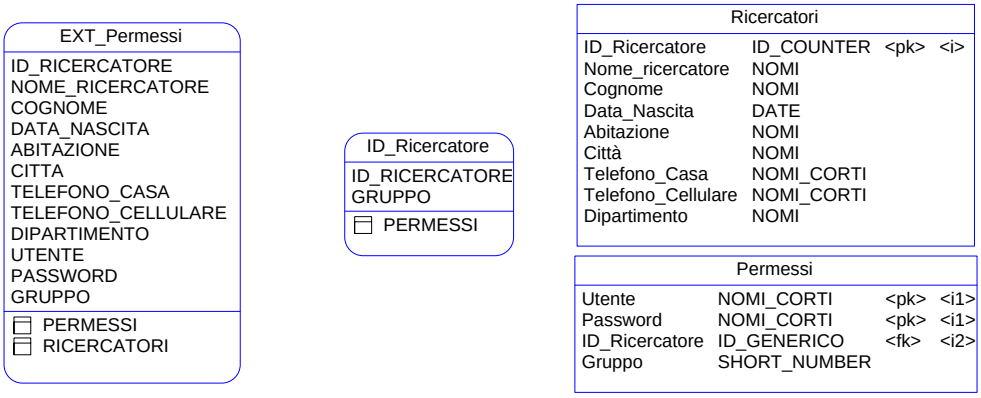
sottoinsieme delle viste adoperate per le relazioni, giacché si riferisce ad alcune ricerche effettuate sulle relazioni *Classificazione* e *Relevance Judgments*.



I nomi delle viste riportate di seguito saranno caratterizzati da sigle iniziali dove con **EXT** s'indicherà che la vista si riferisce ad una tabella, **FE** che la vista è utilizzata dal *modulo d'In/Out* per effettuare un filtraggio esterno, **Test** (già introdotto in precedenza) che è utilizzata per verificare un vincolo d'esistenza dipendente da altre entità. Si osservi che a volte le viste EXT sono adoperate per i filtraggi esterni.

I seguenti sottoparagrafi riportano la struttura delle viste e, dove necessario, anche il relativo comando richiesto per realizzarle.

6.6.1 Viste per i Permessi



La vista ID_Ricercatore è parametrizzata:

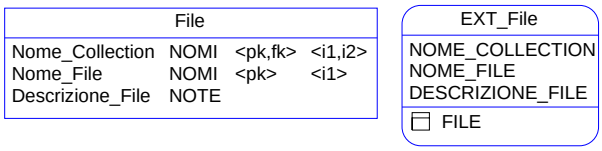
```
select PERMESSI.ID_RICERCATORE, PERMESSI.GRUPPO
from PERMESSI
where PERMESSI.UTENTE = [USER] AND PERMESSI.PASSWORD = [PASS]
```

Permette di ricavare l'identificativo ed il gruppo di un particolare Utente tramite il suo User e la sua Password. Il *Gruppo* sarà utilizzato per individuare l'utente del DBMS necessario per aprire la connessione con i permessi ***DBA*** o ***Ricercatore XXX***; mentre l'*ID_Ricercatore* sarà utilizzato all'occorrenza per ricercare gli Esperimenti appartenenti all'utente.

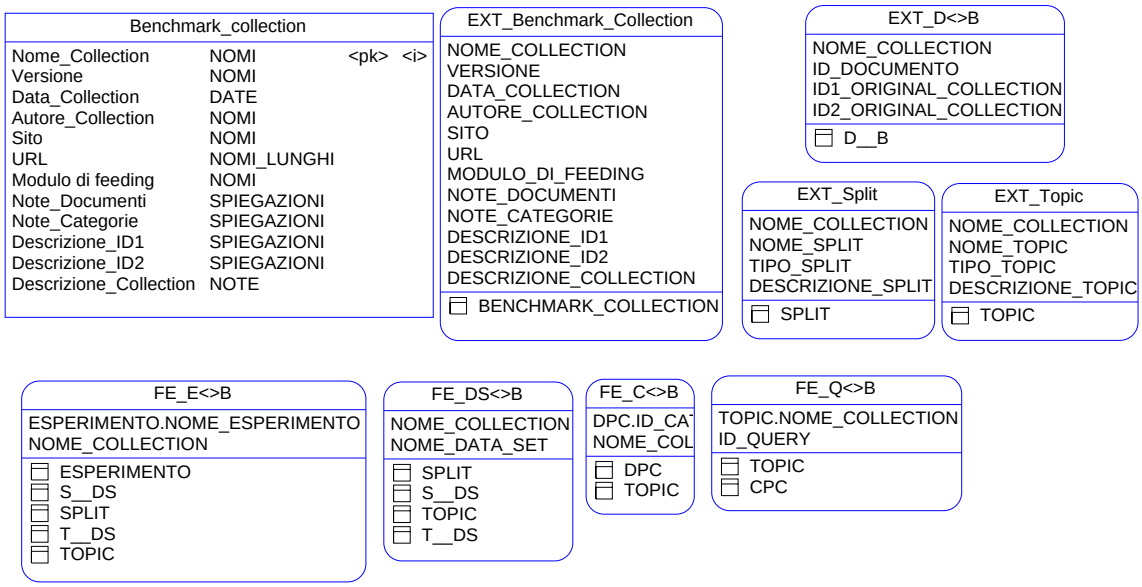
6.6.2 Viste per le Entità

Per ogni entità si riporta la tabella di riferimento e le viste adoperate per i filtraggi esterni. La vista *EXT_nome-tabella* è quella adoperata dal *modulo* per la gestione dell'entità mentre le altre sono adoperate per i filtraggi.

- **File**



- **Collection**



(FE_E<>B)
select DISTINCT ESPERIMENTO.NOME_ESPERIMENTO, SPLIT.NOME_COLLECTION AS NOME_COLLECTION
from (ESPERIMENTO inner join S__DS on ESPERIMENTO.NOME_DATA_SET = S__DS.NOME_DATA_SET) inner join SPLIT on SPLIT.NOME_SPLIT = S__DS.NOME_SPLIT
where SPLIT.NOME_COLLECTION is not null

UNION

select DISTINCT ESPERIMENTO.NOME_ESPERIMENTO, TOPIC.NOME_COLLECTION AS NOME_COLLECTION
from (ESPERIMENTO inner join T__DS on ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET) inner join TOPIC on TOPIC.NOME_TOPIC = T__DS.NOME_TOPIC
where TOPIC.NOME_COLLECTION is not null

(FE_DS<>B)
SELECT DISTINCT SPLIT.NOME_COLLECTION AS NOME_COLLECTION, S__DS.NOME_DATA_SET AS NOME_DATA_SET
FROM SPLIT inner join S__DS on SPLIT.NOME_SPLIT = S__DS.NOME_SPLIT
where NOME_COLLECTION is not null

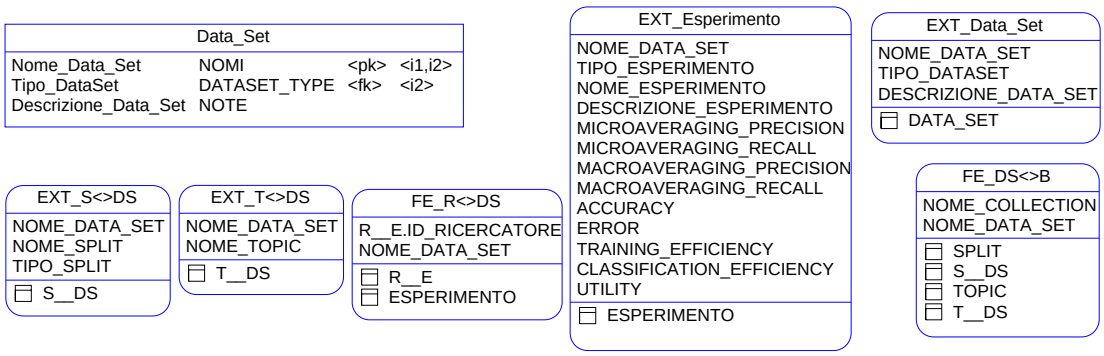
UNION

SELECT DISTINCT TOPIC.NOME_COLLECTION AS NOME_COLLECTION, T__DS.NOME_DATA_SET AS NOME_DATA_SET
FROM TOPIC inner join T__DS on TOPIC.NOME_TOPIC = T__DS.NOME_TOPIC
where NOME_COLLECTION is not null

(FE_C<>B)
select DISTINCT DPC.ID_CATEGORIA, TOPIC.NOME_COLLECTION
from DPC inner join TOPIC on TOPIC.NOME_TOPIC = DPC.NOME_TOPIC
where TOPIC.NOME_COLLECTION is not null

(FE_Q<>B)
select DISTINCT TOPIC.NOME_COLLECTION, CPC.ID_QUERY
from TOPIC inner join CPC on TOPIC.NOME_TOPIC = CPC.NOME_TOPIC
where TOPIC.NOME_COLLECTION is not null

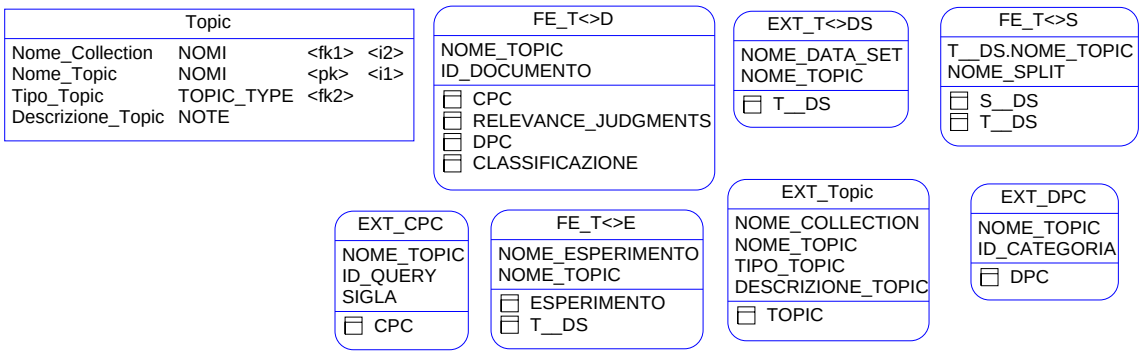
- # Data Set



(FE_R<>DS)
select DISTINCT R__E.ID_RICERCATORE, ESPERIMENTO.NOME_DATA_SET
from R__E inner join ESPERIMENTO on R__E.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO

(FE_DS<>B) *vedi Collection*

- # Topic



(FE_T<>S)
select DISTINCT T__DS.NOME_TOPIC, S__DS.NOME_SPLIT
from S__DS inner join T__DS on S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET

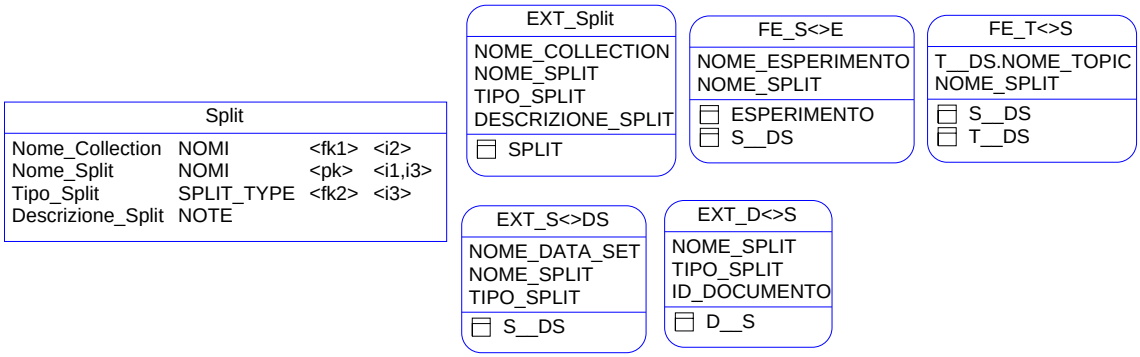
(FE_T<>D)
select DISTINCT NOME_TOPIC, ID_DOCUMENTO
from CPC inner join RELEVANCE_JUDGMENTS on CPC.ID_QUERY = RELEVANCE_JUDGMENTS.ID_QUERY

UNION

select DISTINCT NOME_TOPIC, ID_DOCUMENTO
from DPC inner join CLASSIFICAZIONE on DPC.ID_Categoria = CLASSIFICAZIONE.ID_Categoria

(FE_T<>E)
select ESPERIMENTO.NOME_ESPERIMENTO, T__DS.NOME_TOPIC
from ESPERIMENTO inner join T__DS on ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET

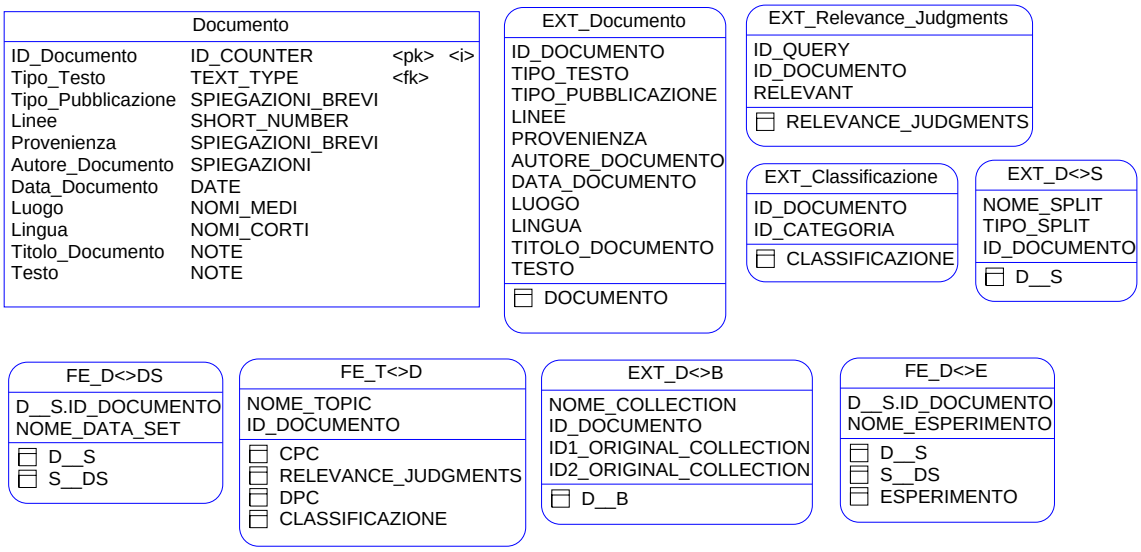
- **Split**



(FE_T<>S) *vedi Topic*

(FE_S<>E)
select ESPERIMENTO.NOME_ESPERIMENTO, S__DS.NOME_SPLIT
from ESPERIMENTO inner join S__DS on ESPERIMENTO.NOME_DATA_SET = S__DS.NOME_DATA_SET

- **Documento**



(FE_T<>D) *vedi Topic*

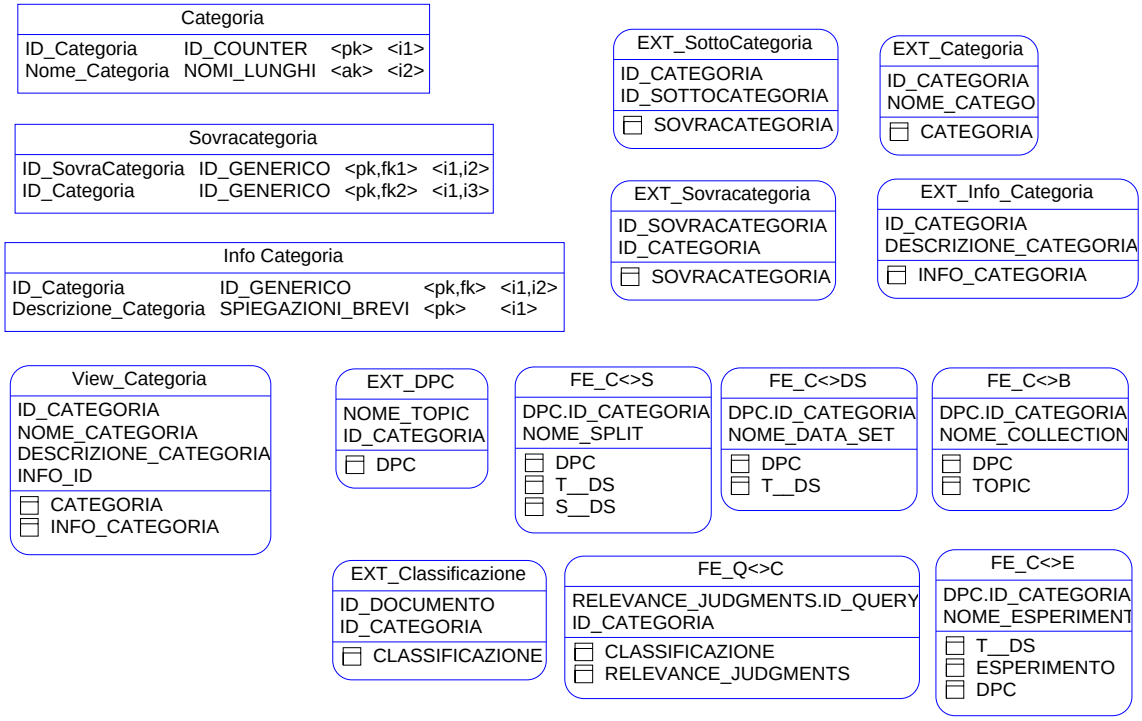
(FE_D<>DS)
select DISTINCT D__S.ID_DOCUMENTO, S__DS.NOME_DATA_SET
from D__S inner join S__DS on D__S.NOME_SPLIT = S__DS.NOME_SPLIT

(FE_D<>E)
select DISTINCT D__S.ID_DOCUMENTO, ESPERIMENTO.NOME_ESPERIMENTO
from (D__S inner join S__DS on D__S.NOME_SPLIT = S__DS.NOME_SPLIT)
inner join ESPERIMENTO on ESPERIMENTO.NOME_DATA_SET = S__DS.NOME_DATA_SET

- **Categorie**

(View_Categoria)
SELECT CATEGORIA.ID_CATEGORIA, CATEGORIA.NOME_CATEGORIA,
INFO_CATEGORIA.DESCRIZIONE_CATEGORIA, INFO_CATEGORIA.ID_CATEGORIA AS INFO_ID
FROM CATEGORIA LEFT JOIN INFO_CATEGORIA
ON CATEGORIA.ID_CATEGORIA = INFO_CATEGORIA.ID_CATEGORIA;

(EXT_SottoCategoria)
```
select SOVRACATEGORIA.ID_SOVRACATEGORIA AS ID_CATEGORIA,
       SOVRACATEGORIA.ID_CATEGORIA AS ID_SOTTOCATEGORIA
from SOVRACATEGORIA
```



(FE_C<>S)
```
select DISTINCT DPC.ID_CATEGORIA, S__DS.NOME_SPLIT
from (DPC inner join T__DS on DPC.NOME_TOPIC = T__DS.NOME_TOPIC)
    inner join S__DS on S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET
```

(FE_C<>DS)
```
select DISTINCT DPC.ID_CATEGORIA, T__DS.NOME_DATA_SET
from DPC inner join T__DS on DPC.NOME_TOPIC = T__DS.NOME_TOPIC
```

(FE_C<>B) *vedi Collection*

(FE_Q<>C)
```
select DISTINCT RELEVANCE_JUDGMENTS.ID_QUERY, CLASSIFICAZIONE.ID_CATEGORIA
from CLASSIFICAZIONE inner join RELEVANCE_JUDGMENTS
      on CLASSIFICAZIONE.ID_DOCUMENTO = RELEVANCE_JUDGMENTS.ID_DOCUMENTO
```

(FE_C<>E)
```
select DISTINCT DPC.ID_CATEGORIA, ESPERIMENTO.NOME_ESPERIMENTO
from (T__DS inner join ESPERIMENTO on T__DS.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET)
      inner join DPC on DPC.NOME_TOPIC = T__DS.NOME_TOPIC
```

- **Query**

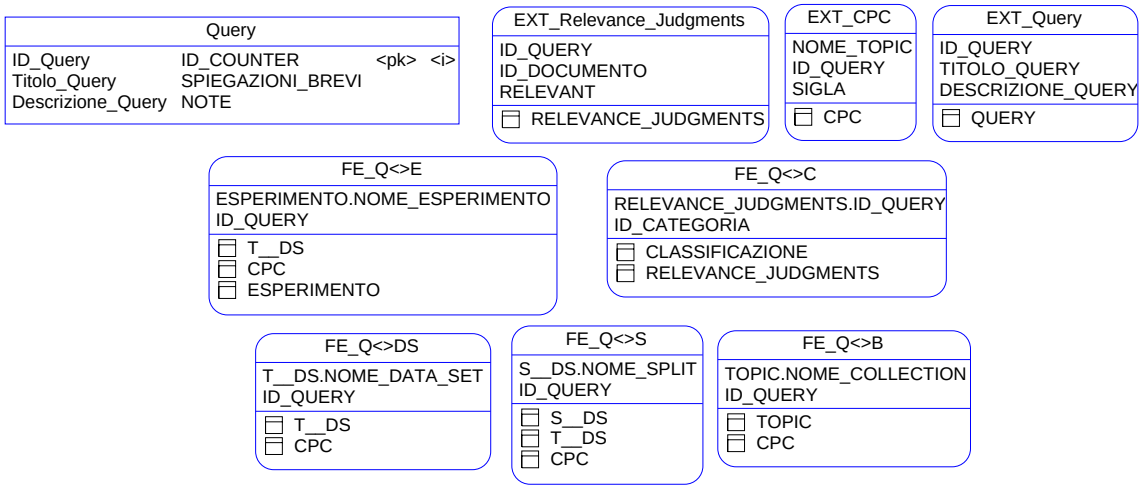
(FE_Q<>E)
```
select DISTINCT ESPERIMENTO.NOME_ESPERIMENTO, CPC.ID_QUERY
from (T__DS inner join CPC on T__DS.NOME_TOPIC = CPC.NOME_TOPIC)
    inner join ESPERIMENTO on ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET
```

(FE_Q<>C) *vedi Categorie*

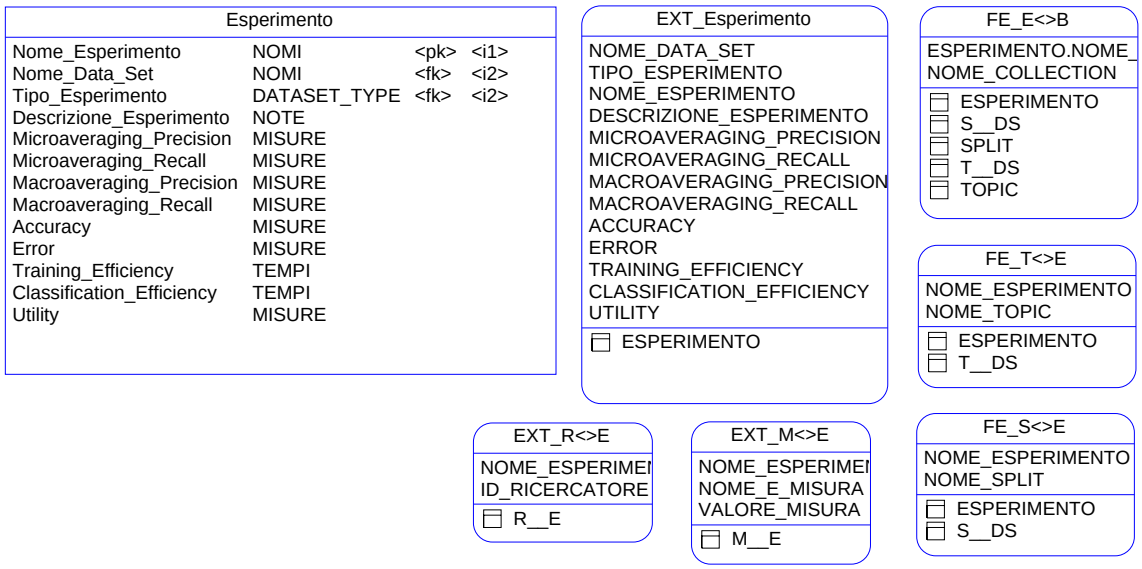
(FE_Q<>DS)
```
select DISTINCT T__DS.NOME_DATA_SET, CPC.ID_QUERY
from T__DS inner join CPC on T__DS.NOME_TOPIC = CPC.NOME_TOPIC
```

(FE_Q<>S)
select DISTINCT S__DS.NOME_SPLIT, CPC.ID_QUERY
from (S__DS inner join T__DS on S__DS.NOME_DATA_SET = T__DS.NOME_DATA_SET)
inner join CPC on T__DS.NOME_TOPIC = CPC.NOME_TOPIC

(FE_Q<>B) *vedi Collection*



- **Esperimento**

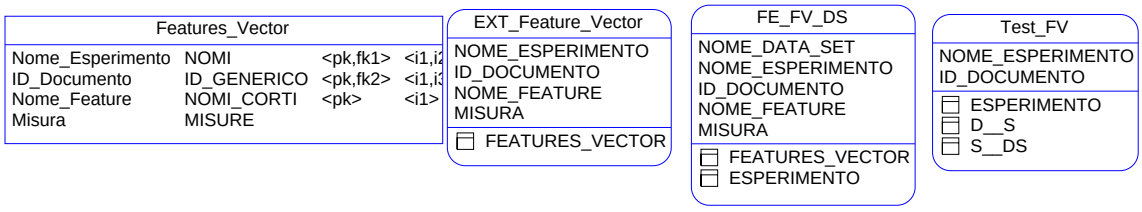


(FE_E<>B) *vedi Collection*

(FE_T<>E) *vedi Topic*

(FE_S<>E) *vedi Split*

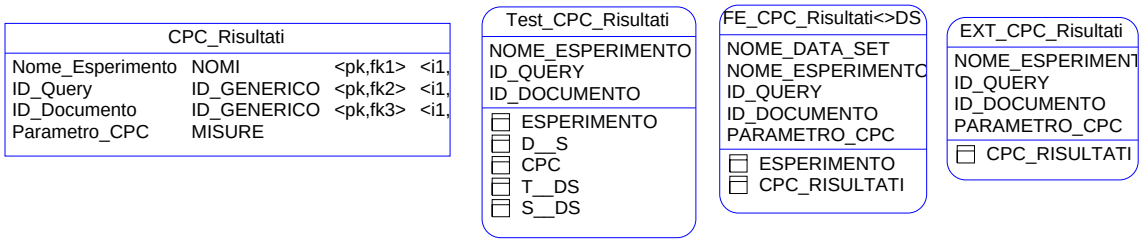
- **Features Vector**



(FE_FV_DS)
select ESPERIMENTO.NOME_DATA_SET, FEATURES_VECTOR.NOME_ESPERIMENTO,
FEATURES_VECTOR.ID_DOCUMENTO, FEATURES_VECTOR.NOME_FEATURE, FEATURES_VECTOR.MISURA
from FEATURES_VECTOR inner join ESPERIMENTO
on FEATURES_VECTOR.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO

(Test_FV) *vedi paragrafo 6.5.1*

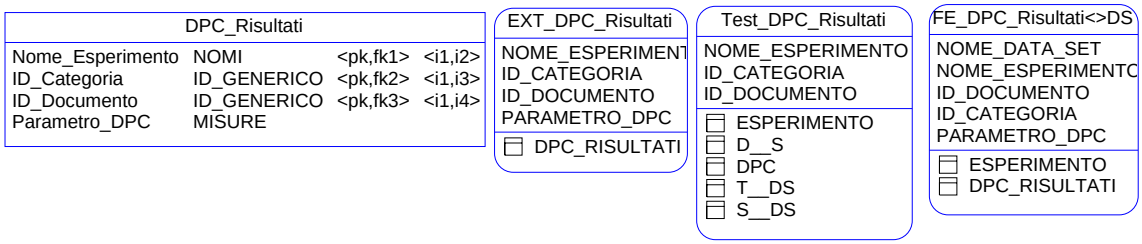
- **CPC_Risultati**



(FE_CPC_Risultati<>DS)
SELECT ESPERIMENTO.NOME_DATA_SET, CPC_RISULTATI.NOME_ESPERIMENTO, CPC_RISULTATI.ID_QUERY,
CPC_RISULTATI.ID_DOCUMENTO, CPC_RISULTATI.PARAMETRO_CPC
FROM ESPERIMENTO INNER JOIN CPC_RISULTATI
ON CPC_RISULTATI.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO

(Test_CPC_Risultati) *vedi paragrafo 6.5.1*

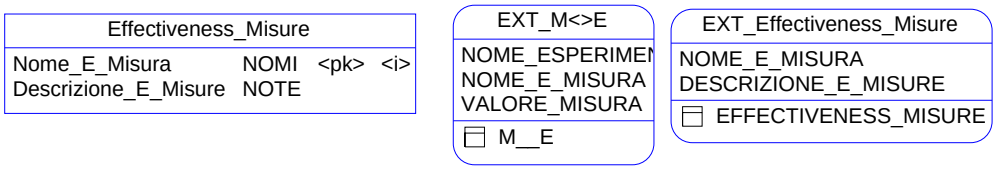
- **DPC_Risultati**



(FE_DPC_Risultati<>DS)
select ESPERIMENTO.NOME_DATA_SET, DPC_RISULTATI.NOME_ESPERIMENTO, DPC_RISULTATI.ID_DOCUMENTO,
DPC_RISULTATI.ID_CATEGORIA, DPC_RISULTATI.PARAMETRO_DPC
from ESPERIMENTO inner join DPC_RISULTATI
on DPC_RISULTATI.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO

(Test_DPC_Risultati) *vedi paragrafo 6.5.1*

- **Effectiveness Misure**



- **Ricercatori**

(FE_R<>B)
select DISTINCT R__E.ID_RICERCATORE, FE_DS__B.NOME_COLLECTION
from (R__E inner join ESPERIMENTO on R__E.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO)
INNER JOIN FE_DS__B on FE_DS__B.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET

(FE_R<>T)
select DISTINCT R__E.ID_RICERCATORE, T__DS.NOME_TOPIC
from (R__E inner join ESPERIMENTO on R__E.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO)
INNER JOIN T__DS on T__DS.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET

(FE_R<>S)
select DISTINCT R__E.ID_RICERCATORE, S__DS.NOME_SPLIT
from (R__E inner join ESPERIMENTO on R__E.NOME_ESPERIMENTO = ESPERIMENTO.NOME_ESPERIMENTO)
INNER JOIN S__DS on S__DS.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET

(FE_R<>DS) *vedi Data Set*

Ricercatori		
ID_Ricercatore	ID_COUNTER	<pk> <i>
Nome_ricercatore	NOMI	
Cognome	NOMI	
Data_Nascita	DATE	
Abitazione	NOMI	
Città	NOMI	
Telefono_Casa	NOMI_CORTI	
Telefono_Cellulare	NOMI_CORTI	
Dipartimento	NOMI	

EXT_Ricercatori
ID_RICERCATORE
NOME_RICERCATORE
COGNOME
DATA_NASCITA
ABITAZIONE
CITTA
TELEFONO_CASA
TELEFONO_CELLULARE
DIPARTIMENTO
RICERCATORI

EXT_R<>E
NOME_ESPERIMENTO
ID_RICERCATORE
R__E

FE_R<>B
R__E.ID_RICERCATORE
NOME_COLLECTION
R__E
ESPERIMENTO
FE_DS__B

FE_R<>T
R__E.ID_RICERCATORE
NOME_TOPIC
R__E
ESPERIMENTO
T__DS

FE_R<>S
R__E.ID_RICERCATORE
NOME_SPLIT
R__E
ESPERIMENTO
S__DS

FE_R<>DS
R__E.ID_RICERCATORE
NOME_DATA_SET
R__E
ESPERIMENTO

6.6.3 Viste per le Relazioni

- **Adaptive Filtering**

Test_AF
NOME_DATA_SET
ID_QUERY
ID_DOCUMENTO
D__S
CPC
T__DS
S__DS
DATA_SET

EXT_Adaptive Filtering
NOME_DATA_SET
ID_QUERY
ID_DOCUMENTO
ADAPTIVE_FILTERING

Adaptive Filtering	
Nome_Data_Set	VARCHAR(30)
ID_Query	LONG
ID_Documento	LONG

- **CPC**

CPC			
Nome_Topic	NOMI	<pk,fk1>	<i1,i2>
ID_Query	ID_GENERICO	<pk,fk2>	<i1,i3>
Sigla	NOMI_CORTI		<i4>

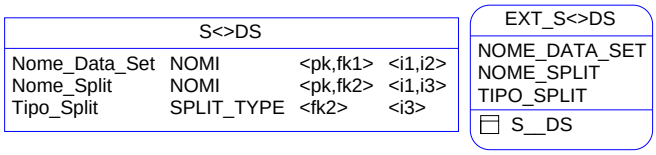
EXT_CPC
NOME_TOPIC
ID_QUERY
SIGLA
CPC

- **DPC**

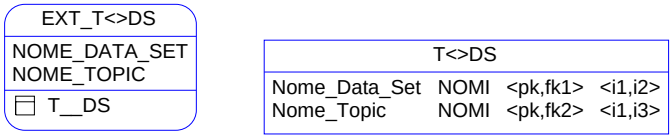
EXT_DPC
NOME_TOPIC
ID_CATEGORIA
DPC

DPC			
Nome_Topic	NOMI	<pk,fk1>	<i1,i2>
ID_Categoria	ID_GENERICO	<pk,fk2>	<i1,i3>

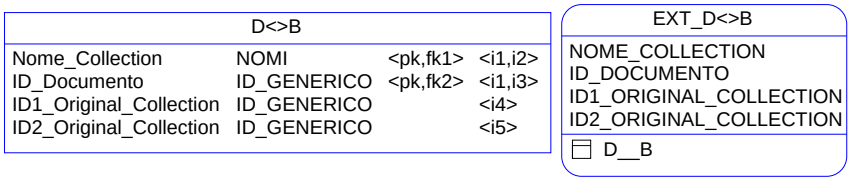
- **S<>DS**



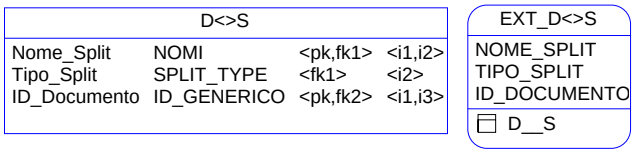
- **T<>DS**



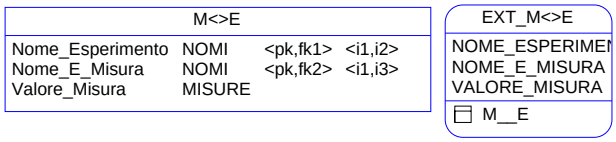
- **D<>B**



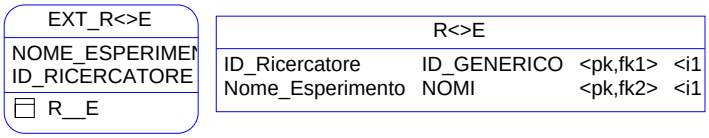
- **D<>S**



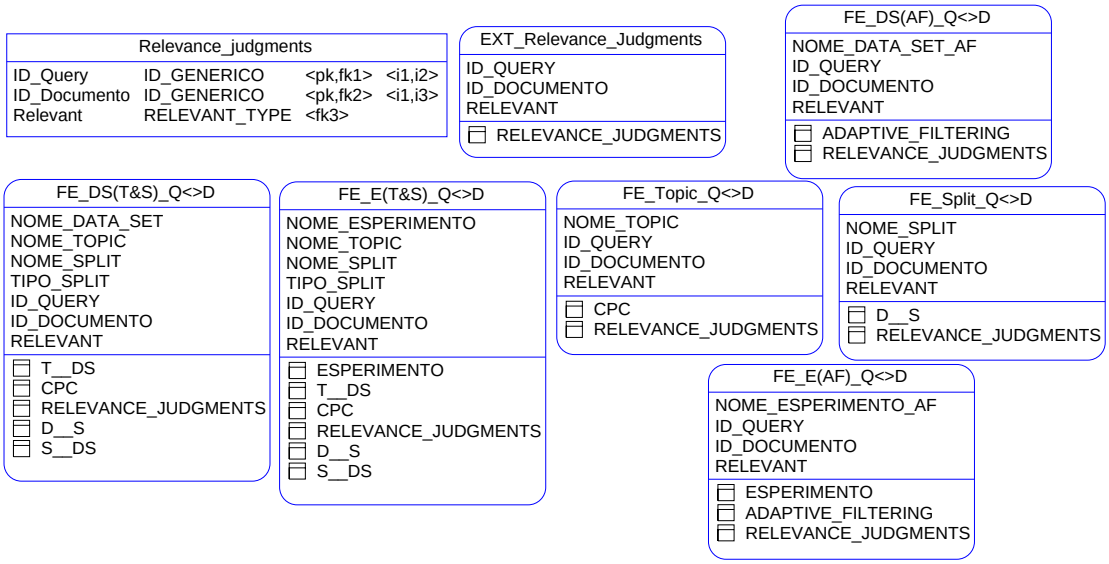
- **M<>E**



- **R<>E**



- **Relevance Judgments**



(FE_Split_Q<>D)

```
select D__S.NOME_SPLIT, RELEVANCE_JUDGMENTS.ID_QUERY, RELEVANCE_JUDGMENTS.ID_DOCUMENTO,
RELEVANCE_JUDGMENTS.RELEVANT
from D__S inner join RELEVANCE_JUDGMENTS
      on RELEVANCE_JUDGMENTS.ID_DOCUMENTO = D__S.ID_DOCUMENTO
```

(FE_Topic_Q<>D)

```
select CPC.NOME_TOPIC, RELEVANCE_JUDGMENTS.ID_QUERY, RELEVANCE_JUDGMENTS.ID_DOCUMENTO,
RELEVANCE_JUDGMENTS.RELEVANT
from CPC inner join RELEVANCE_JUDGMENTS on RELEVANCE_JUDGMENTS.ID_QUERY = CPC.ID_QUERY
```

(FE_DS(AF)_Q<>D)

```
select ADAPTIVE_FILTERING.NOME_DATA_SET AS NOME_DATA_SET_AF, RELEVANCE_JUDGMENTS.ID_QUERY,
RELEVANCE_JUDGMENTS.ID_DOCUMENTO, RELEVANCE_JUDGMENTS.RELEVANT
from ADAPTIVE_FILTERING, RELEVANCE_JUDGMENTS
where ADAPTIVE_FILTERING.ID_QUERY = RELEVANCE_JUDGMENTS.ID_QUERY
    and ADAPTIVE_FILTERING.ID_DOCUMENTO = RELEVANCE_JUDGMENTS.ID_DOCUMENTO
```

(FE_DS(T&S)_Q<>D)

```
SELECT T__DS.NOME_DATA_SET, CPC.NOME_TOPIC, D__S.NOME_SPLIT, D__S.TIPO_SPLIT,
RELEVANCE_JUDGMENTS.ID_QUERY, RELEVANCE_JUDGMENTS.ID_DOCUMENTO,
RELEVANCE_JUDGMENTS.RELEVANT
FROM T__DS, CPC, RELEVANCE_JUDGMENTS, D__S, S__DS
WHERE T__DS.NOME_DATA_SET = S__DS.NOME_DATA_SET
  AND T__DS.NOME_TOPIC = CPC.NOME_TOPIC
  AND RELEVANCE_JUDGMENTS.ID_QUERY = CPC.ID_QUERY
  AND RELEVANCE_JUDGMENTS.ID_DOCUMENTO = D__S.ID_DOCUMENTO
  AND D__S.NOME_SPLIT = S__DS.NOME_SPLIT
```

(FE_E(AF)_Q<>D)

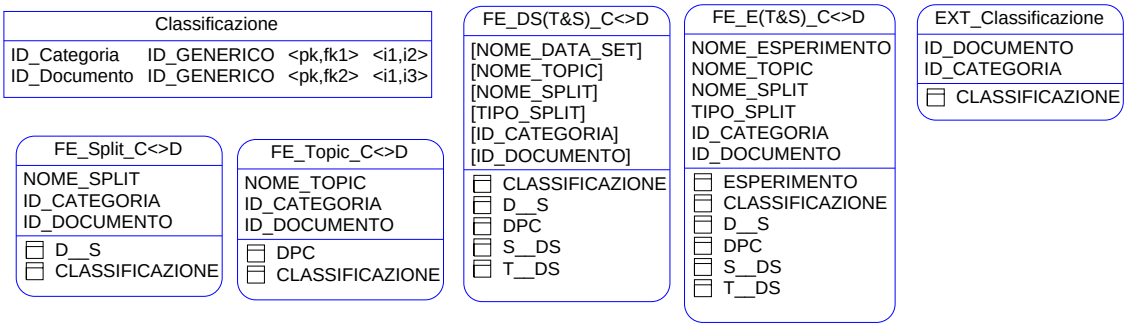
```
select ESPERIMENTO.NOME_ESPERIMENTO AS NOME_ESPERIMENTO_AF, RELEVANCE_JUDGMENTS.ID_QUERY,
RELEVANCE_JUDGMENTS.ID_DOCUMENTO, RELEVANCE_JUDGMENTS.RELEVANT
from ESPERIMENTO, ADAPTIVE_FILTERING, RELEVANCE_JUDGMENTS
where ADAPTIVE_FILTERING.ID_QUERY = RELEVANCE_JUDGMENTS.ID_QUERY
    and ADAPTIVE_FILTERING.ID_DOCUMENTO = RELEVANCE_JUDGMENTS.ID_DOCUMENTO
    and ADAPTIVE_FILTERING.NOME_DATA_SET = ESPERIMENTO.NOME_DATA_SET
```

(FE_E(T&S)_Q<>D)

```
SELECT ESPERIMENTO.NOME_ESPERIMENTO, CPC.NOME_TOPIC, D__S.NOME_SPLIT, D__S.TIPO_SPLIT,
RELEVANCE_JUDGMENTS.ID_QUERY, RELEVANCE_JUDGMENTS.ID_DOCUMENTO,
RELEVANCE_JUDGMENTS.RELEVANT
FROM ESPERIMENTO, T__DS, CPC, RELEVANCE_JUDGMENTS, D__S, S__DS
WHERE T__DS.NOME_DATA_SET = S__DS.NOME_DATA_SET
  AND T__DS.NOME_TOPIC = CPC.NOME_TOPIC
  AND RELEVANCE_JUDGMENTS.ID_QUERY = CPC.ID_QUERY
  AND RELEVANCE_JUDGMENTS.ID_DOCUMENTO = D__S.ID_DOCUMENTO
  AND D__S.NOME_SPLIT = S__DS.NOME_SPLIT
```

```
AND ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET
```

- **Classificazione**



(FE_Split_C<>D)
select D__S.NOME_SPLIT, CLASSIFICAZIONE.ID_CATEGORIA, CLASSIFICAZIONE.ID_DOCUMENTO
from D__S inner join CLASSIFICAZIONE on D__S.ID_DOCUMENTO = CLASSIFICAZIONE.ID_DOCUMENTO

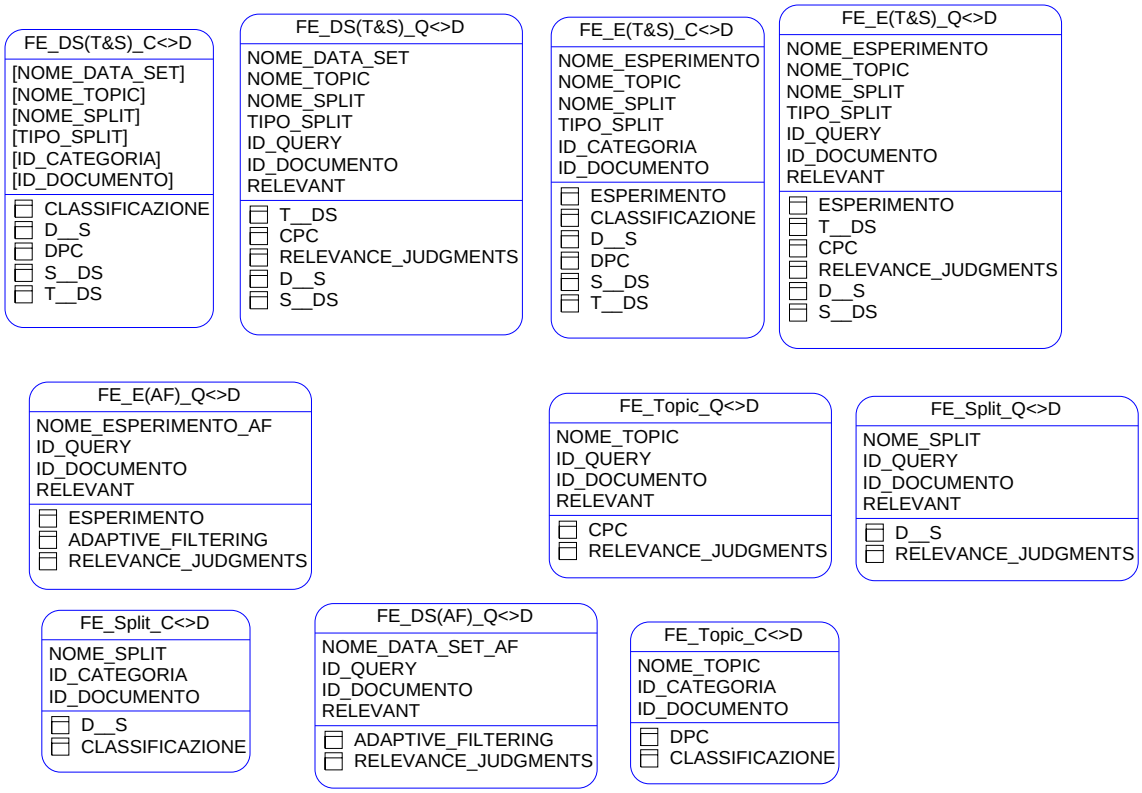
(FE_Topic_C<>D)
select DPC.NOME_TOPIC, CLASSIFICAZIONE.ID_CATEGORIA, CLASSIFICAZIONE.ID_DOCUMENTO
from DPC inner join CLASSIFICAZIONE on DPC.ID_CATEGORIA = CLASSIFICAZIONE.ID_CATEGORIA;

(FE_DS(T&S)_C<>D)
SELECT [S__DS].[NOME_DATA_SET], [DPC].[NOME_TOPIC], [D__S].[NOME_SPLIT], [D__S].[TIPO_SPLIT],
[CLASSIFICAZIONE].[ID_CATEGORIA], [CLASSIFICAZIONE].[ID_DOCUMENTO]
FROM CLASSIFICAZIONE, D__S, DPC, S__DS, T__DS
WHERE [S__DS].[NOME_DATA_SET]=[T__DS].[NOME_DATA_SET]
And [D__S].[NOME_SPLIT]=[S__DS].[NOME_SPLIT]
And [DPC].[NOME_TOPIC]=[T__DS].[NOME_TOPIC]
And [D__S].[ID_DOCUMENTO]=[CLASSIFICAZIONE].[ID_DOCUMENTO]
And [DPC].[ID_CATEGORIA]=[CLASSIFICAZIONE].[ID_CATEGORIA];

(FE_E(T&S)_C<>D)
select ESPERIMENTO.NOME_ESPERIMENTO, DPC.NOME_TOPIC, D__S.NOME_SPLIT, D__S.TIPO_SPLIT,
CLASSIFICAZIONE.ID_CATEGORIA, CLASSIFICAZIONE.ID_DOCUMENTO
from ESPERIMENTO, CLASSIFICAZIONE, D__S, DPC, S__DS, T__DS
WHERE [S__DS].[NOME_DATA_SET]=[T__DS].[NOME_DATA_SET]
And [D__S].[NOME_SPLIT]=[S__DS].[NOME_SPLIT]
And [DPC].[NOME_TOPIC]=[T__DS].[NOME_TOPIC]
And [D__S].[ID_DOCUMENTO]=[CLASSIFICAZIONE].[ID_DOCUMENTO]
And [DPC].[ID_CATEGORIA]=[CLASSIFICAZIONE].[ID_CATEGORIA]
AND ESPERIMENTO.NOME_DATA_SET = T__DS.NOME_DATA_SET

6.6.4 Viste delle QUERY d'interesse

Queste QUERY sono utilizzate dalle associazioni "Classificazione" e "Relevant Judgments" e sono già state descritte in precedenza.



