SQL

Structured Query Language

Il linguaggio

- SQL è un linguaggio di interrogazione per database progettato per
  - leggere,
  - modificare
  - gestire **dati** memorizzati in un sistema basato sul **modello relazionale**
  - creare e modificare **schemi** di database
  - creare e gestire strumenti di **controllo** ed **accesso** ai dati.

Evoluzione del linguaggio

- Le origini di SQL si trovano in un documento del **1970** realizzato da Edgar Codd, "*A Relational Model of Data of Large Shared Data Banks*".
- La prima versione fu sviluppata da IBM all'inizio degli anni settanta. Chiamata originariamente SEQUEL era progettata per manipolare dati memorizzati nel database relazionale ideato e brevettato da IBM.
- Primo **standard SQL-86** pubblicato da ANSI e ratificato da ISO nel 1987
  - (ANSI e ISO sono due organismi internazionali che si occupano della standardizzazione delle tecnologie).
- **SQL-92 (SQL 2)** è lo standard a cui fanno riferimento la maggior parte dei DBMS.
- L'evoluzione del linguaggio ha portato a due ulteriori versioni: **SQL:1999** (oggetti) e **SQL:2003** (xml).

I linguaggi "dentro" SQL

- **DDL**
  - (Data Definition Language, linguaggio di definizione dei dati).
  - Consente di descrivere la struttura delle tabelle e di tutti gli elementi di supporto (come indici, vincoli, trigger, viste ecc.).
  - Viene utilizzato per realizzare lo schema logico e lo schema fisico del database.
- **DML**
  - (Data Manipulation Language, linguaggio per la manipolazione dei dati).
  - Operazioni di inserimento, modifica e cancellazione dei dati.
- **DCL**
  - (Data Control Language, linguaggio di controllo dei dati).
  - Limiti sui dati (permessi di accesso, vincoli di integrità).
- **QL**
  - (Query Language, linguaggio di interrogazione)
  - Interrogare il database al fine di individuare i dati che corrispondono ai parametri di ricerca dell'utente.

Utilizzo di SQL

- **Interattivo**
  - L'utente utilizza un software, in genere fornito con il DBMS, in cui introdurre comandi SQL che vengono inviati al DBMS.
- **All'interno di applicazioni software**
  - L'interazione con il database è scritta in SQL mentre il resto dell'applicazione software in un comune linguaggio di programmazione.
  - I comandi SQL possono essere poi collegati nel programma in due modalità differenti:
  - "**ospitati**" nel codice del software e inviati al DBMS all'occorrenza.
  - **memorizzati all'interno del DBMS** e quindi richiamati dal programma.

I dialetti SQL

- Alcune delle critiche più frequenti rivolte ad SQL riguardano la mancanza di portabilità del codice fra implementazioni diverse.
- Sono spesso differenti alcuni tipi di dato e la sintassi di alcuni operatori particolari (es. LIKE)

Tipi di dato (SQL Server)

- bigint (8 bytes) Contiene valori numerici interi da -4294967296 a 4294967294.
- binary(n) (lunghezza fissa) Contiene dati binari (1 byte) fino ad un massimo di 8000 dati.
- bit (1 bit) Rappresenta i flag (vero/falso o true/false o si/no). Non possono avere valori nulli e non possono avere indici.
- char(n) (lunghezza fissa) Contiene caratteri ANSI (1 byte) fino ad un massimo di 8000 caratteri.
- datetime (8 bytes) Contiene date tra il 1/gen/1753 e il 31/dic/9999 (precisione al trecentesimo di secondo).
- decimal(p, s) (da 2 bytes a 17 bytes) Contiene valori tra 10^38 - 1 e - 10^38 -1. Con p cifre di precisione (massimo 28), e s cifre decimali dopo la virgola (scala).
- float (8 bytes) Contiene numeri reali positivi da 2.23E-308 a 1.79E308 e negativi da -2.23E-308 a -1.79E308 (massimo 15 cifre di precisione).
- image Contiene fino a 2147483647 bytes di dati binari (è solitamente usato per le immagini).
- int (4 bytes) Contiene valori numerici interi da -2147483648 a 2147483647.
- money (8 bytes) Contiene valori monetari da -922337203685477.5808 a 922337203685477.5807

