IL LINGUAGGIO SQL (STRUCTURED QUERY LANGUAGE)

Marco Masseroli, PhD

IL LINGUAGGIO SQL

Premessa:

L'SQL (Linguaggio Strutturato per le Interrogazioni) è uno standard per tutti i DBMS (Data Base Management System) grazie alla sua semplicità, potenza e flessibilità.

Tuttavia ogni DBMS utilizza dialetti diversi dello standard SQL per interagire con i propri DB (Data Base) e i dati in essi contenuti.

IL LINGUAGGIO SQL

Istruzioni SQL:

In SQL vi sono istruzioni che agiscono:

- sulla struttura di un DB e quindi sulla struttura delle sue tabelle e sulle loro relazioni;
- sui dati contenuti in una o più tabelle di un DB.

IL LINGUAGGIO SQL

Istruzioni SQL sulla struttura:

Le operazioni che si possono eseguire sulla struttura di un DB sono creazione/cancellazione di tabelle (con relativa struttura dati e relazioni) e di indici. Per ogni operazione vi è un istruzione operativa in SQL:

- CREATE TABLE ...
- CREATE INDEX … ON ...
- ALTER TABLE ...
- DROP TABLE … / DROP INDEX …ON ...

IL LINGUAGGIO SQL

CREATE TABLE …

Crea una tabella in un DB con i campi ed il relativo tipo di dati e dimensioni (se tipo testo) specificati.

```
CREATE TABLE tabella (campo1 tipo1 [(dimensione)]
    [NOT NULL] [DEFAULT defaultValue1] [indice1]
    [, campo2 tipo2 [(dimensioni)] [NOT NULL] [DEFAULT
    defaultValue2] [indice2] [, ...]]
    [, CONSTRAINT nomeVincolo [, ...]]
    [, ON DELETE azioneCancellazione]
    [, ON UPDATE azioneModifica …]);
```

IL LINGUAGGIO SQL

tipo: tipo di dato (es. text, byte, integer, float, double, …);

dimensione: solo per tipo text, indica il numero di caratteri;

NOT NULL: definisce un campo con valori richiesti;

DEFAULT: definisce il valore predefinito per il campo;

indice: nome dell'indice da creare sul relativo campo;

CONSTRAINT: definisce un vincolo creato su uno o più campi;

ON DELETE: definisce l'azione da svolgere a seguito della cancellazione di un valore in un campo (es. set default);

ON UPDATE: definisce azione svolta a seguito della modifica di un valore in un campo (es. aggiorna campi collegati).

IL LINGUAGGIO SQL

Vincolo su un singolo campo:

CONSTRAINT nomeVincolo {PRIMARY KEY | UNIQUE | NOT NULL | REFERENCES tabellaEsterna [(campoEsterno1, campoEsterno2)]};

PRIMARY KEY: definisce un campo chiave primaria;

UNIQUE: definisce un campo a valori univoci;

NOT NULL: definisce un campo con valori richiesti;

REFERENCES: definisce la tabella esterna e relativo/i campo/i in relazione con il campo su cui agisce il vincolo.

IL LINGUAGGIO SQL

Vincolo su più campi:

```
CONSTRAINT nomeVincolo {
   PRIMARY KEY (primaria1[, primaria2 [, ...]]) |
   UNIQUE (univoca1[, univoca2 [, ...]]) |
   NOT NULL (nonNull1[, nonNull2 [, ...]]) |
   FOREIGN KEY (rif1[, rif2 [, ...]]) REFERENCES tabellaEsterna
   [(campoEsterno1 [, campoEsterno2 [, ...]])]};
```

primaria1, …: campo/i chiave primaria;

univoca1, …: campo/i a valori univoci;

nonNull1, …: campo/i con valori richiesti;

IL LINGUAGGIO SQL

FOREIGN KEY: definisce campo/i (rif1[, rif2 [, ...]]) di relazione con altra tabella esterna;

REFERENCES: definisce la tabella esterna e relativo/i campo/i (campoEsterno1 [, campoEsterno2 [, ...]]) in relazione con il/i campo/i rif1[, rif2 [, ...]].