6.7 Restrizioni del codice SQL dovute ad ACCESS

Già nella realizzazione del prototipo del database si sono riscontrate delle restrizioni a causa dell'utilizzo del *DBMS Microsoft Access 2000*. I file indicati nel paragrafo sono presenti nella directory "*Progettazione DB"* (si veda il *paragrafo 1.6*).

Il PowerDesigner permette di settare un particolare modello in grado di definire il *DBMS* da adoperare per il database progettato. Una volta individuato il DBMS è poi possibile eseguire dei comandi per realizzare il codice SQL necessario alla generazione del database (che nel nostro caso è quello schematizzato fin ora ed, come detto ad inizio capitolo, implementato nel file *PhysicalDataModel_DBDSATC v2.0.pdm*).

I primi problemi si sono riscontrati proprio a causa del modello "Microsoft Access 2000" implementato nella versione di PowerDesigner 9.0. Tale modello presenta principalmente dei difetti riguardanti la creazione delle chiavi primarie, difetti che sono stati corretti tramite l'utilizzo del modello ***MIOaccess2k.xdb***.

In seguito si sono riscontrate delle limitazioni a casua del *DBMS Access* che, anche se molto diffuso, presenta mancanze e restrizioni che hanno reso inattuabile la realizzazione di alcuni dei vincoli del prototipo. Tali mancanze sono state aggravate anche dall'impossibilita di utilizzare alcune DDL del *DBMS Access* a causa di carenze nei *driver ODBC*.

Tra le varie difficoltà incontrate si riportano di seguito solo quelle più di rilievo, individuando quando possibile la fonte del problema e la tecnica di risoluzione:

- Access 2000, in seguito ad un DELETE o ad un UPDATE, non consente il *Set Null* sui dati referenziati.
- Tramite ODBC è possibile realizzare vincoli referenziali, ma non dichiarare il comportamento in caso di DELETE o UPDATE (che per default sarà riportato come RESTRICT).
- Durante la realizzazione del codice del prototipo si è scoperta la mancanza della possibilità di verificare i Check e le ForeignKeys sui commit. Per realizzare le cancellazioni richieste nel capitolo precedente è stato necessario eliminare e ricreare i vincoli referenziali di volta in volta (operazione che, anche avenendo solo a livello transazionale, per la sua pericolosità può essere effettuata solo da un *DBA* e preferibilmente quando il sistema non è accessibile agli utenti). La distruzione dei vincoli ha imposto (per la mancanza del settaggio tramite codice del comportamento sul DELETE e sull'UPDATE) di porre a Restrict i vincoli referenziali legati ad i Documenti, Categorie, Query, Topic e Split. Nella realizzazione del codice in seguito alla distruzione del vincolo si procederà alla sua ricostruzione utilizzando, naturalmente, il nome precedente del vincolo.
- Non è possibile utilizzare tramite ODBC i comandi *CREATE USER* e *GRANT*, da qui la necessità di creare la struttura alternativa già discussa nel *paragrafo 6.3.3* e di aggiungere manualmente dei gruppi e degli utenti che permettano di gestire l'accesso alle diverse tabelle create.
- Mancanza dell'utilizzo dei *check* tramite ODBC. Ciò ha indotto ad aggiungere alcuni vincoli manualmente, e di realizzare alcuni dei domini tramite l'utilizzo di entità e vincoli referenziali.
- Mancanza della possibilità in Access 2000 di vincolare l'inserzione delle tuple alle *viste Test* (si vedano i *paragrafi 6.4.2* e *6.5.1*). Tale carenza dovrà essere risolta via codice tramite il *modulo d'In/Out*.

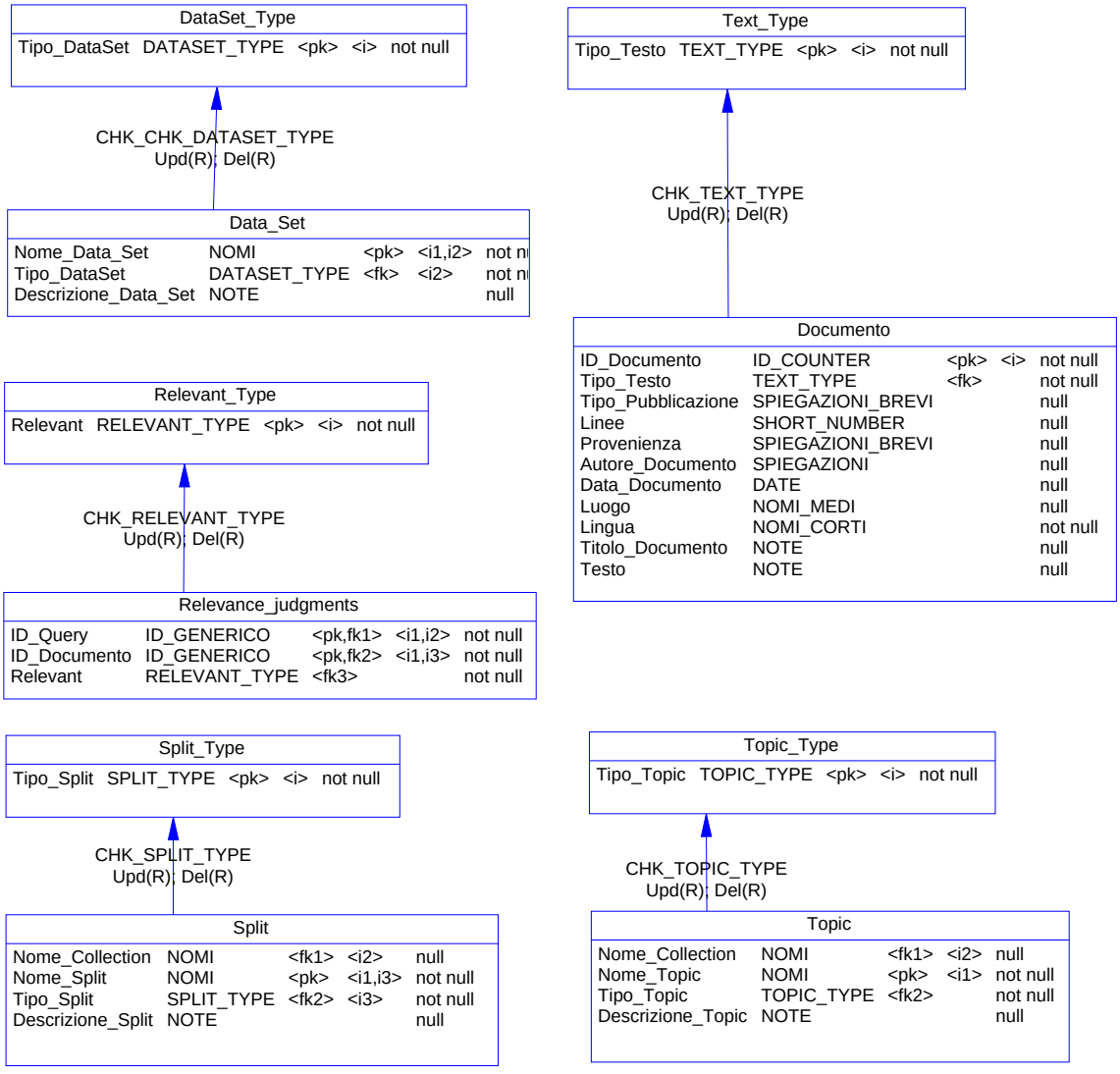
Il codice SQL generato dal PowerDesigner 9.0 per creare il database privo di vincoli manuali è riportato nel file "*ODBC DBDSATC v2.0.sql*", mentre uno script più completo, generato sempre tramite PowerDesigner 9.0 e contenente anche informazioni

riguardanti i vincoli che saranno inseriti manualmente, è presente nel file "*SCRIPT DBDSATC v2.0.sql*".

6.7.1 Implementazione Domini "Type"

L'impossibilita in Access di generare automaticamente i check ci suggerisce di adoperare al posto dei domini enumerativi (*DATASET_TYPE, TOPIC_TYPE, SPLIT_TYPE, TEXT_TYPE e RELEVANT_TYPE*) delle Tabelle che conterranno i loro valori, in tal modo sarà anche possibile aggiungerne di nuovi per descrivere informazioni future prelevate da altri benchmark.

Oltre alla dichiarazione delle precedenti tabelle e dei loro vincoli referenziali il codice SQL, presente nel file "*ODBC DBDSATC v2.0.sql*", utilizza alcuni comandi DML, facenti parte dello standard SQL, per popolare le tabelle con i valori dei domini indicati (si veda il *paragrafo 6.2.1*).



6.7.2 Vincoli Manuali Aggiuntivi

I vincoli manuali da aggiungere al database, realizzato in Access, sono riportati nel file "*Vincoli Manuali v2.0.txt*" mentre i gruppi e gli utenti nel file "*\Progettazione DB\DB settato\DBDSATC.snp*". Un elenco di questi vincoli è descritto di seguito:

1) **CHECK:**
   a. Nella Tabella Sovracategoria si deve aggiungere il check che pone ID_SOVRACATEGORIA <> ID_CATEGORIA
   b. Tutti gli attributi che si riferiscono ai domini MISURE e SHORT_NUMBER devono avere il loro valore ≥ 0.
2) **VINCOLI REFERENZIALI:**
   a. Per default gli UPDATE e i DELETE sono CASCADE.
   b. FK_S__DS_S: Upd(R),Del(R)
   c. FK_T__DS_T: Upd(R),Del(R)
   d. FK_DPC_C: Upd(R),Del(R)
   e. FK_CPC_Q: Upd(R),Del(R)
   f. FK_D__B_D: Upd(R),Del(R)
   g. FK_D__S_D: Upd(R),Del(R)
   h. FK_M__E_M: Upd(R),Del(R)
   i. FK_DS__E: Upd(R),Del(R)
      [non si può cancellare nulla che sia referenziato da un esperimento]
   j. CHK_DATASET_TYPE: Upd(R),Del(R)
   k. CHK_TEXT_TYPE: Upd(R),Del(R)
   l. CHK_TOPIC_TYPE: Upd(R),Del(R)
   m. CHK_SPLIT_TYPE: Upd(R),Del(R)
   n. CHK_RELEVANT_TYPE: Upd(R),Del(R)
3) **VINCOLI USER:**
   a. ***Gruppi***:
      i. ***Admins***: default Access che permette di effettuare tutto.
      ii. **DBA:** può tutto (in realtà modifica ed amministra le strutture solo nel caso sia necessario, ad esempio per gestire l'eliminazione delle ForeignKeys).
      iii. **Users:** default Access a cui non si associa nessun permesso.
      iv. **RICERCATORI**:
         1. Può Aprire e usare il database.

2. Sulle viste può tutto tranne modificarne la struttura ed amministrarle. Fanno eccezione le viste *Test_?* che può solo leggere.
3. Può leggere tutte le Tabelle
4. Può inserire in Effectiveness_Misure
5. Può inserire in Esperimenti
6. Non può nulla su la tabella Permessi, e sulle viste EXT_Permessi e ID_Rirecatori.

v. **RICERCATORIXXX** (da aggiungere come gruppo insieme a RICERCATORI, si noti che nel codice si limiteranno i permessi solo agli esperimenti cui ha accesso l'utente):

1. Può inserire, cancellare e modificare (non le strutture) nelle tabelle:
   a. *R__E*
   b. *M__E*
   c. *Feature_Vector*
   d. *CPC_Risultati*
   e. *DPC_Risultati*
   f. *Esperimenti*

b. ***Utenti**:* come già detto in precedenza servono solo due user effettivi, che sono portati a tre (**TESTER**) per permettere al *modulo d'In/Out* di connettersi inizialmente al database e di effettuare le sue verifiche sulla tabella permessi per poi, se l'utente esiste, creare una connessione tramite l'user **DBAall** o **RICERCATOREall**:

i. **DBAall**: appartiene ai gruppi DBA, RICERCATORE, RICERCATOREXXX.
ii. **RICERCATOREall**: appartiene ai gruppi RICERCATORE, RICERCATOREXXX.
iii. **TESTER**: appartiene al gruppo ***Admins***
iv. **ZAR**: appartiene al gruppo ***Admins*** ed è stato inserito per gestire facilmente il database tramite l'interfaccia fornita dal *DBMS Access 2000*. Allo stato attuale tale utente è privo di password.

6.8 I file DBDSATC ed il loro utilizzo

I file realizzati, contenenti il database privo di dati, sono *"\Progettazione DB\DB settato\DBDSATC.mdb"* per quanto riguarda i dati del database (si osservi che i file Access non possono avere una dimensione maggiore di 2GB e ciò è un limite dal momento che in seguito al feeding tale file sarà circa di 1,2GB) ed *"\Progettazione DB\DB settato\DBDSATC.mdw"* per quanto riguarda i dati di sistema (Il DBMS Access di norma gestisce gli user adoperando i settaggi standard, ma è possibile realizzare un file che permetta di rifarsi ad utenze che esulino quelle della macchina in cui è installato il pacchetto applicativo, tale file ne è un esempio).

Il database realizzato potrà essere adoperato tramite l'interfaccia grafica fornita col pacchetto *Access* o utilizzando il *modulo d'In/Out* realizzato nel prossimo capitolo. Di seguito si descrivono le due modalità di funzionamento ed i relativi settaggi preliminari richiesti:

a. L'utilizzo *per mezzo dell'interfaccia Access* richiede di creare un link col formalismo riportato di seguito ed adoperare i dati degli utenti descritti nel paragrafo precedente:

   file programma:<"><path> <\> <MSACCESS.EXE><">

   file dati: <"><path> <\>< *DBDSATC.mdb*><">

   file di sistemai: </WRKGRP><"><path> <\>< *DBDSATC.mdw*><">

b. L'utilizzo *tramite il modulo d'In/Out*, che sarà descritto nel prossimo capitolo, richiede di settare il database (*DBDSATC*) come fonte ODBC e di adoperare i dati degli utenti inseriti nella tabella ***Permessi***. Un esempio sul processo di settaggio della fonte ODBC è riportato di seguito:

   a. Aprire *amministrazione delle origine dati ODBC*
   b. Aggiungere un *DSN di sistema*
   c. Selezionare come driver dell'origine dati *Microsoft Access Driver*
   d. Tramite **Seleziona (1)** inserire il file *DBDSATC.mdb*
   e. Abilitare il *check Database (2)* nel riquadro Database di sistema.
   f. Tramite **Database di sistema (3)** inserire il file *DBDSATC.mdw*
   g. Inserire il nome desiderato per la fonte.
   h. *Dare l'OK*


Configurazione ODBC per Microsoft Access
Nome origine dati:
inserire nome fonte
Descrizione:
Database
Database: E:\...\GOOD\DBDSATC.mdb
Seleziona...
Crea...
Ripristina...
Compatta...
1
Database di sistema
Nessuno
Database:
E:\...\GOOD\DBDSATC.mdw
2
Database di sistema...
3
OK
Annulla
?
Avanzate...
Opzioni >>

CAPITOLO 7:

IL MODULO DI IN/OUT

7.1 Architettura del Modulo

7.2 Core del Modulo

7.3 Modello Object-Oriented del database

7.4 Classe Permessi

7.5 Classe Abstract

7.6 Classe Vincoli

7.7 Classe DBDSATC

7.8 Organizzazione in Libreria del Modulo

Il capitolo ha il fine di descrivere la realizzazione dell'Object-Oriented Model alla base del Modulo d'In/Out (libreria DBDSATC) e di offrire un manuale per l'utilizzo delle classi contenute nel modulo. Il codice implementato sarà riportato solo nei suoi punti salienti.

I diagrammi e tutta la documentazione del capitolo sono stati prelevati dal workspace "DataSetDB.sws" (si veda il paragrafo 1.6) ed in particolare dai modelli presenti in "Progettazione DB\ObjDataModel_DBDSATC v2.2.oom".

7.1 Architettura del Modulo

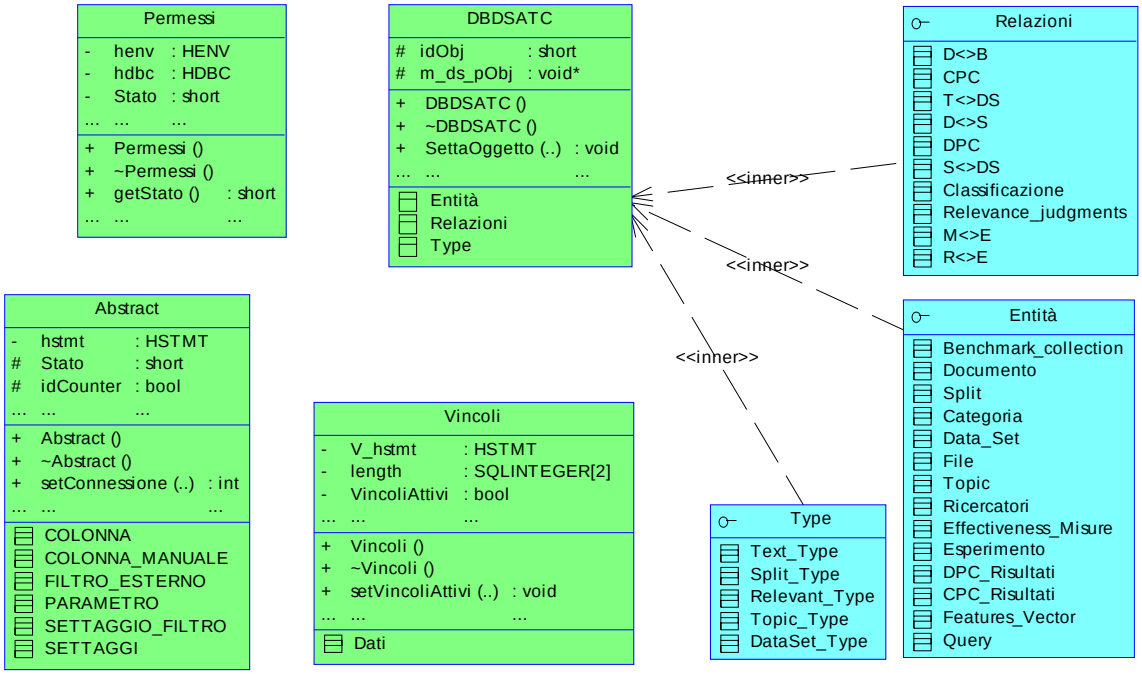


Figura 18: Diagramma Principale del Modulo di In/Out

Il Modulo contiene le classi presentate in figura, dove le interfacce introdotte sono state adoperate solo per raggruppare le definizioni di classi concettualmente simili sotto un'unica classe.

La classe ***Permessi*** è il cuore del modulo, poiché permette la connessione al database ed il settaggio degli *Statement* che saranno adoperati dalle altre classi.

Dalla classe ***Abstract*** derivano tutte quelle relative alle Entità e alle Relazioni del database; queste sono poi organizzate nelle classi d'interfaccia ***Relazioni***, ***Entità*** e ***Type***.

La classe ***Vincoli*** permette di aggiungere la gestione di vincoli non contenuti nel database ed è, all'occorrenza, adoperata al posto della classe Abstract (da cui deriva).

Infine la classe ***DBDSATC*** contiene le interfacce introdotte e permette, tramite un settaggio iniziale, di funzionare come un template (non implementato come tale dal momento che avrebbe richiesto ad eventuali interfacce grafiche gestite con un architettura Documento/Vista la dichiarazione e la creazione di un documento diverso per ogni possibile elemento del template). Si osservi che tramite la classe DBDSATC è possibile creare anche direttamente oggetti definiti tramite le interfacce.

7.2 Core del Modulo

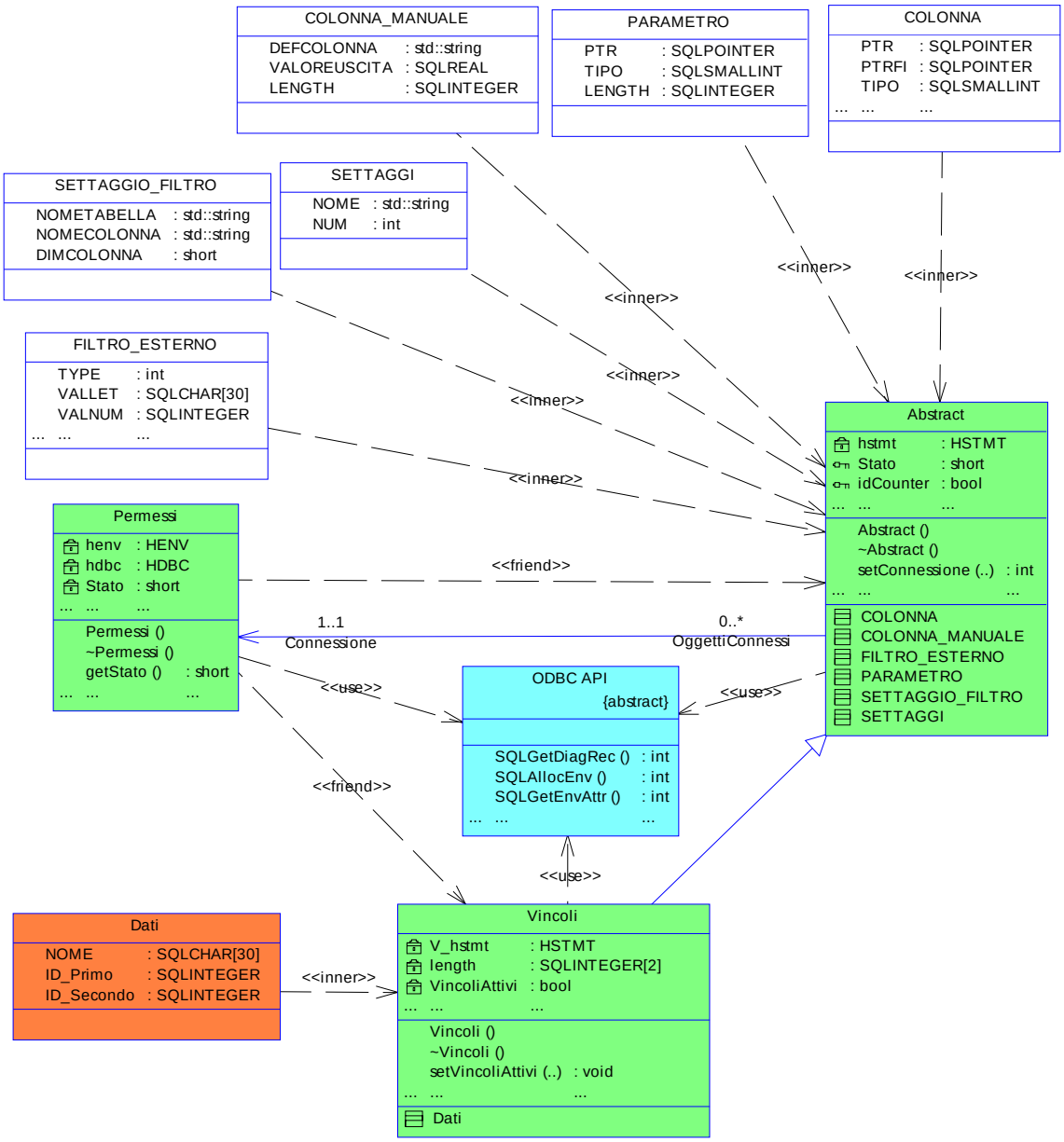


Figura 19: Core del Modulo di In/Out

Il diagramma, riportato in figura, ci permette di comprendere le dipendenze e le associazioni esistenti tra le varie classi che costituiscono il cuore del sistema.

Le tre classi create adoperano una classe fittizia, ***ODBC API***, definita solo per rappresentare le librerie adoperate per la realizzazione del prodotto. Con tale classe astratta ci si riferisce al seguente insieme di header:

- *set.h* (adoperato da ***Permessi*** e ***Abstract***)
- *string.h*
- *windows.h*
- *sqlext.h (include sql.h)*

All'interno di tale classe si sono inserite anche tutte le API che sono state adoperate per implementare il codice. La classe è stata utilizzata anche per la realizzazione dei diagrammi sequenziali delle funzionalità più critiche.
La classe ***Permessi*** è realizzata in modo da consentire solo alle classi ***Abstract*** e ***Vincoli***, dichiarate come *friend*, la creazione di statement legati alla sua connessione; ciò evita l'esecuzione, esternamente al modulo, d'operazioni non desiderate. Dunque solo le classi ***Abstract*** e ***Vincoli*** sono abilitate alla gestione degli *statement* relativi alla connessione aperta da un oggetto di ***Permessi***.
La classe ***Abstract*** al suo interno contiene un membro (*Connessione*) che permette di associarla ad un particolare oggetto di ***Permessi*** e di ricavare così le informazioni necessarie per: creare e richiamare comandi riguardanti gli statement; gestire gli errori; conoscere se sono stati effettuati dei *Rollback* o *Commit* (utilizzando *set.h*) ecc...

7.3 Modello Object-Oriented del database

Tramite l'utilizzo di PowerDesigner 9.0 è stato possibile realizzare dei *Modelli Object-Oriented* partendo dagli schemi logici del database ricavato nel capitolo precedente. I diagrammi ricavati si rifanno a classi definite come *Elementi* (che saranno adoperate in quelle implementate nelle interfacce del *paragrafo 7.1*) e ci permettono di mostrare in grandi linee i legami tra di esse ed i membri attributo necessari per definirle. L'uso delle *API ODBC* ha richiesto, inoltre, la definizione di domini che siano in grado di convertire, utilizzando i tipi definiti in *sqlext.h* e *sql.h*, i campi SQL in variabili C++.

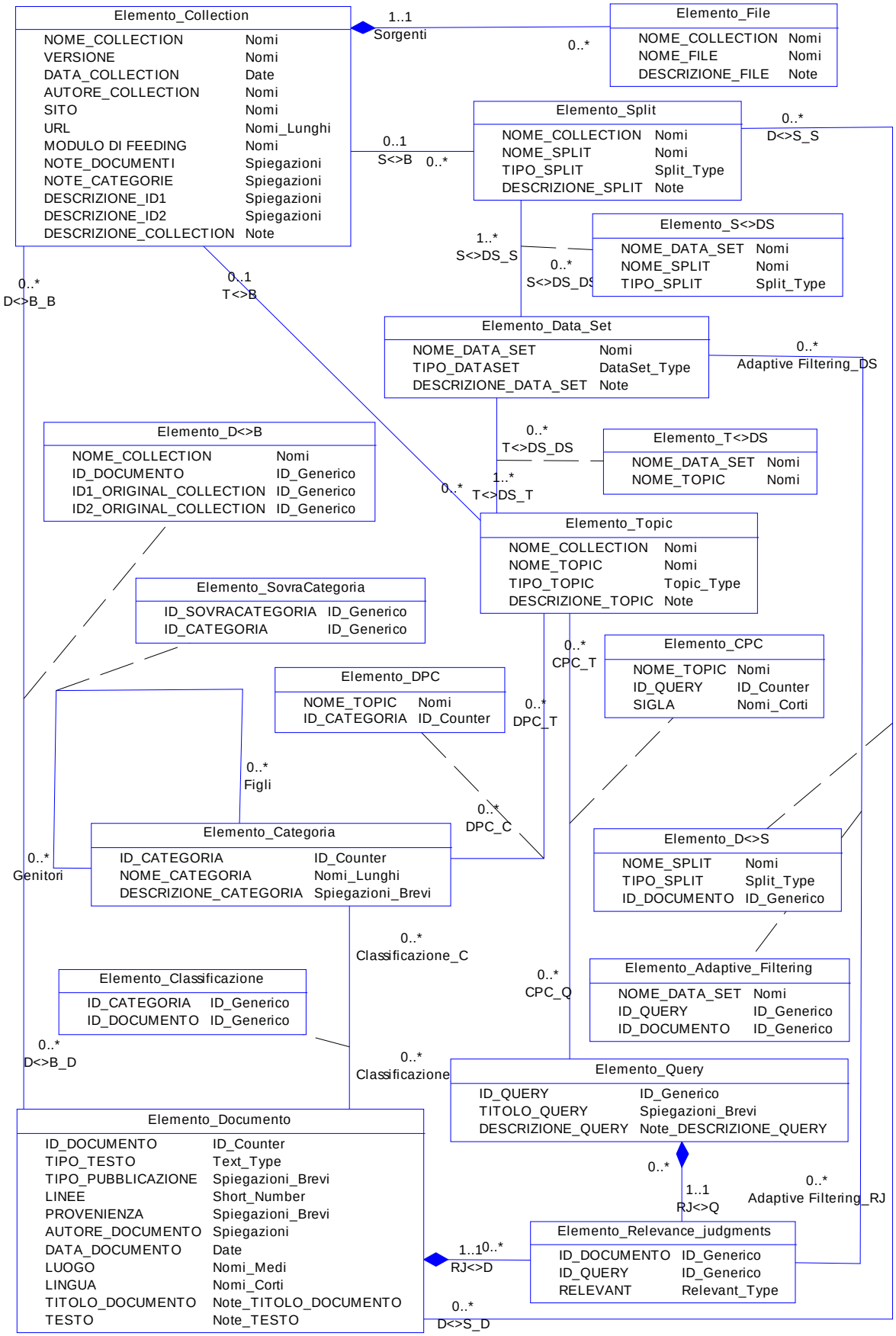


Figura 20: Modello Object-Oriented del database (Benchmarks)

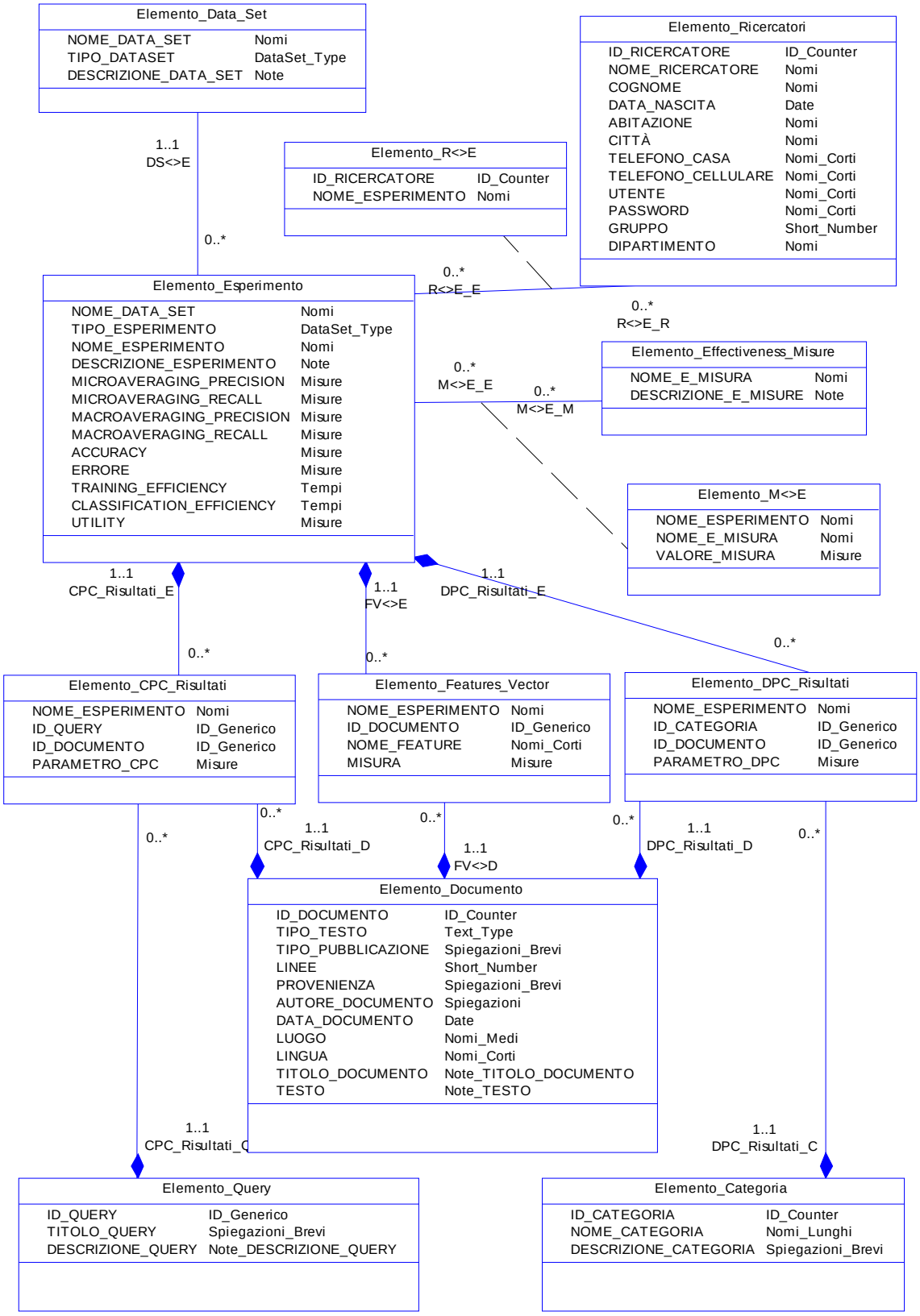


Figura 21: Modello Object-Oriented del database (Esperimenti)

Mostrati gli elementi che saranno adoperati come strutture base dalle classi derivate da **Abstract**, si procede con la dichiarazione dei domini relativi ai tipi utilizzati per i diversi attributi introdotti:

Dominio	dimensione	Tipo API SQL	Tipo C++
Date		DATE_STRUCT	typedef struct tagDATE_STRUCT { SQLSMALLINT year; SQLUSMALLINT month; SQLUSMALLINT day; } DATE_STRUCT;
Tempi		TIMESTAMP_STRUCT	typedef struct tagTIMESTAMP_STRUCT { SQLSMALLINT year; SQLUSMALLINT month; SQLUSMALLINT day; SQLUSMALLINT hour; SQLUSMALLINT minute; SQLUSMALLINT second; SQLUINTEGER fraction; } TIMESTAMP_STRUCT;
ID_Counter		SQLINTEGER	long
ID_Generico		SQLINTEGER	long
Short_Number		SQLSMALLINT	short
Misure		SQLREAL	float
Nomi_Corti	15	SQLCHAR	unsigned char
DataSet_Type	15	SQLCHAR	unsigned char
Split_Type	15	SQLCHAR	unsigned char
Text_Type	15	SQLCHAR	unsigned char
Topic_Type	15	SQLCHAR	unsigned char
Relevant_Type	15	SQLCHAR	unsigned char
Nomi	30	SQLCHAR	unsigned char
Nomi_Medi	60	SQLCHAR	unsigned char
Nomi_Lunghi	90	SQLCHAR	unsigned char
Note	**50000**	**SQLCHAR**	**unsigned char**
Note_DESCRIZIONE_QUERY	**1000**	**SQLCHAR**	**unsigned char**
Note_TESTO	**170000**	**SQLCHAR**	**unsigned char**
Note_TITOLO_DOCUMENTO	**450**	**SQLCHAR**	**unsigned char**
Spiegazioni_Brevi	255	SQLCHAR	unsigned char
Spiegazioni	160	SQLCHAR	unsigned char

L'introduzione di più domini *Note* è stata richiesta in seguito ad alcune problematiche presentatesi nella realizzazione del prototipo del *Modulo d'In/Out*. In un primo momento si era pensato di adoprare per gli attributi testuali il tipo ***std::string*** implementato in *string.h*, ma tale scelta non è risultata felice dal momento che il tipo è implementato per mezzo di puntatori dinamici e quindi i bind effettuati per preparare le QUERY risultavano falsati. Nel prototipo tale problema era stato risolto passando i

valori degli attributi di tipo ***std::string*** ad appositi buffer che poi venivano bindati, ma tale soluzione causava un ulteriore perdita di tempo per effettuare il passaggio dei dati dalla stringa ai buffer.

Nella versione finale si sono utilizzati direttamente attributi che potessero essere adoperati come buffer, ma ciò ha richiesto un'analisi dettagliata anche delle dimensioni dei campi note in modo da ottimizzare i buffer. Tale studio ancora una volta è effettuato tramite l'analisi dei risultati del codice "*Test Dimensioni.cpp*" già introdotto nel *paragrafo 6.2.2* e che sarà meglio approfondito nel *paragrafo 8.4.* Di seguito si riportano i risultati riguardanti i campi "note" memorizzati nel file "*Progettazione DB\Dimensioni utili per il codice.txt":*

Note [NOTE(oltre 255)]:
(alcuni dei campi indicati sono poco ricorrenti e la ridondanza da il vantaggio di avere una maggiore sicurezza nel poter inserire tutte le informazioni necessarie)
per implementare il Modulo si aggiungono i domini:

Note_TITOLO_DOCUMENTO: 450
Note_DESCRIZIONE_QUERY: 1000
Note_DESCRIZIONE: 50000 (Per tutte le descrizioni dove la ridondanza è accettata)
Note_TESTO: 170000

DimMax_DESCRIZIONE_COLLECTION: 271
DimMax_DESCRIZIONE_FILE: 306
DimMax_TITOLO_DOCUMENTO: 402 (solo 285 >255)
DimMax_TESTO: 160499
DimMax_DESCRIZIONE_QUERY: 942
DimMax_DESCRIZIONE_SPLIT: 249 (ridondanza a NOTE accettata)
DimMax_DESCRIZIONE_TOPIC: 98 (ridondanza a NOTE accettata)
DimMax_DESCRIZIONE_DATA_SET: 2319

7.4 Classe Permessi

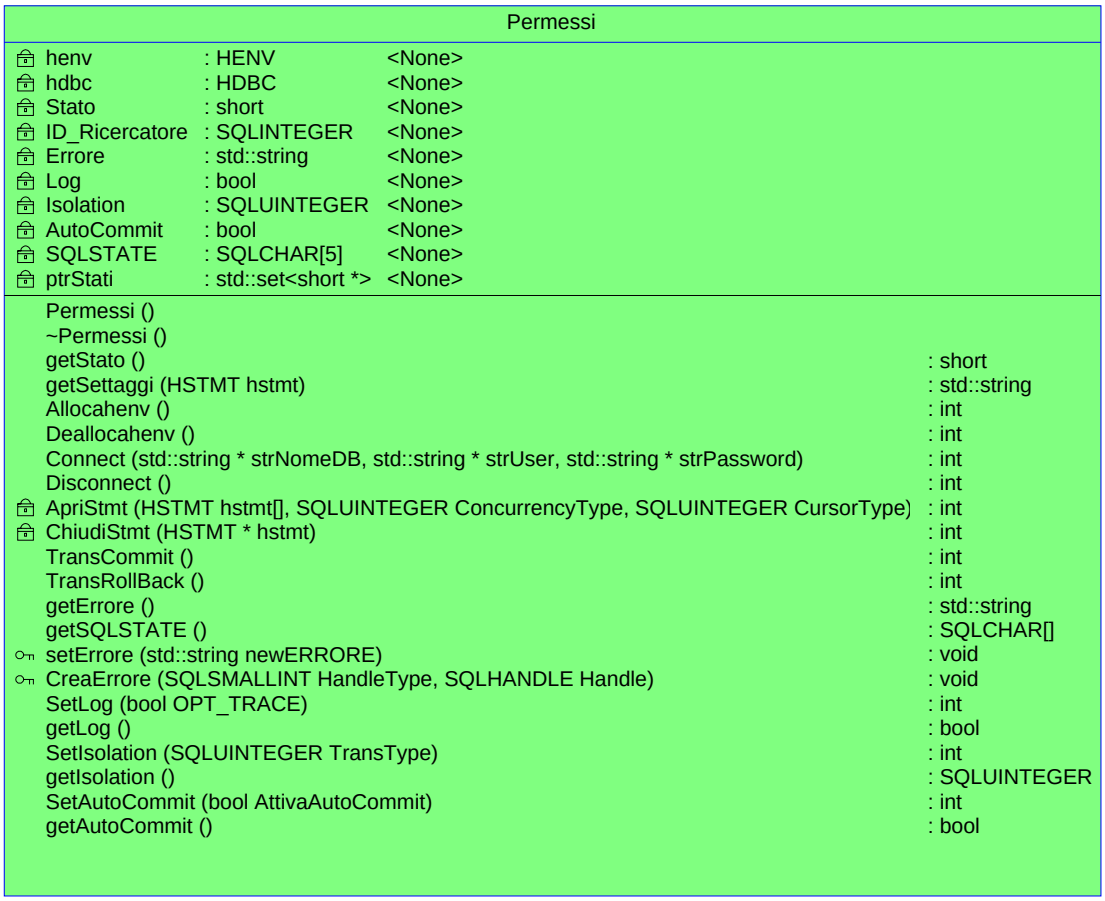


Figura 22: Classe Permessi

La Classe usa la tabella Permessi per concedere l'accesso agli esperimenti, ma non serve ad implementare tale tabella che invece sarà gestita nella classe Ricercatori.