Tipi di dato (2)

- nchar(n) (lunghezza fissa) Contiene caratteri UNICODE (2 bytes) fino ad un massimo di 4000 caratteri.
- ntext (lunghezza variabile) Contiene caratteri UNICODE fino ad un massimo di 1073741823 caratteri.
- numeric(p, s) E' equivalente al tipo 'decimal(p, s)'
- nvarchar(n) (lunghezza variabile) Contiene caratteri UNICODE (2 bytes) fino ad un massimo di 4000 caratteri.
- real (4 bytes) Contiene numeri reali positivi da 1.18E-38 a 3.40E38 e negativi da -1.18E-38 a -3.40E38 (massimo 7 cifre di precisione).
- smalldatetime (4 bytes) Contiene date tra il 1/gen/1753 e il 31/dic/9999 (precisione al minuto).
- smallint (2 bytes) Contiene valori numerici interi da -32768 a 32767.
- smallmoney (4 bytes) Contiene valori monetari da - 214748.3648 a 214748.3647
- sql_variant Tipo che può contenere tipi di dati diversi (int, binary, char).
- text (lunghezza variabile) Contiene caratteri ANSI (1 byte) fino ad un massimo di 2147483647 caratteri.
- timestamp (8 bytes) È un contatore incrementale per colonna assegnato automaticamente da SQL Server 7.
- tinyint (1 byte) Contiene valori numerici interi da 0 a 255.
- uniqueidentifier (16 bytes) E' un identificatore unico a livello globale E' generato automaticamente da SQL Server.
- varbinary(n) (lunghezza variabile) Contiene dati binari (1 byte) fino ad un massimo di 8000 dati.
- varchar(n) (lunghezza variabile) Contiene caratteri ANSI (1 byte) fino ad un massimo di 8000 caratteri.
- xml ◊ è equivalente al tipo 'ntext'.

Operatori (SQL Server)

- + ◊ Addizione
- - ◊ Sottrazione
- * ◊ Prodotto
- / ◊Divisione
- % ◊ Modulo
- < ◊ Minore
- > ◊ Maggiore
- <= ◊ Minore o Uguale
- >= ◊ Maggiore o Uguale
- = ◊ Uguaglianza
- <> ◊ Disuguaglianza
- AND ◊ E logico
- OR ◊ O logico
- NOT ◊ Negazione

DDL

Data Definition Language

Creazione database

- CREATE DATABASE <NomeDB>
- Es.
  - CREATE DATABASE Cinema

Creazione tabella

```
CREATE TABLE <NomeTabella> (
     <NomeCampo1> <Tipo1> [NOT NULL],
     <NomeCampo2> <Tipo2> [NOT NULL],
     …
     <NomeCampoN> <TipoN> [NOT NULL],
);
```

Modifica tabella

- Aggiungere un nuovo campo ad una tabella:

```
ALTER TABLE <NomeTabella>
      ADD <NomeCampo1> <Tipo1> [NOT NULL];
```

- Modificare il tipo di un campo:

```
ALTER TABLE <NomeTabella>
      ALTER COLUMN <NomeCampo> <NuovoTipo>;
```

- Eliminare un campo

```
ALTER TABLE <NomeTabella>
      DROP COLUMN <NomeCampo1>;
```

Eliminazione tabella

```
DROP TABLE <NomeTabella>;
```

- Attenzione: Non è possibile eliminare una tabella a cui fa riferimento un vincolo FOREIGN KEY. È prima necessario eliminare il vincolo FOREIGN KEY o la tabella di riferimento.

