Laboratorio di Basi di dati 1

Suggerimenti per lo svolgimento della II esercitazione

18 marzo 2003

Note Introduttive

Questa seconda esercitazione ha lo scopo di illustrare l'uso di SQL Query Analyzer, mediante l'uso del Data Manipulation Language di SQL Server e di alcune semplici query. Dal query Analyser di SQL Server è possibile eseguire statement T-SQL, ossia comandi scritti nella particolare estensione imperativa di SQL che utilizza Microsoft SQL Server.

NB: Lo standard SQL:1999 non è pienamente supportato.

Importante! Si ribadisce che l'installazione di SQL Server fornita per l'esercitazione è corredata di manuale, richiamabile utilizzando la voce di menù `Books On Line`. Per questa esercitazione, risulterà particolarmente utile l'uso del capitolo dedicato a Transact-SQL (T-SQL). Utilizzare la scheda `Search` per reperire la sintassi dei comandi SQL. Attenzione: il capitolo completo consta di **1420** pagine... stampate i riferimenti solo per i comandi principali.

Punto 1.

- Per lanciare SQL Query Analyzer:
  - utilizzare la voce di menù `Start - Programs - Microsoft SQL Server - Query Analyser` oppure
  - utilizzare la voce di menù `Start - Programs - Microsoft SQL Server - Enterprise Manager` e, dopo aver digitato login e password, dal menù `Tools` scegliere la voce `SQL Query Analyzer`.

  NB: Ricordarsi di modificare il database sul quale vengono effettuate le query:
  - dalla finestra del Query Analyzer, scrivere il comando T-SQL

    ```
    USE nomeDB
    ```

    oppure
  - dal menù `Query` scegliere l'opzione `Change Database`, oppure
  - utilizzare l'unica casella combinata (combo box) che compare nella toolbar e selezionare il database corretto, oppure
  - premere `CTRL + U` e selezionare il database corretto.
- I comandi T-SQL vanno scritti all'interno della finestra bianca che viene aperta all'avvio del Query Analyzer. I comandi possono essere scritti indifferentemente in maiuscolo o in minuscolo. È possibile scrivere più comandi di seguito, da eseguire in sequenza, nell'ordine in cui sono scritti. Da una sequenza di comandi è possibile eseguirne un sottoinsieme, selezionandolo. Una sequenza di comandi può essere salvata per essere ri-eseguita in seguito. I file creati hanno estensione **.sql**. Nella parte inferiore della finestra sono presenti due schede. La scheda **Messages** riporta i messaggi dell'interprete T-SQL. Possono essere messaggi d'errore oppure di conferma dell'avvenuta esecuzione di un comando o di una sequenza di comandi. La scheda **Grids** riporta le tuple risultato di un'interrogazione (statement `SELECT`). Possono essere salvati sia i messaggi di errore che il risultato delle interrogazioni.
- Per eseguire un comando T-SQL:
  - dal menù `Query` scegliere l'opzione `Execute`, oppure
  - cliccare sul pulsante della toolbar con icona una freccia verde, oppure
  - premere `F5`

È possibile verificare la correttezza della sintassi di un comando T-SQL senza eseguire la query. Questo è molto utile soprattutto per provare le diverse forme della sintassi per i comandi del DDL e del DML, che non possono essere correttamente eseguiti più di una volta.

Per verificare la sintassi di un comando T-SQL:

- dal menù `Query` scegliere l'opzione `Parse`, oppure
- cliccare sul pulsante della toolbar con icona una "v" blu, a sinistra della freccia verde, oppure
- premere `Ctrl+F5`

- Il comando T-SQL per cancellare una tabella è `DROP TABLE` *`nometabella`*
- Il comando T-SQL per creare un database è `CREATE DATABASE` *`nomedb`*

Punto 2.

- Il comando T-SQL per la creazione di una tabella è `CREATE TABLE`, che ha sintassi di base:

```
CREATE TABLE nomeTabella
  (specificacolonna_1 [, ..., specificacolonna_n]);
```

dove `specificacolonna_i`, con $i = 1,\ldots,n$, è:

```
NomeColonna_i Dominio_i [DEFAULT valore][NULL(def)|NOT NULL]
   [PRIMARY KEY|UNIQUE][CHECK condizione_su_Colonna_i]
```

È possibile specificare i vincoli anche alla fine della definizione della tabella, come vincoli di tabella, secondo la sintassi vista a lezione. Ad esempio:

```
CREATE TABLE
  (NomeColonna_1 Dominio_1 [, ..., NomeColonna_n Dominio_n],
    PRIMARY KEY (elenco_nomi_colonne));
```

Questa seconda possibilità permette di definire chiavi su più attributi.

È possibile anche assegnare nomi ai vincoli, definendoli tramite l'argomento `CONSTRAINT`. Questo permette di recuperarli in seguito più facilmente (i nomi assegnati automaticamente da SQL Server sono piuttosto lunghi), ad esempio per cancellarli. Vedere il manuale in linea per la sintassi specifica.

- Nella definizione della tabella, utilizzare, se e dove necessario, i vincoli `PRIMARY KEY`, `NOT NULL`.
- Per verificare la struttura della tabella, è sufficiente fare doppio click sul nome della tabella dall'elenco delle tabelle del database `GestioneCorsi`*N*. I vincoli di default sono visibili dalla finestra Design Table, i check dalle proprietà della tabella (finestra Properties, sempre dalla finestra Design Table) .