L'obiettivo della classe è quello di mettere a disposizione dell'utente del sistema un oggetto che possa gestire i collegamenti al database in modo rapido e che sia in grado di offrire alle classi che adoperano la connessione una serie di informazioni e servizi in modo automatico, senza che l'utente debba interessarsene.

Per far meglio comprendere la classe si riportano delle brevi spiegazioni riguardanti i suoi membri cercando di raggruppare le funzionalità legate tra loro:

7.4.1 I Membri

Nel descrivere i parametri dei membri della classe si adopera il formalismo <*CORSIVO_E_MAIUSCOLO*> per individuare i valori definiti tramite direttive negli header *sql.h* e *sqlext.h*, e <SOTTOLINEATO_E_MAIUSCOLO> per individuare i valori definiti tramite direttive nell'header della classe:

- Il **costruttore** **Permessi()** non fa altro che inizializzare gli attributi della classe, mentre il **distruttore** **~Permessi**() verifica se l'oggetto non è disconnesso ed in tal caso procede richiamando il membro **Disconnect()**.

- Il membro ***henv*** contiene l'Environment handle adoperato dalla connessione, tale membro è statico e quindi basta inizializzarlo una sola volta per tutti gli oggetti della classe. Questo comportamento è giustificato dal fatto che le transazioni avvengono a livello di connessione e che quindi non ci interessa avere più environment.. Per poter creare una connessione deve quindi esistere almeno un oggetto che abbia richiamato il membro **Allocahenv()**. Per quanto riguarda invece il resettaggio di ***henv*** *si* adopera il membro **Deallocahenv()** che dovrà essere richiamato solo quando non ci saranno più oggetti della classe **Permessi** con una connessione attiva.
    - **Allocahenv()**: I valori restituiti sono PERMESSI_ERROR_SQL oppure PERMESSI_HENV_ALLOCATO
    - **Deallocahenv()**: I valori restituiti sono PERMESSI_ERROR_SQL oppure PERMESSI_HENV_DEALLOCATO

- Il membro ***hdbc*** contiene il Connection handle relativo all'attuale connessione aperta dall'oggetto. Per attivare una connessione è necessario richiamare il membro **Connect()** e passare la fonte ODBC relativa al database, e la User e la Password dell'utente che desidera adoperare il Modulo. Tale Membro si preoccupa di effettuare i settaggi base e di aprire una prima connessione tramite l'Utente **TESTER** introdotto nel *paragrafo 6.7.2*. A questo punto, dopo aver effettuato i dovuti controlli sulla tabella Permessi, si procede alla chiusura dell'attuale connessione ed all'eventuale apertura di quella dell'utente **RICERCATOREall** o **DBAall,** secondo il gruppo d'appartenenza dell'utente contenuto nella tabella Permessi. Dopo tale operazione si procede col salvataggio in ***ID_Ricercatore*** dell'identificativo dell'utente rintracciato ed in ***Stato*** dell'attuale modalità di funzionamento dell'oggetto (DISATTIVO, DBA o RICERCATORE).
    - **Connect()**: I valori restituiti sono PERMESSI_USER_SCONOSCIUTO, PERMESSI_ERROR_SQL, PERMESSI_GIA_CONNESSO, ed infine PERMESSI_CONNESSO.
    - **GetStato()**: Permette di prelevare lo Stato attuale dell'oggetto.

  Tramite i membri ***Log***, ***Isolation*** e ***AutoCommit*** è inoltre possibile gestire alcune funzionalità opzionali sulla connessione:

- **SetLog()**: Permette di attivare/disattivare l'uso del file Log "ODBC_DBDSATC.log", i valori di ritorno sono PERMESSI_TRACE_OK oppure PERMESSI_ERRORE_SQL.
- **SetIsolation()**: Permette di settare il "transaction isolation level" della connessione per mezzo del parametro TransType i cui valori permessi sono :

  SQL_TRANSACTION_READ_UNCOMMITTED,

  SQL_TRANSACTION_READ_COMMITTED,

  SQL_TRANSACTION_REPEATABLE_READ,

  SQL_TRANSACTION_SERIALIZABLE;

  i possibili valori di ritorno sono PERMESSI_ISOLATION_OK oppure PERMESSI_ERRORE_SQL.
- **SetAutoCommit()**: Permette di attivare/disattivare l' AutoCommit della connessione, i valori di ritorno sono PERMESSI_ERRORE_SQL oppure PERMESSI_AUTOCOMMIT_OK.
- **GetLog()**, **getIsolation()** e **getAutoCommit()** permettono di prelevare i valori dei relativi membri.

Il membro **Disconnect()** invece si preoccupa di disconnettere ***hdbc*** e di avvisare dell'evento tutti gli oggetti **Abstract** che adoperano l'attuale connessione. Tale operazione avviene settando a PERMESSI_INESISTENTE i membri ***Stato*** degli oggetti **Abstract** che sono puntati dal set presente nel membro ***ptrStati***.

- **Disconnect()**: I valori restituiti sono PERMESSI_GIA_DISCONNESSO, PERMESSI_ERROR_SQL oppure PERMESSI_DISCONNESSO.

- Il membro ***Errore*** contiene una stringa con informazioni sugli errori che si sono verificati. La creazione della stringa può avvenire tramite due funzioni membro: **CreaErrore()** che vi aggiunge informazioni, ricavate tramite API ODBC, attinenti all'ultimo errore generato dalle API di comando che si riferiscono ad un particolare *HandleType* che può avere uno dei seguenti tre valori: *SQL_HANDLE_ENV*, *SQL_HANDLE_DBC* e *SQL_HANDLE_STMT* (viene spesso adoperata dalle classi amiche di Permessi); e **SetErrore()** che permette di inserire direttamente la stringa di errore che poi sarà fornita in uscita.

  Tramite **CreaErrore()** si setta anche la stringa ***SQLSTATE*** che contiene l'**ODBC Error Code** dell'ultimo errore riscontrato.

  Le uniche due funzioni membro a cui avrà accesso l'utente sono **getErrore()** e **getSQLSTATE()**, che gli permettono di prelevare le informazioni necessarie a

comprendere la causa dell'ultimo errore. Gli altri membri fin qui introdotti sono invece adoperati dalle classi del *Core della libreria* per generare le informazioni necessarie all'utente.

- Altre informazioni utili si possono prelevare per mezzo di **getSettaggi()** che restituisce una stringa contenente tutti i settaggi relativi all'Environment, alla Connection ed allo Statement dato in ingresso; se non vi è nessuno statement da verificare si deve dare in ingresso *SQL_NULL_HANDLE*.

- Una volta creata la connessione, è necessario offrire alle classi **Abstract** e **Vincoli** dei membri che permettano di gestire gli *statement*.
  - **ApriStmt()** permette di aprire uno statement legato alla connessione attivata dalla particolare istanza della classe Permessi. I valori consentiti al parametro ConcurrencyType: sono:

    SQL_CONCUR_ROWVER,,
    SQL_CONCUR_LOCK,
    SQL_CONCUR_READ_ONLY,
    SQL_CONCUR_VALUES;

    mentre quelli permessi al parametro CursorType sono:

    SQL_CURSOR_FORWARD_ONLY,
    SQL_CURSOR_STATIC,
    SQL_CURSOR_KEYSET_DRIVEN,
    SQL_CURSOR_DYNAMIC;

    infine i valori di uscita del membro sono PERMESSI_STMT_APERTO, PERMESSI_NON_CONNESSO, PERMESSI_STMT_GIA_ALLOCATO oppure PERMESSI_ERROR_SQL.
  - **ChiudiStmt()** permette di chiudere uno statement legato alla connessione attivata dalla particolare istanza della classe Permessi. I valori di uscita sono definiti come: PERMESSI_STMT_CHIUSO, PERMESSI_ERROR_SQL oppure PERMESSI_STMT_GIA_DEALLOCATO.

- Una volta creata la connessione ed i vari statement, è possibile adoperarli per eseguire la transazione d'interesse, alla cui fine sarà necessario comunicare all'oggetto ***hdbc*** se la transazione deve essere confermata (commit) o annullata (rollback). Nel caso del **DBMS Access 2000** le **API ODBC** gestiscono il comando

SQLEndTran in modo distruttivo, in altre parole chiudono i cursori ed i plan (Prepare) di tutti gli statement relativi alla connessione aperta.

I membri **TransCommit()** e **TransRollBack()** dunque si preoccupano non solo di richiamare la particolare API per effettuare il commit o il rollback, ma anche di avvisare tutte le istanze di **Abstract** adoperanti statement connessi all'oggetto che è stata effettuata una delle due operazioni. Tale operazione avviene settando a PERMESSI_EndTrans i membri ***Stato***, degli oggetti **Abstract,** che sono puntati dal set presente nel membro ***ptrStati***.

- **TransCommit ()**: i valori restituiti sono PERMESSI_ERROR_SQL oppure PERMESSI_COMMIT
- **TransRollBack ()**: i valori restituiti sono PERMESSI_ERROR_SQL oppure PERMESSI_ROLLBACK

7.4.2 Le Direttive

Codice delle direttive relative ai valori utilizzati nel paragrafo precedente.

```
/*definizioni degli stati */
#define DISATTIVO 0
#define DBA         1
#define RICERCATORE 2
/*definizioni dei valori di uscita dei membri di Permessi */
#define PERMESSI_HENV_ALLOCATO 1
#define PERMESSI_ERROR_SQL -1
#define PERMESSI_HENV_DEALLOCATO 2
#define PERMESSI_CONNESSO 3
#define PERMESSI_USER_SCONOSCIUTO 0
#define PERMESSI_GIA_CONNESSO -3
#define PERMESSI_DISCONNESSO 4
#define PERMESSI_GIA_DISCONNESSO -4
#define PERMESSI_STMT_APERTO 5
#define PERMESSI_NON_CONNESSO -50
#define PERMESSI_STMT_GIA_ALLOCATO -51
#define PERMESSI_STMT_CHIUSO 6
#define PERMESSI_STMT_GIA_DEALLOCATO -6
#define PERMESSI_COMMIT 7
#define PERMESSI_ROLLBACK 8
#define PERMESSI_ISOLATION_OK 9
#define PERMESSI_TRACE_OK  10
#define PERMESSI_AUTOCOMMIT_OK  11
/* Definizione dei valori di uscita delle funzioni membro
delle classi che adopereranno Permessi */
#define ERRORE_SQL              PERMESSI_ERROR_SQL
#define NO_ERRORE               1
#define NO_DATO                 100
#define PERMESSI_NON_ABILITATI   -10
#define PERMESSI_ABILITATI       10
#define NO_ACCESS               -100
/* Definizione di un valore indicante la fine di una transazione */
#define PERMESSI_EndTrans       0
#define PERMESSI_INESISTENTE     -99
```

7.4.3 Sequence Diagram di una Transazione

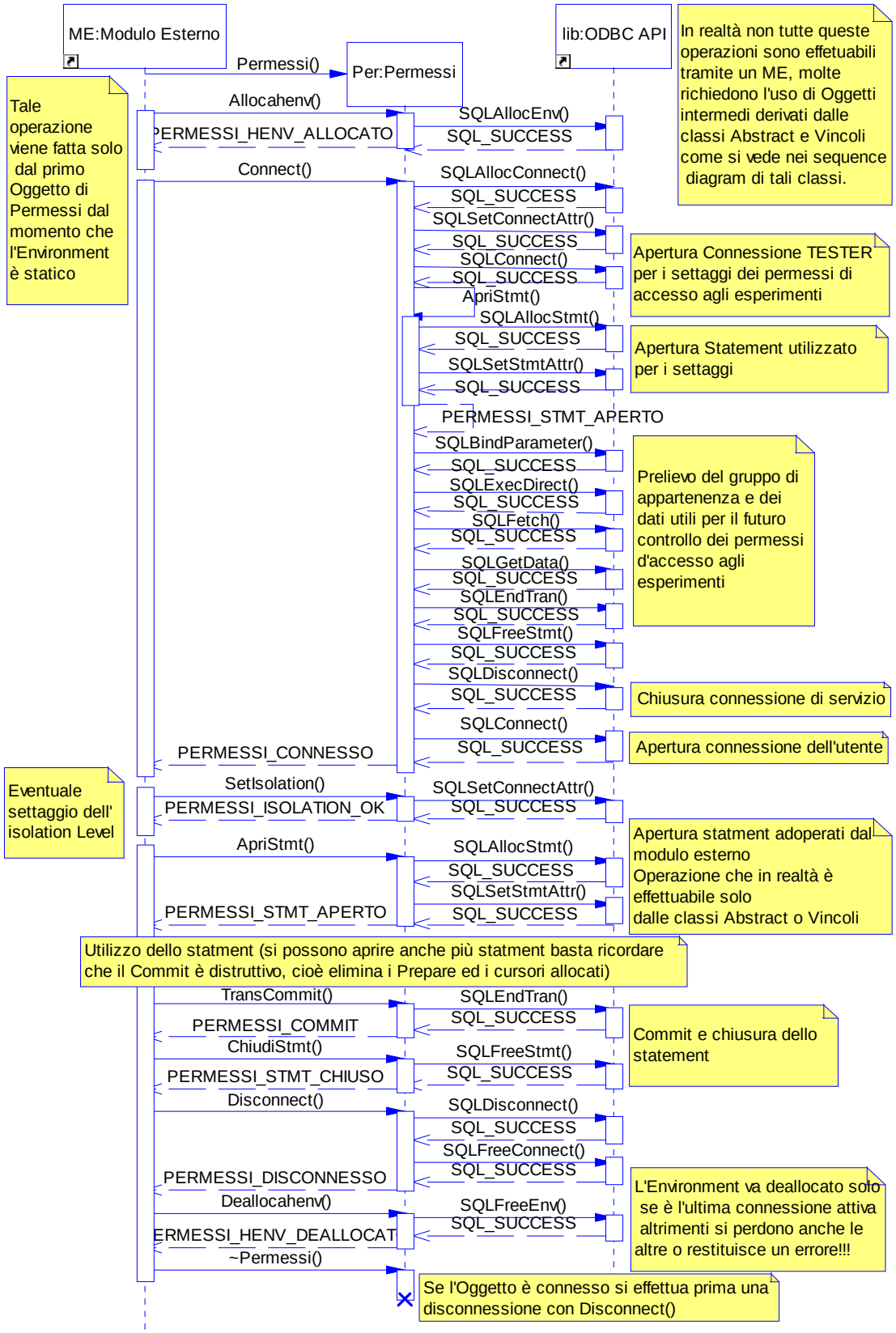


Figura 23: Sequence Diagram di una Transazione

7.5 Classe Abstract

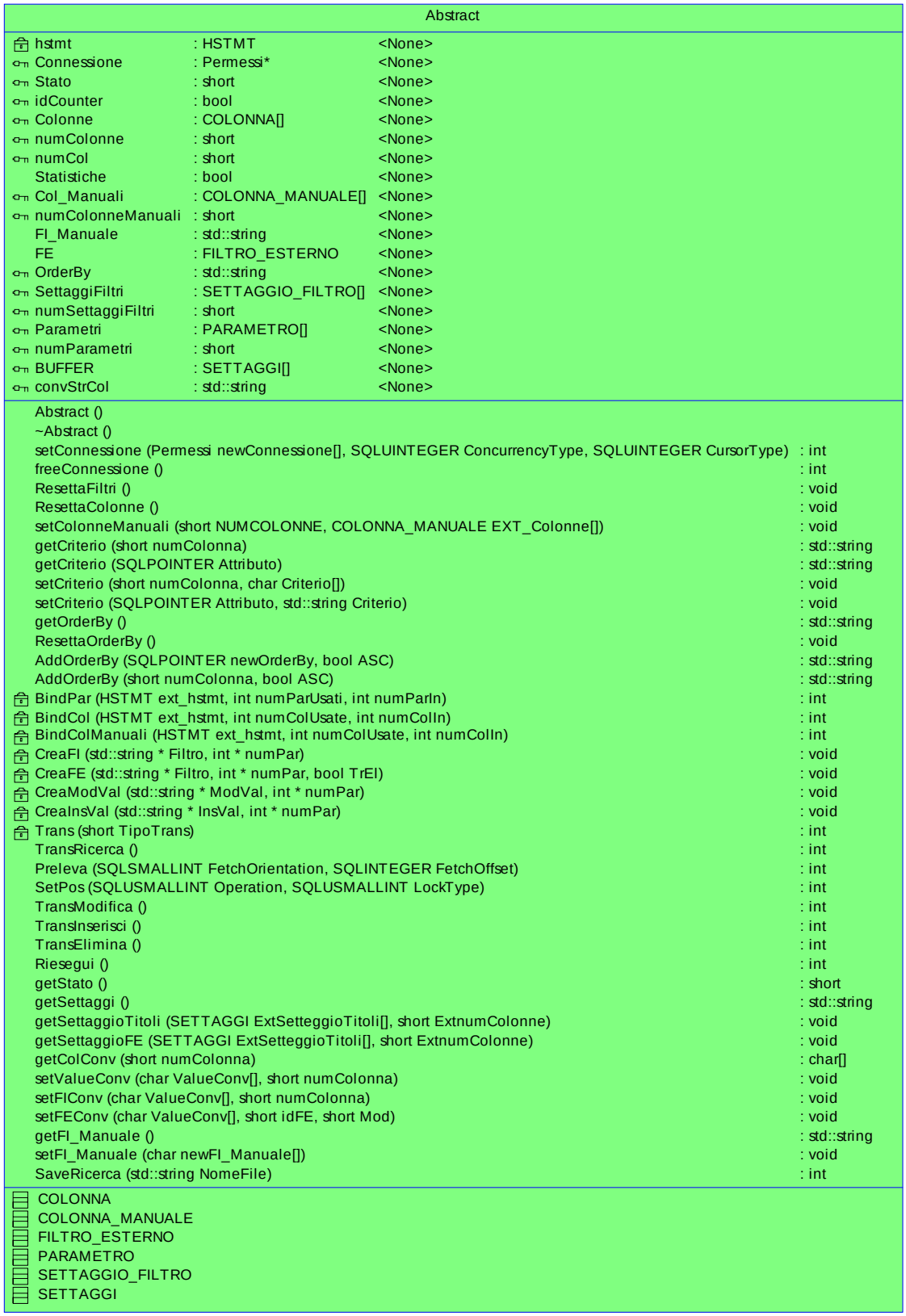


Figura 24: Classe Abstract

La classe **Abstract** definisce le strutture di supporto e le funzionalità necessarie alla realizzazione di classi derivate che siano in grado di gestire i dati presenti in una particolare tabella di un qualsiasi database.

Le funzionalità realizzate non sono vincolate ad un particolare database e l'unica accortenza richiesta per il loro utilizzo tramite le classi derivate è quella di realizzare delle QUERY che permettano di ricavare i dati necessari ai filtraggi esterni richiesti dalla particolare tabella implementata (nel nostro caso le viste del *paragrafo 6.6*).

7.5.1 Il Settaggio

Settando alcuni dei membri di **Abstract** è possibile far sì che la struttura dell'oggetto possa riferirsi ad una particolare tabella. Tali operazioni di settaggio sono possibili solo generando delle classi derivate, poiché i membri adoperati per conservare le informazioni necessarie sono stati dichiarati intenzionalmente privati.
I comandi descritti di seguito sono implementati nei costruttori delle classi derivate da **Abstract** (alcuni esempi sono riportati nel *paragrafo 7.7.1*).

- Prima di tutto è necessario creare un array di ***numSettaggiFiltri*** oggetti del tipo SETTAGGIO_FILTRO da associare al puntatore ***SettaggiFiltri***:

  SettaggiFiltri = new SETTAGGIO_FILTRO[numSettaggiFiltri];

  La classe SETTAGGIO_FILTRO, definita all'interno d'**Abstract,** ha una struttura in grado di contenere i dati necessari per l'individuazione della tabella cui si riferisce l'oggetto e delle QUERY adoperate per i filtraggi esterni:

  Le informazioni riguardanti la tabella sono contenute nell'elemento FE_Assente (si vedano le direttive definite nel paragrafo) dell'array SettaggiFiltri e sono organizzate come di seguito:

Membro	Descrizione
std::string NomeTabella	Nome della tabella di riferimento
std::string NomeColonna	Nomi delle colonne contenenti la chiave primaria (separati da una virgola)
short DimColonna	Indica il numero di colonne presenti in NomeColonna

  Nel caso dei filtraggi esterni le informazioni contenute nell'elemento ***id_el*** (adoperato come si vedrà di seguito per identificare il tipo di filtraggio esterno richiesto) dell'array hanno il significato:

Membro	Descrizione
std::string NomeTabella	Il nome della QUERY adoperata per il filtraggio esterno.
std::string NomeColonna	Nome della colonna cui si riferisce il filtraggio esterno.
short DimColonna	Numero di caratteri della colonna adoperata per l'apposito filtraggio esterno (0 se la colonna di filtraggio non è testuale)

  Le QUERY adoperate per i filtraggi esterni definiti dall'elemento ***id_el*** *dell'array devono possedere come campi sia quelli che si riferiscono alla chiave primaria della tabella implementata (****SettaggiFiltri[FE_Assente].NomeColonna****, può*

*contenere anche più campi come già detto) sia quello che si riferisce al campo indicante il filtro esterno (**SettaggiFiltri[id_el].NomeColonna**).*

- Poi è necessario creare un array di ***numColonne*** oggetti del tipo COLONNA da associare al puntatore ***Colonne***:

  Colonne = new COLONNA[numColonne];

  La classe COLONNA, definita all'interno d'**Abstract,** ha una struttura in grado di contenere i dati necessari per la gestione di una singola colonna di una tabella:

Membro	Descrizione
SQLPOINTER ptr	Puntatore al valore della colonna relativo all'attuale record selezionato o da inserire
SQLPOINTER ptrFI	Puntatore all'attuare valore della colonna adoperato come filtro per le ricerche, le modifiche o le eliminazioni.
SQLSMALLINT tipo	Indica il tipo di dati C puntati da ptr e ptrFI: SQL_C_CHAR, SQL_C_TIMESTAMP, SQL_C_DATE, SQL_C_FLOAT, SQL_C_SLONG, SQL_C_SSHORT
SQLINTEGER dim	Indica la dimensione del campo nel caso sia del tipo SQL_C_CHAR (0 altrimenti)
SQLINTEGER length	Variabile usata per effettuare il bind della colonna tramite l'API ODBC **SQLBindCol**
std::string NomeColonna	Nome della colonna dichiarato nella struttura del database
std::string Criterio	Permette di modificare il Criterio di filtraggio relativo alla colonna

  I primi elementi dell'array ***Colonne*** devono essere quelli che si riferiscono alle colonne della chiave primaria. I puntatori *ptr* e *ptrFI* dichiarati all'interno degli elementi ***Colonne*** si riferiscono ad attributi membri delle classi derivate.

- Una volta definito il numero di colonne si deve creare un array di oggetti PARAMETRO che saranno adoperati per tenere memoria dei parametri adoperati dopo aver eseguito le **API ODBC SQLPrepare()** ed **SQLExecute()**:

  ***Parametri** = new PARAMETRO[(numColonne * 2) +1];*

  Le dimensioni si riferiscono al massimo numero di parametri consentito. I membri del tipo PARAMETRO sono:

Membro	Descrizione
SQLPOINTER ptr	Puntatore al valore del parametro da utilizzare
SQLSMALLINT tipo;	Indica il tipo di dati C puntati da ptr e ptrFI: SQL_C_CHAR, SQL_C_TIMESTAMP, SQL_C_DATE, SQL_C_FLOAT, SQL_C_SLONG, SQL_C_SSHORT
SQLINTEGER length;	Variabile usata per effettuare il bind della colonna tramite l'API ODBC **SQLBindParameter**

- Infine è necessario settare ***idCounter*** in modo da informare i membri della classe **Abstract** adoperanti la tabella se la chiave primaria è di tipo contatore o no.

7.5.2 I Membri

- Il **costruttore** **Abstract()** non fa altro che inizializzare gli attributi della classe, mentre il **distruttore** **~Abstract**() verifica se esiste ancora un collegamento ad un oggetto della classe **Permessi** ed in tal caso procede richiamando il membro **freeConnessione()**.

- Il membro ***hstmt*** contiene lo Statement handle adoperato per gestire la particolare tabella. Il membro è dichiarato privato, in modo che solo le funzioni di tale classe abbiano un completo controllo sulle operazioni adoperabili tramite il passaggio del valore di ***hstmt*** alle **API ODBC**.

  Per poter creare uno statement si usa la funzione membro **setConnessione()** che prima di tutto associa al membro ***Connessione*** il puntatore all'oggetto di tipo **Permessi** passato come parametro, poi tramite **Connessione->ApriStmt()** crea l'***hstmt*** che sarà adoperato dall'oggetto **Abstract** per le sue operazioni, ed infine setta il membro ***Stato*** a Trans_Null e passa il suo puntatore all'oggetto puntato da ***Connessione*** tramite il membro ***Connessione->ptrStati***.

  Per chiudere lo statement si richiama il membro **freeConnessione()** che si preoccupa di richiamare **Connessione->ChiudiStmt()** nel caso in cui tale operazione sia ancora effettuabile (cioè quando l'oggetto di tipo Permessi puntato da ***Connessione*** non è ancora stato distrutto o disconnesso dal database), di settare ***Stato*** col valore PERMESSI_INESISTENTE e di eliminare il puntatore a tale membro tramite ***Connessione->ptrStati***.

  - **setConnessione()** associa l'oggetto ad una connessione e crea un suo statement. I valori consentiti per il parametro ConcurrencyType sono:

    SQL_CONCUR_ROWVER,
    SQL_CONCUR_LOCK,
    SQL_CONCUR_READ_ONLY,
    SQL_CONCUR_VALUES;

    mentre quelli permessi al parametro CursorType:sono:

    SQL_CURSOR_FORWARD_ONLY,
    SQL_CURSOR_STATIC,
    SQL_CURSOR_KEYSET_DRIVEN,
    SQL_CURSOR_DYNAMIC;

    infine i valori di uscita del membro sono PERMESSI_STMT_APERTO, PERMESSI_NON_CONNESSO, PERMESSI_STMT_GIA_ALLOCATO, PERMESSI_ERROR_SQL.

- **freeConnessione()** chiude lo statement dell'oggetto (e' opportuno richiamarlo prima di disconnettere l'oggetto di tipo Permessi puntato da ***Connessione***) i possibili valori di uscita della funzione membro sono definiti come: PERMESSI_STMT_CHIUSO, PERMESSI_ERROR_SQL, PERMESSI_STMT_GIA_DEALLOCATO, PERMESSI_INESISTENTE (l'oggetto Permessi puntato da Connessione è stato distrutto prima o non è mai stato attivato).

Infine il membro **getSettaggi()** restituisce una stringa contenente tutti i settaggi relativi all' Environment, alla Connection ed allo Statement adoperati dall'oggetto in seguito ad una connessione (si adopera **Connessione->getSettaggi(hstmt)**)

- La classe permette di definire dei **settaggi aggiuntivi** in grado di modificare le QUERY richiamate per mezzo dei membri **TransRicerca()**, **TransModifica()**, **TransElimina()** e **TransInserisci()**:
  - **I Criteri**: per effettuare i filtraggi dei dati è necessario decidere il tipo di criterio di confronto adoperato per gli attributi selezionati per il filtraggio. Tali criteri (memorizzati in ***Colonne[i].Criterio***) possono essere settati per mezzo dei due membri **setCriterio()** definiti tramite l'*overloading degli operatori*:
    - Il primo permette di modificare il Criterio di filtraggio relativo ad un elemento dell'array ***Colonne*** passando come parametri di ingresso una stringa contenente il valore del criterio di filtraggio e il numero della colonna cui si riferisce il criterio.
    - Il secondo permette di svincolarsi dalla conoscenza del numero della colonna passando come parametri di ingresso una stringa contenente il valore del criterio di filtraggio e il puntatore del valore della colonna cui si riferisce il criterio (cioè ***Colonne[i].ptr***).

  Nel caso dell'utilizzo del criterio "**LIKE"** il campo che si riferisce a ***Colonne[i].ptrFI*** dovrà contenere una stringa formattata a dovere con simboli di ***escape*** e di <***%***> e <***_***> o <*******> e <***?***> secondo il DBMS adoperato.

  A questi due membri si affiancano i **getCriterio()** che permettono di prelevare i valori di ***Colonne[i].Criterio***; anche in questo caso uno avrà come parametro di ingresso il numero della colonna e l'altro il puntatore al valore della colonna.

- **Le Colonne Manuali:** il membro **TransRicerca()** può restituire anche delle colonne aggiuntive oltre alle colonne della tabella definite in ***Colonne***, ciò è possibile se prima di effettuare la ricerca si adopera il membro **setColonneManuali()**.

  I parametri d'input permettono di passare un array d'oggetti COLONNA_MANUALE, il cui puntatore è memorizzato in ***Col_Manuali*** (*si osservi che, se l'oggetto passato viene distrutto, il puntatore si riferirà ad un'area di memoria non più valida ed il programma in esecuzione darà un errore non previsto*) e le dimensioni in ***numColonneManuali***. È possibile inoltre eliminare le definizioni di tali colonne adottando sempre lo stesso membro ma passando come parametri *numColonne* ≤ 0 e un *EXT_Colonne* qualsiasi.

  L'oggetto COLONNA_MANUALE contiene tre attributi membri:

Membro	Descrizione
std::string DefColonna	Stringa contenente la definizione della colonna (ad esempio *len(TESTO)* oppure *sin(PARAMETRO_CPC)*)
SQLREAL ValoreUscita	Variabile contenente il valore che sarà restituito per la colonna del record corrente di una ricerca
SQLINTEGER length;	Variabile usata per effettuare il bind della colonna tramite l'API ODBC **SQLBindCol** (l'utente non deve modificare tale valore)

- **L'ordinamento dei risultati**: il membro ***OrderBy*** contiene una stringa che definisce il tipo di ordinamento che sarà adoperato dal membro **TransRicerca()**. Il valore iniziale è una stringa nulla cui si può aggiungere un ordinamento ascendente o discendente per una particolare colonna.

  Un ordinamento primario può essere aggiunto per mezzo dei due membri **AddOrderBy()** definiti tramite l'*overloading degli operatori*. Per entrambi il parametro ASC specifica se l'ordinamento è ascendente (true) o discendente (false). I due membri invece differenziano per il secondo parametro che in un caso (numColonna) è il numero della colonna a cui si riferisce il criterio e nell'altro (newOrderBy) il puntatore del valore della colonna a cui si riferisce il criterio (cioè ***Colonne[i].ptr***). Richiamare una seconda volta il membro **AddOrderBy()** permette di definire l'ordinamento secondario, una terza volta quello terziario e cosi via… Per resettare, e quindi ridefinire i vari ordinamenti, sarà necessario richiamare **ResettaOrderBy()**.

Naturalmente esiste anche una funzione che permette di prelevare l'attuale ordinamento settato: **getOrderBy()**.

- **I filtri interni manuali**: il membro ***FI_Manuali*** contiene una stringa che permette di aggiungere un ulteriore filtraggio (ad esempio *len(TESTO) > 10*) alle QUERY eseguite per mezzo dei membri **TransRicerca()**, **TransModifica()** e **TransElimina()**. Il settaggio di tale membro avviene per mezzo di **setFI_Manuale()**, mentre il prelievo del suo attuale valore richiede l'uso di **getFI_Manuale()**.
- **I filtri Esterni**: per settare il filtraggio esterno desiderato occorre adoperare il membro ***FE*** la cui struttura è:

Membro	Descrizione
SQLCHAR ValLet[DIM_Nomi]	Valore del filtro esterno nel caso sia testuale (si adopera un buffer per poterlo bindare).
SQLINTEGER ValNum	Valore del filtro esterno nel caso sia un ID_Counter
int Type	Se ha come valore ***FE_Assente*** significa che non si vuole tener conto d'alcun filtraggio esterno; se si desidera invece considerare un filtraggio esterno sarà necessario utilizzare il particolare ***id_el*** di riferimento (si veda il sottoparagrafo precedente).
short mod	Permette di decidere se l'insieme dei risultati deve appartenere al filtro esterno (ModIn) non appartenervi (ModNotIn) o essere solo nel filtro esterno (ModSolo)

Ulteriori informazioni riguardanti i filtraggi esterni settati si possono ricavare utilizzando il membro **getSettaggioFE()** che permette di prelevare un array di oggetti SETTAGGI i cui membri sono:

Membro	Descrizione
std::string Nome	Restituisce il nome del campo che si riferisce al filtro esterno
int num	Indica il valore di ***FE.Type*** necessario per utilizzare come filtro esterno il campo indicato in Nome.

*(si osservi che tale membro adopera il puntatore **BUFFER** che distrugge e ricrea l'array di SETTAGGI ogni qualvolta si richiami **getSettaggioFE()** o **getSettaggioTitoli()**).*

Una volta conosciuti i possibili valori per ***FE.Type*** ed i relativi campi a cui si riferiscono è possibile settare il Filtro Esterno adoperando **setFEConv()**. Tale membro permette di dare in ingresso una stringa che automaticamente,

secondo il parametro idFE (***id_el***) passato, sarà trasformata in ***FE.ValLet*** o ***FE.ValNum***. Inoltre le modifiche effettuate tramite i parametri idFE e/o Mod possono essere ignorate ponendoli uguali a ***–1***.

- **Modalità Statistiche**: si osservi che, settando a true il membro ***Statistiche***, è possibile far sì che in uscita si prelevino solo i dati relativi alla chiave primaria della tabella ed alle colonne manuali. Tale opzione permette di risparmiare tempo nel prelievo dei dati quando all'utente servono solo i valori ricavati tramite calcoli sugli attributi dei record.

- Come già anticipato i membri **TransRicerca()**, **TransModifica()**, **TransElimina()** e **TransInserisci()** permettono di effettuare le rispettive QUERY, ma queste richiedono, a seconda della tabella definita tramite i settaggi già presentati, un certo numero di **parametri**:
  - **I parametri di input** possono essere passati o per mezzo delle variabili associate a ***Colonne[i].ptr*** (adoperate come valori di ingresso per l'inserzione e le modifiche) ed a ***Colonne[i].ptrFI*** (adoperate come valori di ingresso per i filtri relativi agli elementi ricercati, da modificare, o da eliminare) o per mezzo dei membri **setFIConv()** e **setValueConv()** che convertono la stringa data in ingresso (ValueConv) nel valore della colonna individuata tramite numColonna e poi lo memorizzano nell'indirizzo puntato da ***Colonne[i].ptr*** o ***Colonne[i].ptrFI*** a seconda dei casi.
    *Si adoperano i valori **strNull** o **numNull** per indicare che un campo è nullo, mentre si adoperano **strNoUtil** e **numNoUtil** per indicare che il campo, se possibile, non deve essere considerato nella QUERY.*
  - **I parametri di output** possono essere passati per mezzo delle variabili associate a ***Colonne[i].ptr*** (adoperate come valori di ritorno delle colonne dell'attuale record selezionato in seguito ad una ricerca) e dei valori dei campi **ValoreUscita** dell'array di oggetti COLONNA_MANUALE passati all'oggetto per mezzo di **setColonneManuali()**.
    Ulteriori informazioni riguardanti i parametri di uscita si possono ricavare utilizzando il membro **getSettaggioTitoli()** che permette di prelevare un array di oggetti SETTAGGI i cui membri sono:

Membro	Descrizione
std::string Nome	Restituisce il nome del campo della tabella.
int num	Indica il massimo numero di caratteri del campo

*(si osservi che tale membro adopera il puntatore **BUFFER** che distrugge e ricrea l'array di SETTAGGI ogni qualvolta si richiami **getSettaggioFE()** o **getSettaggioTitoli()**).*

Una volta conosciuto il numero di colonne dei record ricercati ed i loro titoli è possibile prelevare i valori dei campi di un record adoperando il membro **getColConv()** che restituisce una stringa contenente il valore della colonna individuata dal parametro di ingresso numColonna (si osservi che nel caso il campo non sia testuale tale membro si preoccupa di convertirlo in una stringa). **getColConv()** restituisce in uscita un puntatore a *char* il cui valore di riferimento è il puntatore al buffer di caratteri del membro ***convStrCol***.

- I membri **TransRicerca()**, **TransModifica()**, **TransElimina()** e **TransInserisci()** richiamano il membro privato **Trans()** a cui richiedono di eseguire tramite il parametro di ingresso la particolare QUERY di interesse. **Trans()** si preoccupa di verificare il parametro di ingresso Trans_Type, nel caso che la QUERY abbia buon esito il valore sarà associato allo ***Stato***:
  - Trans_Inserisci (inserisce un elemento, i cui valori sono Colonne[i].ptr, si noti che alcune colonne non saranno usate)
  - Trans_Ricerca (effettua una ricerca in base agli attributi che si riferiscono a Colonne[i].ptrFI e Colonne[i].Criterio)
  - Trans_Elimina (elimina degli elementi usando come filtro gli attributi che si riferiscono a Colonne[i].ptrFI e Colonne[i].Criterio)
  - Trans_Modifica (modifica, per mezzo degli stessi attributi adoperati da **TransIserisci()**, gli oggetti definiti per mezzo degli attributi che si riferiscono a Colonne[i].ptrFI e Colonne[i].Criterio)

  Trans() crea una stringa di comando da passare all' **API ODBC SQLPrepare()** tramite l'utilizzo dei membri **CreaFI()** (che permette di inserire i criteri di ricerca dopo un where), **CreaFE()** (che permette di inserire, nelle stringhe delle QUERY che richiedono filtraggi esterni, un criterio di ricerca adoperante la clausola **IN** o **EXISTS** con i settaggi relativi a ***SettaggiFiltri[FE.Type]***), **CreaModVal()** (che inserisce i parametri richiesti, da una QUERY di modifica, riguardanti i campi da modificare) e **CreaInsVal()** (che inserisce i parametri dei campi del un nuovo elemento). Tutti e quattro i membri adoperati utilizzano il puntatore ***Parametri*** per gestire i nuovi parametri individuati (un valore, puntato da Colonne[i].ptrFI oppure

Colonne[i].ptro, sarà considerato un parametro se diverso da **numNoUtil** o **strNoUtil**).

Dopo aver preparato la QUERY tramite la stringa costruita si procede prima a richiamare il membro **BindPar()** e poi l'**API SQLExecute()**. Dopo se l'operazione richiamata è **TransRicerca()** si procede anche con l'eseguire i membri **BindCol()** e **BindColManuali()**.

- **BindPar()**: permette di effettuare il Bind dei parametri; numParIn è la prima colonna da iniziare ad associare ai parametri di uscita, numParUsati è il numero di parametri usati dallo statement dell'oggetto attuale. I valori restituiti saranno: ERRORE_SQL oppure NO_ERRORE.
- **BindCol()**: permette di effettuare il Bind delle colonne della tabella desiderate in uscita; numColIn è la prima colonna da iniziare ad associare ai parametri di uscita; numColUsate è il numero di colonne usate dallo statement dell'oggetto attuale (serve nel caso di QUERY con ***Statistiche*** uguale a true). I valori restituiti saranno: ERRORE_SQL oppure NO_ERRORE.
- **BindColManuali()**: permette di effettuare il Bind delle colonne manuali definite dall'utente; numColIn è la prima colonna da iniziare ad associare ai parametri di uscita; numColUsate è il numero di colonne manuali usate. I valori restituiti saranno: ERRORE_SQL oppure NO_ERRORE.

Inoltre alla fine dell'esecuzione del membro Trans() si procede a memorizzare il numero di parametri di input individuati in ***numParametri*** ed il numero di colonne di output in ***numCol*** (valori che saranno adoperati dai membri **Preleva()** e **Riesegui()**).

- **Trans()**:I valori restituiti saranno: ERRORE_SQL, NO_ERRORE oppure PERMESSI_INESISTENTE (nel caso non sia attiva una connessione).

Tra i quattro membri che utilizzano **Trans()** solo **TransRicerca()** ha un valore d'uscita in più: NO_DATO (che dipende dal membro **Preleva()** richiamato dopo **Trans(Trans_Ricerca)** per prelevare i dati del primo elemento della ricerca effettuata).

- Una volta effettuata una Ricerca è possibile prelevare i dati di un qualsiasi Record individuato tramite il membro **Preleva()** ed inoltre è possibile adoperare il membro **SetPos()** per effettuare inserimenti, refresh, cancellazioni o modifiche che si riferiscano al record attivo del recordset individuato.

- **Preleva()**: permette di prelevare i dati e inserirli negli attributi puntati da ***Colonne[i].ptr*** ed in quelli individuati da ***Col_Manuali[i].ValoreUscita***. I valori di *FetchOrientation* sono:

  SQL_FETCH_NEXT,
  SQL_FETCH_PRIOR,
  SQL_FETCH_FIRST,
  SQL_FETCH_LAST,
  SQL_FETCH_ABSOLUTE,
  SQL_FETCH_RELATIVE

  Il valore di *FetchOffset* è il numero della riga nel caso di ABSOLUTE ed il numero di righe fetched nel caso di RELATIVE.

  I valori di ritorno sono: ERRORE_SQL, NO_ERRORE, NO_DATO oppure NO_ACCESS (nel caso in cui lo ***Stato*** sia diverso da Trans_Ricerca).

- **SetPos()**: permette di modificare, eliminare o rileggere il record attualmente prelevato (adoperano l'**API SQLSetPos**), inoltre è possibile anche aggiungere nuovi record (adopera l'**API SQLBulkOperaction**).

  I valori validi per *Operation* sono: *SQL_REFRESH, SQL_UPDATE, SQL_DELETE, SQL_ADD*.

  I valori validi per *LockType* (ignorato se *SQL_ADD*), che specifica il tipo di lock effettuato dopo aver eseguito l'operazione indicata in *Operation*, sono: SQL_LOCK_UNLOCK, SQL_LOCK_NO_CHANGE (*sembrerebbe l'unico funzionante con Access!*), SQL_LOCK_EXCLUSIVE.

  I valori di ritorno sono: ERRORE_SQL, NO_ERRORE, e NO_ACCESS (nel caso in cui lo ***Stato*** sia diverso da Trans_Ricerca).

Il membro **SaveRicerca()** invece permette di salvare in un file, avente il nome indicato nel parametro di ingresso, l'elenco dei record ricercati secondo il formalismo: <valore campo> <\t> <valore campo> <\t> … <valore campo> <\n>

- **SaveRicerca()**: I valori di ritorno sono: ERRORE_SQL, NO_ERRORE, NO_DATO oppure NO_ACCESS (nel caso in cui lo ***Stato*** sia diverso da Trans_Ricerca o ci sia un errore nella creazione del file).

- Un membro molto utile è **Riesegui()** che permette di sfruttare l'ultimo **comando SQLPrepare()** (che si riferisce ad uno tra i membri **TransRicerca()**, **TransInserisci()**, **TransModifica()** e **TransElimina()**) per rieseguire una QUERY che adoperi i suoi stessi parametri (in pratica si riesegue la QUERY preparata in precedenza considerando gli attributi puntati da ***Colonne[i].ptr*** e ***Colonne[i].ptrFI***

ed i valori ***FE.ValNum*** o ***FE.ValLet*** che in precedenza non avevano come valore ***numNoUtil*** o ***strNoUtil***; inoltre si osservi che se un filtro aveva valore ***NumNull*** o ***strNull*** questo sarà sempre conservato tale) con l'accortezza che non abbiano valori *NoUtil* e che non siano *nulli* nel caso si riferiscano a ***Colonne[i].ptrFI***. Inoltre se la QUERY è stata dichiarata Statistica (***Statistiche*** = true) non sarà possibile aggiungere le colonne non considerate e se sono presenti delle colonne manuali o dei filtri interni manuali questi non potranno essere modificati.

  - **Riesegui()**: I valori di ritorno sono: ERRORE_SQL, NO_ERRORE, NO_DATO (restituito solo nel caso di ripetizione di **TransRicerca()**), PERMESSI_INESISTENTE (se non è presente una connessione), PERMESSI_EndTrans (se vi è stata la chiusura della transazione e non è possibile ripetere l'operazione).

- Infine vi sono i due membri **ResettaColonne()** (che resetta tutti i campi puntati da ***Colonne[i].ptr*** portandoli a *NoUtil*, riporta a false ***Statistiche*** e cancella il puntatore ***Colonne_Manuali***) e **ResettaFiltri()** (che resetta tutti i campi puntati da ***Colonne[i].ptrFI*** portandoli a *NoUtil*, pone ***Colonne[i].Criterio*** a `"="` e setta ***FE.Type*** = FE_Assente).