Creazione nuovi domini

- Definire un nuovo tipo di dato:
- `CREATE TYPE <NomeTipo> FROM <TipoBase> <Condizione>;`

- <NomeTipo> è il nome del tipo di dato da creare.
- <TipoBase> è il tipo già esistente dal quale deriva il nuovo tipo.
- <Condizione> è la condizione che i dati del nuovo tipo devono soddisfare.

I vincoli

- I vincoli consentono di specificare **controlli** sui dati, al fine di assicurare la **correttezza** e **consistenza** dell'informazione.
- I vincoli possono essere:
  - **interni** (o intrarelazionali) specificano controlli sulla singola tabella intesa come entità a se stante
  - di **integrità referenziale** riguardano i rapporti tra una tabella e l'altra.

Vincoli interni

- NOT NULL
  - Impedisce di inserire un dato nullo nel campo in cui viene specificato.
  - <NomeCampo> <Tipo> NOT NULL;
- PRIMARY KEY
  - Imposta un campo (o più campi) come chiave primaria della tabella.
  - PRIMARY KEY (<NomeCampo>);
- CHECK
  - Indica un controllo su un'espressione tra i campi della tabella.
  - CHECK (<NomeCampo> VALUE IN (<valori>));
  - CHECK (<NomeCampo> VALUE BETWEEN (<valore1> AND <valore2>));

Vincoli di integrità referenziale

- FOREIGN KEY
  - Imposta una chiave esterna in una tabella, con campi che fanno riferimento ad un'altra tabella del DataBase.
  - FOREIGN KEY (<ElencoCampi>) REFERENCES <NomeTabella> (<ElencoCampiTabella>);

- <ElencoCampi>
  - Elenco dei campi della tabella corrente.
- <NomeTabella>
  - Tabella in cui sono presenti i campi esterni.
- <ElencoCampiEsterni>
  - Elenco dei campi della tabella di riferimento.

Integrità referenziale

- L'integrità referenziale viene controllata anche dalle parole chiave RESTRICT, CASCADE e SET NULL, che consentono di controllare la risposta del database a un vincolo.
- RESTRICT
  - Il database rifiuta le modifiche violano un vincolo
- CASCADE
  - Il database propaga a cascata le modifiche
- SET NULL
  - E' consentita la modifica alla tabella principale, eventuali riferimenti in altre tabelle non più validi vengono posti a NULL

QL

Query Language

SELECT

- Per estrarre informazioni dalla base di dati si utilizza l'istruzione SELECT.
- La sintassi completa dell'istruzione SELECT è complessa perché l'istruzione implementa varie funzionalità.

SELECT (proiezione)

SELECT [DISTINCT]
<Campo1> [AS "Alias1"],
<Campo2> [AS "Alias2"],
…
<CampoN> [AS "AliasN"]
FROM <Tabella1>, <Tabella2>, … <TabellaN>

- DISTINCT - Questa opzione permette di ottenere solo tuple differenti tra loro.
- <Campo> - Elenco dei campi da estrarre.
- <Tabella> - Tabella in cui sono contenuti i campi da estrarre.
- "Alias" - Etichetta da assegnare al campo nella selezione (facoltativa).
- * Sostituendolo ai nomi dei campi implica la selezione di tutti i campi della tabella specificata.

Esempi

- Selezione di un'intera tabella

```
SELECT *
FROM Genere
```

- Selezione di alcuni campi di una tabella (proiezione)

```
SELECT fi_titolo, fi_regia
FROM Film
```

- Selezione (senza duplicazione)

```
SELECT DISTINCT fi_titolo
FROM Film
```

SELECT (restrizione)

- Per estrarre informazioni dal DB, limitate da una condizione:

SELECT [DISTINCT]
<Campo1>, <Campo2>, … <CampoN>
FROM <Tabella>
[WHERE <Condizione>]

- <Condizione> - Indica la condizione che devono soddisfare le tuple estratte. All'interno di questa espressione è possibile specificare:
- nomi dei campi della tabella;
- operatori di confronto, come =, <>, >, >=, <=, <;
- operatori logici come NOT, AND, OR;
- l'operatore LIKE;
- la parola chiave IS NULL o IS NOT NULL.