IL LINGUAGGIO SQL

Creazione tabella e suo inserimento dati (SELECT ... INTO ...):

Si può creare una tabella in un DB con i record selezionati da un'altra tabella in base a dati criteri.

```
SELECT campo1 [, campo2, campo3]
INTO nuovaTabella [IN DBesterno]
FROM tabella
WHERE campo1 = 6;
```

IL LINGUAGGIO SQL

CREATE INDEX … ON …

Crea un nuovo indice in una tabella esistente.

```
CREATE [UNIQUE] INDEX indice
ON tabella (campo1 [ASC | DESC][, campo2 [ASC | DESC], ...])
[WITH {PRIMARY | DISALLOW NULL | IGNORE NULL }];
```

UNIQUE: crea indice a valori univoci;

ASC | DESC: crea indice crescente o decrescente;

WITH: definisce le regole di integrità dei dati;

IL LINGUAGGIO SQL

{... | ...}: indica uno a scelta tra i successivi elementi tra parentesi:

PRIMARY: crea l'indice come chiave primaria;

DISALLOW NULL: impedisce l'esistenza di valori "null" nel/i campo/i indicizzato/i;

IGNORE NULL: impedisce che i record con valori "null" nel/i campo/i indicizzato/i siano inclusi nell'indice.

IL LINGUAGGIO SQL

ALTER TABLE …

Modifica la struttura di una tabella creata con CREATE TABLE, aggiungendo o eliminando un campo o un indice multi campo.

```
ALTER TABLE tabella
  {ADD {COLUMN campo tipo [(dimensioni)] [NOT NULL]
  [CONSTRAINT indice1] | CONSTRAINT nomeVincolo} |
  DROP {COLUMN campo | CONSTRAINT indice2}};
```

IL LINGUAGGIO SQL

ADD COLUMN: aggiunge un campo alla tabella dandone nome, tipo dati (es. text, integer, …), e dimensioni (solo per campi testo);

NOT NULL: definisce un campo con valori richiesti;

indice: nome dell'indice da creare (CONSTRAINT indice) sul relativo campo;

ADD CONSTRAINT: aggiunge un vincolo su più campi (nomeVincolo);

DROP COLUMN: elimina un campo dandone il nome;

DROP CONSTRAINT: elimina un indice su più campi .

IL LINGUAGGIO SQL

DROP TABLE …

Elimina una tabella esistente in un DB.

```
DROP TABLE tabella;
```

DROP INDEX … ON ...

Elimina un indice esistente su un campo di una tabella.

```
DROP INDEX indice ON tabella;
```

IL LINGUAGGIO SQL

Istruzioni SQL sui dati:

Le operazioni che si possono eseguire sui dati di un DB sono essenzialmente quattro: ricerca, inserimento, modifica e cancellazione. Per ognuna di esse vi è un istruzione operativa in SQL:

- SELECT … FROM ...
- INSERT INTO …
- UPDATE … SET …
- DELETE FROM …

IL LINGUAGGIO SQL

SELECT … FROM …

Effettua le ricerche selezionando i record in base a criteri specifici.

```
SELECT campo1, campo2, campo3 FROM tabella;
```

DISTINCT: qualora esistano, permette non estrarre i valori duplicati dei campi selezionati.

```
SELECT DISTINCT campo1, campo2 FROM tabella;
```

IL LINGUAGGIO SQL

WHERE: definisce i criteri per la selezione. Vi si possono usare tutti gli operatori logici e di confronto, oltre agli operatori: EXISTS e NOT EXISTS, IS NULL, IN () e NOT IN (), ALL ().

```
SELECT campo1, campo2 FROM tabella
WHERE (campo1 <= 4 AND campo2 EXISTS) OR
      (campo1 = 12 AND campo2 IS NULL);
```