Punto 3.

- Il comando T-SQL per la modifica della struttura di una tabella è `ALTER TABLE`. La sintassi per aggiungere una colonna a una tabella è (il ; finale è opzionale):

```
ALTER TABLE
  ADD specificaColonna;
```

- Una definizione di tipo corretta per il campo `Stipendio` potrebbe essere `decimal(8,2)`. **Oss:** In T-SQL i tipi di dato `numeric` e `decimal` hanno un'implementazione equivalente.
- Nella modifica della tabella, utilizzare, i vincoli `DEFAULT` *`valore costante`* e `CHECK` *`condizione`*.
- In T-SQL il tipo di dato `boolean` non è definito. Utilizzare il tipo di dato `bit`, che ha semantica equivalente e può assumere valori 1 per true e 0 per false.

- La sintassi della `ALTER TABLE` per modificare una colonna a una tabella è (il ; finale è opzionale):

```
ALTER TABLE
  ALTER COLUMN specificaColonna;
```

- La sintassi della `ALTER TABLE` per cancellare una colonna a una tabella è (il ; finale è opzionale):

```
ALTER TABLE
  DROP COLUMN NomeColonna;
```

Punto 4.

Il vincolo di `FOREIGN KEY` può essere impostato, secondo la sintassi T-SQL, come vincolo di colonna o come vincolo di tabella.

Di seguito viene riportata una versione più estesa, rispetto a quella precedentemente vista, della sintassi della `CREATE TABLE`, in simil BNF, che mostra entrambi le possibilità. Per la sintassi completa si veda il manuale in linea, con riferimento al capitolo su Transact SQL (effettuare una ricerca utilizzando le parole chiave `CREATE TABLE`.

```
CREATE TABLE nomeTabella
    ( < definizione_di_colonna >
    | < vincolo_di_tabella  >
    )


< definizione_di_colonna > ::= { nome_di_colonna dominio }
    [ DEFAULT costante ]
    [ < vincolo_di_colonna > ]

< vincolo_di_colonna > ::=
    { [ NULL | NOT NULL ]
    | [ { PRIMARY KEY | UNIQUE }]
    | [ [ FOREIGN KEY ]
          REFERENCES nome_tabella_riferita [ (lista_nomi_colonne_riferite) ]
          [ ON DELETE { CASCADE | NO ACTION } ]
          [ ON UPDATE { CASCADE | NO ACTION } ]]
    | CHECK ( condizione )
    }

< vincolo_di_tabella > ::=
   { [ { PRIMARY KEY | UNIQUE } (lista_nomi_colonne)]
   | FOREIGN KEY
     [ ( lista_nomi_colonne) ]
        REFERENCES nome_tabella_riferita [ (lista_nomi_colonne_riferite) ]
        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]
    | CHECK ( condizioni )
    }
```

`CASCADE` imposta la propagazione di una modifica o di una cancellazione, mentre `NO ACTION`, che è il valore di default, non permette né la modifica né la cancellazione se esistono tuple riferite (semantica equivalente a RESTRICT, nello standard SQL:1999). Si noti l'assenza degli argomenti `MATCH`, `SET NULL`, `SET DEFAULT`, `RESTRICT` che sono invece previsti dallo standard SQL:1999.

Punto 5.

Per l'attributo `Iscrizione` si usi il tipo `char`, di 9 caratteri.

Punto 6.

- La forma generale del comando T-SQL per eseguire interrogazioni è:

```
SELECT [DISTINCT] lista_colonne
FROM lista_tabelle | espressione_di_join
[WHERE condizione ]
[ORDER BY order_expression [ ASC | DESC ] ]
```

dove:

```
< condizione > ::=
    {    [ NOT ] < predicato > | ( < condizione > ) }
        [ { AND | OR } [ NOT ] { < predicato > | ( < condizione > ) } ]
    }

< predicato > ::=
    {    espressione1 { = | < > | ! = | > | > = | ! > | < | < = | ! < } espressione2
        | stringa [ NOT ] LIKE stringa_pattern_matching
        | espressione1 [ NOT ] BETWEEN espressione2 AND espressione3
        | espressione1 IS [ NOT ] NULL
        | espressione1 [ NOT ] IN (elenco di valori| sottoquery)
    }
```

dove `espressionei` può essere un nome di colonna, stringa è una stringa di caratteri alfanumerici.

- L'argomento opzionale `LIKE` permette di effettuare i confronti fra stringhe. Una stringa, in SQL Server, è una sequenza di caratteri e numeri fra '' (apici singoli).

- Nelle stringhe di pattern matching dell'argomento opzionale `LIKE`, "_" (underscore) indica un carattere qualunque, mentre "%" indica una stinga da 0 a n caratteri. `LIKE`, in T-SQL, utilizza anche altri caratteri jolly per esprimere stringhe di pattern matching più complesse. Si veda il manuale in linea per l'uso di questi caratteri.

- Gli innner join previsti da SQL Server sono il cross join e il theta-join, con sintassi:

```
relazione1 CROSS JOIN relazione2

relazione1 JOIN relazione2 ON predicato_di_join.
```