Esempi

- Selezione delle righe che soddisfano una condizione (restrizione)

```
SELECT *
FROM Film
WHERE fi_durata>100
```

- Selezione con condizione composta

```
SELECT *
FROM Film
WHERE fi_durata>100 AND fi_titolo LIKE 'M%'
```

Esempi

- Selezione di alcuni campi delle righe che soddisfano una condizione (restrizione e proiezione)

```
SELECT fi_titolo, fi_regia
FROM Film
WHERE fi_titolo LIKE '%K'
```

- Alias per le colonne

```
SELECT fi_titolo,
fi_regia AS Regista
FROM Film
WHERE fi_titolo LIKE '_L%'
```

Esempi

- Selezione di valori NULL

```
SELECT *
FROM Film
WHERE fi_titoloOriginale IS NULL
```

- Selezione di valori NOT NULL

```
SELECT *
FROM Film
WHERE fi_titoloOriginale IS NOT NULL
```

SELECT (join)

- Per concatenare due tabelle in base ad un campo comune (JOIN) può essere utilizzata l'istruzione SELECT-WHERE, con una particolare condizione:

```
SELECT [DISTINCT]
<Campo1>, <Campo2>, ... <CampoN>
FROM <Tabella1>, <Tabella2>, ... <TabellaN>
WHERE <Tabella1>.<Campo1> = <Tabella2>.<Campo2>
```

Esempi

- Primo formato

```
SELECT *
FROM Film, Genere
WHERE Film.fi_genere = Genere.ge_codice
```

- Formato esplicito

```
SELECT *
FROM Film INNER JOIN Genere
        ON Film.fi_genere = Genere.ge_codice
```

Left Outer Join

- Oltre alle righe che soddisfano la condizione vengono anche incluse tutte le righe della prima tabella

```
SELECT *
FROM Film LEFT OUTER JOIN Premio
ON Premio.pr_film = Film.fi_codice
```

- In questo caso anche i film che non hanno vinto premi
- Esistono anche Right Outer Join … Full Outer Join …

Unione di due tabelle

- Per accodare i campi due tabelle compatibili (con campi omogenei):

```
( SELECT
 <Campo1>
  FROM <Tabella1>
UNION
 SELECT
 <Campo2>
 FROM <Tabella2> );
```

Esempio

```
SELECT fi_titolo, fi_regia
FROM   Film
WHERE Film.fi_regia='Fellini Federico'

UNION

SELECT fi_titolo, fi_regia
FROM   Film INNER JOIN Premio
       ON Premio.pr_film = Film.fi_codice
WHERE  Premio.pr_anno='1975'
```

Differenza

- Per estrarre da due tabelle compatibili (con campi omogenei) solo i record presenti nella prima ma non nella seconda:

```
( SELECT
        <Campo1>
  FROM <Tabella1>
EXCEPT
 SELECT
        <Campo2>
  FROM <Tabella2> );
```

Esempio

```
SELECT fi_titolo, fi_regia
FROM   Film
WHERE Film.fi_regia='Fellini Federico'

EXCEPT

SELECT fi_titolo, fi_regia
FROM   Film INNER JOIN Premio
       ON Premio.pr_film = Film.fi_codice
WHERE  Premio.pr_anno='1975'
```

Intersezione

- Per estrarre da due tabelle compatibili (con campi omogenei) i record che entrambe le tabelle hanno in comune:

```
( SELECT
        <Campo1>
  FROM <Tabella1>
INTERSECT
 SELECT
        <Campo2>
  FROM <Tabella2> );
```

Esempio

```
SELECT fi_titolo, fi_regia
FROM   Film
WHERE Film.fi_regia='Fellini Federico'

INTERSECT

SELECT fi_titolo, fi_regia
FROM   Film INNER JOIN Premio
       ON Premio.pr_film = Film.fi_codice
WHERE  Premio.pr_anno='1975'
```

Funzioni di aggregazione

- SQL dispone di alcune modalità per effettuare calcoli sui dati, senza per questo modificare i dati in tabella: il calcolo di espressioni e l'utilizzo di funzioni predefinite.