Se la SELECT è su più tabelle e non vi sono condizioni (WHERE), si ottiene il prodotto cartesiano tra le tabelle!

```
SELECT tabella1.campo1, tabella2.campo2,
FROM tabella1, tabella2;
```

IL LINGUAGGIO SQL

IN () e NOT IN (): vengono usati per indicare che i record di interesse devono (o non devono) essere inclusi nell'elenco tra parentesi; *: indica tutti i campi.

```
SELECT * FROM tabella1
WHERE campo1 IN (SELECT campo2 FROM tabella2);
```

ALL () : viene usato per indicare che la condizione deve essere verificata per tutti i record inclusi nell'elenco tra parentesi.

```
SELECT * FROM tabella1
WHERE campo1 > ALL (SELECT campo2 FROM tabella2);
```

IL LINGUAGGIO SQL

COUNT(), SUM(), AVG(), MAX(), MIN(), ...: sono funzioni di raggruppamento che permettono calcolare totali, medie, massimi, minimi, … di un insieme di dati.

```
SELECT COUNT(*), SUM(campo1), AVG(campo2),
MAX(campo3) FROM tabella WHERE campo4 = 12;
```

COUNT(*) restituisce il numero di record che soddisfano la condizione della select.

L'uso degli indici (sui campi criterio del raggruppamento) velocizza l'azione delle funzioni di raggruppamento.

IL LINGUAGGIO SQL

GROUP BY (): raggruppa i dati. Viene usata assieme a una funzione di raggruppamento.

```
SELECT COUNT(*) AS Quanti, Year(data) AS Anno,
Month(data) AS Mese FROM tabella
GROUP BY Year(data), Month(data);
```

AS assegna un nome al campo risultato di una espressione.

Year(), Month(), Day() restituiscono l'anno, il mese, il giorno della data passata come argomento.

IL LINGUAGGIO SQL

HAVING (): specifica la condizione di selezione di dati raggruppati mediante GROUP BY.

```
SELECT COUNT(*) AS Quanti, Year(data) AS Anno,
Month(data) AS Mese FROM tabella
GROUP BY Year(data), Month(data)
HAVING Year(data) > 1990;
```

IL LINGUAGGIO SQL

ORDER BY: ordina i record risultato della selezione.

```
SELECT * FROM tabella
ORDER BY campo1, campo2;
```

L'ordinamento che si ottiene è crescente. Per ordinare in ordine decrescente si deve specificare DESC:

```
SELECT campo1, campo2 FROM tabella
ORDER BY campo1 DESC;
```

IL LINGUAGGIO SQL

INSERT INTO …

Inserisce nei campi specificati di una tabella i valori (VALUES) specificati.

```
INSERT INTO tabella (campo1, campo2, campo3)
VALUES (18, 344.5, 'Giovanni');
```

Possono anche venire inseriti interi record presi da un'altra tabella:

```
INSERT INTO tabella1 SELECT * FROM tabella2;
```

IL LINGUAGGIO SQL

UPDATE … SET …

Modifica i valori nei campi.

```
UPDATE tabella SET campo1 = 0;
```

Nell'istruzione UPDATE si possono inserire anche degli operatori e delle condizioni:

```
UPDATE tabella SET campo1 = campo1 + campo2 * 25
WHERE campo1 = 18;
```

IL LINGUAGGIO SQL

DELETE FROM …

Cancella i record di una tabella.

```
DELETE FROM tabella WHERE campo1 = 0;
```

Per cancellare tutti i record di una tabella:

```
DELETE FROM tabella;
```

IL LINGUAGGIO SQL

Istruzioni composte sui dati:

Le istruzioni SQL possono venire combinate e innestate tra loro:

```
INSERT INTO tabella1 SELECT * FROM tabella2;

INSERT INTO tabella1 SELECT * FROM tabella2
WHERE campo1 IN (SELECT campo2 FROM tabella3);
```

IL LINGUAGGIO SQL

Altre istruzioni SQL sui dati di più tabelle:

Tra le altre istruzioni SQL su più tabelle esistono:

- INNER JOIN,
- LEFT JOIN e RIGHT JOIN,
- UNION,
- EXCEPT,
- INTERSECT.