7.5.3 Le Direttive

In tale sottoparagrafo si riportano le direttive relative ai valori utilizzati nel paragrafo precedente.

```
//Valori per disattivare i campi
//Per i campi DATA e TIMESTAMP tale valore è associato a year
#define strNoUtil "\a"
#define numNoUtil -1
//Valori per i NULL
//Per i campi DATA e TIMESTAMP tale valore è associato a year
#define strNull ""
#define numNull -2

//Valori di Stato
//#define PERMESSI_INESISTENTE
#define Trans_Null   PERMESSI_EndTrans
#define Trans_Inserisci 1
#define Trans_Ricerca 2
#define Trans_Elimina 3
#define Trans_Modifica 4

//Valore relativo alla mancanza di filtro esterno
#define FE_Assente 0
//Modalità relative all'uso del filtro esterno
#define ModIn     0
#define ModNotIn  1
#define ModSolo   2

//Dimensioni dei campi testo
//DIM_Nomi_Corti è adoperato anche per tutti i Type
#define DIM_Nomi_Corti 15
#define DIM_Nomi 30
#define DIM_Nomi_Medi 60
#define DIM_Nomi_Lunghi 90
#define DIM_Spiegazioni_Brevi 160
#define DIM_Spiegazioni 255
//(Sperimentali)
#define DIM_Note_TITOLO_DOCUMENTO          450
#define DIM_Note_DESCRIZIONE_QUERY         1000
#define DIM_Note       50000
#define DIM_Note_TESTO              170000
```

7.5.4 Sequence Diagram delle Funzionalità Principali

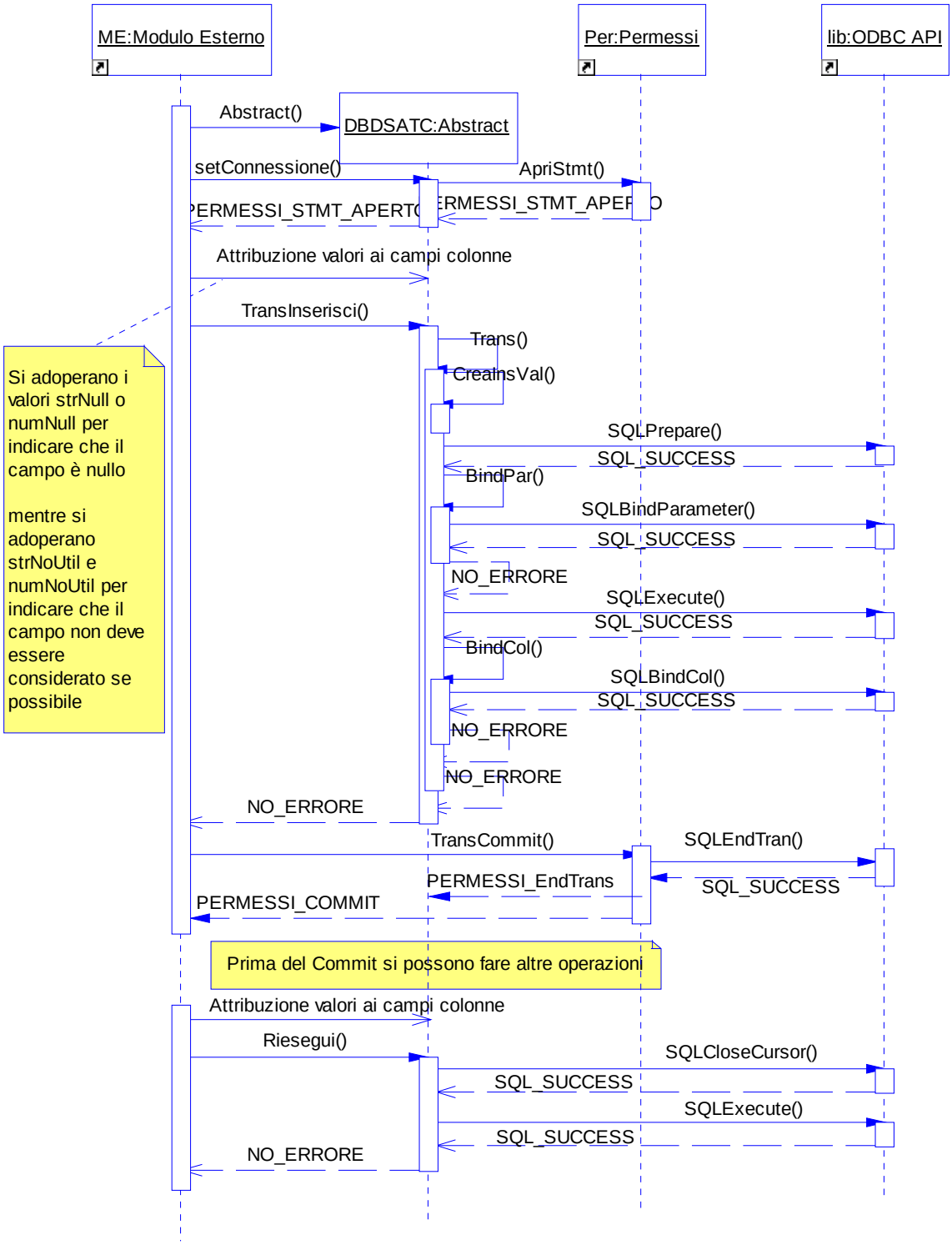


Figura 25: Sequence Diagram di un Inserzione

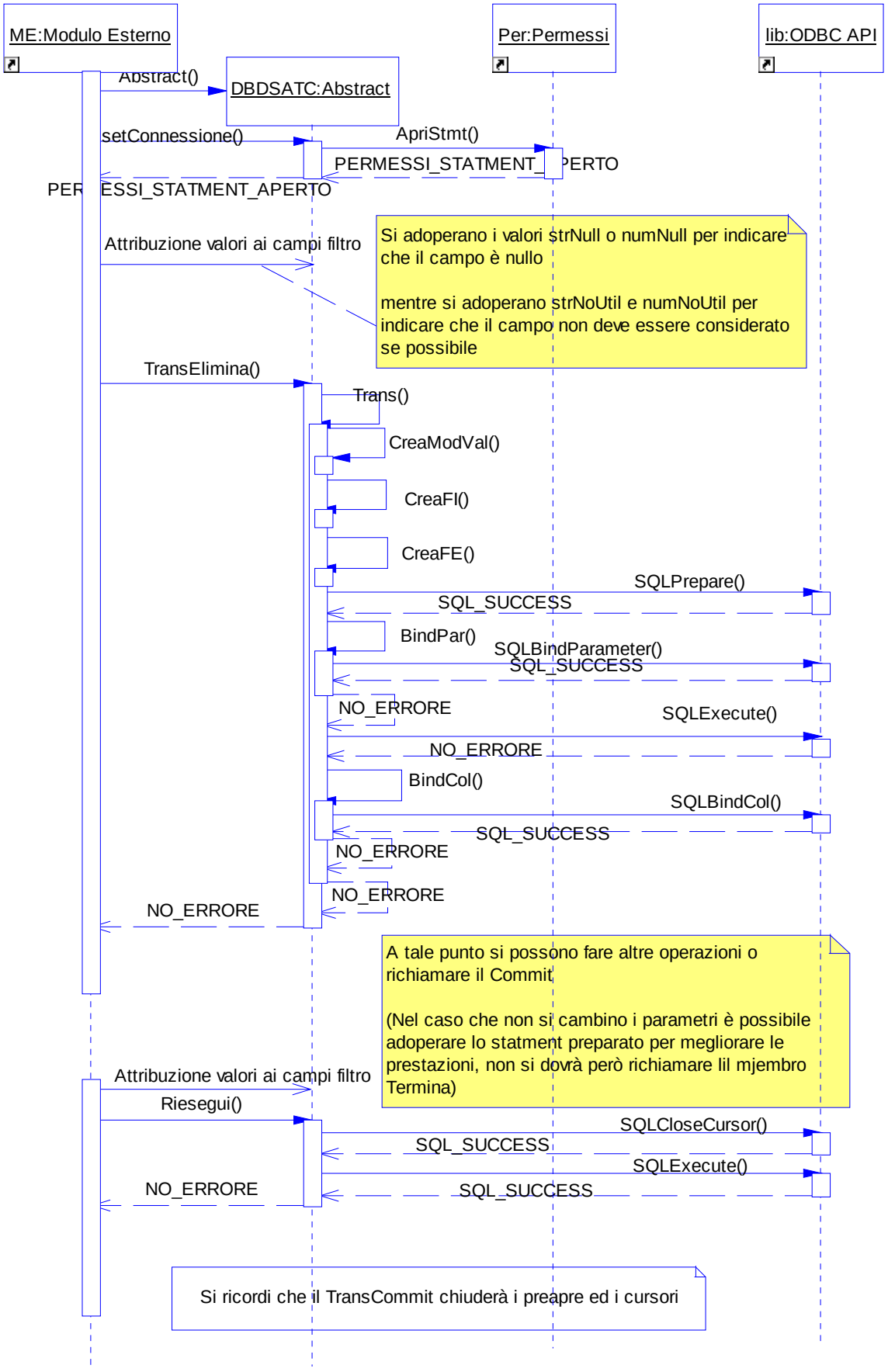


Figura 26: Sequence Diagram di una Cancellazione

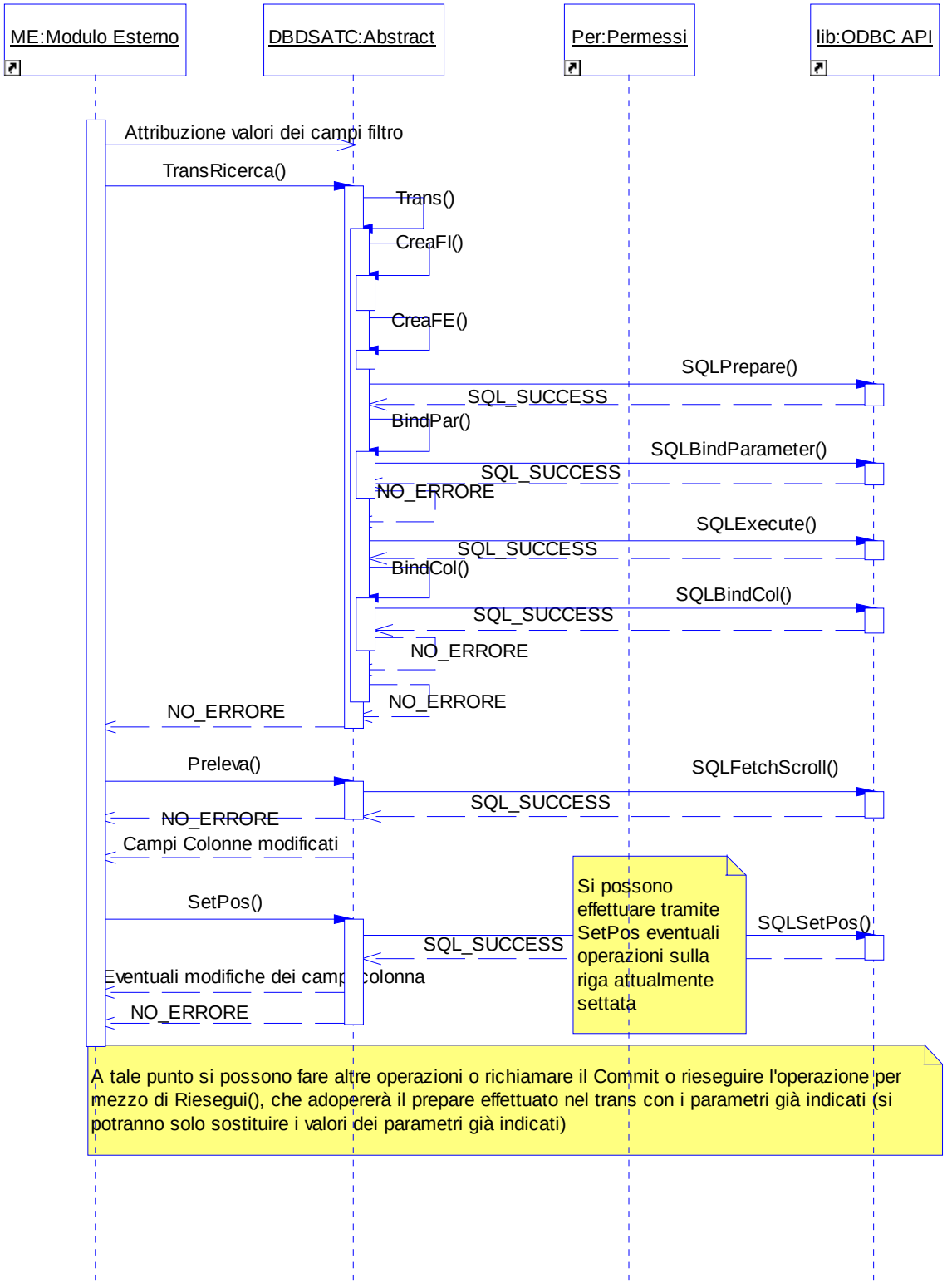


Figura 27: Sequence Diagram di una Ricerca

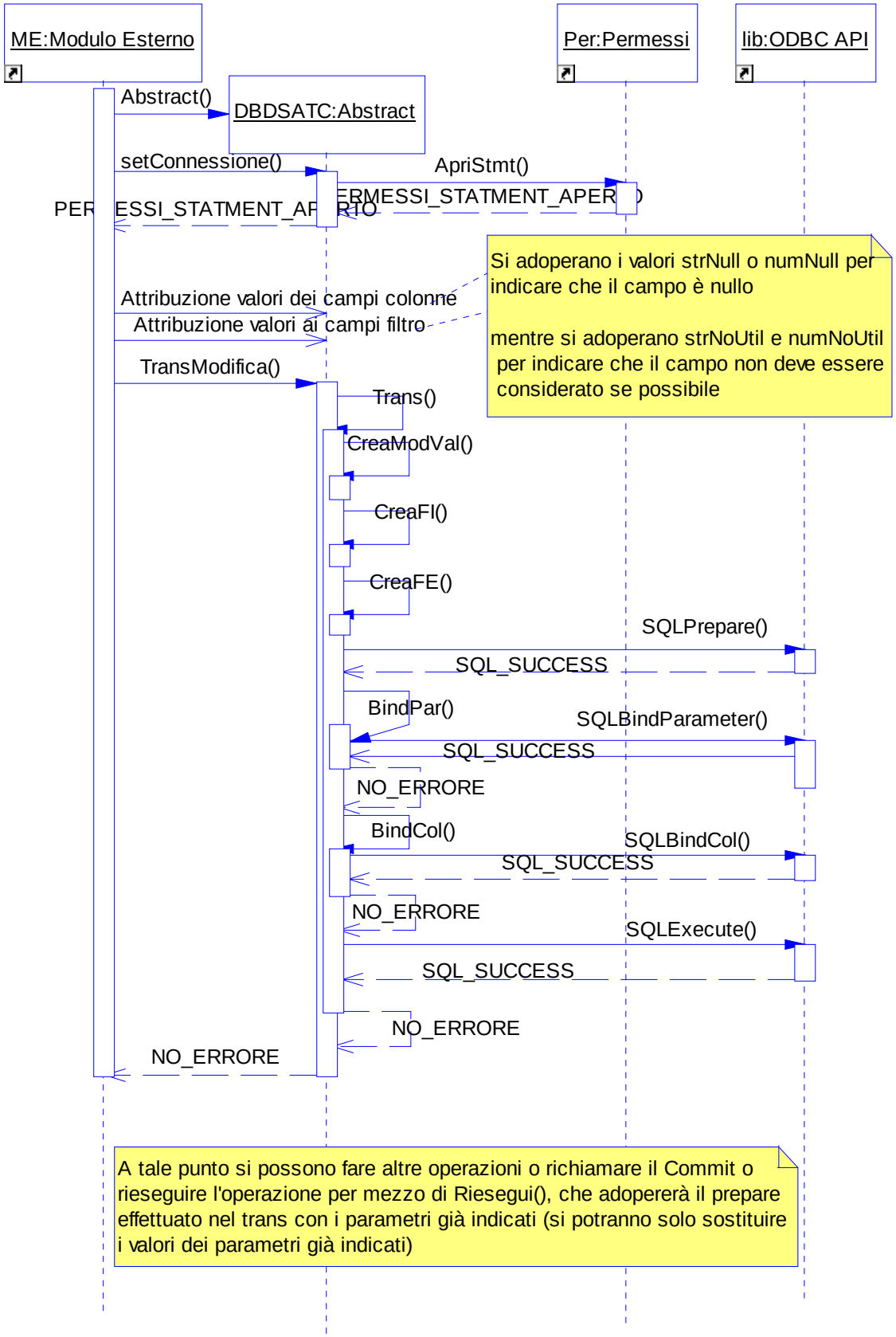


Figura 28: Sequence Diagram di una Modifica

7.6 Classe Vincoli

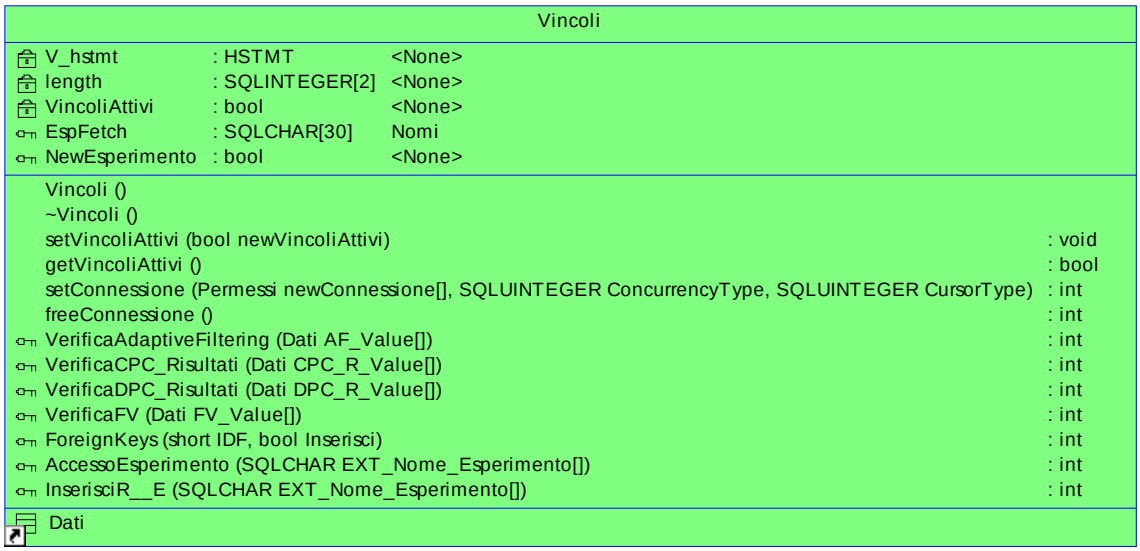


Figura 29: Classe Vincoli

Tale classe è realizzata per implementare tutti quei vincoli che non è stato possibile gestire tramite il **DBMS Access 2000**. A differenza della classe **Abstract** i membri realizzati in tale caso sono molto dipendenti dal database.

Vincoli sarà adoperato solo per realizzare classi che devono gestire tabelle richiedenti particolari vincoli, in caso contrario basterà adoperare **Abstract**..

Procediamo con la descrizione dei nuovi membri, tenendo conto del fatto che a tale classe appartengono anche di tutti i membri d'**Abstract**, da cui deriva:

- Il **costruttore** **Vincoli()** non fa altro che inizializzare i nuovi attributi della classe, mentre il **distruttore** **~Vincoli**() verifica se esiste ancora un collegamento ad un oggetto della classe **Permessi** ed in tal caso procede richiamando il membro **freeConnessione()**.

- Il membro ***V_hstmt*** contiene lo Statement handle utilizzato dalla classe per eseguire i comandi relativi al particolare vincolo (che quando possibile si gestisce tramite l'**API SQLPrepare()**). Il membro è dichiarato privato, in modo che (così come avviene per ***hstmt***) solo le funzioni di tale classe abbiano un completo controllo sulle operazioni adoperabili tramite il passaggio del valore di ***V_hstmt*** alle **API ODBC**.

  Per poter creare lo statement si adopera la funzione membro **setConnessione()** che, tramite **Connessione->ApriStmt(V_hstmt)**, crea il ***V_hstmt*** che sarà adoperato

dall'oggetto **Vincoli** per gestire i vincoli e richiama **Abstract::setConnessione()** per collegare l'oggetto base **Abstract**.

Per chiudere lo statement si richiama il membro **freeConnessione()** che si preoccupa di richiamare **Connessione->ChiudiStmt(V_hstmt)** e poi **Abstract::freeConnessione()**.

- **setConnessione()** e **freeConnessione()** utilizzano gli stessi parametri e gli stessi valori di uscita dei relativi membri di **Abstract**.

- Il membro ***VincoliAttivi*** permette di memorizzare se la verifica dei vincoli è attiva o no (non tutti i vincoli possono essere disattivati). Per modificare tale valore si adopera la funzione membro **setVincoliAttivi()** che può essere usata solo nel caso che lo stato dell'oggetto puntato da ***Connessione*** sia DBA (in caso contrario non apporta nessuna modifica). Invece per prelevare il valore del membro si adopera **getVincoliAttivi()**.

- **ForeignKeys()**: Permette di inserire ed eliminare le ForeignKeys così come richiesto dal paragrafo 6.7 quando si parla dell'impossibilità di spostare la verifica dei vincoli di ForeignKeys sul commit. I parametri di ingresso sono:

  IDF che permette di individuare una particolare ForeignKeys:

  1 per FK_D__S_D
  2 per FK_CPC_Q
  3 per FK_DPC_C
  4 per FK_T__DS_T
  5 per FK_S__DS_S
  6 per FK_D__B_D
  7 per FK_M__E_M

  Inserisci che se posto a true indicherà al membro di inserire la ForeignKeys, altrimenti di eliminarla.

  I valori di ritorno sono: NO_ERRORE, ERRORE_SQL, NO_ACCESS (se l'operazione non è stata richiesta da un **DBA**) e PERMESSI_INESISTENTE (se è stato distrutto o scollegato l'oggetto puntato da ***Connessione***)

- Il membro ***length*** definisce un array di due elementi, utilizzati per effettuare i bind dei parametri necesarri all'interno di tutte le seguenti funzioni membro.

- Le funzioni membro **VerificaAdaptiveFiltering()**, **VerificaCPC_Risultati()**, **VerificaDPC_Risultati()** e **VerificaFV()** permettono di verificare se un particolare elemento è inseribile in una tabella, in pratica implementano quanto richiesto dai *paragrafi 6.4.2* e *6.5.1*. Tali membri hanno gli stessi parametri e valori di ritorno:
  - In ingresso si ha un puntatore ad un oggetto la cui struttura permette di memorizzare i valori delle colonne appartenenti alla tabella da verificare (serve per effettuare i bind);
  - I valori di ritorno sono:

    ERRORE_SQL,

    NO_ACCESS (il vincolo non è verificato),

    PERMESSI_ABILITATI (il vincolo è verificato),

    PERMESSI_INESISTENTE (se non è attiva una connessione)

  Tali membri gestiscono il riciclo dell'**API ODBC SQLPrepare()** effettuando delle verifiche sul membro ***Stato***.

  Tutti questi membri utilizzano come parametro di ingresso un oggetto della classe **Dati** il cui modello è contenuto nella classe **Vincoli**. I membri dell'oggetto hanno significato diverso secondo il particolare membro che li adopera (l'utilizzo di tale parametro sarà spiegato nelle diverse classi derivate):

Membro	Descrizione
SQLCHAR NOME[DIM_Nomi]	Stringa adoperata come buffer per un campo testuale
SQLINTEGER ID_Primo	Valore adoperato come buffer per un primo ID_Generico
SQLINTEGER ID_Secondo	Valore adoperato come buffer per un secondo ID_Generico

- Il membro **InserisciR__E()** permette di inserire una tupla della tabella **R__E** (relazione Ricercatore Esperimento) in seguito alla creazione di un nuovo esperimento. Tale funzionalità è richiesta poiché i vincoli impostati sulla classe **R__E** non permettono di creare tuple relative ad un particolare esperimento ad utenti che non siano già in relazione con l'esperimento (in pratica solo chi è associato ad un esperimento può decidere se un altro utente può essere abilitato a modificarne i dati). Tale membro adopera il flag ***NewEsperimento*** ed il valore di ***Stato*** per gestire il riciclo dell'**API ODBC SQLPrepare()**.

- **AccessoEsperimento()**: Verifica se l'utente attivo ha accesso alle modifiche su un record di una tabella contenente i dati relativi ad un particolare Esperimento. I valori di uscita sono definiti come: PERMESSI_NON_ABILITATI, ERROR_SQL, PERMESSI_ABILITATI, e PERMESSI_INESISTENTE (se ***Connessione*** punta ad

un oggetto non più connesso o che è stato distrutto). Tale membro adopera il flag ***NewEsperimento*** ed il valore di ***Stato*** per gestire il riciclo dell'**API ODBC SQLPrepare()**.

- Infine si ha il membro ***EspFetch*** che sarà adoperato per conservare le informazioni prelevate da **ClasseDerivata::Preleva()** e necessarie a **ClasseDerivata::SetPos()** per verificare i permessi sull'ultimo record prelevato.

7.7 Classe DBDSATC

La classe è organizzata in modo da contenere i modelli di tutte le classi che sono state derivare da **Abstract** e **Vincoli** per implementare le entità e le relazioni del database. Oltre a contenere tali classi essa permette, come già detto, di dichiarare degli oggetti **DBDSATC** che, all'occorrenza, possono essere settati per definire una qualsiasi delle classi modello contenute. Inizieremo quindi a descrivere le classi derivate da **Abstract** e **Vincoli** per poi concludere con la classe **DBDSATC**.
Nella maggior parte dei casi, esclusi i costruttori ed i distruttori, i membri delle classi derivate sono gli stessi di quella base, a volte però può capitare di doverli ridefinire per includere al loro interno la gestione di particolarità o di vincoli, oppure di doverne aggiungerne di nuovi.

7.7.1 Implementazione dei Benchmarks

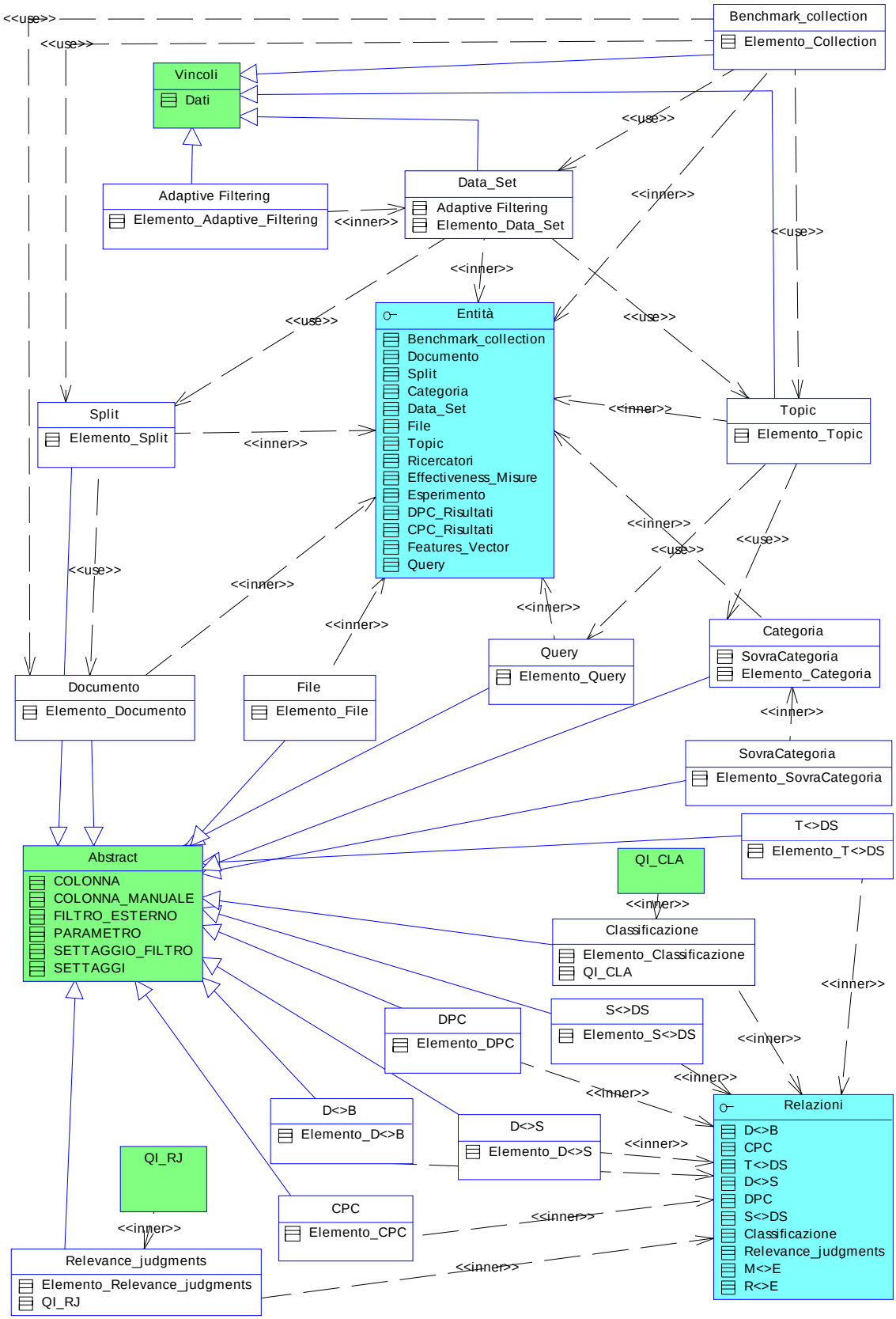
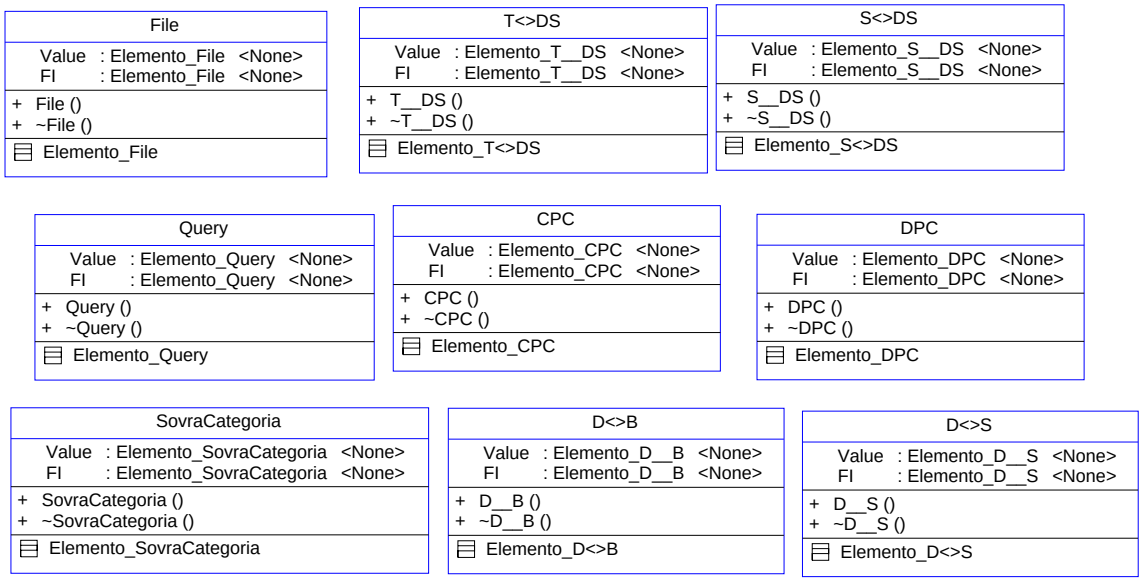


Figura 30: Modello Object-Oriented (1° Parte)

∅ Le classi derivate più semplici sono quelle riportate di seguito che non fanno altro che definire ognuna due membri (relativi alle classi dei loro elementi), uno utilizzato per i filtri interni i cui puntatori agli attributi saranno passati ai valori ***Colonne[i].ptrFI*** di **Abstract**, e l'altro per i campi di input (o di output nel caso di ricerche) i cui puntatori agli attributi saranno passati ai valori ***Colonne[i].ptr***.



Di seguito, per mostrare un tipico settaggio, si utilizza com'esempio indicativo il **costruttore** della classe **Query**:

```
//Valori relativi ai filtri esterni
numSettaggiFiltri = 8;
SettaggiFiltri = new SETTAGGIO_FILTRO[numSettaggiFiltri];
SettaggiFiltri[FE_Assente].NomeTabella = "EXT_QUERY";
SettaggiFiltri[FE_Assente].NomeColonna = "ID_QUERY";
SettaggiFiltri[FE_Assente].DimColonna = 1;
SettaggiFiltri[QRY_da_Documento].NomeTabella = "EXT_RELEVANCE_JUDGMENTS";
SettaggiFiltri[QRY_da_Documento].NomeColonna =
  "EXT_RELEVANCE_JUDGMENTS.ID_DOCUMENTO";
SettaggiFiltri[QRY_da_Documento].DimColonna = 0;
SettaggiFiltri[QRY_da_Categoria].NomeTabella = "FE_Q__C";
SettaggiFiltri[QRY_da_Categoria].NomeColonna = "FE_Q__C.ID_CATEGORIA";
SettaggiFiltri[QRY_da_Categoria].DimColonna = 0;
SettaggiFiltri[QRY_da_Topic].NomeTabella = "EXT_CPC";
SettaggiFiltri[QRY_da_Topic].NomeColonna = "EXT_CPC.NOME_TOPIC";
SettaggiFiltri[QRY_da_Topic].DimColonna = DIM_Nomi;
SettaggiFiltri[QRY_da_Split].NomeTabella = "FE_Q__S";
SettaggiFiltri[QRY_da_Split].NomeColonna = "FE_Q__S.NOME_SPLIT";
SettaggiFiltri[QRY_da_Split].DimColonna = DIM_Nomi;
SettaggiFiltri[QRY_da_Esperimento].NomeTabella = "FE_Q__E";
SettaggiFiltri[QRY_da_Esperimento].NomeColonna =
  "FE_Q__E.NOME_ESPERIMENTO";
SettaggiFiltri[QRY_da_Esperimento].DimColonna = DIM_Nomi;
SettaggiFiltri[QRY_da_DataSet].NomeTabella = "FE_Q__DS";
```

```
SettaggiFiltri[QRY_da_DataSet].NomeColonna = "FE_Q__DS.NOME_DATA_SET";
SettaggiFiltri[QRY_da_DataSet].DimColonna = DIM_Nomi;
SettaggiFiltri[QRY_da_Collection].NomeTabella = "FE_Q__B";
SettaggiFiltri[QRY_da_Collection].NomeColonna = "FE_Q__B.NOME_COLLECTION";
SettaggiFiltri[QRY_da_Collection].DimColonna = DIM_Nomi;

//Creazione dati colonne
numColonne=3;
Colonne = new COLONNA[numColonne];

idCounter = true;
Colonne[0].ptr = &Value.ID_QUERY;
Colonne[0].ptrFI =&FI.ID_QUERY;
Colonne[0].tipo = SQL_C_SLONG;
Colonne[0].dim = 0;
Colonne[0].NomeColonna = "ID_QUERY";
Colonne[1].ptr = &Value.TITOLO_QUERY;
Colonne[1].ptrFI =&FI.TITOLO_QUERY;
Colonne[1].tipo = SQL_C_CHAR;
Colonne[1].dim = sizeof(Value.TITOLO_QUERY);
Colonne[1].NomeColonna = "TITOLO_QUERY";
Colonne[2].ptr = &Value.DESCRIZIONE_QUERY;
Colonne[2].ptrFI =&FI.DESCRIZIONE_QUERY;
Colonne[2].tipo = SQL_C_CHAR;
Colonne[2].dim = sizeof(Value.DESCRIZIONE_QUERY);
Colonne[2].NomeColonna = "DESCRIZIONE_QUERY";

//Creazione puntatore ai Parametri
Parametri = new PARAMETRO[(numColonne * 2) +1];

ResettaFiltri();
ResettaColonne();
numCol = numColonne;          //individua il numero di colonne attualmente adoperate
```

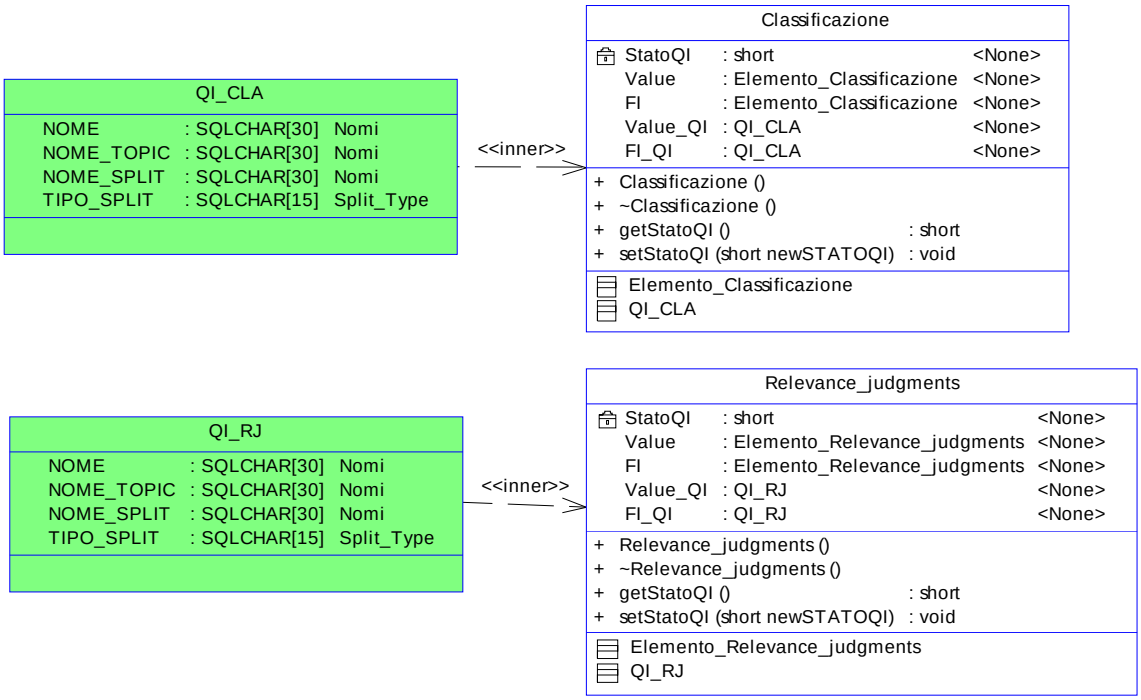
Per quanto riguarda gli indici di ***SettaggiFiltri*** si è deciso di utilizzare dei parametri definiti per mezzo di direttive (e che naturalmente avranno valori tra 1 e ***numSettaggiFiltri*** –1).

I **distruttori** invece sono tutti uguali e non fanno altro che deallocare gli array creati dai costruttori:

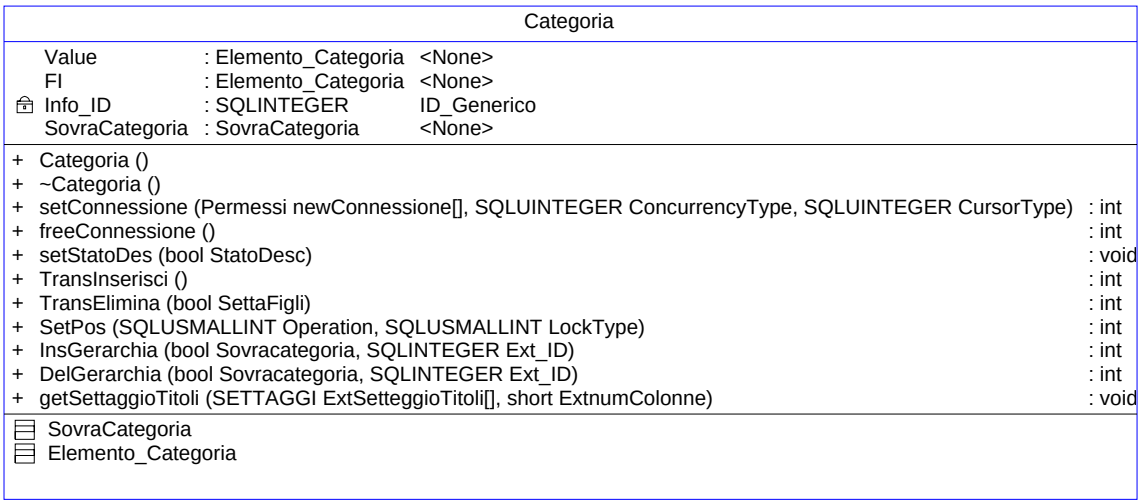
```
delete[] SettaggiFiltri;
delete[] Parametri;
delete[] Colonne;
```

∅ Le classi **Classificazione** e **Relevance_Judgments** sono simili a quelle presentate fin ora se non per la particolarità di poter funzionare in tre diverse modalità settabili per mezzo del membro **setStatoQI()** (lo stato viene conservato nel membro ***StatoQI*** che inizialmente è uguale a StatoQI_Normale). I possibili ingressi sono:

StatoQI_Normale, StatoQI_Esperimento e StatoQI_DataSet. Nelle ultime due modalità sarà possibile adoperare i membri ***Value_QI*** e ***FI_QI*** i cui attributi ***NOME*** indicheranno il DataSet o l'Esperimento di interesse secondo i casi. Cambiando modalità è possibile, dunque, abilitare all'occorrenza l'uso dei campi presenti nei membri ***Value_QI*** e ***FI_QI*** al fine di effettuare le QUERY d'interesse introdotte nei capitoli precedenti. È possibile inoltre prelevare la modalità conservata in ***StatoQI*** per mezzo del membro **getStatoQI()**.



∅ La classe **Categoria** è quella che ha richiesto più modifiche, a causa della sua organizzazione gerarchica e del fatto che doveva gestire tre tabelle mascherandone il più possibile la struttura.



I membri **setConnessione()** e **freeConnessione()** sono ridefiniti poiché la classe deve poter gestire anche il suo membro ***SovraCategoria*** cosa che richiede di eseguire i membri **Sovracategoria.setConnessione()** o **Sovracategoria.freeConnessione()** a seconda dei casi.

Il membro **setStatoDes()** permette di abilitare/disabilitare il campo DESCRIZIONE_CATEGORIA (l'oggetto inizializzato ha il campo abilitato).

La ridefinizione di **TransInserisci()** adotta degli accorgimenti necessari nel caso le descrizioni siano abilitate:

- Per inserire in tale modalità delle nuove categorie il valore del campo ID_CATEGORIA deve essere settato a *numNoUtil*,
- mentre se si vogliono aggiungere solo delle informazioni ad una categoria già esistente deve avere l'ID della Categoria di interesse.

Il membro **TransElimina()** *dà errore nel caso le descrizioni siano abilitate!* Dunque è evidente che l'eliminazione di una particolare categoria non adopera DESCRIZIONE_CATEGORIA come filtro. Al membro è stato aggiunto un parametro d'ingresso (SettaFigli) che se è true indica di associare i padri delle categorie da eliminare ai propri figli, altrimenti resetta il campo padre dei figli. Tale operazione, se SettaFigli è uguale a true, non può essere rieseguita con **Riesegui()**. Attenzione nel caso di errore il database potrebbe essere in uno stato inconsistente e si consiglia di eseguire il membro **TransRollBack()** dell'istanza della classe **Permessi** associata all'oggetto **Categoria**.

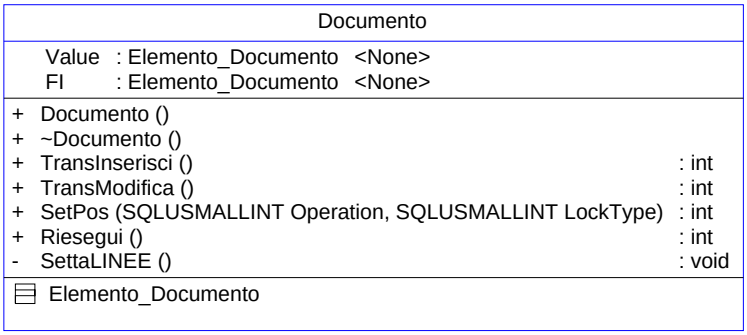
La ridefinizione di **SetPos()** adotta degli accorgimenti necessari nel caso le descrizioni siano abilitate:

- Per inserire in tale modalità delle nuove categorie il valore del campo ID_CATEGORIA deve essere settato a *numNoUtil*,
- mentre se si vogliono aggiungere solo delle informazioni ad una categoria già esistente deve avere l'ID della Categoria di interesse.
- le cancellazioni in tale modalità eliminano solo le descrizioni (*si osservi che è l'unico modo per eliminare una delle descrizione di una categoria*).