Funzioni per i calcoli sui dati

- COUNT([DISTINCT] <Campo>)
  - Conta il numero di elementi del campo indicato.
- MIN(<Campo>)
  - Restituisce il valore minimo del campo indicato.
- MAX(<Campo>)
  - Restituisce il valore massimo del campo indicato.
- SUM([DISTINCT] <Campo>)
  - Calcola e restituisce la somma dei valori presenti nel campo indicato.
- AVG([DISTINCT] <Campo>)
  - Calcola e restituisce la media aritmetica dei valori presenti nel campo indicato.

Ordinamento

- Per raggruppare i campi selezionati in base al valore di uno o più campi:

```
SELECT [DISTINCT]
        <Campo1>, <Campo2>, ... <CampoN>
FROM <Tabella1>, <Tabella2>, ... <TabellaN>
[WHERE <Condizione>]
[ORDER BY <CampoOrdine1> [ASC | DESC], <CampoOrdine2>
[ASC | DESC], ... <CampoOrdineN> [ASC | DESC]];
```

- <CampoOrdine> - Campo(i) in base al(ai) quale(i) ordinare il risultato ottenuto dalla SELECT.
- ASC | DESC - Indicano l'ordinamento crescente [ASC] o decrescente [DESC] dei campi. Di default viene impostato il modificatore ASC.

Raggruppamento

- GROUP BY raggruppa le righe sulla base del valore di uno o più attributi, in genere per effettuare calcoli aggregati su dati omogenei.

Raggruppamento (esempio)

- Per ordinare i campi selezionati:

```
SELECT [DISTINCT]
        <Campo1>, <Campo2>, ... <CampoN>
FROM <Tabella1>, <Tabella2>, ... <TabellaN>
[WHERE <Condizione>]
[GROUP BY <CampoGruppo1>, < CampoGruppo2>, ...
<CampoGruppoN>
        [HAVING <CondizioneGruppo>]];
```

- <CampoGruppo> - Campo(i) in base al(ai) quale(i) raggruppare tutti i record ottenuti dalla SELECT.
- <CondizioneGruppo> - Specifica la condizione secondo la quale verranno raggruppati i record.

HAVING

- È anche possibile restringere il risultato specificando una condizione che può considerare sia i campi sia il valore di funzioni di aggregazione.

Interrogazioni nidificate

- Talvolta le operazioni di interrogazione si rivelano particolarmente complesse; in questo caso, è necessario utilizzare più istruzioni SELECT al fine di ottenere tutti i dati voluti.

```
SELECT
        <Campo1>
FROM <Tabella1>
WHERE <Campo1> = (
                SELECT
                        <Campo2>
                FROM <Tabella2>
                WHERE <Condizione2>);
```

ANY - ALL

- ANY ritorna vero se il confronto indicato è vero per almeno uno degli elementi identificati dalla query nidificata
- ALL ritorna vero se il confronto indicato è vero per tutti gli elementi individuati dalla query nidificata.
- ANY e ALL sono più potenti di IN, in quanto consentono di utilizzare operatori di confronto >, >=, <= e <

DML

Data Manipulation Language

Inserimento dati

```
INSERT INTO <NomeTabella>
        [(<Campo1>, <Campo2>, … <CampoN>)]
VALUES
        (<Valore1>, <Valore2>, … <ValoreN>);
```

- <NomeTabella> - Nome della tabella in cui inserire i dati.
- <Campo> - Lista dei campi della tabella in cui inserire i valori specificati di seguito.
- <Valore> - Lista dei valori da inserire nei rispettivi campi.
- L'elenco dei campi è opzionale; se non viene specificato è necessario inserire un valore per tutti i campi della tabella.