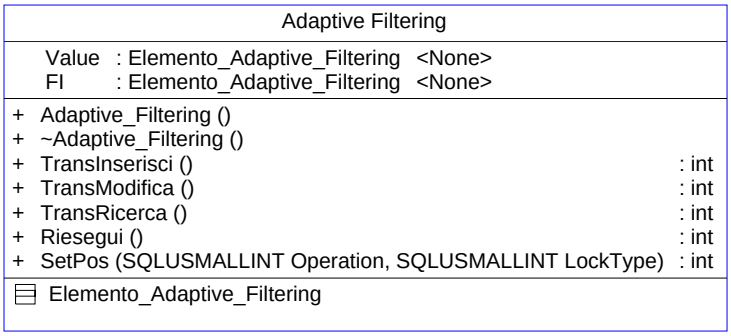
InsGerarchia() e **DelGerarchia()** permettono di inserire o eliminare la categoria individuata dal parametro di ingresso Ext_ID come Sovracategoria (SottoCategoria se il parametro di ingresso SovraCategoria è false) dell'attuale categoria individuata tramite **Preleva()**. Nel caso il membro richiamato sia **DelGerarchia()** se Ext_ID ha valore **0** si

procede con l'eliminare tutte le Sovracategoria (o Sottocategorie a seconda dei casi). Per entrambi i membri, i valori d'uscita sono ERRORE_SQL oppure NO_ERRORE. Infine l'ultimo membro modificato è **getSettaggioTitoli()**, implementato per far si che le informazioni restituite in uscita si riferiscano sempre solo ai campi attualmente disponibili all'utente.

∅ La particolarità della classe **Documento** è quella di implementare un membro **SettaLINEE()** che permette di gestire il vincolo relativo al fatto che il campo LINEE ha come valore il numero di linee del campo TESTO. Tale membro viene poi utilizzato dalle nuove versioni dei membri **TransInserisci()**, **TransModifica()**, **SetPos()** e **Riesegui()**.



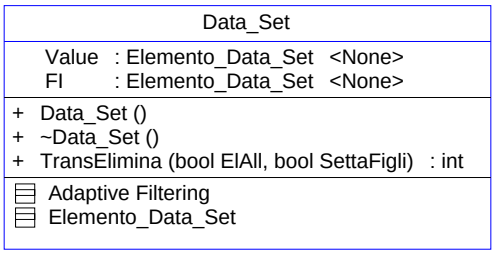
∅ La classe **Adaptive_Filtering** deriva da **Vincoli** e tutti i membri che ridefinisce non fanno altro che gestire la verifica del vincolo presentato nel *paragrafo 6.4.2*, tramite la riga di codice *VerificaAdaptiveFiltering((Dati *)&Value)*. Più in particolare mentre i membri **TransInserisci()**, **TransModifica()**, **SetPos()** e **Riesegui()** sono ridefiniti per verificare i vincoli (la cui mancata soddisfazione causerà in uscita un valore uguale a NO_ACCESS, non previsto nei membri base), **TransRicerca()** lo è solo per preparare la QUERY che poi sarà utilizzata nel gestire il vincolo quando sarà richiamato **SetPos()**.

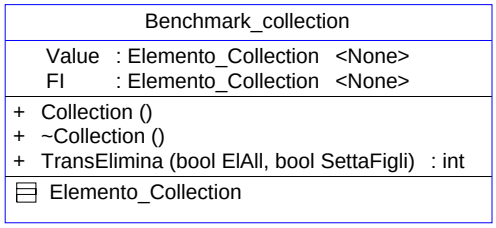


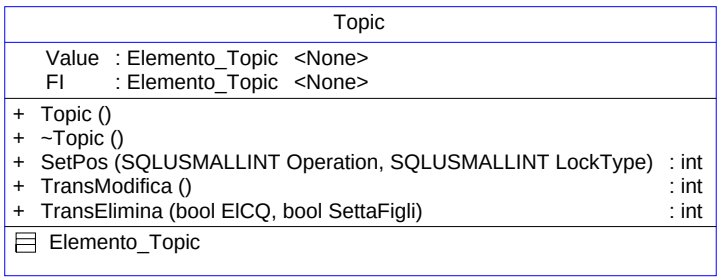
∅ Le classi **Data_Set**, **Collection**, **Split** e **Topic** ridefiniscono il membro **TransElimina()** al fine di poter implementare le cancellazioni anche delle entità a loro connesse. I parametri d'ingresso sono facilmente interpretabili ma vediamo di individuare altre particolarità di questi membri:

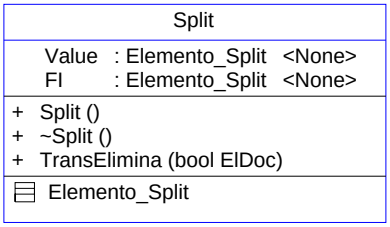
- Innanzi tutto, se si abilitano gli input ElAll (ElDoc o ElCQ), non è possibile rieseguire l'operazione cosi come richiesta la prima volta.
- Nel caso di errori si consiglia di effettuare un **TransRollBack()** dal momento che si potrebbero essere cancellati solo alcuni dei record sottoposti all'elemento da eliminare.
- Si aggiunge a tutti i membri **TransElimina()** il valore di ritorno NO_DATO che individua il caso in cui non vi siano elementi da eliminare.
- A tutti i membri (tranne quello di **Split**,che non deriva da Vincoli e dunque non adopera il membro **Vincoli::ForeignKeys()**) si aggiunge in uscita il valore NO_ACCESS, restituito quando l'operazione non è richiesta da un **DBA**.(tale operazione infatti è rischiosa e se mal gestita potrebbe cancellare le ForeignKeys senza ricrearle).

∅ La classe **Topic,** rispetto a **Data_Set**, **Collection** e **Split,** ridefinisce anche i membri **SetPos()** e **TransModifica()** in modo da soddisfare il vincolo secondo cui il valore del campo TIPO_TOPIC dei Topic esistenti non possa essere modificato.









7.7.2 Implementazione Domini “Type”

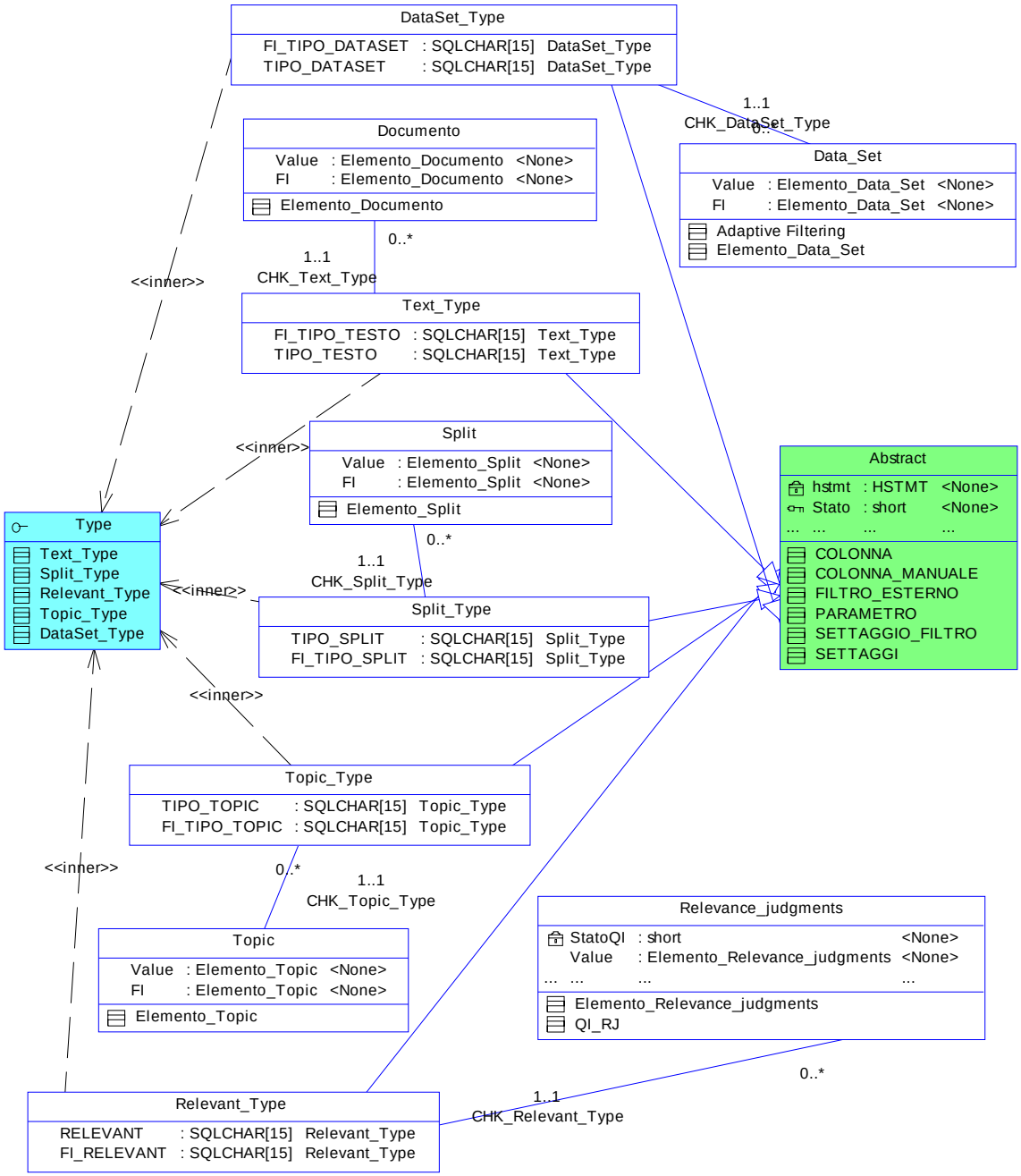


Figura 31: Modello Object-Oriented (2° Parte)

Tali classi definiscono solo i loro **costruttori** e **distruttori** (così come visto all’inizio del sottoparagrafo precedente); inoltre, la semplicità delle tabelle, fa sì che basti definire solo due membri: un primo da associare al puntatore ***Colonne[0].ptr*** e l’altro a ***Colonne[0].ptrFI***.

7.7.3 Implementazione degli Esperimenti

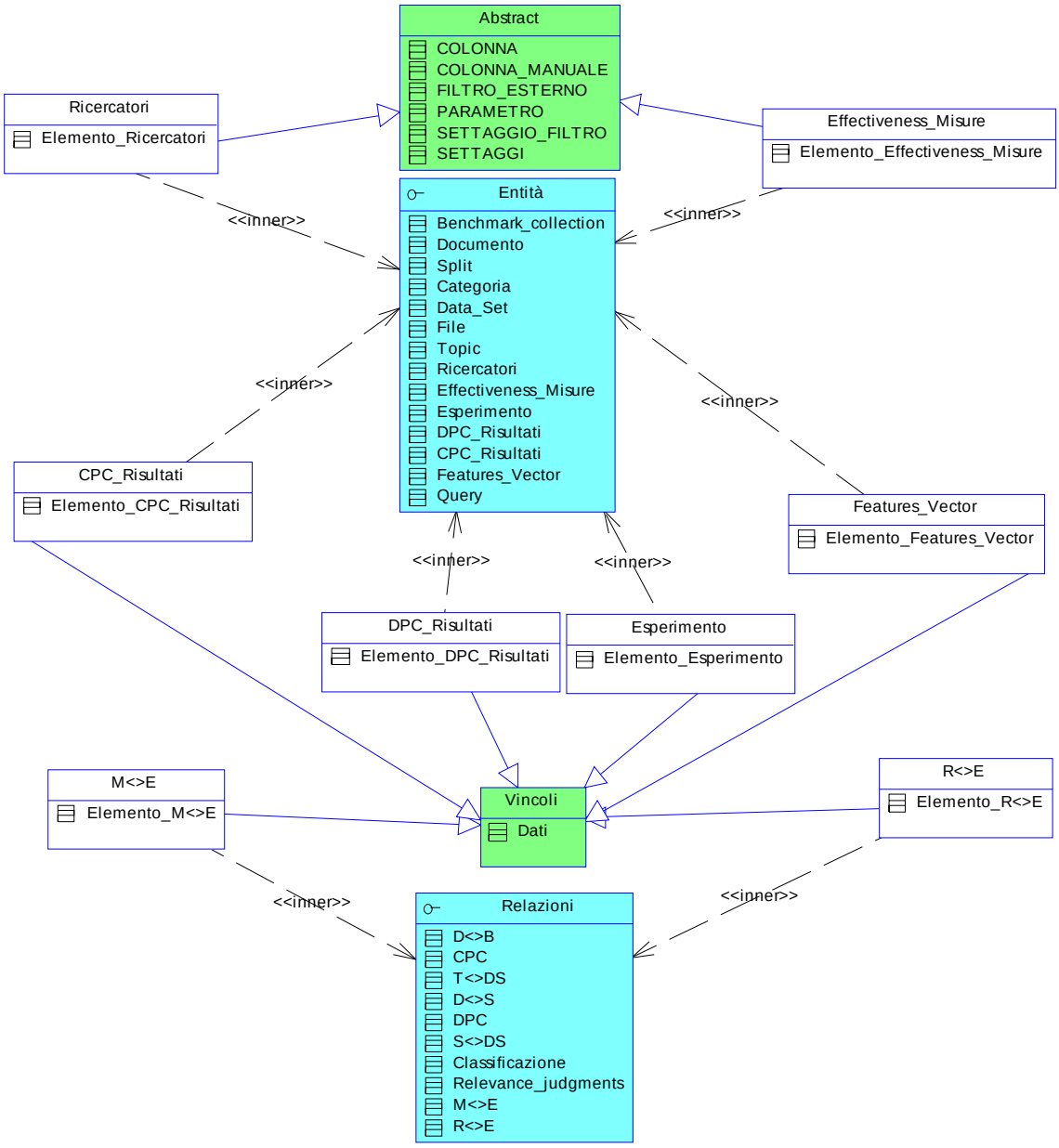
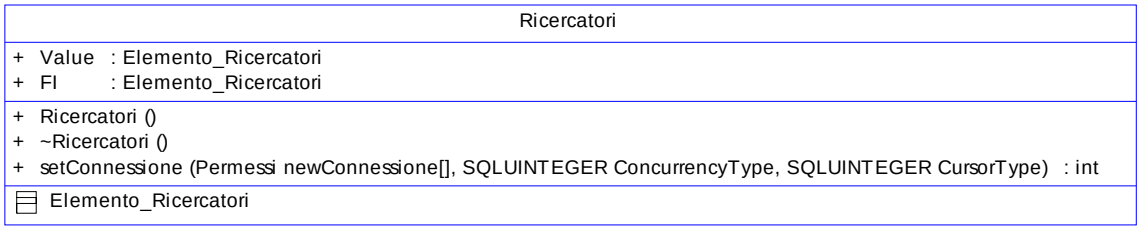
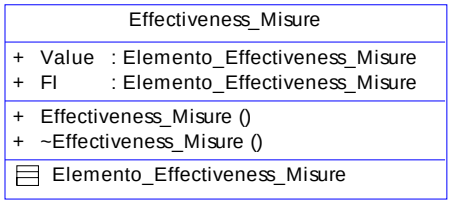


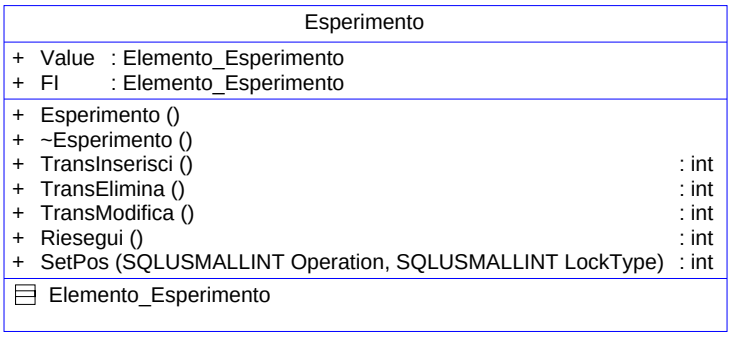
Figura 32: Modello Object-Oriented (3° Parte)

∅ La classe derivata **Effectiveness_Misure** non fa altro che definire i propri **costruttori** e **distruttori**. La classe **Ricercatori** invece ridefinisce anche il membro **setConnessione()**, poichè i campi ***Value.UTENTE***, ***Value.PASSWORD*** e ***Value.GRUPPO*** sono significativi solo se l'oggetto Permessi, passato come parametro di ingresso, è in uno ***Stato*** DBA. (in tale caso si adopera la QUERY contenente sia i dati di PERMESSI che di RICERCATORI, cioè *EXT_PERMESSI*, al posto di quella di default *EXT_RICERCATORI*).



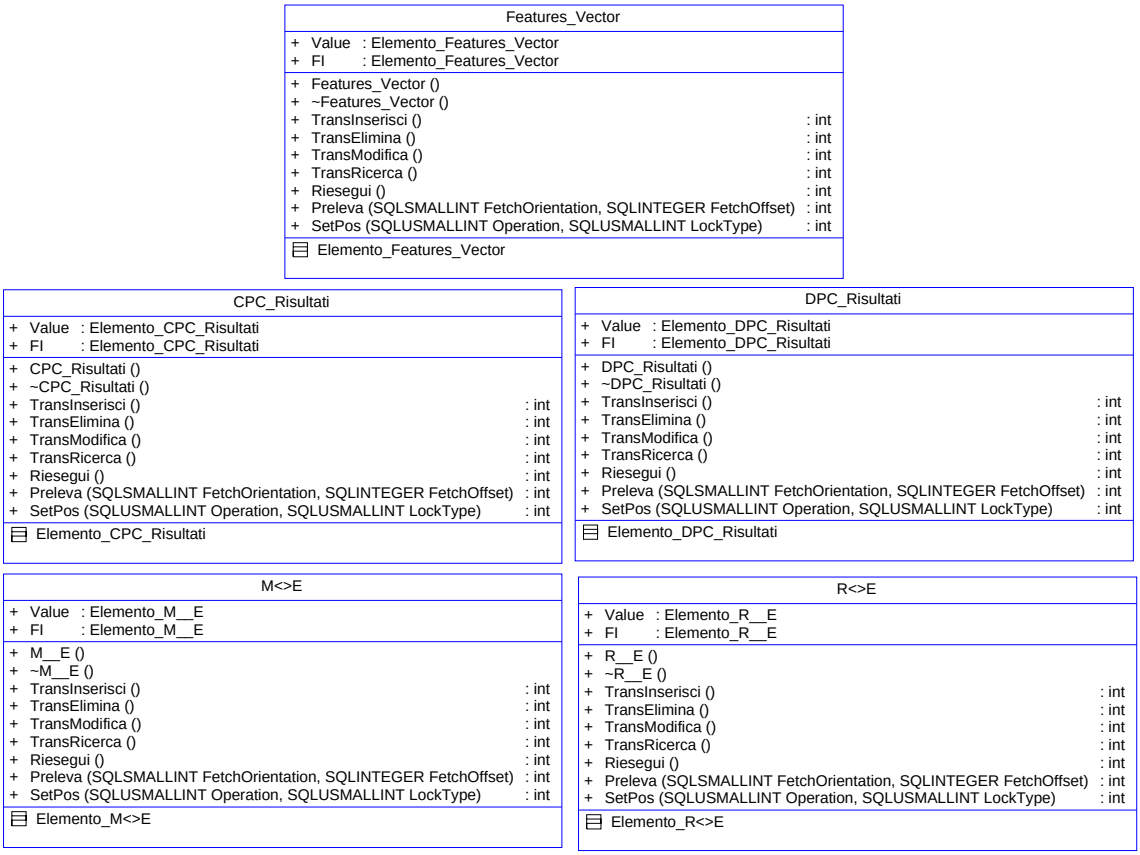


∅ La classe **Esperimenti** (come tutte le altre ad essa legate) modifica molti dei membri definiti in **Abstract**. In **TransInserisci()** si utilizza il comando ***Vincoli::InserisciR__E(Value.NOME_ESPERIMENTO)*** in modo da poter aggiungere, oltre all'esperimento, anche l'informazione riguardante il suo primo proprietario. I membri **TransModifica()** e **TransElimina()** adoperano invece la funzione membro **Vincoli::AccessoEsperimento()** per verificare se si ha accesso al particolare esperimento. Infine i membri **SetPos()** e **Riesegui()** adoperano entrambe le funzionalità della classe base **Vincoli** di cui si è parlato. L'ultima osservazione riguarda l'insieme dei valori di ritorno, relativi agli ultimi quattro membri, che si arricchisce di PERMESSI_NON_ABILITATI, valore dato in uscita nel caso in cui si stia in uno stato RICERCATORE e non si abbiano i permessi necessari (tale valore è restituito anche se si cerca di eseguire delle QUERY senza specificare un particolare esperimento).



∅ Le classi rimaste ridefiniscono tutte le funzionalità principali. I membri **TransInserisci()**, **TransElimina()**, **TransModifica()** si preoccupano tutti di utilizzare **Vincoli::AccessoEsperimento()** per effettuare le verifiche. **TransInserisci()** e **TransModifica()** inoltre adoperano, nel caso la particolare classe derivata richieda la verifica dei vincoli discussi nel *paragrafo 6.5.1*, **Vincoli::VerificaDPC_Risultati()**,

Vincoli::VerificaCPC_Risultati() o **Vincoli::VerificaFV()**. I membri **TransRicerca()** e **Preleva()** sono ridefiniti (come già visto nella *paragrafo 7.6*) semplicemente per salvare in ***EspFetch*** il valore del campo ***Value.NOME_ESPERIMENTO*** (ciò è necessario poiché la stessa variabile potrebbe contenere un eventuale nuovo valore cancellando quello vecchio necessario per effettuare le verifiche). **Riesegui()** e **SetPos()** adoperano, a seconda dell'operazione a loro richiesta, entrambi od uno dei vincoli fin qui introdotti. Infine tutti i membri qui riportati, ad eccezione di **Preleva()** e **TransRicerca()**, restituiscono in uscita il valore PERMESSI_NON_ABILITATI se si sta in uno stato RICERCATORE e non si hanno i permessi necessari o non si è specificato un particolare esperimento nei filtri.



7.7.4 I Membri della Classe DBDSATC

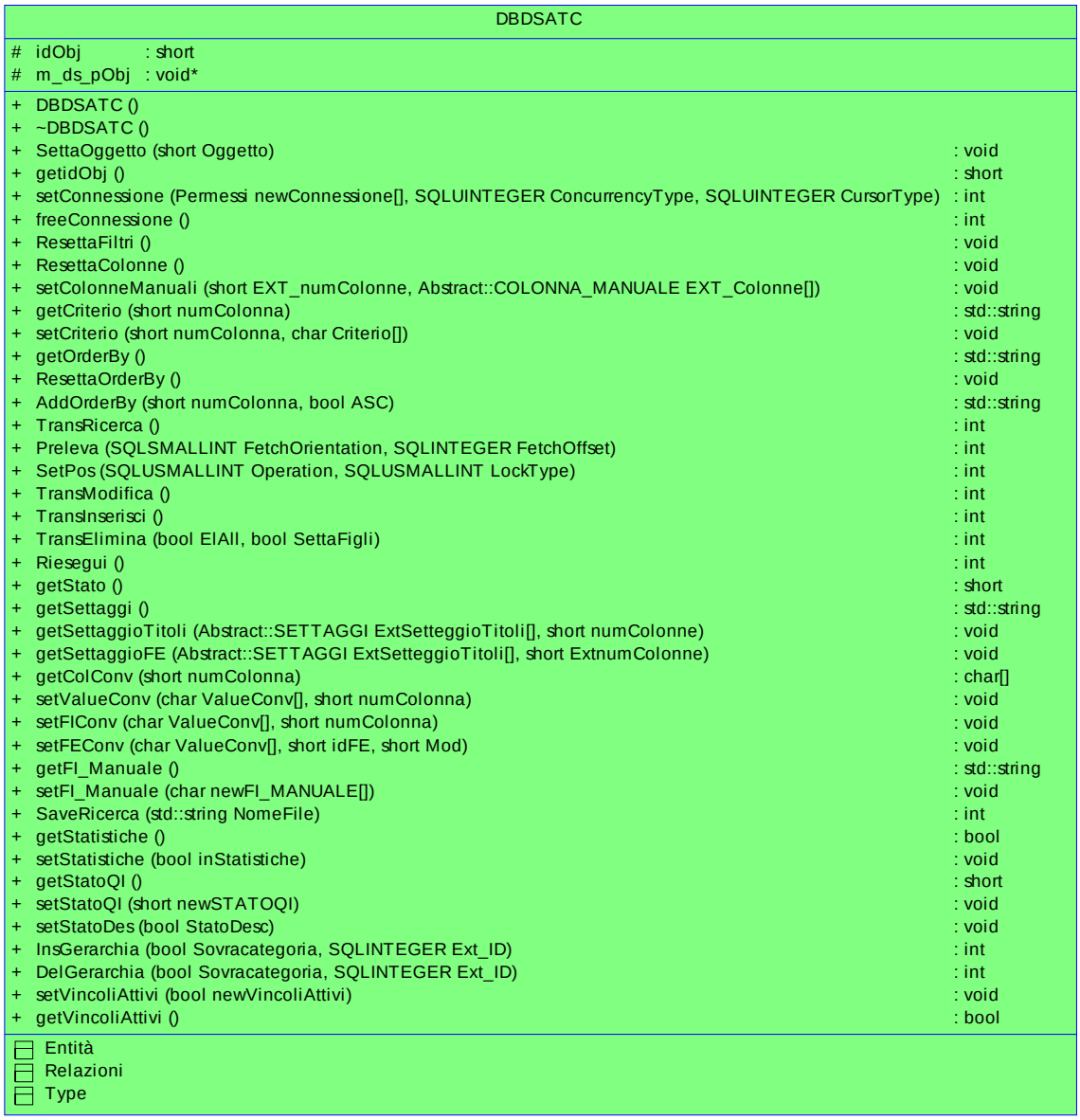


Figura 33: Classe DBDSATC

Il **costruttore** **DBDSATC()** genera, di default, dinamicamente un oggetto della classe Collection da associare al membro ***m_ds_pObj***; il **distruttore** **~DBDSATC()** invece si preoccupa di liberare la memoria dall'oggetto allocato in precedenza.

Il membro **SettaOggetto()** distrugge l'ultimo oggetto puntato da ***m_ds_pObj*** e si preoccupa di crearne uno nuovo identificato tramite il parametro di ingresso Oggetto, che sarà memorizzato nel membro ***idObj*** (i possibili valori di idObj sono definiti per mezzo di direttive nel paragrafo seguente). In seguito sarà possibile ricavare il tipo di oggetto settato adoperando il membro **getidObj()**.

Tutti gli altri membri definiti non fanno altro che richiamare, secondo l'oggetto settato, il relativo membro da adoperare. Nel caso i membri richiedano, per classi diverse, un

numero discordante di parametri, si è preferito inserirli tutti e lasciare inutilizzati quelli ridondanti quando non necessari. Per quanto riguarda i membri adoperati da solo alcune classi questi saranno sempre richiamabili, ma solo quando effettivamente necessari daranno qualche risultato, altrimenti, se è presente un parametro d'uscita, restituiranno il valore NO_ACTIVE.

7.7.5 Le Direttive.

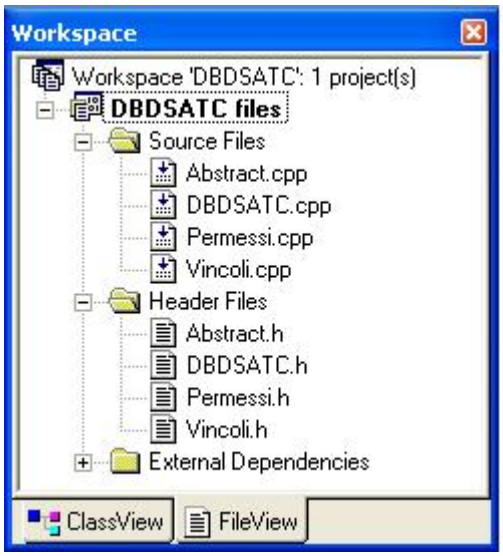
In tale sottoparagrafo si riportano le direttive relative ai valori consentiti per ***idObj***, per gli ***FE.Type*** di ogni classe definita in precedenza e per il membro ***StatoQI*** quando presente.

```
/*Definizione idObj di DBDSATC*/
#define idObj_Collection        1
#define idObj_File              2
#define idObj_DataSet           3
#define idObj_AF                4
#define idObj_Topic             5
#define idObj_Split             6
#define idObj_Query             7
#define idObj_Categoria         8
#define idObj_Documento         9
#define idObj_Classificazione   10
#define idObj_RJ                11
#define idObj_DPC               12
#define idObj_CPC               13
#define idObj_T__DS             14
#define idObj_S__DS             15
#define idObj_D__B              16
#define idObj_D__S              17

#define idObj_Ricercatori       18
#define idObj_EM                19
#define idObj_Esperimento       20
#define idObj_FV                21
#define idObj_DPC_R             22
#define idObj_CPC_R             23
#define idObj_M__E              24
#define idObj_R__E              25

#define idObj_Text_Type         26
#define idObj_Split_Type        27
#define idObj_Relevant_Type     28
#define idObj_Topic_Type        29
#define idObj_DataSet_Type      30

/* Valore di ritorno quando l'operazione non è attiva */
#define NO_ACTIVE               -999

/*Definizione valori validi per i FE.Type delle diverse entità*/

#define BNC_da_Documento        1
#define BNC_da_Categoria        2
#define BNC_da_Query            3
#define BNC_da_Topic            4
#define BNC_da_Split            5
#define BNC_da_Esperimento      6
#define BNC_da_DataSet          7

#define DTS_da_Topic            1
#define DTS_da_Split            2
#define DTS_da_Esperimento      3
#define DTS_da_Ricercatore      4
#define DTS_da_Collection       5
```

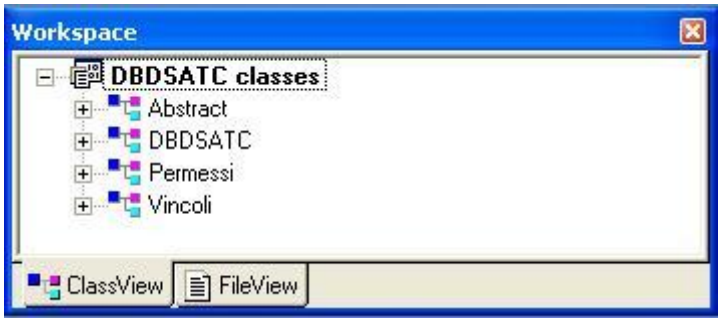
```
#define TPC_da_Documento        1
#define TPC_da_Categoria        2
#define TPC_da_Query            3
#define TPC_da_Split            4
#define TPC_da_Esperimento      5
#define TPC_da_DataSet          6

#define SPT_da_Documento        1
#define SPT_da_Topic            2
#define SPT_da_Esperimento      3
#define SPT_da_DataSet          4

#define CAT_da_Documento        1
#define CAT_da_Query            2
#define CAT_da_Topic            3
#define CAT_da_Split            4
#define CAT_da_Esperimento      5
#define CAT_da_Collection       6
#define CAT_da_DataSet          7
#define CAT_da_SovraCategoria   8
#define CAT_da_SottoCategoria   9

#define QRY_da_Documento        1
#define QRY_da_Categoria        2
#define QRY_da_Topic            3
#define QRY_da_Split            4
#define QRY_da_Esperimento      5
#define QRY_da_DataSet          6
#define QRY_da_Collection       7

#define DOC_da_Categoria        1
#define DOC_da_Query            2
#define DOC_da_Topic            3
#define DOC_da_Split            4
#define DOC_da_DataSet          5
#define DOC_da_Esperimento      6
#define DOC_da_Collection       7

#define RIC_da_Collection       1
#define RIC_da_DataSet          2
#define RIC_da_Split            3
#define RIC_da_Topic            4
#define RIC_da_Esperimento      5

#define EM_da_Esperimento       1

#define CPCR_da_DataSet         1

#define DPCR_da_DataSet         1

#define ESP_da_EffMisure        1
#define ESP_da_Ricercatore      2
#define ESP_da_Collection       3
#define ESP_da_Split            4
#define ESP_da_Topic            5

#define FV_da_DataSet           1

/*Definizione valori validi per i FE.Type delle diverse relazioni*/

#define RJ_da_Topic             1
#define RJ_da_Split             2
#define RJ_da_Esperimento       3
#define RJ_da_DataSet           4
#define RJ_da_EsperimentoAF     5
#define RJ_da_DataSetAF         6

#define CLA_da_Topic            1
#define CLA_da_Split            2
#define CLA_da_DataSet          3
#define CLA_da_Esperimento      4

/*Definizione valori validi per le funzioni membro setStatoQI */
#define StatoQI_Normale         0
#define StatoQI_Esperimento     1
#define StatoQI_DataSet         2
```

7.8 Organizzazione in Libreria del Modulo

La documentazione relativa al modulo e l'implementazione dei vari membri sono state incluse nel file "*Progettazione DB\ObjDataModel_DBDSATC v2.2.oom*". Per ricavare i file sorgente e gli header necessari alla realizzazione della libreria basta richiamare il comando "Language\Generate C++ Code…" messo a disposizione in PowerDesigner 9.0 e selezionare le sole classi **Permessi**, **Abstract**, **Vincoli** e **DBDSATC**. I file realizzati sono stati posti nella directory *"Progettazione DB\Codice"* (si veda il *paragrafo 1.6*) e sono *Permessi.cpp*, *Abstract.cpp*, *Vincoli.cpp*, *DBDSATC.cpp* ed i rispettivi header *Permessi.h*, *Abstract.h*, *Vincoli.h*, *DBDSATC.h* (al loro interno sono già presenti i riferimenti alle librerie adottate).



Per mezzo di questi file è dell'ambiente di sviluppo **Microsoft Visual C++ 6.0** si è poi realizzata la libreria statica **DBDSATC.lib,** che per poter essere utilizzata richiede i quattro file header riportati precedentemente. Il workspace adoperato per la realizzazione della libreria è contenuto nella directory *"GestioneDS\DBDSATC"*.



CAPITOLO 8:

I MODULI DI FEEDING

8.1 ReutersIn

8.2 NewsgroupsIn

8.3 OhsumedIn

8.4 Test Dimensioni

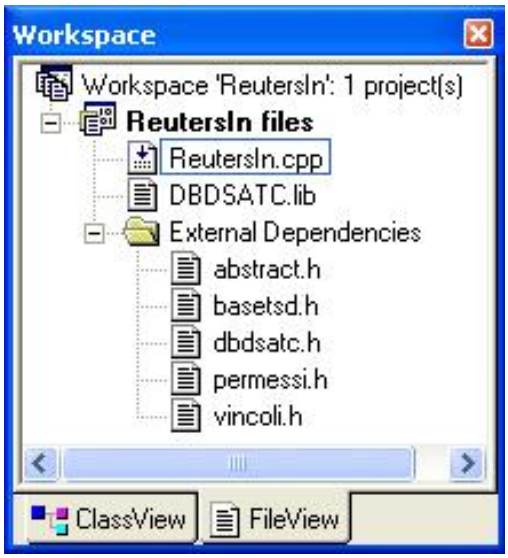
8.5 Analisi del database

Da questo punto in poi il codice realizzato adopera il workspace "DataSetDB.sws" (vedi paragrafo 1.6*) solo come punto di riferimento delle strutture dati da estrarre, e dunque non esiste alcuna documentazione esterna al codice in grado di far comprendere il lavoro effettuato per la sua realizzazione. Questo capitolo ha il compito di sopperire alla mancanza di documentazione esterna descrivendo i moduli di feeding,, il loro funzionamento e le difficoltà incontrate nell'estrazione dei dati.*

8.1 ReutersIn

Il modulo è costituito da un eseguibile realizzato utilizzando la libreria **DBDSATC.lib**. Il workspace adoperato, relativo all'ambiente di sviluppo **Microsoft Visual C++ 6.0,** è contenuto nella directory *"GestioneDS\ReutersIn"*.

Per realizzare il modulo basterebbe comunque considerare il solo file sorgente *ReutersIn.cpp*, contenuto nella directory *"Progettazione DB\Codice\ReutersIn"*, che include a sua volta solo l'header **DBDSATC.h** (al cui interno vi sono i richiami agli altri header della libreria e a quelli presentati nel *paragrafo 7.2*). Naturalmente si dovrà anche inserire nell'ambiente di sviluppo il file della libreria **DBDSATC.lib**.



Il modulo durante la sua fase di esecuzione richiederà di poter accedere ad alcuni file, definiti internamente al codice per mezzo di alcune direttive:

```
//La directory del DataSet va inserita nella stessa directory dell'eseguibile
#define FileCategorie "reuters21578\\cat-descriptions_120396.txt"
//I file dei documenti vanno da FileDocumenti + "00.sgm" a FileDocumenti + "021.sgm"
#define FileDocumenti "reuters21578\\reut2-0"
```

8.1.1 Descrizione

Il codice adopera le classi messe a disposizione dalla libreria DBDSATC, gran parte di esso è ripetitivo ed utilizzato per gestire eventuali errori. L'inserimento dei dati nel database avviene per mezzo di transazioni che saranno risolte in alcuni punti del codice con un commit (*Per.TransCommit()*). Nel caso di un errore sarà possibile ripristinare, in un secondo momento, l'esecuzione dall'ultimo commit effettuato rieseguendo il modulo

e passandogli un particolare valore in ingresso (nel flusso che sarà descritto si trascura la descrizione di tale funzionalità).

Prima di procedere con la descrizione del flusso (considerato un buon esempio d'utilizzo delle funzionalità d'inserzione e ricerca della libreria), riportiamo il codice adoperato per gestire e verificare la presenza d'errori nei membri.

```
//con Per s'indica l'oggetto della classe Permessi adoperato
if (TestReturn == PERMESSI_ERROR_SQL) {   //Condizione dipendente dal membro
  strTmp = Per.getErrore();               //Prelievo dell'errore riscontrato
  strTmp += "\nErrore LINEA: ";
  strTmp += itoa(__LINE__ -4,(char *)Err,10);     //Prelievo linea dove si è verificato l'errore
  printf("%s",strTmp.data());
  printf("\nSi desidera Continuare(G) o uscire con un RollBack(R) o Commit(C)\? [G/R/C] ");
  scanf("%s",Err);                        //Prelievo decisione dell'utente
  if (Err[0] == 'R') {
          Per.TransRollBack();            //Effettua un RollBack
          Per.Disconnect();               //Disconnette Per
          Per.Deallocahenv();             //Dealloca l'Environment
          return -1;
  }  else if (Err[0] == 'C') {
          Per.TransCommit();              //Effettua un Commit
          Per.Disconnect();               //Disconnette Per
          Per.Deallocahenv();             //Dealloca l'Environment
          return -1;
  }
}
```

1) Dopo aver prelevato i dati relativi al nome della fonte ODBC, dell'User e della Password (che dovranno essere quelli di un DBA) si procederà con la creazione di un oggetto permessi e con la sua inizializzazione:

```
Permessi Per;
TestReturn = Per.Allocahenv ();
TestReturn = Per.Connect(&strNameDB, &Utente, &Pass);
```

2) Inserimento dei dati sulla Collection

```
DBDSATC::Entita::Collection Collezione;

Collezione.setConnessione(&Per,SQL_CONCUR_LOCK,SQL_CURSOR_KEYSET_DRIVEN);

strcpy((char*)Collezione.Value.NOME_COLLECTION,"Reuters-21578");
strcpy((char*)Collezione.Value.VERSIONE,"Distribution 1.0");
…

TestReturn = Collezione.TransInserisci();
```

*(il richiamo del membro **setConnessione(…)** e l'assegnamento dei valori ai campi **Value…** sarà effettuato anche nelle inserzioni relative alle classi seguiti, ma in tali casi per snellire la descrizione tali comandi saranno sottointesi).*

La tabella sottostante riporta i dati inseriti relativamente all'entità benchmark *(ogniqualvolta si descriveranno delle inserzioni di dati, ricavati da informazioni estratte tramite l'interpretazione umana, si procederà all'inserimento di una tabella contenente i dati inseriti).*

Elemento BENCHMARK_COLLECTION	
Nome_Collection	Reuters-21578
Versione	Distribution 1.0
Data_Collection	26-set-97
Autore_Collection	David D. Lewis
Sito	UCI KDD Archive
URL	http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
Modulo_di_feeding	ReutersIn.exe
Note_Documenti	Newswire collection
Note_Categorie	Categories related to economics
Descrizione_ID1	OLDID: The identification number (ID) the story had in the Reuters-22173 collection
Descrizione _ID2	NEWID: The identification number (ID) the story has in the Reuters-21578, Distribution 1.0 collection. These IDs are assigned to the stories in chronological order
Descrizione__Collection	

3) Inserimento dei file della collection

```
BDSATC::Entita::File File;
…
TestReturn = File.Riesegui();              //Dal secondo inserimento in poi usiamo Riesegui()
…
```

(Il membro Riesegui(…) sarà adoperato ogni qual volta si presenterà la necessità di inserire più elementi, anche l'utilizzo di tali comandi sarà sottinteso).

Elementi FILE		
NOME_COLLECTION	**NOME_FILE**	**DESCRIZIONE_FILE**
Reuters-21578	readme.txt	File contenente informazioni sul Data Set e sulla sua organizzazione
Reuters-21578	reuters21578.tar.gz	File contenente il Data Set formattato in SGML e alcuni file che elencano le Categorie

4) Inserimento degli Split della collection

```
DBDSATC::Entita::Split Split;
…
```

Elementi SPLIT			
NOME_COLLECTION	**NOME_SPLIT**	**TIPO_SPLIT**	**DESCRIZIONE_SPLIT**
Reuters-21578	CGISPLIT Te	TEST SET	Indica il Test Set adoperato da Hayes nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazione avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti HAYES89, HAYES90b.
Reuters-21578	CGISPLIT Tr	TRAINING SET	Indica il Training Set adoperato da Hayes nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazione avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti HAYES89, HAYES90b.
Reuters-21578	LEWISSPLIT Te	TEST SET	Indica il Test Set adoperato da Lewis nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazione avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b.
Reuters-21578	LEWISSPLIT Tr	TRAINING SET	Indica il Training Set adoperato da Lewis nel Reuters-22173. Tale Split non è completo a causa delle correzioni e cancellazioni avute per passare dal Reuters-22173 al Reuters-21578. Adoperato negli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b.
Reuters-21578	ModApte Te	TEST SET	The Modified Apte (ModApte) Split Te indica il Test Set adoperato da Apt adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti APTE94 e APTE94b con il Reuters-22173.
Reuters-21578	ModApte Tr	TRAINING SET	The Modified Apte (ModApte) Split Tr indica il Training Set adoperato da Apt adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti APTE94 e APTE94b con il Reuters-22173.
Reuters-21578	ModHayes Te	TEST SET	The Modified Hayes (ModHayes) Split indica il Test Set adoperato da Hayes adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti HAYES89, HAYES90b con il Reuters-22173.
Reuters-21578	ModHayes Tr	TRAINING SET	The Modified Hayes (ModHayes) Split indica il Training Set adoperato da Hayes adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti HAYES89, HAYES90b con il Reuters-22173.
Reuters-21578	ModLewis Te	TEST SET	The Modified Lewis (ModLewis) Split Te indica il Test Set adoperato da Lewis adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b con il Reuters-22173.
Reuters-21578	ModLewis Tr	TRAINING SET	The Modified Lewis (ModLewis) Split Tr indica il Training Set adoperato da Lewis adattato al Reuters-21578. La versione originale era adoperata per eseguire gli esperimenti LEWIS91d, LEWIS92b, LEWIS92e, LEWIS94b con il Reuters-22173.
Reuters-21578	TOPICS assenti	NO	Indica l'assenza di elementi della classe TOPICS nel Reuters-22173
Reuters-21578	TOPICS Bypass	BYPASS	Indica la mancata determinazione del TOPICS nel Reuters-22173
Reuters-21578	TOPICS presenti	YES	Indica la presenza di elementi della classe TOPICS nel Reuters-22173

5) Inserimento dei Topic della collection

```
DBDSATC::Entita::Topic Topic;
…
```

Elementi TOPIC			
NOME_COLLECTION	**NOME_TOPIC**	**TIPO_TOPIC**	**DESCRIZIONE_TOPIC**
Reuters-21578	Economics (HAYES90b)	CATEGORIE	Il totale di 674 categories menzionate nel HAYES90b, è la somma delle 135 del HAYES89 ed altre 539
Reuters-21578	Topics (HAYES89)	CATEGORIE	Una lista di 135 categorie di tipo economico che sono state usate nel HAYES89

6) Inserimento dei Data Set e delle associazioni

```
DBDSATC::Entita::Data_Set Data_Set;

/*Classi necessarie per le associazioni con altre entità*/
DBDSATC::Relazioni::T__DS T__DS;
DBDSATC::Relazioni::S__DS S__DS;
…
```

Elementi DATA_SET		
NOME_ DATA_SET	**TIPO_ DATASET**	**DESCRIZIONE_DATA_SET**
ModApte	**DPC**	This replaces the 10645/3672 split (7,856 not used) of the Reuters-22173 collection. These are our best approximation to the training and test splits used in APTE94 and APTE94b. Note the following: 1. As with the ModLewis, those documents removed in forming Reuters-21578 are not present, and BYPASS documents are not used. 2. The intent in APTE94 and APTE94b was to use the Lewis split, but restrict it to documents with at least one TOPICS categories. However, but it was not clear exactly what Apte, et al meant by having at least one TOPICS category (e.g. how was "bypass" treated, whether this was before or after any fixing of typographical errors, etc.). We have encoded our interpretation in the TOPICS attribute. ***Note that, as discussed above, some TOPICS="YES" stories have no TOPICS categories, and a few TOPICS="NO" stories have TOPICS categories. These facts are irrelevant to the definition of the split.*** If you are using a learning algorithm that requires each training document to have at least TOPICS category, you can screen out the training documents with no TOPICS categories. Please do NOT screen out any of the 3,299 documents - that will make your results incomparable with other studies. 3. As with ModLewis, it may be desirable to use the 8,676 Unused documents for gathering statistical information about feature distribution. As with ModLewis, this split assigns documents from April 7, 1987 and before to the training set, and documents from April 8, 1987 and after to the test set. The difference is that only documents with at least one TOPICS category are used. The rationale for this restriction is that while some documents lack TOPICS categories because no TOPICS apply (i.e. the document is a true negative example for all TOPICS categories), it appears that others simply were never assigned TOPICS categories by the indexers. (Unfortunately, the amount of time that has passed since the collection was created has made it difficult to establish exactly what went on during the indexing.)

		WARNING: Given the many changes in going from Reuters-22173 to Reuters-21578, including correction of many typographical errors in category labels, results on the ModApte split cannot be compared with any published results on the Reuters-22173 collection!
ModHayes	**DPC**	This is the best approximation we have to the training and test splits used in HAYES89, HAYES90b, and Chapter 8 of LEWIS91d. It replaces the 21450/723 split of the Reuters-22173 collection. Note the following: 1. As with the other splits, the duplicate documents removed in forming Reuters-21578 are not present. 2. "Training" in HAYES89 and HAYES90b was actually done by human beings looking at the documents and writing categorization rules. We can not be sure which of the document files were actually looked at. 3. We specify that the BYPASS stories and the TOPICS=NO stories are part of the training set, since they were used during manual knowledge engineering in the original Hayes experiments. That does not mean researchers are obliged to give these stories to, for instance, a supervised learning algorithm. As mentioned in the other splits, they may be more useful for getting distributional information about features.
ModLewis	**DPC**	This replaces the 14704/6746 split (723 unused) of the Reuters-22173 collection, which was used in LEWIS91d (Chapters 9 and 10), LEWIS92b, LEWIS92c, LEWIS92e, and LEWIS94b. Note the following: 1. The duplicate documents removed in forming Reuters-21578 are of course not present. 2. The documents with TOPICS="BYPASS" are not used, since subsequent analysis strongly indicates that they were not categorized by the indexers. 3. The 1,765 unused documents should not be tested on and should not be used for supervised learning. However, they may useful as additional information on the statistical distribution of words, phrases, and other features that might used to predict categories. This split assigns documents from April 7, 1987 and before to the training set, and documents from April 8, 1987 and after to the test set. WARNING: Given the many changes in going from Reuters-22173 to Reuters-21578, including correction of many typographical errors in category labels, results on the ModLewis split cannot be compared with any published results on the Reuters-22173 collection!

Elementi T__DS	
NOME_DATA_SET	**NOME_TOPIC**
ModApte	Topics (HAYES89)
ModHayes	Economics (HAYES90b)
ModLewis	Topics (HAYES89)

Elementi S__DS		
NOME_DATA_SET	**NOME_SPLIT**	**TIPO_SPLIT**
ModApte	ModApte Te	TEST SET
ModApte	ModApte Tr	TRAINING SET
ModHayes	ModHayes Te	TEST SET
ModHayes	ModHayes Tr	TRAINING SET
ModLewis	ModLewis Te	TEST SET
ModLewis	ModLewis Tr	TRAINING SET

7) Inserimento delle Categorie e delle associazioni Categorie-Topic

```
DBDSATC::Entita::Categoria Categoria;
DBDSATC::Relazioni::DPC DPC;
…
```

I dati delle Categorie vengono prelevati per mezzo del file individuato nella direttiva **#define FileCategorie** (si veda la fine del *paragrafo 8.1*). Le maggiori difficoltà di tale parte del codice sono legate ad una pessima formattazione delle informazioni contenute nel file. La suddivisione delle categorie in cinque gruppi ha permesso di individuare le sovracategorie principali (EXCHANGES, ORGS, PEOPLE, PLACES e TOPICS) le cui sottocategorie sono organizzate ognuna secondo un formalismo differente che, secondo i casi, può contenere anche informazioni riguardo alle categorie.

La sovracategoria TOPICS ha sottocategorie organizzate in due livelli, lo stesso si ha per quella PEOPLE, con la non trascurabile osservazione che le sottocategorie di primo livello possono coincidere con quelle di PLACES, giacché gli individui presenti nella categoria sono raggruppati per nazione.

Le forti differenze di formattazione e le osservazioni riguardo ai livelli della gerarchia, hanno richiesto l'organizzazione del prelievo dei dati in tre cicli principali. Di seguito si riportano le Categorie relative alle cinque sovracategorie principali, inserite nel database all'inizio dei vari cicli che saranno descritti.

ID_CATEGORIA	NOME_CATEGORIA	DESCRIZIONE_CATEGORIA
1	TOPICS	E' un insieme di codici riferiti a soggetti economici.
142	Organization Codes	
199	Exchanges Codes	
230	Country Codes	
415	People Codes	

- **Inserimento Categorie di TOPICS**

  Di seguito si riporta l'inizio del testo relativo alle categorie di **TOPICS**

```
****Subject Codes (135)

Money/Foreign Exchange (MONEY-FX)
Shipping (SHIP)
Interest Rates (INTEREST)

**Economic Indicator Codes (16)

Balance of Payments (BOP)
Trade (TRADE)
…
```

  I passi principali del ciclo, trascurando il codice relativo al recupero dei dati dal file, saranno:

  a) Verifica della presenza della categoria nel database.

```
strcpy((char*)Categoria.FI.NOME_CATEGORIA,…);
TestReturn = Categoria.Riesegui();                    //si riferisce a TransRicerca()
```

b) Inserzione della categoria, o solo della sua informazione se questa è già presente.

```
/*Categoria.Value.ID_CATEGORIA viene prelevato da TransRicerca() se la categoria è presente, altrimenti non è necessario e si crea una nuova tupla*/
strcpy((char*)Categoria.Value.NOME_CATEGORIA,...);
strcpy((char*)Categoria.Value.DESCRIZIONE_CATEGORIA,...);
TestReturn = Categoria.SetPos(SQL_ADD,NULL);
```

c) Prelievo dell'ID della categoria creata nel caso che la verifica del punto **a)** non abbia avuto buon esito.

```
TestReturn = Categoria.Preleva(SQL_FETCH_LAST,NULL);
```

d) Gestione ed inserzione delle sovracategorie.

```
Categoria.InsGerarchia(true,tmpID_SovraCategoria1);
```

e) Inserzione dell'associzione tra la categoria ed i due Topic del Benchmark

```
DPC.Value.ID_CATEGORIA = tmpID_SovraCategoria2;
strcpy((char*)DPC.Value.NOME_TOPIC,"Topics (HAYES89)");
TestReturn = DPC. Riesegui ();                    //si riferisce a TransInserisci();
...
DPC.Value.ID_CATEGORIA = tmpID_SovraCategoria2;
strcpy((char*)DPC.Value.NOME_TOPIC,"Economics (HAYES90b)");
TestReturn = DPC.Riesegui();                      //si riferisce a TransInserisci();
```

- **Inserimento Categorie di ORGS, EXCHANGE, PLACES**

Di seguito si riporta l'inizio del testo relativo alle categorie di **ORGS**

```
@heading[Organization Codes (56)]
@begin[format]
African Development Bank (ADB-AFRICA)@*
Agency for International Development  (AID)@*
```

alle categorie di **EXCHANGE**

```
@heading[Exchange Codes (39)]
@begin[format]
American Stock Exchange (AMEX)@*
Amsterdam Stock Exchange/Bourse (ASE)@*
```

e di **PLACES**

```
@heading[Country Codes (176)]
@begin[format]
AFGHANISTAN@*
ALBANIA @*
```

Come si può notare le ultime classi non hanno informazioni aggiuntive. Il ciclo adoperato è simile al precedente a parte le seguenti osservazioni:

a) Il recupero dei dati è differente a causa delle diverse formattazioni

b) Ogni categoria sarà associata alla sovracategoria principale di riferimento

c) Le categorie saranno inserite solo nel Topic "Economics (HAYES90b)"

- **Inserimento Categoria di PEOPLE**

  Di seguito si riporta l'inizio del testo relativo alle categorie di **PEOPLE**

```
@heading[People Codes (269)]
@begin[itemize]
Argentina
@begin[itemize]
   President Raul Alfonsin  (ALFONSIN)

   Economy Minister Juan Sourrouille  (SOURROUILLE)

   Finance Secretary Mario Brodersohn  (BRODERSOHN)

   Central Bank Governor Jose Luis Machinea  (MACHINEA)
@end[itemize]

Australia
@begin[itemize]
```

  Ancora una volta il ciclo è simile a quello adoperato per inserire le categorie di TOPICS, ma anche qui ci sono delle osservazioni da effettuare:

  a) Il recupero dei dati è differente a causa delle diverse formattazioni
  b) L'individuazione dell'attuale sovracategoria da considerare varia a causa della diversa formattazione.
  c) Le categorie saranno inserite solo nel Topic "Economics (HAYES90b)"

8) Inserimento dei Documenti e delle relative associazioni

A tal punto si procede all'esecuzione, su ogni file che sarà definito per mezzo della direttiva **#define FileDocumenti** (si veda la fine del *paragrafo 8.1*), di un ciclo che permetterà di inserire i dati dei documenti e le associazioni alle tuple delle entità fin qui inserite nel database. Innanzi tutto si procede con la dichiarazione degli oggetti necessari per gestire le tabelle d'interesse:

```
DBDSATC::Entita::Documento Documento;

/*Classi necessarie per le associazioni con altre entità*/
DBDSATC::Relazioni::Classificazione Classificazione;
DBDSATC::Relazioni::D__B D__B;
DBDSATC::Relazioni::D__S D__S;
```

Poi si procede coll'estrazione delle informazioni del documento. Un esempio di formattazione dei dati di un documento (i cui campi sono descritti nel *paragrafo 2.2*) è riportato di seguito:

```
<REUTERS TOPICS="NO" LEWISSPLIT="TEST" CGISPLIT="TRAINING-SET" OLDID="4034" NEWID="15051">
<DATE> 8-APR-1987 11:15:28.57</DATE>
<TOPICS></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
```

```
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f1353&#31;reute
r f BC-AUTOSPA-&lt;LUBE>-TO-RED   04-08 0041</UNKNOWN>
<TEXT>&#2;
<TITLE>AUTOSPA &lt;LUBE> TO REDEEM WARRANTS</TITLE>
<DATELINE>    NEW YORK, April 8 - </DATELINE><BODY>AutoSpa corp said it will redeem all
its common stock purchase warrants on May Five at 7.5 cts each.
    Through May Four, each warrant may be exercised into one
common share at 1.75 dlrs.
 Reuter
&#3;</BODY></TEXT>
```

L'organizzazione sequenziale degli attributi ha facilitato il prelievo dei dati. Una volta individuato il particolare campo è stato necessario tener conto della presenza di eccezioni relative ad alcuni errori riscontrati nella formattazione. L'individuazione di tali eccezioni è stata possibile grazie allo sviluppo di un prototipo del *modulo d'In/Out* (presente nella directory "*Progettazione DB\Prototipi\ReutersIn*") ed all'analisi dei dati prelevati ogni qual volta l'esecuzione del codice, in modalità di debug, presentava qualche errore inatteso. Di seguito si riportano le principali osservazioni a riguardo:

a) Il prelievo del valore da associare a Documento.Value.LUOGO ha presentato diverse difficoltà per la mancanza di una formattazione precisa del campo DATELINE. Infatti non è stato possibile definire immediatamente come suddividere il luogo e la data, o individuare l'assenza di uno dei due (si osservi che la stessa formattazione della data varia di volta in volta).
b) A volte è capitato di trovare i tag relativi a DATELINE all'interno del BODY. In questi casi si sono semplicemente eliminati i tag, poiché si è riscontrato che il contenuto del campo risultava essere parte del testo e non un luogo seguito da una data.
c) Nel prelievo dei valori di Documento.Value.TITOLO_DOCUMENTO e Documento.Value.TESTO si è constatato al loro interno la presenza di alcuni termini di escape tipici del formalismo html. Il problema è stato risolto sostituendo i valori con il loro significato originale:

   "**<**" si sostituisce con "<"

   "**&**" si sostituisce con "**&**"

Durante l'estrazione dei dati si procede anche alla ricerca delle categorie associate al particolare documento, in modo da poter realizzare un elenco di valori relativi agli ID_CATEGORIA da inserire nella tabella *Classificazione*. Estratti i dati si procede

all'inserzione del documento ed al prelievo del suo identificativo in modo da poterlo utilizzare in futuro per inserire le tuple delle relazioni individuate.

```
TestReturn = Documento.SetPos(SQL_ADD,NULL);
…
TestReturn = Documento.Preleva(SQL_FETCH_LAST,NULL);
…
/*Inserimento associazione Collection-Documento*/
D__B.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
TestReturn = D__B.SetPos(SQL_ADD,NULL);
…
/*Inserimento associazione Categorie-Documento*/
Classificazione.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
Classificazione.Value.ID_CATEGORIA = Ctgr[0];

if (numCtgr > 0) {
TestReturn = Classificazione.SetPos(SQL_ADD,NULL);
…
for (int i=1; i<numCtgr; i++) {
        Classificazione.Value.ID_CATEGORIA = Ctgr[i];
        TestReturn = Classificazione.SetPos(SQL_ADD,NULL);
…
```

Poi si procede alla verifica delle condizioni, presentate nel *paragrafo 2.4.2*, che stabiliscono le relazioni Documento Split da inserire nel database tramite il comando:

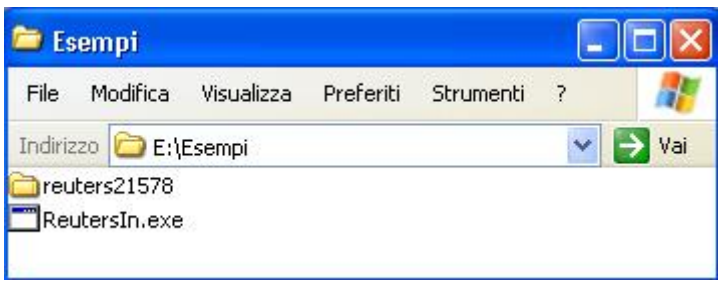
```
TestReturn = D__S.SetPos(SQL_ADD,NULL);
```

9) Infine si procede alla chiusura della connessione ed alla deallocazione dell'environment (si osservi che ogni qualvolta un oggetto della libreria non era più necessario si è dissociato da **Per** per mezzo del membro **freeConnessione()**).

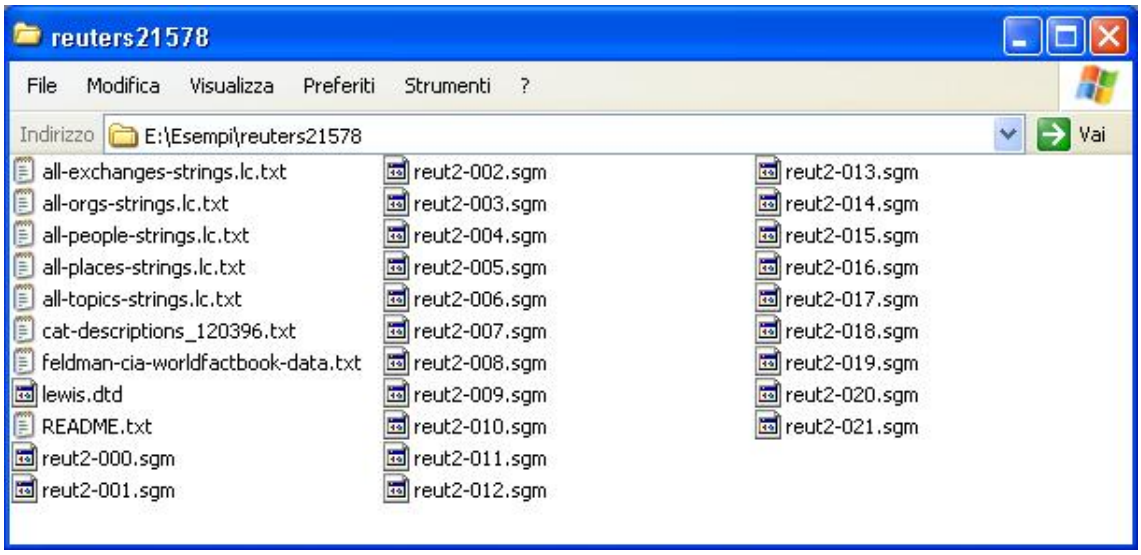
```
TestReturn = Per.Disconnect();
…
TestReturn = Per.Deallocahenv();
```

8.1.2 Utilizzo

Per utilizzare il modulo basta creare una directory di nome "reuters21578" nello stesso path dell'eseguibile **ReutersIn.exe**.

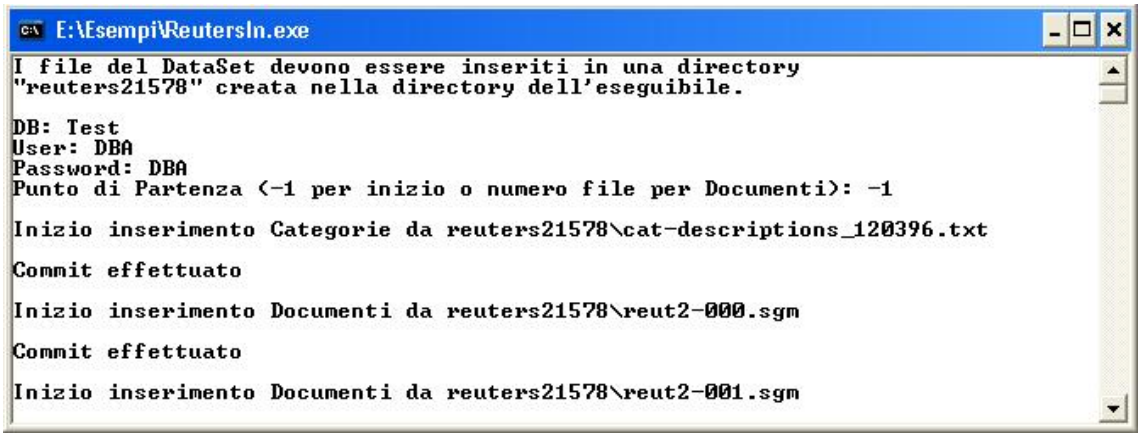


e poi scompattare al suo interno il file contenente la collection (“Archivio File\(01) reuters21578\da UCI KDD Archive\ reuters21578.tar.gz”)



Fatto ciò sarà sufficiente mandare in esecuzione **ReutersIn.exe** che farà partire una shell dos. Inizialmente sono richiesti i dati relativi alla fonte ODBC adoperata (si veda il *paragrafo 6.8*), il nome dell’user e la usa password.

Infine si richiede se iniziare l’inserimento del benchmark dall’inizio (inserendo il valore ***–1***) o da un particolare file “*reut2-???.sgm*” (inserendo il numero relativo al file). La seconda opzione è stata molto utile in fase di debug, giacché, dopo aver corretto l’eventuale errore, permetteva di ripristinare l’esecuzione dall’ultimo commit effettuato.



Durante l’esecuzione del modulo si restituiscono alcune informazioni riguardo al file attualmente in elaborazione. Nel caso si dovesse verificare un errore, la sua descrizione sarà presentata all’utente, il quale potrà scegliere come comportarsi.

Si desidera Continuare(G) o uscire con un RollBack(R) o Commit(C)? [G/R/C]

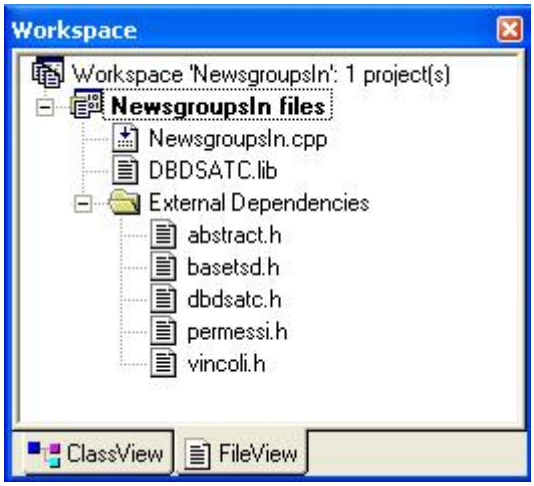
Si consiglia in tale situazione di uscire con un RollBack dall’esecuzione del modulo, di risolvere il problema e di ripristinare l’esecuzione inserendo alla richiesta del punto di partenza: ***–1***, se nell’esecuzione precedente dopo il segnale “*inizio inserimento*

categorie" non si è avuto quello di "*Commit effettuato*", oppure un numero tra ***0*** e ***21*** (il cui valore corrisponde all'ultimo file da cui si è cominciato ad inserire dati ma sul quale non si è effettuato il commit).

8.2 NewsgroupsIn

Il procedimento seguito per realizzare il modulo è simile a quello effettuato per la collection Reuters. Il workspace adoperato, relativo all'ambiente di sviluppo **Microsoft Visual C++ 6.0,** è contenuto nella directory "*GestioneDS\NewsgroupsIn* ".

Per realizzare il modulo basterebbe considerare solo il file *NewsgroupsIn.cpp,* contenuto nella directory *"Progettazione DB\Codice\NewsgroupsIn",* che include a sua volta l'header **DBDSATC.h** (al cui interno vi sono i richiami agli altri header della libreria e a quelli presentati nel *paragrafo 7.2*). Naturalmente si dovrà anche inserire nell'ambiente di sviluppo il file della libreria **DBDSATC.lib**.



Il modulo, durante la sua fase di esecuzione, richiederà di poter accedere ad alcuni file, definiti internamente al codice per mezzo di alcune direttive:

```
//Le directory vanno inserite nella stessa directory dell'eseguibile
//File contenente le categorie
#define FileCategorie "gruppi.txt"
//File contenente gli errori di calcolo delle Linee e delle lunghezze delle Keywords
#define FileLog "Error.log"
//File contenente lo Stato delle sovracategorie
#define FileStato "Stato.dat"
//File contenenti i documenti associati alle categorie
//"nome categoria".txt
//Directory con i file dei vari Split
#define Split_All "20_newsgroups\\"              //tutti i file della collection
#define Split_Mini "mini_newsgroups\\"           //verifico se il file è presente nella directory della categoria
#define Split_Date_Tr "20news-bydate-train\\"    //verifico se il file è presente nella directory della categoria
#define Split_Date_Te "20news-bydate-test\\"     //verifico se il file è presente nella directory della categoria
```

L'organizzazione in directory del banchmark ha richiesto la realizzazione di alcuni file per velocizzare l'inserimendo dei dati delle categorie e delle relazioni Documento-Split (**D__S**). Il file "**gruppi-txt**" è creato per individuare le categorie da inserire nel database; il formalismo utilizzato è:

<numero livello gerarchia (1 indica che è un nodo principale)> <\n>

<nome categoria> <\n>

<descrizione categoria (obbligatoria)> <\n>

<"******Inserire Dati****" (inserito solo se la categoria ha dei documenti) > <\n>

Di seguito si riporta l'inizio del file:

```
1
alt
Elenco degli argomenti disponibili
2
alt.atheism
Ateismo
***Inserire Dati***
1
comp
Hardware, software, info per i consumatori
```

I file **"nome categoria".txt** sono creati tramite l'utilizzo di "*Crea_Liste.bat*" e contengono i risultati di un comando dir effettuato sulla directory della categoria di interesse (il funzionamento sarà spiegato nel *paragrafo 8.2.2*).

Il file "**Error.log**" conterrà, a fine esecuzione, le informazioni sulle incongruenze tra il numero di linee del testo calcolate ed inserite nel database per mezzo della classe ***Documento*** e quelle presenti nel campo *Lines* della collezione.

Il file "**Stato.dat**" è gestito, dal *modulo NewsgroupsIn.exe*, per ripristinare l'esecuzione dell'operazione di feeding dall'ultimo commit effettuato.

8.2.1 Descrizione

Il codice adopera le classi messe a disposizione dalla libreria DBDSATC, gran parte di esso è ripetitivo ed utilizzato per gestire eventuali errori. L'inserimento dei dati nel database avviene per mezzo di transazioni che saranno terminate in alcuni punti del codice con un commit (*Per.TransCommit()*) e col salvataggio di uno stato che permetterà, nel caso d'errori, di recuperare l'esecuzione dall'ultimo commit effettuato (nel flusso che sarà descritto si trascura la descrizione di tale funzionalità).

I primi sei passi del flusso sono quasi del tutto identici a quelli effettuati nel *paragrafo 8.1.1*, di seguito quindi si procederà con la sola descrizione dei dati inseriti in questi passi:

Elemento BENCHMARK_COLLECTION	
Nome_Collection	20 Newsgroups
Versione	
Data_Collection	09-set-99
Autore_Collection	Ken Lang; Tom Mitchell
Sito	UCI KDD Archive
URL	http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
Modulo_di_feeding	NewsgroupsIn
Note_Documenti	Messages posted to Usenet newsgroups
Note_Categorie	Categories are the newsgroups
Descrizione_ID1	Identifica il numero del File che contiene il documento, si noti che tale numero non è univoco poiché in realtà dipende dalla categoria a cui appartiene il Documento e che tale informazione viene inserita per completezza.
Descrizione _ID2	Non adoperato
Descrizione__Collection	I campi dei documenti sono da interpretare come: PROVENIENZA: rappresenta il campo "Organization" AUTORE_DOCUMENTO: rappresenta il campo "From" TITOLO_DOCUMENTO: rappresenta il campo "Subject" LINEE: rappresenta il campo "Lines" (nel Data Set originale ci sono alcuni errori)

Elementi FILE		
NOME_COLLECTION	**NOME_FILE**	**DESCRIZIONE_FILE**
20 Newsgroups	20_newsgroups.tar.gz	File contenente il Data Set originale
20 Newsgroups	20news-bydate.tar.gz	Questo file è scaricabile all'indirizzo http://www.ai.mit.edu/~jrennie/20Newsgroups/index.html. In tale file i documenti sono ordinati per data e divisi in un training(60%) e test(40%) sets, non includono cross-posts (duplicates) e newsgroup-identifying headers (Xref, Newsgroups, Path, Followup-To, Date).
20 Newsgroups	mini_newsgroups.tar.gz	A tarred and gzipped subset of the Newsgroup data which contains 100 randomly selected messages from each newsgroup. This is a useful dataset for learning to use Rainbow

Elementi SPLIT			
NOME_COLLECTION	**NOME_SPLIT**	**TIPO_SPLIT**	**DESCRIZIONE_SPLIT**
20 Newsgroups	20 Newsgroups by date Te	TEST SET	Il Test Set contiene il 40% dei documenti della collection. Rimuove alcuni duplicati e campi. Tale split è recuperabile da Jason Rennie all' URL www.ai.mit.edu/~jrennie/20Newsgroups/.
20 Newsgroups	20 Newsgroups by date Tr	TRAINING SET	Il Training Set contiene il 60% dei documenti della collection. Rimuove alcuni duplicati e campi. Tale split è recuperabile da Jason Rennie all' URL www.ai.mit.edu/~jrennie/20Newsgroups/.

Elementi SPLIT			
NOME_COLLECTION	**NOME_SPLIT**	**TIPO_SPLIT**	**DESCRIZIONE_SPLIT**
20 Newsgroups	All 20 Newsgroups	DATA SET	Contiene tutti i documenti della collection 20 Newsgroups
20 Newsgroups	Mini 20 Newsgroups	DATA SET	Un tarred e gzipped subset del Newsgroup data che contiene 100 messaggi selezionati a caso per ogni newsgroup. Questo split è usualmente adoperato per gli addestramenti.

Elementi TOPIC			
NOME_COLLECTION	**NOME_TOPIC**	**TIPO_TOPIC**	**DESCRIZIONE_TOPIC**
20 Newsgroups	Newsgroups	CATEGORIE	Insieme di categorie di newsgroups (le descrizioni sono state prelevate da Google)
20 Newsgroups	Newsgroups Keywords	KEYWORDS	Insieme di Keyword adoperate dai documenti della collection 20 NewsGroups

Elementi DATA_SET		
NOME_DATA_SET	**TIPO_DATASET**	**DESCRIZIONE_DATA_SET**
20 Newsgroups by date	DPC	I documenti sono ordinati per data e divisi in un training(60%) e test(40%) sets, non includono cross-posts (duplicates)
All 20 Newsgroups	DPC	Contiene tutti i documenti della collection 20 Newsgroups associati a tutte le categorie.
Mini 20 Newsgroups	DPC	Contiene un sottoinsime di documenti della collection 20 Newsgroups associati a tutte le categorie.

Elementi T__DS	
NOME_DATA_SET	**NOME_TOPIC**
20 Newsgroups by date	Newsgroups
All 20 Newsgroups	Newsgroups
Mini 20 Newsgroups	Newsgroups
20 Newsgroups by date	Newsgroups Keywords
All 20 Newsgroups	Newsgroups Keywords
Mini 20 Newsgroups	Newsgroups Keywords

Elementi S__DS		
NOME_DATA_SET	**NOME_SPLIT**	**TIPO_SPLIT**
20 Newsgroups by date	20 Newsgroups by date Te	TEST SET
20 Newsgroups by date	20 Newsgroups by date Tr	TRAINING SET
All 20 Newsgroups	All 20 Newsgroups	DATA SET
Mini 20 Newsgroups	Mini 20 Newsgroups	DATA SET

7) Inserimento delle Categorie, Documenti e varie associazioni

Gli inserimenti delle categorie e dei documenti sono molto legati tra loro. Per mezzo del file "**gruppi.txt**" si procede all'individuazione delle categorie, delle loro sovracategorie e dell'esistenza di documenti attinenti ad esse:

```
/* Verifica Esistenza Categoria */
TestReturn = Categoria.TransRicerca();
…
        /* Inserzione nuova categoria o nuova descrizione */
        TestReturn = Categoria.SetPos(SQL_ADD,NULL);
        …
        if (!trovato) {
                /* Prelievo ID nuova categoria */
                TestReturn = Categoria.Preleva(SQL_FETCH_LAST,NULL);
…
/* Inserzione eventuale Sovracategoria */
```

```
if (ID_Liv_Cat[Liv_Cat- 1] != numNoUtil) {
TestReturn = Categoria.InsGerarchia(true,ID_Liv_Cat[Liv_Cat- 1]);
…
//Inserzione delle associazioni in DPC
DPC.Value.ID_CATEGORIA = Categoria.Value.ID_CATEGORIA;
strcpy((char*)DPC.Value.NOME_TOPIC,"Newsgroups");
…
if (strcmp(Buffer,"***Inserire Dati***\n") == 0) {
…
```

Nel caso in cui la categoria presenta dei documenti annessi si procede con l'apertura prima del file contenente le informazioni sulla directory della categoria (ad esempio "*sci.electronics.txt*") e poi di ognuno dei file, contenenti i dati di un particolare documento, riportati al suo interno (ad esempio "*53974*"):

```
From: pauls@trsvax.tandy.com
Date: 21 Apr 93 09:36 CDT
Newsgroups: sci.electronics
Subject: Re: Need source for old Radio Shack ste
Message-ID: <288200083@trsvax>
Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!euro [continua…]
Nf-ID: #R:acs.ucalgary.ca:27323:trsvax:288200083:000:125
Nf-From: trsvax.tandy.com!pauls  Apr 21 09:36:00 1993
References: <27323@acs.ucalgary.ca>
Lines: 5


 It's made by Rohm. (as is all BAxxx parts). Call 714-855-2131 and ask if
 you can get a sample (it's only like a $2 part).
```

L'organizzazione non sequenziale degli attributi ha richiesto di verifica, ad ogni prelievo di una riga, se l'attributo contenutovi appartenesse a quelli considerati significativi per i documenti della collection. L'utilizzo del prototipo del modulo (presente nella directory "*Progettazione DB\Prototipi\NewsgroupsIn*") e l'analisi dei dati prelevati ogni qual volta l'esecuzione del codice, in modalità di debug, presentava qualche errore inatteso, ha permesso di individuare alcuni errori. Di seguito si riportano le principali osservazioni a riguardo:

a) Durante il prelievo delle Keywords di un documento può capitare di trovarne alcune riportate più di una volta. Tal errore ha richiesto delle verifiche sulle liste di keywords ricavate per ogni documento.
b) Il numero di LINEE calcolato per mezzo del membro **SettaLINEE()** della classe ***Documento*** non coincide sempre con quello riportato nel campo Lines dei documenti (quest'ultimo a volte risulta essere errato). Tale problema è

risolto inserendo nel database il numero di linee corretto e creando il file "**Error.log**" che contiene le discrepanze tra i due valori.

```
…
Nome_Categoria: comp.graphics
Numero File: 38753
Lines: 139        LineeNew: 138

Nome_Categoria: comp.graphics
Numero File: 38848
Lines: non presente       LineeNew: 209
…
```

c) Alcuni attributi hanno riscontrato valori di dimensioni superiori a quelle attese. Fin quando è stato possibile si è risolto il problema aumentando le dimensioni dei campi, ma una volta raggiunti valori eccessivi si è preferito non inserire i particolari dati che avrebbero richiesto un ulteriore ritocco delle dimensioni. Le informazioni, riguardanti questi casi, sono state inserite nel file "**Error.log**", che con gran sorpresa a fine esecuzione ha permesso di riscontrare il verificarsi di uno solo di questi casi.

```
Keywords troppo lunga: proportional representation majority europe european parliament
election voting ballots Britain Italy Holland Belgium Luxemburg Germany Spain France
Greece Portugal Denmark EEC EC EP commission
Nome_Categoria: talk.politics.misc
Numero File: 178988
```

Durante l'estrazione dei dati si procede anche alla ricerca (e nel caso di assenza, inserzione) delle Keywords associate al particolare documento in modo da poter realizzare un elenco di valori relativi agli ID_CATEGORIA da inserire nella tabella *Classificazione*. Per il problema riscontrato al punto **a)** ogni qual volta s'inserisce un nuovo ID_CATEGORIA occorre verificare se questo è già presente.

```
/* Verifica Esistenza Categoria */
TestReturn = Categoria.Riesegui();        //si riferisce a TransRicerca()
…
/* Inserimento Keywords */
if (TestReturn == NO_DATO) {
  strcpy((char*)Categoria.Value.NOME_CATEGORIA,(char*)Categoria.FI.NOME_CATEGORIA);
  strcpy((char*)Categoria.Value.DESCRIZIONE_CATEGORIA,strNoUtil);
  Categoria.Value.ID_CATEGORIA = numNoUtil;

  /* Inserzione nuova categoria */
  TestReturn = Categoria.SetPos(SQL_ADD,NULL);
  …
  /* Prelievo ID nuova categoria */
  TestReturn = Categoria.Preleva(SQL_FETCH_LAST,NULL);
  …
```

```
}// fine inserimento

//Inserzione delle associazioni in DPC
DPC.Value.ID_CATEGORIA = Categoria.Value.ID_CATEGORIA;
strcpy((char*)DPC.Value.NOME_TOPIC,"Newsgroups Keywords");

TestReturn = DPC.Riesegui();              //si riferisce a TransInserisci()
if (TestReturn == ERRORE_SQL) {
   strTmp = Per.getErrore();
   //Controllo verifica falso Errore a causa di reciclo di Categoria già presente
   if (strcmp((char *)Per.getSQLSTATE(),"23000") != 0) {
…
//Conservazione dell'ID (capita che siano riportati più di una volta)
…
```

Estratti i dati si procede all'inserzione del documento ed al prelievo del suo identificativo in modo da poterlo utilizzare in futuro per inserire le tuple delle relazioni individuate.

```
/*Inserimento del Documento*/
TestReturn = Documento.SetPos(SQL_ADD,NULL);
…
/* Prelievo ID del documento */
TestReturn = Documento.Preleva(SQL_FETCH_LAST,NULL);
…
/*Inserimento associazione Collection-Documento*/
D__B.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
TestReturn = D__B.SetPos(SQL_ADD,NULL);
…
/* Verifica numero linee */
if (Lines == numNull) {
   …
else if (Lines != Documento.Value.LINEE) {
   …
}
/*Inserimento associazione Categorie-Documento*/
Classificazione.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
Classificazione.Value.ID_CATEGORIA = tmp_ID_CATEGORIA;

TestReturn = Classificazione.SetPos(SQL_ADD,NULL);
…
/* Inserimento associazioni con le Keywords */
for (i = 0; i < numKey; i++) {
   Classificazione.Value.ID_CATEGORIA = Key[i];
   TestReturn = Classificazione.SetPos(SQL_ADD,NULL);
…
}
/*Inserimento associazione Split-Documento*/
D__S.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
strcpy((char*)D__S.Value.NOME_SPLIT,"All 20 Newsgroups");
strcpy((char*)D__S.Value.TIPO_SPLIT,"DATA SET");
```

```
TestReturn = D__S.SetPos(SQL_ADD,NULL);
…
```

Infine per verificare se un documento appartiene ad un particolare Split (diverso da quello principale "All 20 Newsgroups" la cui associazione è già stata riportata nel codice sopraindicato) si adoperano le direttive che permettono di definire i file degli split (ad esempio ***#define Split_Mini "mini_newsgroups****"*) e si prova ad aprire il file per appurarne l'esistenza.

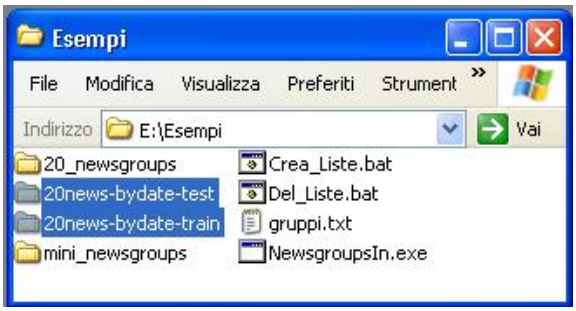
```
_itoa(D__B.Value.ID1_ORIGINAL_COLLECTION,Buffer,10);

/*Controllo appartenenza allo Split Mini*/
strTmp2 = Split_Mini;
strTmp2 += tmp_NOME_CATEGORIA;
strTmp2 += "\\";
strTmp2 += Buffer;

/*Apertura File del documento*/
fileDoc = fopen(strTmp2.data(),"r");
if (fileDoc != NULL) {

  strcpy((char*)D__S.Value.NOME_SPLIT,"Mini 20 Newsgroups");
  strcpy((char*)D__S.Value.TIPO_SPLIT,"DATA SET");

  TestReturn = D__S.SetPos(SQL_ADD,NULL);
  …
```

8) Infine si procede alla chiusura della connessione ed alla deallocazione dell'environment così com'effettuato nel passo **9)** del flusso del modulo precedente.

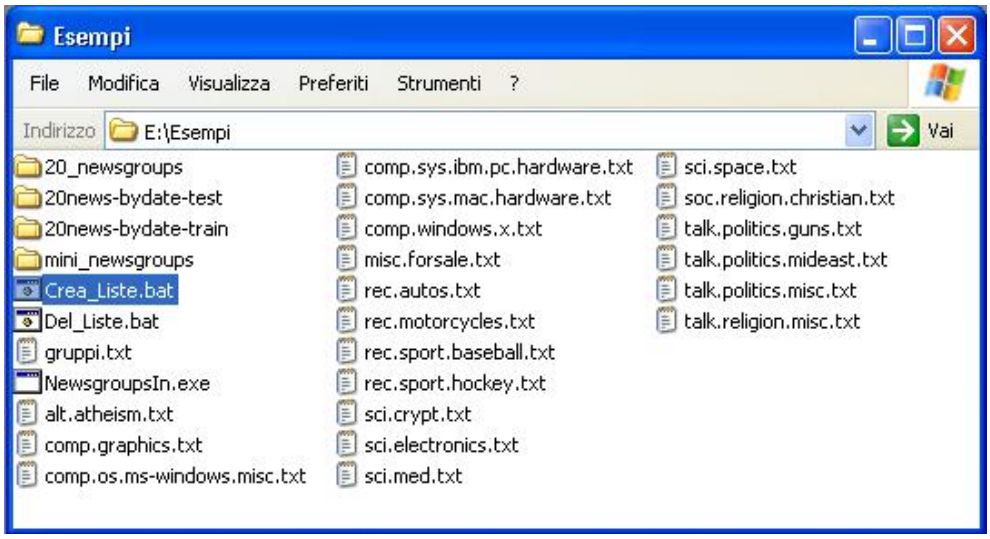
8.2.2 Utilizzo

Per utilizzare il modulo si devono scompattare i file contenenti la collection (presenti nella directory "*\Archivio File\(03) 20 Newsgroups collection*") nello stesso path dell'eseguibile **NewsgroupsIn.exe**:

\da UCI KDD Archive\20_newsgroups.tar.gz"	Da inserire interamente
\da UCI KDD Archive\mini_newsgroups.tar.gz	
\MIT Artificial Intelligence Lab - Jason Renne\20news-bydate.tar.gz	Si devono inserire solo le cartelle evidenziate in figura

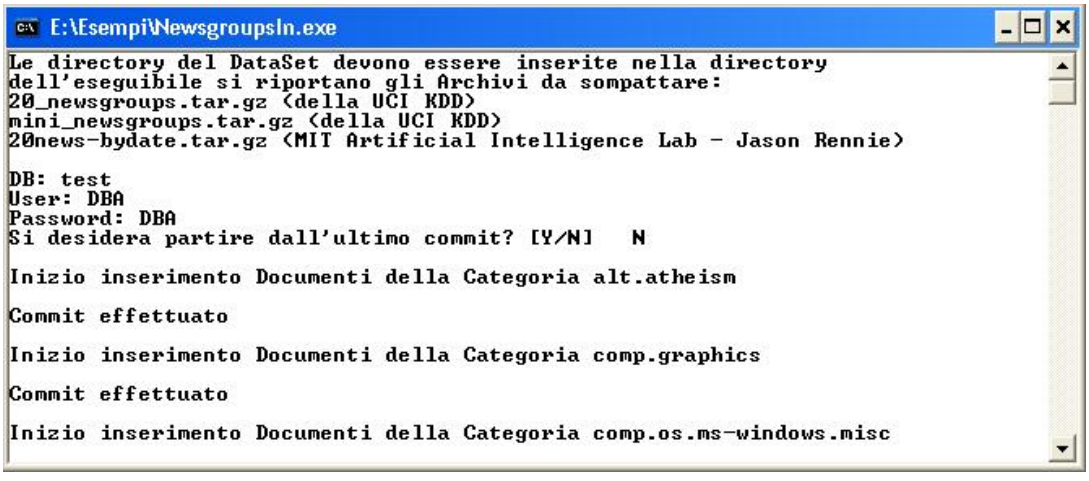


In seguito mandando in esecuzione "*Crea_Liste.bat*" si realizzano i file contenenti le informazioni recuperate tramite il comando **dir** effettuato sulle directory presenti nella cartella **20_newsgroups**.