Modifica dati

```
UPDATE
        <NomeTabella>
SET
        <Campo1> = <Valore1>,
        <Campo2> = <Valore2>,
        …
        <CampoN> = <ValoreN>
[WHERE <Condizione>];
```

- <NomeTabella> - Nome della tabella in cui modificare i dati.
- <Campo> - Lista dei campi della tabella in cui modificare i dati esistenti con i valori seguenti.
- <Valore> - Lista dei valori da sostituire a quelli dei rispettivi campi.
- Se non viene specificata alcuna condizione WHERE, il valore inserito viene sostituito ai valori di ogni campo.

Eliminazione dati

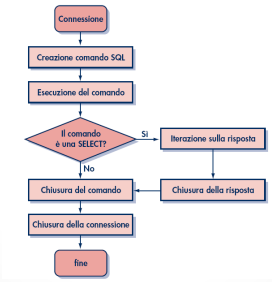
```
DELETE FROM <NomeTabella>
[WHERE <Condizione>];
```

- <NomeTabella> - Nome della tabella dalla quale verranno eliminati i dati.
- <Condizione> - Condizione che deve essere soddisfatta dai campi che verranno eliminati.
- Se non viene specificata alcuna condizione WHERE, viene eliminato il valore di ogni campo.

SQL come linguaggio ospitato

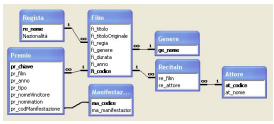
- Per eseguire comandi SQL da un programma scritto in un linguaggio differente è necessario effettuare alcune operazioni aggiuntive.
- **Connessione**: per ottenere un oggetto che consentirà di eseguire uno o più comandi SQL. La connessione è necessaria per stabilire con quale database si vuole operare e per fornire dati di autenticazione (in genere utente/password).
- **Creazione** di un **comando SQL**: viene creato un oggetto che rappresenta un'istruzione SQL e che viene impostato con uno specifico comando.
- **Esecuzione** del **comando**: comporta il passaggio dell'oggetto che rappresenta il comando a quello che rappresenta la connessione, in modo che il comando venga eseguito.
- **Iterazione sulla risposta**: il risultato di un'istruzione SELECT è una tabella e in genere un programma deve scorrere le righe del risultato per elaborarle.
- **Chiusura** della **risposta**: l'oggetto che rappresenta la risposta, una volta utilizzato, deve essere chiuso e rilasciato dalla memoria.
- **Chiusura** del **comando**: l'oggetto che rappresenta il comando SQL, una volta utilizzato, deve essere chiuso e rilasciato dalla memoria.
- **Chiusura** della **connessione**: l'oggetto che rappresenta la connessione SQL, prima della conclusione del programma, deve essere chiuso e rilasciato dalla memoria.

SQL ospitato



Data Base "Cinema"

Esercizi



1. Modificare in "ITA" la Nazionalità di tutti i registi che hanno nome che termina per "Mario".
2. Recuperare Titolo e Titolo originale di tutti i film di un certo regista.
3. Recuperare Titolo e nome del regista di film di genere commedia diretti da registi di Nazionalità "ITA".
4. Recuperare Titolo e Regista dei film che hanno avuto una candidatura all'Oscar per la Regia.
5. Recuperare Titolo, Regista e anno dei film che hanno vinto l'Oscar per la Regia.
6. Recuperare Nome degli attori che hanno recitato in un certo film.
7. Recuperare Titolo e Regista dei film in cui ha recitato un certo attore.
8. Recuperare il Titolo dei film candidati ai premi del festival di Venezia in un certo anno.
9. Aggiungere il campo at_foto di 30 caratteri alla tabella Attore
10. Modifica il campo at_foto in Attore ponendolo uguale ad at_nome+".jpg"