Fatto ciò sarà sufficiente mandare in esecuzione **NewsgrupsIn.exe** che farà partire una shell dos. Inizialmente sono richiesti i dati relativi alla fonte ODBC adoperata (si veda il *paragrafo 6.8*), il nome dell'user e la usa password.

Infine si richiede se iniziare l'inserimento del benchmark dall'ultimo commit (inserendo il valore ***Y***) o dall'inizio (inserendo ***N***). La prima scelta è stata molto utile in fase di debug giacché dopo aver corretto l'eventuale errore, permetteva di ripristinare l'esecuzione dall'ultimo commit effettuato.



Durante l'esecuzione del modulo si restituiscono alcune informazioni riguardo al file attualmente in elaborazione. Nel caso si dovesse verificare un errore, la sua descrizione sarà presentata all'utente, il quale potrà scegliere come comportarsi.

Si desidera Continuare(G) o uscire con un RollBack(R) o Commit(C)? [G/R/C]

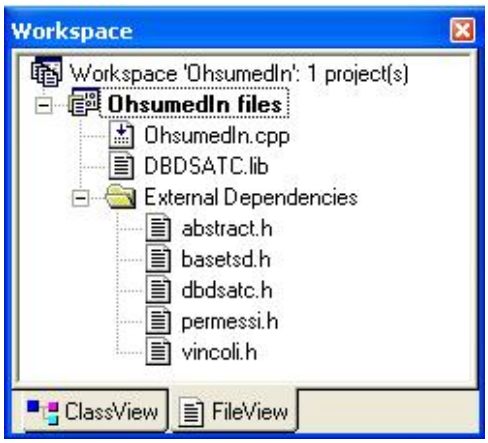
Si consiglia in tale situazione di uscire con un RollBack dall'esecuzione del modulo, di risolvere il problema e di ripristinare l'esecuzione inserendo alla richiesta del punto di partenza ***Y*** (si osservi che in seguito all'interruzione del modulo viene creato all'interno della directory il file **"Stato.dat"** necessario per il ripristino dello stato relativo all'ultimo commit).

Il modulo inoltre restituisce il file "**Error.log**" di cui si è discusso nel paragrafo precedente. Nel caso d'interruzione e riavvio i nuovi risultati saranno appesi a quelli precedenti. Tramite l'utilizzo del sorgente "*numErrori.cpp*" presente nella directory "*\Progettazione DB\Codice\NewsgroupsIn*" è stato possibile calcolare che vi sono 310 errori (il file compilato **numErrori.exe** deve trovarsi nella stessa cartella del file "**Error.log**").

8.3 OhsumedIn

Il procedimento seguito per realizzare il modulo è simile a quello effettuato nei casi precedenti. Il workspace adoperato, relativo all'ambiente di sviluppo **Microsoft Visual C++ 6.0,** è contenuto nella directory *"GestioneDS\OhsumedIn"*.

Per realizzare il modulo basterebbe considerare il sorgente *OhsumedIn.cpp,* contenuto nella directory *"Progettazione DB\Codice\ OhsumedIn"*, che include a sua volta l'header **DBDSATC.h** (al cui interno vi sono i richiami agli altri header della libreria e a quelli presentati nel *paragrafo 7.2*). Naturalmente si dovrà anche inserire nell'ambiente di sviluppo il file della libreria **DBDSATC.lib**.



Il modulo durante la sua fase di esecuzione richiederà di poter accedere ad alcuni file, definiti internamente al codice tramite l'utilizzo di alcune direttive:

```
//La directory del Data Set va inserita nella stessa directory dell'eseguibile
//File contenenti le Query
#define QueryOHSU_pre_test "ohsu-trec\\pre-test\\query.ohsu.test.1-43"
#define QueryMeSH_pre_test "ohsu-trec\\pre-test\\query.mesh.test.1-119"
#define QueryOHSU "ohsu-trec\\trec9-train\\query.ohsu.1-63"
#define QueryMeSH "ohsu-trec\\trec9-train\\query.mesh.1-4904"
#define QueryMeSH_subset "ohsu-trec\\sample.map"     //richiama le sigle di QueryMeSH
//tutti i Documenti del Training Set
#define Split_Training "ohsu-trec\\trec9-train\\ohsumed.87"
//i documenti del Test Set a cui si aggiungono le ultime due cifre dell'anno
#define Split_Test "ohsumed."
//File contenenti associazioni Query-Documenti
#define AssOHSU_pre_test "ohsu-trec\\pre-test\\qrels.ohsu.test.87"
#define AssMeSH_pre_test "ohsu-trec\\pre-test\\qrels.mesh.test.87"
#define AssOHSU_Batch_Tr "ohsu-trec\\trec9-train\\qrels.ohsu.batch.87"
#define AssMeSH_Batch_Tr "ohsu-trec\\trec9-train\\qrels.mesh.batch.87"
#define AssOHSU_Adapt_Tr "ohsu-trec\\trec9-train\\qrels.ohsu.adapt.87"
                    //usato per definire lo split OHSU adaptive filtering
#define AssMeSH_Adapt_Tr "ohsu-trec\\trec9-train\\qrels.mesh.adapt.87"
                    //usato per definire lo split MeSH adaptive filtering
#define AssOHSU_Te "ohsu-trec\\trec9-test\\qrels.ohsu.88-91"
#define AssMeSH_Te "ohsu-trec\\trec9-test\\qrels.mesh.88-91"
//File contenente lo Stato di avanzamento
#define FileStato "Stato.dat"
```

Il file "**Stato.dat**" è gestito, dal *modulo OhsumedIn.exe*, per ripristinare l'esecuzione dell'operazione di feeding dall'ultimo commit effettuato.

8.3.1 Descrizione

Il codice adopera le classi messe a disposizione dalla libreria DBDSATC, gran parte di esso è ripetitivo ed utilizzato per gestire eventuali errori. L'inserimento dei dati nel database avviene per mezzo di transazioni che saranno completate in alcuni punti del codice con un commit (*Per.TransCommit()*) e col salvataggio di uno stato che permetterà, nel caso d'errori, di recuperare l'esecuzione dall'ultimo commit effettuato (nel flusso che sarà descritto si trascura la descrizione di tale funzionalità).

I primi sei passi del flusso sono quasi del tutto identici a quelli effettuati per i due benchmark precedenti, quindi la loro descrizione sarà effettuata riportando solo l'elenco dei dati inseriti in questi passi:

Elemento BENCHMARK_COLLECTION	
Nome_Collection	OHSUMED
Versione	
Data_Collection	06-lug-94
Autore_Collection	William Hersh
Sito	Text REtrieval Conference
URL	http://trec.nist.gov/data.html
Modulo_di_feeding	ohsumedIn
Note_Documenti	Subset of the Medline document base
Note_Categorie	Categories are the postable terms of the MeSH thesaurus
Descrizione_ID1	Sequential identifier (important note: documents should be processed in this order).
Descrizione _ID2	MEDLINE identifier (UI) (<DOCNO> used for relevance judgments).
Descrizione__Collection	I campi dei documenti sono da interpretare come: PROVENIENZA: rappresenta il campo "Source" DATA_DOCUMENTO: è rappresentativo solo l'anno TESTO: rappresenta il campo "Abstract"

Elementi FILE		
NOME_COLLECTION	**NOME_FILE**	**DESCRIZIONE_FILE**
OHSUMED	filtering.tar.gz	File contenente il Data Set
OHSUMED	ohsumed.88	File, contenente i documenti del 1988, prelevato da "ftp://medir.ohsu.edu/pub/ohsumed/"
OHSUMED	ohsumed.89	File, contenente i documenti del 1989, prelevato da "ftp://medir.ohsu.edu/pub/ohsumed/"
OHSUMED	ohsumed.90	File, contenente i documenti del 1990, prelevato da "ftp://medir.ohsu.edu/pub/ohsumed/"
OHSUMED	ohsumed.91	File, contenente i documenti del 1991, prelevato da "ftp://medir.ohsu.edu/pub/ohsumed/"
OHSUMED	readme.txt	File contenente informazioni sul Data Set e sulla sua organizzazione

Elementi SPLIT			
NOME_COLLECTION	**NOME_SPLIT**	**TIPO_SPLIT**	**DESCRIZIONE_SPLIT**
OHSUMED	TREC-9 Test Set	TEST SET	Il Test Set si rifa ai riferimenti della MEDLINE tra il 1988-1991
OHSUMED	TREC-9 Training Set	TRAINING SET	Indica il Training Set adoperato dalla TREC-9 per il batch ed il routing filtering, l'adaptive filtering e per i pre-test. Il Training Set si rifa ai riferimenti della MEDLINE del 1987. Si noti che il Training Set è anche adoperato per dei pre-test

Elementi TOPIC			
NOME_COLLECTION	**NOME_TOPIC**	**TIPO_TOPIC**	**DESCRIZIONE_TOPIC**
OHSUMED	MeSH pre-test	QUERY	set of 119 MeSH test topics
OHSUMED	MeSH terms	KEYWORDS	A subset of the MeSH 2000 (Human-assigned MeSH terms)
OHSUMED	OHSU pre-test	QUERY	Set of 43 OHSU test topics
OHSUMED	Subset TREC-9 MeSH	QUERY	subset of the TREC-9 MeSH topics
OHSUMED	TREC-9 MeSH	QUERY	set of 4904 TREC-9 MeSH topics
OHSUMED	TREC-9 OHSU	QUERY	set of 63 TREC-9 OHSU topics

Elementi DATA_SET		
NOME_DATA_SET	**TIPO_DATASET**	**DESCRIZIONE_DATA_SET**
MeSH adaptive filtering	ADAPTIVE	
OHSU adaptive filtering	ADAPTIVE	Il Topic MeSH term è inserito come elenco di termini Thesaurus dei documenti
MeSH batch&routing filtering	CPC	
MeSH pre-test	CPC	Non viene adoperato per ottimizzare i parametri del sistema, ma giusto per generare dei test per assicurarsi che il sistema funzioni correttamente.
OHSU batch&routing filtering	CPC	Il Topic MeSH term è inserito come elenco di termini Thesaurus dei documenti
OHSU pre-test	CPC	Non viene adoperato per ottimizzare i parametri del sistema, ma giusto per generare dei test per assicurarsi che il sistema funzioni correttamente. Il Topic MeSH term è inserito come elenco di termini Thesaurus dei documenti

Elementi T__DS	
NOME_DATA_SET	**NOME_TOPIC**
MeSH adaptive filtering	TREC-9 MeSH
MeSH batch&routing filtering	TREC-9 MeSH
MeSH pre-test	MeSH pre-test
OHSU adaptive filtering	MeSH terms
OHSU adaptive filtering	TREC-9 OHSU
OHSU batch&routing filtering	MeSH terms
OHSU batch&routing filtering	TREC-9 OHSU
OHSU pre-test	MeSH terms
OHSU pre-test	OHSU pre-test

Elementi S__DS		
NOME_DATA_SET	**NOME_SPLIT**	**TIPO_SPLIT**
MeSH adaptive filtering	TREC-9 Test Set	TEST SET
MeSH batch&routing filtering	TREC-9 Test Set	TEST SET
OHSU adaptive filtering	TREC-9 Test Set	TEST SET
OHSU batch&routing filtering	TREC-9 Test Set	TEST SET
MeSH adaptive filtering	TREC-9 Training Set	TRAINING SET
MeSH batch&routing filtering	TREC-9 Training Set	TRAINING SET
MeSH pre-test	TREC-9 Training Set	TRAINING SET
OHSU adaptive filtering	TREC-9 Training Set	TRAINING SET
OHSU batch&routing filtering	TREC-9 Training Set	TRAINING SET
OHSU pre-test	TREC-9 Training Set	TRAINING SET

I passi seguenti differiscono dalle collezioni precedenti non solo perché si riferiscono a Query e non a Categorie, ma soprattutto perché le relazioni tra le entità trattate sono individuate tramite file assestanti e non, come nei due casi precedenti, da informazioni interne ai documenti o relative alla loro organizzazione in directory.

7) Inserimento delle Query

Si procede all'apertura, ad uno ad uno, dei file dei singoli Topic (utilizzando le direttive presentate nel *paragrafo 8.3*) in modo da potervi prelevare i dati delle Query da inserire nel database:

```
/* Inserzione nuova query */
TestReturn = Query.SetPos(SQL_ADD,NULL);
…
/* Prelievo ID nuova query */
TestReturn = Query.Preleva(SQL_FETCH_LAST,NULL);
…
//Inserisco l'associazione in CPC
CPC.Value.ID_QUERY = Query.Value.ID_QUERY;

TestReturn = CPC.SetPos(SQL_ADD,NULL);
```

Per quanto riguarda il "*Subset TREC-9 MeSH*" si procede aprendo il file individuato dalla direttiva *#define QueryMeSH_subset "ohsu-trec\\sample.map"* ed eseguendo su ogni sua riga i comandi presentati di seguito:

```
strTmp = Buffer;
iPos1 = strTmp.find_first_of("\t",0);
strTmp2 = strTmp.substr(iPos1 +1,strTmp.size() - iPos1 -2);      //Sigla MeSH_Topic
strTmp =  strTmp.substr(0,iPos1);                                //Sigla MeSH_Sub_Topic

/*Ricerca dell'associazione con la sigla di MeSH_Topic*/
strcpy((char*)CPC.FI.SIGLA,strTmp2.data());
TestReturn = CPC.Riesegui();                                     //si riferisce a TransRicerca()
…
//Inserisco l'associazione con la sigla MeSH_Sub_Topic*/
strcpy((char*)CPC.Value.SIGLA,strTmp.data());
strcpy((char*)CPC.Value.NOME_TOPIC,"Subset TREC-9 MeSH");
TestReturn = CPC.SetPos(SQL_ADD,NULL);
```

8) Inserimento dei documenti, dei MesH term e delle associazioni documenti-Split

A questo punto si procede all'esecuzione, su ogni file che sarà definito per mezzo delle direttive **#define Split_Training** e **#define Split_Test** (si veda la fine del *paragrafo 8.3*), di un ciclo che permetterà di inserire i dati dei documenti e le associazioni alle tuple delle entità **Collection** e **Split**..

Inanzitutto si procede coll'estrazione delle informazioni del documento. Un esempio di formattazione dei dati di un documento (i cui campi sono descritti nel *paragrafo 4.2*) è riportato di seguito:

```
.I 54711
.U
88000001
.S
Alcohol Alcohol 8801; 22(2):103-12
.M
Acetaldehyde/*ME; Buffers; Catalysis; HEPES/PD; Nuclear  [continua…]
.T
The binding of acetaldehyde to the active site of [continua…]
.P
JOURNAL ARTICLE.
.W
Ribonuclease A was reacted with [1-13C,1,2-14C]acetaldehyde  [continua…]
.A
Mauch TJ; Tuma DJ; Sorrell MF.
```

Nonostante l'organizzazione sequenziale degli attributi, è stato necessario, poiché tramite il prototipo si è osservata in alcuni casi la mancanza di alcuni di questi, effettuare delle verifiche dell'identità dei dati prelevati al fine di non causare degli errori nell'inserirli nel database. L'uso di un prototipo del modulo (presente nella directory "*Progettazione DB\Prototipi\OhsumedIn*") e l'analisi dei dati prelevati ogni qual volta l'esecuzione del codice, in modalità di debug, presentava qualche errore inatteso, ha consentito di verificare non solo la sottostima delle dimensioni dei campi, ma anche l'individuazione degli errori che si verificavano porpio a causa della mancanza di alcuni dei campi.

Durante l'estrazione dei dati si procede anche alla ricerca (e nel caso di assenza, inserzione) delle Keywords associate al particolare documento in modo da poter realizzare un elenco di valori relativi agli ID_CATEGORIA da inserire nella tabella *Classificazione*.

```
//Verifico se il MeSH term è presente
TestReturn = Categoria.TransRicerca();
if (TestReturn == ERRORE_SQL) {
…
}
else if (TestReturn == NO_DATO) {

   strcpy((char*)Categoria.Value.NOME_CATEGORIA,(char*)Categoria.FI.NOME_CATEGORIA);
   strcpy((char*)Categoria.Value.DESCRIZIONE_CATEGORIA,strNoUtil);
   Categoria.Value.ID_CATEGORIA = numNoUtil;
```

```
   /* Inserzione nuova Keywords */
   TestReturn = Categoria.SetPos(SQL_ADD,NULL);
   …
   /* Prelievo ID nuova Keywords */
   TestReturn = Categoria.Preleva(SQL_FETCH_LAST,NULL);
   …
}// fine inserimento

/* Ricerca Presenza */
DPC.FI.ID_CATEGORIA = Categoria.Value.ID_CATEGORIA;
strcpy((char*)DPC.FI.NOME_TOPIC,"MeSH terms");
TestReturn = DPC.Riesegui();                    //si riferisce a TransRicerca();
if (TestReturn == ERRORE_SQL) {
   …
}
else if (TestReturn == NO_DATO) {
   //Inserzione delle associazioni in DPC
   DPC.Value.ID_CATEGORIA = Categoria.Value.ID_CATEGORIA;
   strcpy((char*)DPC.Value.NOME_TOPIC,"MeSH terms");
   TestReturn = DPC.SetPos(SQL_ADD,NULL);;
   …
}

//Conservazione dell'ID (capita che siano riportati più di una volta)
```

Estratti i dati si procede all'inserzione del documento ed al prelievo del suo identificativo in modo da poterlo utilizzare in futuro per inserire le tuple delle relazioni individuate.

```
/*Inserimento del Documento*/
TestReturn = Documento.SetPos(SQL_ADD,NULL);
…
/* Prelievo ID del documento */
TestReturn = Documento.Preleva(SQL_FETCH_LAST,NULL);
…
/*Inserimento associazione Collection-Documento*/
D__B.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;
TestReturn = D__B.SetPos(SQL_ADD,NULL);
…
/* Inserimento associazioni con le Keywords */
Classificazione.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;

for (int j=0; j<numCtgr; j++) {

   Classificazione.Value.ID_CATEGORIA = Ctgr[j];

   TestReturn = Classificazione.SetPos(SQL_ADD,NULL);
   …
}

/*Inserimento associazione Split-Documento*/
```

```
D__S.Value.ID_DOCUMENTO = Documento.Value.ID_DOCUMENTO;

TestReturn = D__S.SetPos(SQL_ADD,NULL);
…
```

9) Inserimento associazioni Documento-Query OHSU

```
DBDSATC::Relazioni::Relevance_judgments Judgments;
```

In questa fase si aprono i file (definiti tramite le direttive del *paragrafo 8.3*) che si riferiscono ai Relevance Judgments che associano i Documenti e le Query OHSU. Vediamo le operazioni effettuate su ogni singolo record individuato:

```
/*Prelievo dell'ID della Query*/
iPos1 = strTmp.find_first_of ("\t",0);
strTmp2 = strTmp.substr(0,iPos1);
strTmp.replace(0,strTmp2.size() +1 ,"");
if (strcmp((char*)CPC.FI.SIGLA,strTmp2.data()) != 0) {
   strcpy((char*)CPC.FI.SIGLA,strTmp2.data());
   strcpy((char*)CPC.FI.NOME_TOPIC,tmp_NOME_TOPIC.data());
   TestReturn = CPC.Riesegui();                    //si riferisce a TransRicerca()
   …
   Judgments.Value.ID_QUERY = CPC.Value.ID_QUERY;
}

/*Prelievo dell'ID del documento*/
iPos1 = strTmp.find_first_of ("\t",0);
strTmp2 = strTmp.substr(0,iPos1);
strTmp.replace(0,strTmp2.size() +1 ,"");

D__B.FI.ID2_ORIGINAL_COLLECTION = atoi(strTmp2.data());
TestReturn = D__B.Riesegui();                      //si riferisce a TransRicerca()
…
Judgments.Value.ID_DOCUMENTO = D__B.Value.ID_DOCUMENTO;

/*Prelievo del valore di relevant*/
…

/*Inserimento associazione*/
TestReturn = Judgments.SetPos(SQL_ADD,NULL);
…
```

10) Inserimento associazioni Documento-Query MeSH

Rispetto al passo precedente l'inserzione di queste associazioni viene complicata a causa della più complicata individuazione del valore del campo RELEVANT, che richiede di verificare la presenza del titolo della Query tra le Keywords del documento:

```
/*Prelievo dell'ID della Query e del Titolo della Query*/
iPos1 = strTmp.find_first_of ("\t",0);
strTmp2 = strTmp.substr(0,iPos1);
```

```
strTmp.replace(0,strTmp2.size() +1 ,"");
if (strcmp((char*)CPC.FI.SIGLA,strTmp2.data()) != 0) {
  …
  /*Prelievo dell'ID della Query*/
  strcpy((char*)CPC.FI.SIGLA,strTmp2.data());
  strcpy((char*)CPC.FI.NOME_TOPIC,tmp_NOME_TOPIC.data());

  TestReturn = CPC.Riesegui();                    //si riferisce a TransRicerca()
  …
  Judgments.Value.ID_QUERY = CPC.Value.ID_QUERY;

  /*Prelievo del Titolo della Query*/
  Query.FI.ID_QUERY = Judgments.Value.ID_QUERY;

  TestReturn = Query.Riesegui();                  //si riferisce a TransRicerca()
  …
  /*Controllo di un'eventuale presenza di un MeSH term collegato alla Query, */
  /*utile per la verifica del Relevant */
  strcpy((char*)Categoria.FI.NOME_CATEGORIA,(char*)Query.Value.TITOLO_QUERY);

  TestReturn = Categoria.Riesegui();              //si riferisce a TransRicerca()
  if (TestReturn == ERRORE_SQL) {
  …
  }
  else if (TestReturn == NO_DATO) Ass_MeSH_Query = false;
  else Ass_MeSH_Query = true;

  /* Settaggio filtro riguardante la Categoria da verificare */
  Classificazione.FI.ID_CATEGORIA = Categoria.Value.ID_CATEGORIA;

  /* Riposizionamento inizio Recordset */
  TestReturn = D__B.Preleva(SQL_FETCH_FIRST,NULL);
  …
}

/*Prelievo dell'ID del documento*/
tmpID2 = atoi(strTmp.data());

while (tmpID2 != D__B.Value.ID2_ORIGINAL_COLLECTION) {

  TestReturn = D__B.Preleva(SQL_FETCH_NEXT,NULL);
  …
}

Judgments.Value.ID_DOCUMENTO = D__B.Value.ID_DOCUMENTO;
strcpy((char*)Judgments.Value.RELEVANT,"POSSIBLY");

/*Controllo del valore di relevant*/
if (Ass_MeSH_Query) {

  Classificazione.FI.ID_DOCUMENTO = Judgments.Value.ID_DOCUMENTO;
```

```
   TestReturn = Classificazione.Riesegui();                    //si riferisce a TransRicerca()
   …
}
/*Inserimento associazione*/
TestReturn = Judgments.SetPos(SQL_ADD,NULL);
```

11) Inserimento associazioni Adaptative Filtering

```
DBDSATC::Entita::Data_Set::Adaptive_Filtering AF;
```

Gli ultimi dati da inserire sono quelli relativi alla relazione **Adaptive Filtering,** presenti, come le altre associazioni, in particolari file (definiti tramite le direttive del *paragrafo 8.3*) che varrano aperti all'occorrenza. Il codice di inserimento utilizzato è:

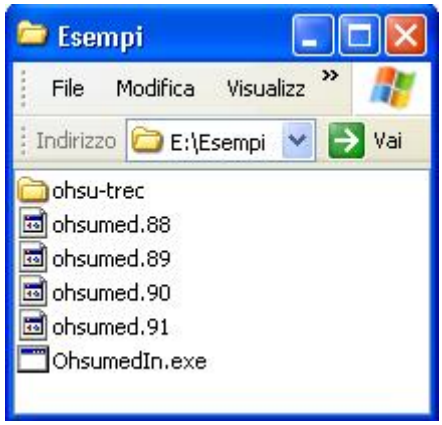
```
/* Inserimento collegamento Relevant Judjments - Data Set */
fgets(Buffer,BufLength,file);

while (!feof(file)) {

   strTmp = Buffer;

   /*Prelievo dell'ID della Query*/
   iPos1 = strTmp.find_first_of ("\t",0);
   strTmp2 = strTmp.substr(0,iPos1);
   strTmp.replace(0,strTmp2.size() +1 ,"");
   if (strcmp((char*)CPC.FI.SIGLA,strTmp2.data()) != 0) {

      strcpy((char*)CPC.FI.SIGLA,strTmp2.data());

      TestReturn = CPC.Riesegui();                             //si riferisce a TransRicerca()
      …
      AF.Value.ID_QUERY = CPC.Value.ID_QUERY;
}

/*Prelievo dell'ID del documento*/
iPos1 = strTmp.find_first_of ("\t\n",0);
strTmp2 = strTmp.substr(0,iPos1);
D__B.FI.ID2_ORIGINAL_COLLECTION = atoi(strTmp2.data());

TestReturn = D__B.Riesegui();  //si riferisce a TransRicerca()
…

AF.Value.ID_DOCUMENTO = D__B.Value.ID_DOCUMENTO;

/*Inserimento Adaptative Filtering*/
TestReturn = AF.SetPos(SQL_ADD,NULL);
…

fgets(Buffer,BufLength,file);

}// fine while
```

12) Infine si procede alla chiusura della connessione ed alla deallocazione dell'environment così com'effettuato per i moduli di feeding precedenti.

8.3.2 Utilizzo

Per utilizzare il modulo si devono scompattare i file contenenti la collection (presenti nella directory "*\Archivio File\(02) OHSUMED*") nello stesso path dell'eseguibile **OhsumedIn.exe**:

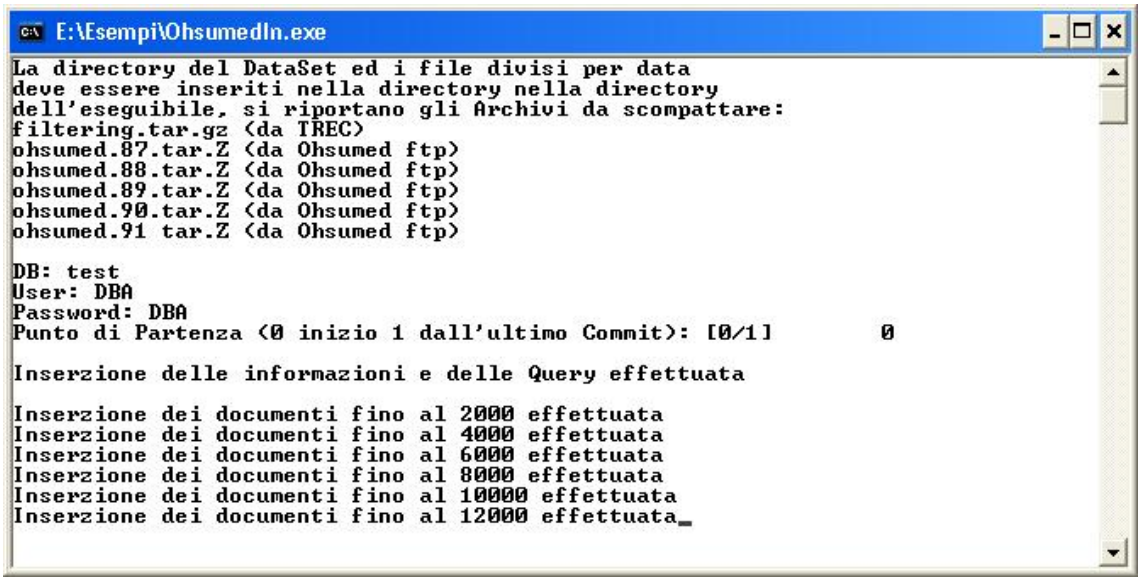
\Da TREC\filtering.tar.gz"
\Da Ohsumed\ohsumed.88.tar.Z
\Da Ohsumed\ohsumed.89.tar.Z
\Da Ohsumed\ohsumed.90.tar.Z
\Da Ohsumed\ohsumed.91.tar.Z



La directory "*ohsu-trec*" deve contenere tutti i dati contenuti nel file della collection "filtering.tar.gz " fornito dalla **TREC**.

Fatto ciò è sufficiente mandare in esecuzione **OhsumedIn.exe** che farà partire una shell dos. Inizialmente sono richiesti i dati relativi alla fonte ODBC adoperata (si veda il *paragrafo 6.8*), il nome dell'user e la usa password.

Infine si richiede se iniziare l'inserimento del benchmark dall'ultimo commit (inserendo il valore ***1***) o dall'inizio (inserendo ***0***). La prima scelta è stata molto utile in fase di debug giacché dopo aver corretto l'eventuale errore, permetteva di ripristinare l'esecuzione dall'ultimo commit effettuato.



Durante l'esecuzione del modulo si restituiscono alcune informazioni riguardo: alle entità inserite fino a quel momento od al numero di documenti o associazioni inserite. Nel caso si dovesse verificare un errore, la sua descrizione sarà presentata all'utente, il quale potrà scegliere come comportarsi.

Si desidera Continuare(G) o uscire con un RollBack(R) o Commit(C)? [G/R/C]

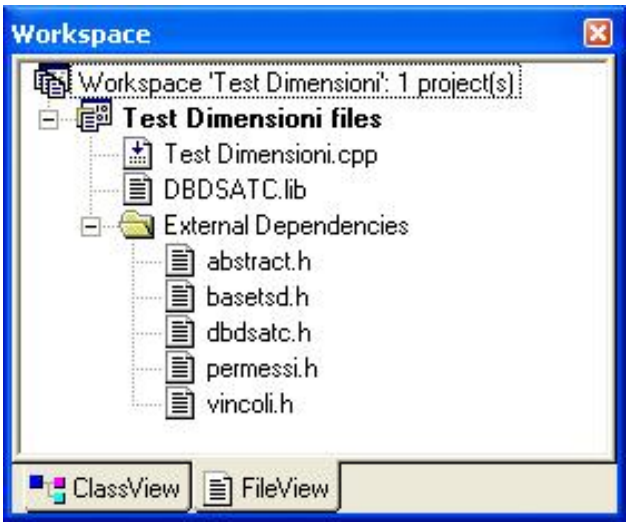
Si consiglia in tale situazione di uscire con un RollBack dall'esecuzione del modulo, di risolvere il problema e di ripristinare l'esecuzione inserendo, alla richiesta del punto di partenza, ***1*** (si osservi che in seguito all'interruzione del modulo viene creato all'interno della directory il file **"Stato.dat"**, necessario per il ripristino dello stato relativo all'ultimo commit).

8.4 Test Dimensioni

Il workspace adoperato, relativo all'ambiente di sviluppo **Microsoft Visual C++ 6.0,** è contenuto nella directory *"GestioneDS\Test Dimensioni"*.

Per realizzare l'eseguibile basta considerare il sorgente *"Test Dimensioni.cpp"*, contenuto nella directory *"Progettazione DB\Codice\Test"*, che include a sua volta solo l'header **DBDSATC.h** (al cui interno vi sono i richiami agli altri header della libreria e a quelli presentati nel *paragrafo 7.2*). Naturalmente si dovrà anche inserire nell'ambiente di sviluppo il file della libreria **DBDSATC.lib**.

Una volta compilato il file "**Test Dimensioni.exe**" basterà mandarlo in esecuzione e poi prelevare i dati mostrati a video nella shell attivata.



I risultati ottenuti sono già stati presentati nel *paragrafo 6.2.2*, di seguito invece riportiamo i risultati memorizzati nel file *"Progettazione DB\Dimensioni utili per il codice.txt"* e solo in parte presentati nel *paragrafo 7.3*:

Nomi_Corti [VARCHAR(15)]:
SIGLA
TELEFONO_CASA
TELEFONO_CELLULARE
USER
PASSWORD
DimMax_LINGUA: 7
DataSet_Type
Relevant_Type
Split_Type
Text_Type
Topic_Type

Nomi [VARCHAR(30)]:
(alcuni campi indicati sono poco ricorrenti e la ridondanza dà solo più sicurezza)

DimMax_NOME_COLLECTION: 13
DimMax_VERSIONE: 16
DimMax_AUTORE_COLLECTION: 22
DimMax_SITO: 25
DimMax_NOME_FILE: 22
DimMax_MODULO_DI_FEEDING: 12
DimMax_NOME_SPLIT: 24 *(era 25, viene ridotto attribuendogli nomi più corti)*
DimMax_NOME_TOPIC: 19
DimMax_NOME_DATA_SET: 30

Nomi_Medi [VARCHAR(60)]:

DimMax_LUOGO: 45

Nomi_Lunghi [VARCHAR(90)]:
(era 80, diventa 90 in seguito all'inserimento delle Keywords del 20 NewsGroups)

DimMax_URL: 63
DimMax_NOME_CATEGORIA: 86

Spiegazioni_Brevi [VARCHAR(160)]:

DimMax_PROVENIENZA: 116
DimMax_TITOLO_QUERY: 148
DimMax_TIPO_PUBBLICAZIONE: 130
(nel prototipo bloccava l'esecuzione a 106, valgono le stesse osservazioni fatte di seguito per il campo AUTORE_DOCUMENTO)
DimMax_DESCRIZIONE_CATEGORIA: 147

Spiegazioni [VARCHAR(255)]:
(i campi indicati sono poco ricorrenti e la ridondanza dà solo più sicurezza)

DimMax_NOTE_DOCUMENTI: 36
DimMax_NOTE_CATEGORIE: 55
DimMax_DESCRIZIONE_ID1: 83
DimMax_DESCRIZIONE_ID2: 164
DimMax_AUTORE_DOCUMENTO: 223
(La scelta iniziale di usare un unico attributo si è rivelata infelice a livello di ottimizzazione dello spazio, ma tenendo conto delle dimensioni dl campo TESTO dei documenti, del risparmio di tempo sulla ricerca dei dati ad essi attinenti, e dell' accettabile perdita di tempo nella ricerca per autore, si continuerà a conservare un campo unico)

Note [NOTE(oltre 255)]:
(alcuni dei campi indicati sono poco ricorrenti e la ridondanza dà solo più sicurezza)
per il Modulo creato si aggiungono i domini:
Note_TITOLO_DOCUMENTO 450
Note_DESCRIZIONE_QUERY 1000
Note_DESCRIZIONE 50000 *(Per le descrizioni dove la ridondanza è accettata)*
Note_TESTO 170000

DimMax_DESCRIZIONE_COLLECTION: 271
DimMax_DESCRIZIONE_FILE: 306
DimMax_TITOLO_DOCUMENTO: 402 *(solo 285>255)*
DimMax_TESTO: 160499
DimMax_DESCRIZIONE_QUERY: 942
DimMax_DESCRIZIONE_SPLIT: 249 *(ridondanza a NOTE accettata)*
DimMax_DESCRIZIONE_TOPIC: 98 *(ridondanza a NOTE accettata)*
DimMax_DESCRIZIONE_DATA_SET: 2319

8.5 Analisi del database

Nella tabella di seguito si riportano le dimensioni utili per i calcoli d'ottimizzazione dei costi. Le dimensioni riportate sono in byte. Per quanto riguarda i campi testuali si fa osservare che in **Access 2000** la codifica di un carattere avviene in **UNICODE**, cioè si adoperano 2 byte invece di uno (delle ottimizzazioni possono essere effettuate abilitando la compressione per tali campi, opzione implicita nel caso di campi **MEMO**).

Tabella	Cardinalità(T)				Size(T) (medio)
BENCHMARK_ _COLLECTION	3				1263,663
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_COLLECTION	3	2	60	22
	VERSIONE	1	2	60	32
	DATA_COLLECTION	3	8	8	8
	AUTORE_COLLECTION	3	2	60	32,666
	SITO	3	2	60	36,666
	URL	3	2	180	104
	MODULO_DI_FEEDING	3	2	60	22,666
	NOTE_DOCUMENTI	3	2	510	60
	NOTE_CATEGORIE	3	2	510	76,666
	DESCRIZIONE_ID1	3	2	510	257,333
	DESCRIZIONE_ID2	3	2	510	160,666
	DESCRIZIONE_COLLECTION	2	2	-	451
FILE	11				249,635
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_FILE	10	2	60	28,545
	NOME_COLLECTION	3	2	60	19,454
	DESCRIZIONE_FILE	10	2	-	201,636
DATA_SET	12				1101,966
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_DATA_SET	12	2	60	34,5
	TIPO_DATASET	3	2	30	7,666
	DESCRIZIONE_DATA_SET	10	2	-	1059,8
ADAPTIVE_ _FILTERING	19792				54
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_DATA_SET	2	2	60	46
	ID_QUERY	4967-	4	4	4
	ID_DOCUMENTO	11297	4	4	4
SPLIT	19				422,629
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_SPLIT	19	2	60	28,526
	NOME_COLLECTION	3	2	60	24,736
	TIPO_SPLIT	6	2	30	17,578
	DESCRIZIONE_SPLIT	19	2	-	351,789
S__DS	20				87,4
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_DATA_SET	12	2	60	35,3
	NOME_SPLIT	12	2	60	32,1
	TIPO_SPLIT	3	2	30	20

TOPIC	10		166,2		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_TOPIC	10	2	60	28,2
	NOME_COLLECTION	3	2	60	18,8
	TIPO_TOPIC	3	2	30	13,6
	DESCRIZIONE_TOPIC	10	2	-	105,6
T__DS	18		62,888		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_DATA_SET	12	2	60	36,333
	NOME_TOPIC	9	2	60	26,555
CATEGORIA	7018		33,127		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_CATEGORIA	7018	4	4	4
	NOME_CATEWGORIA	7018	2	180	29,127
INFO_CATEGORIA	458		72,694		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_CATEGORIA	458	4	4	4
	DESCRIZIONE_CATEGORIA	457	2	320	68,694
SOVRACATEGORIA	758		8		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_SOVRACATEGORIA	78	4	4	4
	ID_CATEGORIA	718	4	4	4
DPC	7259		41,408		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_TOPIC	5	2	60	25,621
	ID_CATEGORIA	7018	4	4	4
QUERY	5129		433,424		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_QUERY	5129	4	4	4
	TITOLO_QUERY	5129	2	320	32,736
	DESCRIZIONE_QUERY	5129	2	-	396,688
CPC	5629		41,408		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_TOPIC	5	2	60	23,358
	ID_QUERY	5129	4	4	4
	SIGLA	5629	2	30	14,05
DOCUMENTO	390141		2668,528		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_DOCUMENTO	390141	4	4	4
	TIPO_TESTO	4	2	30	8,324
	TIPO_PUBBLICAZIONE	242	2	320	31,971
	LINEE	420	2	2	2
	PROVENIENZA	347111	2	320	65,273
	AUTORE_DOCUMENTO	304987	2	510	77,438
	DATA_DOCUMENTO	133	8	8	8
	LUOGO	1872	2	120	20,129
	LINGUA	1	2	30	14
	TITOLO_DOCUMENTO	390141	2	-	156,813
	TESTO	390141	2	-	2280,58

D__B	390141		34		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_COLLECTION	3	2	60	22
	ID_DOCUMENTO	390141	4	4	4
	ID1_ORIGINAL_COLLECTION	348566	4	4	4
	ID2_ORIGINAL_COLLECTION	370145	4	4	4
D__S	507706		52,119		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	NOME_SPLIT	19	2	60	30,265
	TIPO_SPLIT	6	2	30	17,854
	ID_DOCUMENTO	390141	4	4	4
RELEVANCE_ _JUDGMENTS	1501378		24,33		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_QUERY	5120	4	4	4
	ID_DOCUMENTO	341457	4	4	4
	RELEVANT	2	2	30	16,33
CLASSIFICAZIONE	411223		8		
	Attributo	**Val(A_i)**	**MIN(A_i)**	**MAX(A_i)**	**SIZE(A_i)**
	ID_CATEGORIA	6762	4	4	4
	ID_DOCUMENTO	388256	4	4	4

CAPITOLO 9:

ESEMPIO DI INTERFACCIA GRAFICA

9.1 Librerie Adoperate

9.2 Menu e sue Funzionalità

9.3 ToolBar e sue Funzionalità

9.4 Descrizione delle Viste

9.5 Funzionalità in Modalità Report

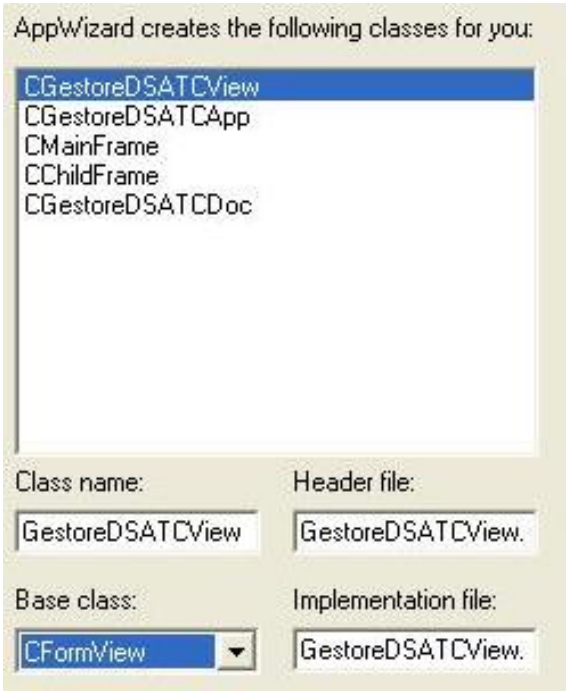
In questo capitolo sì descrizione un'interfaccia grafica realizzata per mezzo della libreria MFC. L'obiettivo è di creare uno interfaccia in grado di verificare le funzionalità implementate per il Modulodi d'In/Out.

Le descrizioni presenti nei paragrafi avranno il compito principale di spiegare l'utilizzo dell'HIC, introducendo solo in grandi linee l'architettura e l'implementazione di quest'ultima.

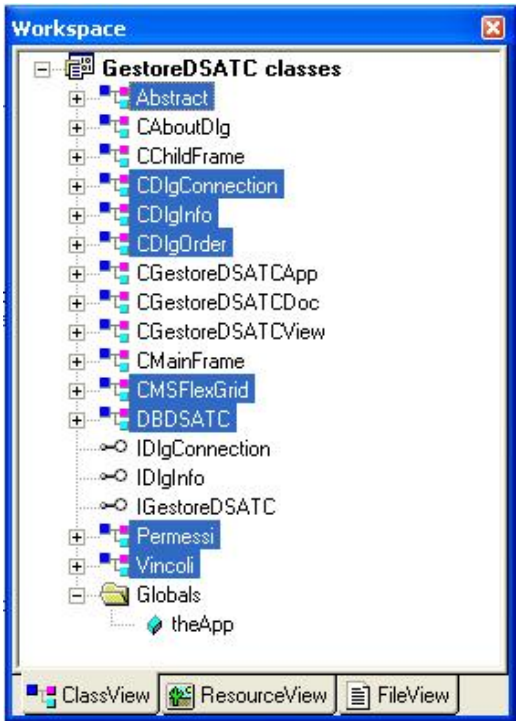
9.1 Librerie Adoperate

Per creare l'HIC si è utilizzato il workspace, relativo all'ambiente di sviluppo **Microsoft Visual C++ 6.0**, contenuto nella directory *"GestioneDS\GestoreDSATC"***.** Il settaggio del workspace ha richiesto la creazione di un progetto di tipo **MFC AppWizard(exe)** abilitato all'utilizzo di *documenti multipli* tramite un ***architettura Documento/Vista***. Per la gestione dei database si richiede solo l'inclusione dei file header. Il resto dei settaggi è quello standard, tranne che per l'inclusione dei supporti per l'*Automation* e gli *ActiveX Controls* e la richiesta di adoperare le librerie MFC a livello di linkaggio statico (opzione necessaria poiché la libreria realizzata è statica e l'ambiente di sviluppo non permette di effettuare contemporaneamente i linkaggi dinamici e statici).

Le classi iniziali sulle quali si è andato a lavorare sono quelle fornite in default dal wizard, con l'unica modifica della classe **CGestoreDSATCView** che si è fatta derivare da **CFormView** invece che da *CView*.



Al progetto poi sono state aggiunte le classi derivate dalla classe base **Cdialog (CAboutDlg** è già inclusa nel progetto per mezzo di **CGestoreDSATCApp**), la classe **CMSFlexGrid** (insieme con quelle da essa adoperate) e naturalmente le classi presenti nella libreria **DBDSATC.lib**.



Il diagramma riportato di seguito (implementato nel file *ObjModel_HIC.oom* ed inserito nel workspace DataSetDB.sws) dà un'idea sui legami esistenti tra le classi fin qui nominate:

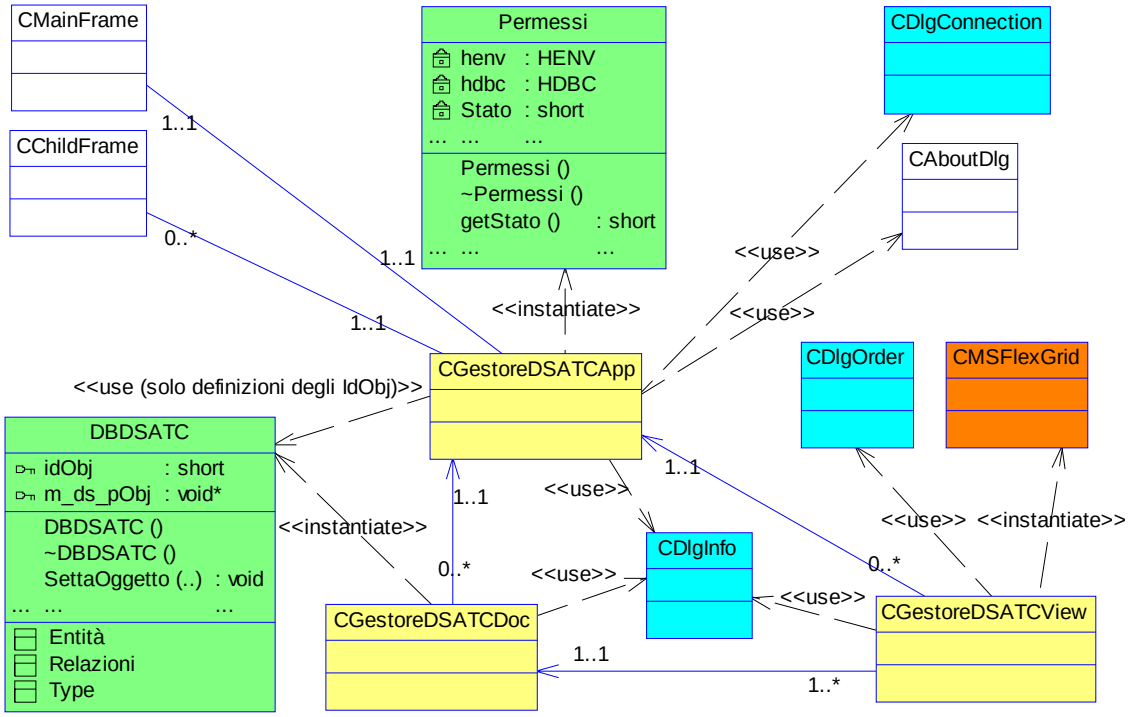


Figura 34: Diagramma di Massima dell'HIC

- Le classi in bianco e giallo sono quelle fornite dal wizard.
- Le classi in giallo sono modificate per gestire i messaggi sopraggiunti dai menu, dalle toolbar e dalle viste.

- Le classi in azzurro sono create per gestire particolari dialoghi con l'utente.
- La classe in arancione individua l'*ActiveX Control* utilizzato per realizzare la griglia necessaria alla visualizzazione dei risultati delle ricerche.

Per adoperare l'HIC sarà sufficiente eseguire il file **GestoreDSATC.exe**, si osservi che: per utilizzare il file help **DSATC.hlp** è necessario che sia inserito nella stessa directory dell'eseguibile. Il file ***\Eseguibili\Gestore DSATC\MSFLXGRD.OCX*** *va inserito nella directory system nel caso di Windows 98 e nella directory system32 nel caso di Windows XP o NT; la registrazione dell'****OCX*** *si ha richiamando da prompt* ***"regsvr32 MSFLXGRD.OCX"***.
Nel resto del capitolo si procederà alla descrizione delle funzionalità offerte dall'interfaccia ed a piccole osservazioni riguardo alla loro realizzazione, la cui comprensione richiede alcune note preliminari:

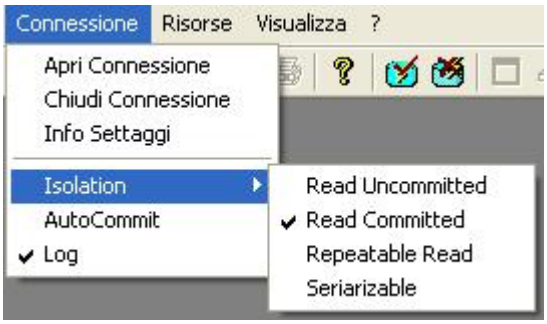
- l'**Environment** dell'unico oggetto Permessi (membro ***m_PConn***) associato a ***theApp*** (che a sua volta è l'unico oggetto della classe CGestoreDSATCApp implementato) viene allocato in automatico appena si apre l'applicativo e deallocato quando si chiude).
- Col termine *Risorse* s'intende un oggetto della classe **DBDSATC,** definito come membro (***m_ds_pObj***) della classe **CGestoreDSATCDoc**.
- Grazie alla presenza della classe **DBDSATC,** è stato possibile definire un solo tipo di documento e di vista (un oggetto della classe **CGestoreDSATCView** che permette di interagire con le informazioni contenute in un oggetto **CGestoreDSATCDoc**).
- ***theApp*** è stato dotato, per aprire una particolare Risorsa, di una struttura dati in grado di passare ad un documento l'*idObj* necessario per richiamare il membro **m_ds_pObj.SettaOggetto()** in grado di specificare la particolare classe che deve essere gestita da ***m_ds_pObj***.

9.2 Menu e sue Funzionalità

All'apertura dell'applicativo il menu non è completo (si riferisce all'unico oggetto **CMainFrame** allocato) e si presenta come in figura. Selezionando **File** gli unici due comandi richiamabili saranno:

- **Last New**: attiva una **Risorsa** (vale a dire una tabella del database) simile all'ultima aperta; l'operazione richiama il membro **theApp.VisualizzaErrore**, che adopera la classe **CDlgInfo**, nel caso in cui non sia stata ancora aperta nessuna connessione oppure si siano presentati degli errori durante la creazione dell'oggetto ***m_ds_pObj***.
- **Esci**: Esce dal programma; si preoccupa di chiudere la connessione e l'environment.

∅ Il pop-up **Connessione** permette di gestire le funzionalità relative alla connessione adoperata dall'applicativo (il membro ***m_PConn*** di ***theApp***):



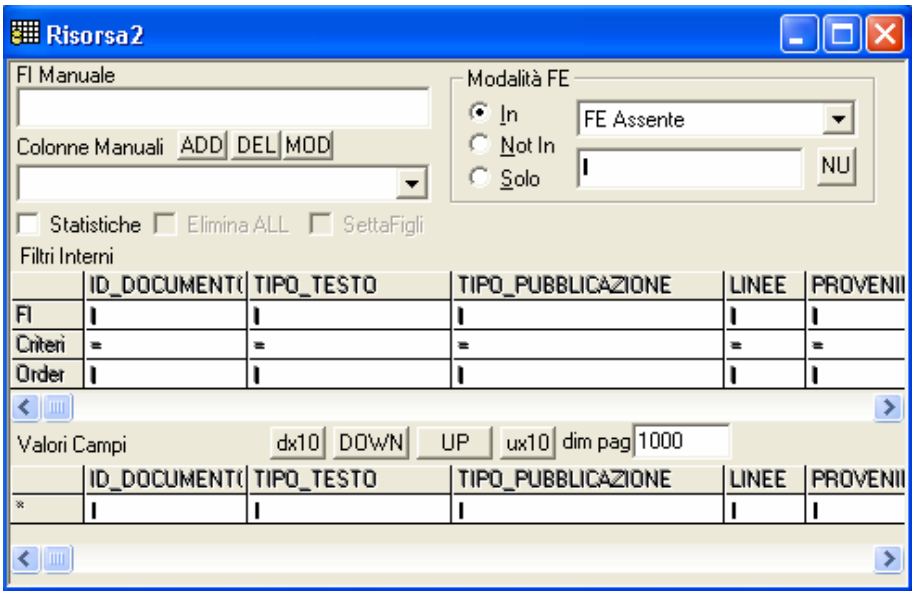
- **Log**: permette di attivare/disattivare l'utilizzo di un file di Log (creato nella stessa directory dell'applicativo col nome "*ODBC_DBDSATC.log*"). L'attivazione del Log abilita una spuntatura a fianco al comando in modo da mostrare all'utente lo stato attuale della funzionalità.
- **AutoCommit**: permette di attivare/disattivare l'utilizzo dell'autocommit. L'attivazione dell'AutoCommit abilita una spuntatura a fianco al comando in modo da mostrare all'utente lo stato attuale della funzionalità. Tale funzionalità rende non annullabili le operazioni effettuate sul database, però in compenso le velocizza giacché risparmia al **DBMS** la gestione di un numero di lock eccessivo e delle informazioni necessarie al ripristino del precedente stato del database.
- **Isolation**: apre un ulteriore finestra di pop-up nella quale è possibile selezionare una delle possibili modalità relative al livello di insolazione della connessione: *Read Uncommitted*, *Read Committed*, *Repeatable Read* e *Serializable*. L'opzione selezionata avrà una spuntatura abilitata al suo fianco.

- **Apri Connessione**: attiva una finestra di dialogo (un oggetto della classe **CDlgConnection**) che permette di inserire il nome della fonte ODBC, l'User e la Password necessari per creare una connessione tramite il membro **m_PConn.Connect(…)**.

- **Chiudi Connessione**: chiude l'attuale connessione per mezzo del membro **m_PConn.Disconnect()**.
- **Info Settaggi**: richiama **m_PConn.getSettaggi()** e visualizza la stringa ricevuta attraverso l'utilizzo di un oggetto della classe **CDlgInfo**.(non vi è alcuna informazione relativa allo statement). Nel caso sia attiva una Risorsa si richiama il membro **m_ds_pObj.getSettaggi()**, che permette di prelevare anche le informazioni relative al particolare statement della Risorsa attualmente attiva.

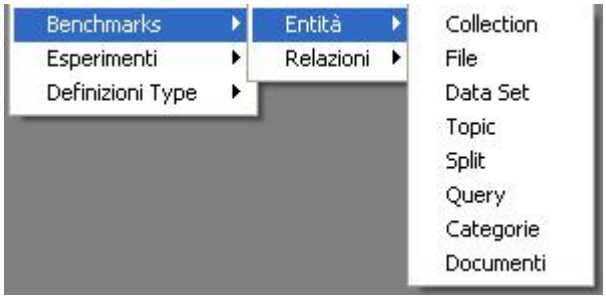
Si osservi che tramite l'utilizzo della ScrollBar è possibile visionare anche il resto delle informazioni, inoltre il testo è selezionabile in modo da poterlo copiare se necessario.

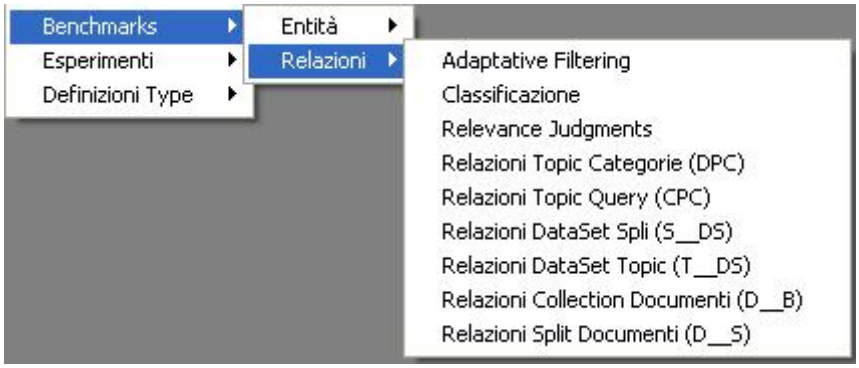
∅ Il pop-up **Risorse** contiene le funzionalità che permettono di aprire un documento e di settare il suo membro ***m_ds_pObj***. L'operazione richiama il membro **theApp.VisualizzaErrore**, che adopera la classe **CDlgInfo**, nel caso in cui non sia stata ancora aperta nessuna connessione oppure si siano presentati degli errori durante la creazione dell'oggetto ***m_ds_pObj***.

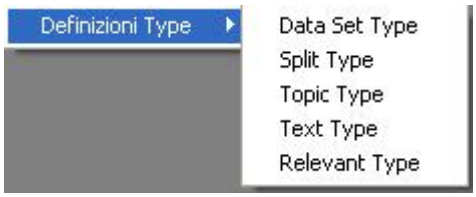


Per identificare la particolare tabella che dovrà essere gestita, si è pensato di inserire nel menu un insieme di comandi in grado di richiamare la creazione di un documento passandogli un particolare *idObj*. Il costruttore della classe **CGestoreDSATCDoc** si preoccuperà di richiamare **m_ds_pObj.SettaOggetto()** inserendogli come parametro di ingresso *l'idObj* individuato.

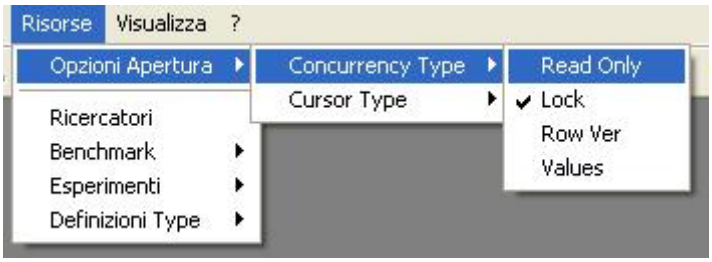
I comandi legati agli *idObj* sono organizzati in Ricercatori (attiva la tabella relativa ai ricercatori), Benchmarks e Esperimenti (divisi a loro volta in due gruppi detti Entità e Relazioni) ed infine Definizioni Type.

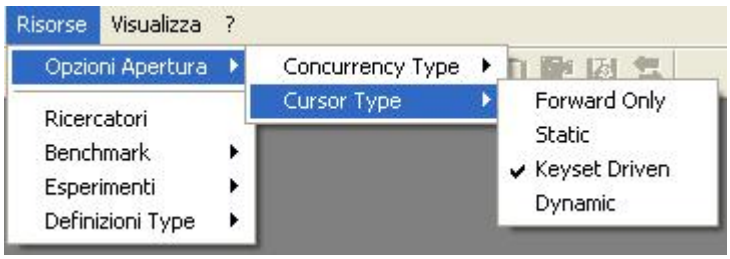






Tramite i sottomenu **Concurrency Type** e **Cursor Type** di **Opzioni Apertura** è possibile settare la concorrenza ed il tipo di cursore che saranno adoperati, dal membro **m_ds_pObj.setConnessione(…)** del documento creato, per aprire la nuova risorsa (quelle già aperte conservano per tali parametri i valori adoperati al momento della loro apertura). I valori attivi presentano al loro fianco una spuntatura.





∅ Il pop-up **Visualizza** permette di abilitare/disabilitare la **barra degli strumenti** o la **barra di stato** dell'applicazione, mentre **?** richiama delle informazioni relative al prodotto software.

Una volta attivata una risorsa si abiliterà il menu relativo agli oggetti **CChildFrame**. Alcune nuove funzionalità vengono inserite in **File** ed in più si aggiungono due nuovi

pop-up: **Modifica** (che permette di effettuare alcune operazioni sul testo selezionato) e **Finestra** (che permette di gestire le finestre relative alle viste attualmente aperte). I nuovi comandi di **File** sono:

- **Chiudi**: che permette di chiudere il documento attualmente aperto, dopo però aver disconnesso il suo membro ***m_ds_pObj*** dalla connessione attivata dal membro ***m_PConn*** di ***theApp***.
- **Salva Report**: se la Risorsa è in modalità Report salva in un file (di default *DSATC.txt*) tutti i record ottenuti dalla ricerca effettuata.
- **Salva Report con nome...**: simile all'operazione precedente ma permette di cambiare il nome del file di default da utilizzare per salvare i dati del particolare documento relativo alla vista attiva.

9.3 ToolBar e sue Funzionalità

Quando nessuna Risorsa è attivata la ToolBar non permette di utilizzare tutti i suoi possibili comandi. Inizieremo quindi descrivendo quelli sempre attivi.


Richiama il comando **Last New** già presentato nella descrizione dei Menu.


Visualizza, tramite l'uso di un oggetto della classe **CdlgInfo,** il file *DSATC.h* contenente un piccolo help ed una lista di comandi relativa alle funzionalità adoperabili sulle griglie delle **Risorse**


Effettua un Commit (richiama **m_Pconn.TransCommit()**). L'operazione restituisce un errore se non vi è alcuna connessione attiva. In seguito ad un commit (o rollback) non sarà più possibile usare il comando **Riesegui**.


Effettua un RollBack (richiama **m_Pconn.TransRollBack()**).. L'operazione restituisce un errore se non vi è alcuna connessione attiva.

Dopo l'apertura di una Risorsa si abilitano gli altri comandi della ToolBar che si riferiscono alle operazioni che possono essere effettuate sulla particolare vista attiva (si osservi che il membro **m_ds_pObj.SetPos(…)** del particolare documento attivato verrà adoperato solo nella modalità **Report** e tramite particolari combinazioni di tasti):


Comando **Salva Report**, restituisce un errore se non si sta in modalità Report

Permette di ripristinare la connessione relativa alla Risorsa attiva. Da utilizzare per ripristinare le viste di una risorsa dopo aver cambiato l'utente del database (operazione che richiede di disconnettersi e poi riconnettersi per mezzo del membro ***m_PConn***, che cosi facendo è dissociato dagli oggetti DBDSATC legati ai documenti attivi).

Comando **Pulisci**, ripristina lo stato iniziale della vista.

Richiama il membro **m_ds_pObj.TransInserisci()** relativo alla vista corrente, i valori di input sono quelli presenti in "*Valori Campi*" dove il simbolo "|" rappresenta il valore strNoUtil che da ora in avanti sarà definito come **NoUtil**.

Richiama il membro **m_ds_pObj.TransElimina()** relativo alla vista corrente, i valori di input sono quelli presenti nei campi che permettono di individuare dei parametri legati ai filtri o alle opzioni di cancellazione:

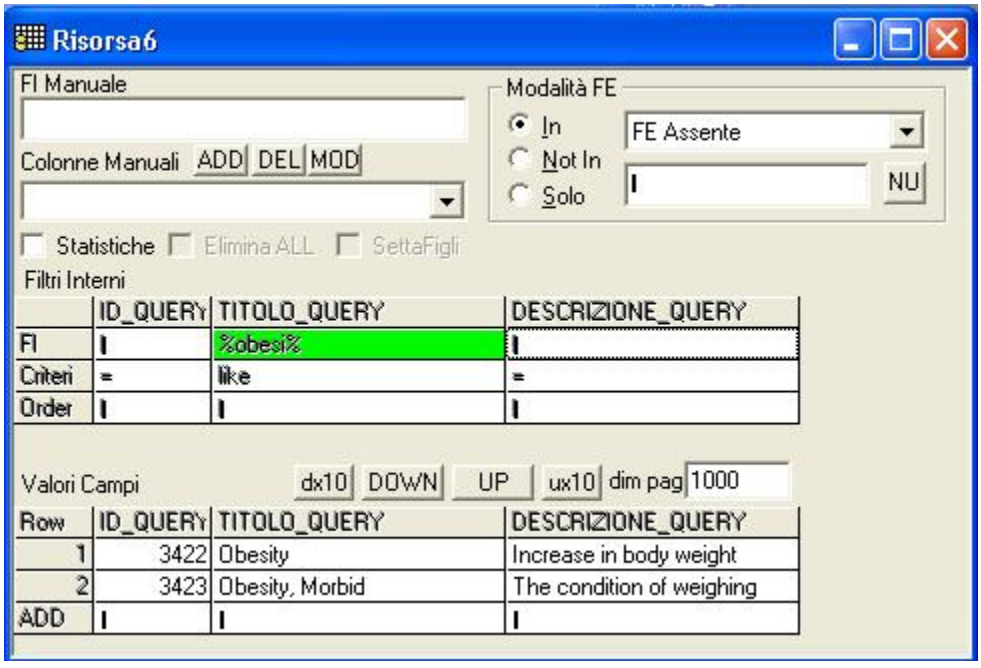
1. Campo contenente un particolare filtro manuale definito dall'utente.
2. Indica di eliminare anche tutte le entità legate a quelle che saranno filtrate; tale campo è settabile solo per alcuni *idObj*.
3. Indica di associare come sovracategoria delle categorie che resterebbero orfane quelle delle loro ex-sovracategorie; tale campo è settabile solo per alcuni *idObj*.
4. *Modalità FE* permette di aggiungere ai normali filtraggi anche quelli dei filtri esterni abilitati per il particolare *idObj*.
5. Permette di definire gli attributi che si vogliono adoperare per il filtraggio ed i particolari criteri da adoperare.

Richiama il membro **m_ds_pObj.TransModifica()** relativo alla vista corrente, i valori di input sono quelli presenti in "*Valori Campi*" per quanto riguarda i nuovi valori da attribuire agli attributi, e quelli dei punti *1,4 e 5* (utilizzati per l'eliminazione) per quanto riguarda l'individuazione dei record da modificare.

Richiama il membro **m_ds_pObj.TransRicerca()** relativo alla vista corrente, i valori di input sono quelli dei punti *1,4 e 5* (utilizzati per l'eliminazione) per quanto riguarda i filtri e quelli dei punti *6 e 7* per quanto riguarda l'individuazione delle colonne restituite dalla ricerca:

5. I filtri interni contengono anche una riga **Order** che permette di definire l'ordinamento adoperato dalla ricerca.
6. Permette di definire delle colonne da aggiungere alla ricerca
7. Permette di decidere se la ricerca deve restituire solo le colonne appartenenti alla chiave della tabella e quelle manuali (check abilitato) o tutte le colonne indiscriminatamente (disabilitato).

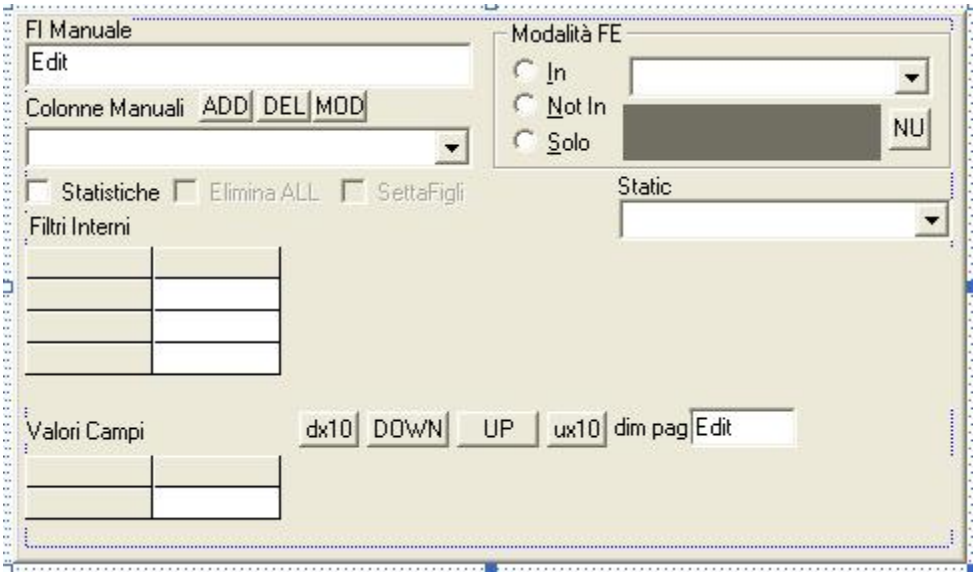
Di seguito si riporta un esempio del risultato di una ricerca



Tale funzione tiene conto solo delle celle in verde o del Valore del filtro esterno nel caso sia visibile il suo bordo; I valori da attribuire non dovranno essere nè stringhe nulle nè **NoUtil**. Solo nel caso dei *"Valori Campi"* si potrà ammettere l'uso di stringhe nulle.

9.4 Descrizione delle Viste

Nel paragrafo precedente si è accennato più volte ai campi delle viste, attraverso i quali è possibile passare i valori di ingresso necessari ad eseguire le varie operazioni richieste. La vista, come già detto, è generica ed utilizzabile da una qualsiasi classe definita tramite il membro ***m_ds_pObj***.



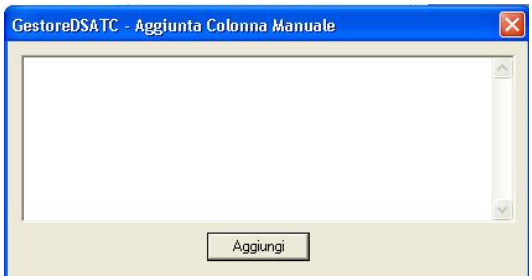
La vista richiama il form "*IDD_GESTOREDSATC_FORM*" e nella fase d'inizializzazione si preoccupa di prelevare dal membro ***m_ds_pObj***, relativo al documento cui è collegata, le informazioni necessarie al settaggio del form.

∅ Il campo **FI Manuale** viene abilitato semplicemente scrivendo qualcosa al suo interno. Il valore assegnato è da considerare come parte della stringa delle condizioni tipicamente inserita dopo la clausola **WHERE** di una QUERY. Questo filtro è aggiunto a tutti quelli abilitati tramite il form. Un esempio per quanto riguarda la Risorsa **Documenti** potrebbe essere:

∅ L'area **Modalità FE** contiene i controlli necessari per definire il tipo di filtro esterno adoperato. Il gruppo di radio button **In**, **Not In** e **Solo** permette di definire se i record ricercati devono essere *contenuti*, *non contenuti*, o si devono trovare *solo* nell'insieme degli elementi in relazione con l'entità, individuata tramite la combo box ***1***, il cui valore è definito in *2*. Il button **NU** permette di settare a **NoUtil** il valore definito in *2*, la cui conseguenza, nel momento in cui il campo *1* è diverso da **FE Assente**, è che il filtraggio considererà tutti i valori del campo come validi (cioè, nel caso di **In** cerca i

record che appartengono almeno ad una delle entità esterne, e nel caso di **Not In** che non appartengono alle entità esterne).

∅ La ComboBox **Colonne Manuali** permette di visualizzare le colonne aggiunte dall'utente. Per inserire una nuova colonna basta premere il button **ADD**, tramite il quale si apre un oggetto della classe **CDlgInfo** (settato ad hoc) che permette di scrivere la nuova colonna e di confermare l'operazione per mezzo del button **Aggiungi**. Il button **DEL** invece elimina la colonna attualmente selezionata nella ComboBox. Infine **Modifica** apre un oggetto della classe **CDlgInfo** (settato ad hoc) con un edit box contenente la vecchia definizione del campo, che una volta modificata richiede la conferma dell'operazione per mezzo del button **Modifica**.



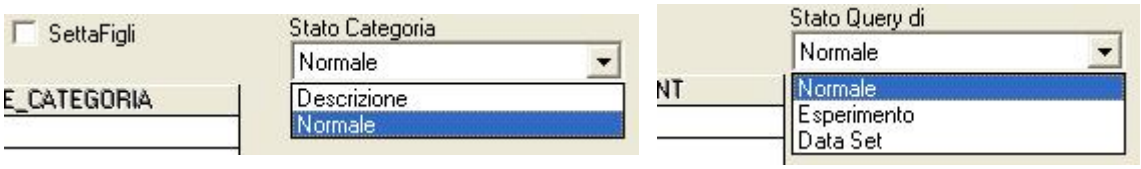
∅ Il check **Statistiche** viene adoperato solo nel caso di ricerche e permette di decidere se devono essere restituite solo le colonne appartenenti alla chiave della tabella e quelle manuali (check abilitato) o tutte le colonne indiscriminatamente (disabilitato).

∅ La ComboBox individuata dall'etichetta **Static** è abilitata solo per alcune classi di DBDSATC e può avere solo due diversi settaggi:

- **Stato Categoria**, abilitato nel caso della vista su una tabella **Categoria**. Permette di decidere se inserire o no il campo DESCRIZIONE_CATEGORIA nella vista (in tale

caso sarà possibile effettuare cancellazioni solo nella modalità Report, e quest'ultime elimineranno solo le descrizioni relative alla categoria selezionata).

- **"Stato Query di"**, abilitato nel caso delle viste relative alle **Classificazioni** od ai **Relevance Judgments**. Permette di scegliere i campi aggiuntivi da considerare per effettuare filtraggi più complessi (la modalità **Normale** mostra la tabella del database e permette di effettuare tutte le operazioni senza alcun problema).



∅ Il check **SettaFigli**, se abilitato, indica di associare come sovracategoria delle categorie che resterebbero orfane in seguito alle cancellazioni effettuate per mezzo d'**Elimina** quelle delle loro ex-sovracategorie; tale campo è settabile solo per le viste di **Categorie**, **Topic**, **Data Set** e **Collection**.

∅ Il check **Elimina All**, se abilitato, indica di eliminare anche tutti i record (di altre tabelle) in relazione a quelli da eliminare per l'attuale vista; tale campo è settabile solo per le viste di **Split**, **Topic**, **Data Set** e **Collection**.

∅ Per **settare a NoUtil una cella** di una griglia basta selezionarla e digitare **C+SHIFT**. Si noti che tale comando non è attivo sulla riga **Criteri**.

∅ Per **modificare una cella** di una griglia è sufficiente selezionarla e digitare "**M**", oppure farci sopra doppio click col mouse. Tali eventi abiliteranno l'apertura di un oggetto della classe **CDlgInfo** (settato ad hoc) contenente il valore attuale della cella selezionata, una volta modificato tale valore basterà confermare la modifica premendo sul button **Modifica,** reso disponibile dall'oggetto **CDlgInfo**. I valori consentiti per le celle **Criteri** sono "=", ">", ">=", "<",…. e **like** (quest'ultimo richiede i caratteri jolly "**%**" e "**_**" nei valori dei campi **FI**).

Nel caso la cella si riferisca ad una riga *Order* si aprirà un oggetto **CDlgOrder** che permette di decidere se il campo deve essere aggiunto o eliminato da quelli utili per l'ordinamento:

- Se il campo non è presente il comando **Aggiungi** permette di inserirlo come ordinamento (n+1)° ascendente o descrescente, dove con (n+1)° si intende che già n campi sono adoperati per definire l'ordinamento.

	NOME_SPLIT	NOME_COLLECTION	TIPO_SPLIT	DESCRIZIONE_SPLIT
FI				
Criteri	=	=	=	=
Order	2° ASC		1° DESC	3° ASC

- Nel caso il campo sia già adoperato per l'ordinamento è possibile eliminarlo tramite **Rimuovi**. Quest'operazione ripristina gli indici d'ordinamento.

	NOME_SPLIT	NOME_COLLECTION	TIPO_SPLIT	DESCRIZIONE_SPLIT
FI				
Criteri	=	=	=	=
Order	1° ASC			2° ASC

- Nel caso il campo sia già adoperato per l'ordinamento è possibile inoltre modificare il tipo d'ordinamento da ascendente e descendente (o viceversa) e selezionare **Aggiungi**, in questo caso però il grado dell'ordinamento resterà immutato.

Criteri	=	=	=	=
Order	1° DESC			2° ASC

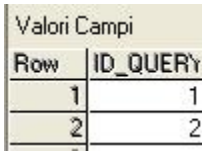
∅ Una funzionalità aggiuntiva molto utile è quella di **Trascinamento**. Tenendo premuto il tasto destro del mouse su un'area priva di oggetti di una vista e mantenendolo premuto fino ad arrivare su un'altra vista si trascina il valore del campo abilitato (della griglia "Valori Campi") dalla prima vista alla seconda: se tale valore esiste nei filtri esterni verrà inserito in questi ultimi, altrimenti se esiste nei filtri interni sarà inserito in questi. Tale funzionalità vale anche nella stessa vista.

9.5 Funzionalità in Modalità Report

Una vista attiva può lavorare in due modalità, quella **Normale** in cui è possibile adoperare solo le funzionalità offerte dalla ToolBar, e quella **Report** nella quale sono

abilitati dei comandi aggiuntivi sulla Griglia *"Valori Campi"*. Quest'ultima modalità, cosi come mostrato in figura, può essere individuata per mezzo d'alcune varianti nei nomi delle colonne e righe della griglia "Valori Campi":

Modalità Normale



Modalità Report

La modalità **Report** è attivata in seguito all'utilizzo del comando **Ricerca** presente nella ToolBar. Per uscire da questa modalità è sufficiente richiamare **Pulisci** oppure uno dei comandi base **Inserisci** (che funziona solo se la riga **ADD** è presente nell'attuale pagina dei record), **Modifica** ed **Elimina**. (che funzionano se la riga abilitata della griglia "*Valori*" non è **ADD**).

Nella modalità **Report** è possibile adoperare dei particolari comandi per muoversi tra le pagine contenenti i record ottenuti per mezzo della ricerca. La riga **ADD,** accennata in precedenza, è presente solo nell'ultima pagina del **Report**

La necessità di questi comandi si deve all'impossibilità da parte del controllo *FlexGrid* di contenere grandi moli di dati e di gestirle efficacemente. Con il campo **dim pag** sì dà la possibilità all'utente di mostrare più dati per pagina a svantaggio della velocità di spostamento da una pagina all'altra. Vediamo ora i *button command* a cosa servono:

- **dx10**: permette di muoversi dieci pagine prima della corrente
- **DOWN**: permette di muoversi alla pagina prima della corrente
- **UP**: permette di muoversi alla pagina dopo la corrente
- **ux10**: permette di muoversi dieci pagine dopo la corrente

Quando ci si trova nella modalità **Report** è possibile richiamare il membro **m_ds_pObj.SetPos(…)** del documento legato alla vista corrente. Ciò è possibile utilizzando una particolare combinazione di tasti dopo aver selezionato una cella della riga d'interesse:

- **D+SHIFT**: elimina la riga selezionata solo se non è quella identificata da **ADD** (in tale modalità EllAll e SettaFigli non sono presi in considerazione)
- **U+SHIFT**: aggiorna la riga solo se non è quella identificata da **ADD**
- **A+SHIFT**: aggiunge una tupla solo se la riga selezionata è identificata da **ADD**

Quando si adopera la tabella **Categoria** ed è settato un filtro esterno *ID_SOVRACATEGORIA* o *ID_SOTTOCATEGORIA* con un numero valido (cioè il campo valore è diverso da **NoUtil**), sulle righe della griglia "*Valori Campi*" sono attive le funzionalità Inserisci/Elimina Gerarchia richiamabili per mezzo delle combinazioni:

- **I+SHIFT**: inserisce la categoria selezionata come sovracategoria (sottocategoria) della categoria indicata in *ID_SOTTOCATEGORIA* (*ID_SOVRACATEGORIA*).
- **O+SHIFT**: elimina la categoria selezionata come sovracategoria (sottocategoria) della categoria indicata in *ID_SOTTOCATEGORIA* (*ID_SOVRACATEGORIA*).

CAPITOLO 10:

CONCLUSIONI

10.1 Limitazioni del Software

10.2 Possibili Sviluppi Successivi

Lo scopo di ogni studio o lavoro effettuato è quello di fornire un servizio utile. La tesi sviluppata ha consentito l'approfondimento e lo studio di tematiche di (mio) interesse finalizzate alla realizzazione di uno strumento che potesse facilitare il lavoro di altri. In questo capitolo vengono presentate le principali limitazioni cui si va incontro adoperando il pacchetto fornito e i possibili sviluppi futuri consigliati.

10.1 Limitazioni del Software

Il software realizzato contiene delle limitazioni legate principalmente all'uso del DBMS Access 2000. Infatti, anche se prevede la possibilità di un utilizzo in ambienti multiutente e dell'istallazione in reti locali (tramite una fonte ODBC, associata a file di pubblico dominio), non permette un accesso a più di un ristretto numero d'utenti a causa di una loro non ottimale gestione.
Per quanto riguarda la struttura del database la principale restrizione si riferisce al vincolo imposto sui campi di tipo *Misure*. L'ipotesi che i valori contenuti debbano essere solo positivi, consigliata dall'utilizzo di features normalizzate e di misure che in base alle loro definizioni risultano positive, potrebbe col tempo risultare troppo restrittiva. L'eliminazione futura del vincolo richiederebbe in ogni modo l'esecuzione d'alcune modifiche, facilmente effettuabili tramite due semplici passi:

1. La cancellazione del vincolo, presente su tutti i campi appartenenti al dominio *Misure*, utilizzando l'interfaccia Access (accessibile tramite l'utente ZAR, DBAall o un altro amministratore creato in un secondo momento, si veda il *paragrafo 6.7.2*).
2. La modifica dei valori relativi a numNoUtil e numNull (definiti tramite le direttive presenti nell'header della classe Abstract, si vade il *paragrafo 7.5.3*) in modo da considerare dei numeri interi negativi raramente adoperati (si noti che tale valore viene usato anche per degli short integer).

I membri **Riesegui()** presentati nel *modulo d'In/Out* potrebbero essere modificati in modo da riconoscere in automatico se le modifiche effettuate sui campi ed i filtri della particolare classe permettano di rieseguire l'ultima QUERY preparata senza violare i vincoli richiesti dal membro. Gli svantaggi di tale modifica sono legati alla memorizzazione, per ogni QUERY preparata, dello stato dei campi adoperati ed alla necessità di effettuare, ad ogni richiamo di **Riesegui()**, una serie di confronti che potrebbe causare un eccessiva perdita di tempo; i vantaggi, invece, sono legati al richiamo in automatico dei comandi in grado di ripreparare la QUERY e quindi ad una gestione più trasparente, da parte dei ricercatori, del membro **Riesegui()**.
L'HIC realizzata ha dei vincoli legati all'utilizzo del controllo FlexGrid, che richiede di paginare i risultati ottenuti. Tale limite si deve alla necessità di offrire, tramite il *modulo d'In/Out*, funzionalità d'accesso ai dati trasparenti all'interfaccia ODBC e svincolate

dalle librerie microsoft. Infatti l'utilizzo del *modulo d'In/Out* per realizzare un'*HIC* non permette di associare direttamente i recordset ottenuti tramite le *API ODBC* ad oggetti griglia. Naturalmente una volta messo a disposizione il database **DBDSATC** chiunque può adoperarlo per realizzare un'HIC senza l'utilizzo del modulo e quindi gestendo i controlli più familiari o idonei alla situazione.

Infine si fa notare che, a causa della presenza di campi testuali che potrebbero contenere caratteri di tabulazione "\t" o di invio a capo "\n", la formattazioni ottenuta tramite i membri **SaveRicerca()** può dare dei problemi, risolvibili modificando il membro ed inserendovi del codice in grado di generare una formattazione più idonea. La decisione di creare tramite il membro file con un formalismo cosi semplice è nata tenendo conto dell'obiettivo cui doveva adempiere, cioè riportare dei listati di feature e di classificazioni utili per effettuare gli addestramenti ed i calcoli delle misure relative agli esperimenti.

10.2 Possibili Sviluppi Successivi

Il sistema realizzato è stato testato su una macchina con le seguenti caratteristiche:

Scheda Madre:	ASUS A7V
Processore:	AMD DURON 800Mhz
RAM:	384 MB a 133Mhz
HD:	Ultra ATA100 "IBM DTLA 307030 SCSI Disk Device"
Sheda di Rete:	NIC Fast Ethernet PCI Realtek RTL8139 Family
Sistema Operativo:	Microsoft Windows XP
DBMS:	Access 2000

L'attenta realizzazione del sistema dal punto di vista della portabilità, l'accurata descrizione del processo di sviluppo e la messa a disposizione di documentazione e moduli PowerDesigner, permette di effettuare delle modifiche in grado di importare quanto realizzato su macchine dotate di DBMS e sistemi operativi differenti.

I primi sviluppi potrebbero essere legati al cambio delle tecnologie adoperate. Utilizzando i modelli conclusivi del database, ottenuti tramite il lavoro realizzato nei primi sei capitoli di questa tesi, è possibile creare automaticamente (tranne che per i vincoli manuali) un database relativo ad un DBMS differente. L'inserimento dei dati nel nuovo database richiederebbe la creazione di un *modulo di scambio dati* che, utilizzando il *modulo d'In/Out*, prelevi le informazioni contenute nel database Access e

le inserisca in quello nuovo (un alternativa sarebbe riutilizzare i *moduli di feeding* relativamente ad una fonte ODBC attinente al nuovo database, ma in tal modo non si inserirebbero tutti i dati delle modifiche effettuate successivamente all'utilizzo dei moduli di feeding).

Un ulteriore sviluppo potrebbe essere il passaggio da macchine dotate di sistemi operativi Microsoft a quelle adoperanti UNIX. Tale operazione (agevolata grazie all'utilizzo, all'interno del *modulo d'In/Out*, dell'interfaccia ODBC e delle sole librerie standard riportate nel *paragrafo 7.2*) non dovrebbe richiedere altro che una ricompilazione del codice sorgente del *modulo d'In/Out* tramite l'utilizzo di uno dei compilatori C++ forniti dagli ambienti UNIX.

Lo schema del database è stato sviluppato tenendo conto della possibilità di aggiungere nuovi benchmark, quindi, nel caso un ricercatore individui una nuova collezione d'interesse, sarebbe opportuno creare ulteriori moduli di feeding in grado di inserire i nuovi dati nel database.

Il sistema mette a disposizione dei ricercatori un enorme mole di dati e un modulo in grado di accedervi, ma senza la realizzazione di ulteriori moduli che permettano di implementare gli addestramenti di sistemi basati sull'approccio machine learning e dei classificatori da essi realizzati, il lavoro effettuato non avrebbe alcuna utilità. Inoltre ogni ricercatore dovrebbe essere in grado di implementare del codice che permetta di inserire nel database i risultati ottenuti, a tal fine il *modulo d'In/Out* potrebbe essere ampliato aggiungendo al suo interno delle funzionalità in grado di calcolare automaticamente le misurazioni relative alle entità degli esperimenti effettuati.

BIBLIOGRAFIA

- Fabrizio Sebastiani *"Machine Learning in Automated Text Categorization"*, ACM Computing Surveys, Vol 34, No 1, Marzo 2002.
- G.M. Di Nunzio e A. Micarelli *"Categorizzazione automatica di testi utilizzando Reti RBF Modificate"*, Dipartimento di Informatica e Automazione, Laboratorio di Intelligenza Artificiale, Università degli Studi "Roma tre", Roma, Italia.
- Alan Simpson e Elizabeth Olson *"Access 97 Manuale d'uso"*, tecniche nuove, Aprile 1998.
- Bruce Eckel *"Programmare in C++"*, Mc Graw Hiil, Marzo 1993.
- Jeff Promise, *"Programmare Microsoft Windows con MFC"*, Mondadori Informatica, settembre 1999.

RINGRAZIAMENTI

Con quest'ultima pagina termino non solo la mia tesi ma con gran probabilità anche il primo libro della mia vita, i cui capitoli pieni di momenti di gioia e tristezza mi hanno permesso di divenire una persona né cattiva né tanto meno buona, ma semplicemente una persona cosciente di ciò che la circonda e pronta, con i vincoli della condizione umana, a rispettare il prossimo.

Ringrazio tutte le persone che hanno creduto in me, quando nel massimo pessimismo desideravo solo fuggire da un mondo che non comprendevo, un mondo troppo assurdo ed ingiusto.

Ringrazio tutte le persone che lungo la mia carriera scolastica mi hanno deriso sostenendo che non avrei mai ottenuto buoni risultati, le ringrazio perché hanno risvegliato sempre la mia gran forza di volontà ed il mio desiderio di dimostrare che essere idealisti ed onesti può e deve essere un giusto modo di vivere.

Ringrazio i miei genitori che non mi hanno mai permesso di accontentarmi di una vita semplice, anche se forse è quella in grado di rendere più felici, poiché l'attaccamento agli oggetti non fa altro che renderci schiavi di cose senza importanza.

Ringrazio tutti i miei amici (e vi assicuro sono tanti e forse anche troppi per una sola persona) che si sono sempre mostrati gentili e che stranamente, anche se mi ritengo uno schizzopatico, mi hanno sempre rispettato senza mai abbandonarmi.

Ringrazio quell'Uomo che ha creato tanto scalpore da essere ancora oggi ricordato da tutta l'umanità, la quale, però, non ha prestato attenzione al fatto che anche se asceso affianco al Suo Padre era pur sempre un uomo e come tale ci ha dimostrato quanto ognuno di noi possa essere in grado di migliorare ciò che lo circonda seguendo le voci che gli giungono direttamente dall'anima.

L'evoluzione di un popolo
non deve essere misurata dal suo grado tecnologico,
ma dalla cultura e dalle interazioni sociali.
Meglio un popolo d'indigeni
dove la maggior parte degli individui può considerarsi felice,
che le società oggi ritenute "*civilizzate*"
dove gli individui sono stressati da ritmi inumani e problemi futili.