IL LINGUAGGIO SQL

… INNER JOIN … ON …

Utilizzato in una proposizione FROM, combina i record di due tabelle quando nei campi collegati di entrambe le tabelle vi sono valori corrispondenti.

FROM tabella1 INNER JOIN tabella2 ON tabella1.campo1 operconf tabella2.campo2;

tabella1, tabella2: le tabelle di cui si combinano i record;

campo1, campo2: i campi tra cui viene creato il join. Devono essere dello stesso tipo di dati;

operconf: un operatore di confronto (<, <=, =, <>, >, >=).

IL LINGUAGGIO SQL

… LEFT JOIN / RIGHT JOIN … ON …

Utilizzati in una proposizione FROM, combinano i record delle tabelle di origine.

```
FROM tabella1 [LEFT | RIGHT] JOIN tabella2
ON tabella1.campo1 operconf tabella2.campo2;
```

tabella1, tabella2: le tabelle di cui si combinano i record;

campo1, campo2: i campi tra cui viene creato il join. Devono essere dello stesso tipo di dati;

operconf: un operatore di confronto (<, <=, =, <>, >, >=).

IL LINGUAGGIO SQL

LEFT JOIN crea un join esterno sinistro che include tutti i record della prima tabella (di sinistra), anche se non hanno valori corrispondenti ai record della seconda tabella (di destra).

RIGHT JOIN crea un join esterno destro che include tutti i record della seconda tabella (di destra), anche se non hanno valori corrispondenti ai record della prima tabella (di sinistra).

IL LINGUAGGIO SQL

È possibile collegare numerose proposizioni ON in un'istruzione JOIN, utilizzando operatori logici con la seguenti sintassi:

```
SELECT campo1, campo2 FROM tabella1
INNER JOIN tabella2
ON tabella1.campo1 operconf tabella2.campo1 AND
ON tabella1.campo2 operconf tabella2.campo2) OR
ON tabella1.campo3 operconf tabella2.campo3)];
```

IL LINGUAGGIO SQL

È possibile nidificare delle istruzioni JOIN come segue:

```
SELECT campo1, campo2
FROM tabella1 INNER JOIN (tabella2 INNER JOIN [( ]tabella3
[INNER JOIN [( ]tabellaX [INNER JOIN ...)]
ON tabella3.campo3 operconf tabellaX.campoX)]
ON tabella2.campo2 operconf tabella3.campo3)
ON tabella1.campo1 operconf tabella2.campo2;
```

LEFT JOIN o RIGHT JOIN possono essere nidificati in un'istruzione INNER JOIN, ma non si può nidificare un INNER JOIN in un LEFT JOIN o RIGHT JOIN.

IL LINGUAGGIO SQL

Esempio:

È possibile utilizzare l'istruzione INNER JOIN con le tabelle Pazienti ed Esami di un ospedale per selezionare tutti gli esami di ciascun paziente.

Al contrario, per selezionare tutti i pazienti anche se alcuni non hanno fatto esami, oppure per selezionare tutti gli esami anche se alcuni non sono dei pazienti dell'ospedale (es. esami ambulatoriali), è possibile utilizzare rispettivamente LEFT JOIN o RIGHT JOIN per creare un join esterno tra le tabelle Pazienti ed Esami.

IL LINGUAGGIO SQL

… UNION …

Crea una query di unione che combina i risultati di due o più tabelle o query indipendenti.

[TABLE] istruzione1 UNION [ALL] [TABLE] istruzione2 […] [UNION [ALL] [TABLE] istruzioneN];

TABLE: indica che "istruzioneX" è il nome di una tabella;

istruzioneX: nome tabella o istruzione SELECT;

ALL: fa in modo che vengano restituiti tutti i record inclusi i duplicati che per default non vengono restituiti.

IL LINGUAGGIO SQL

Tutte le tabelle o query indipendenti combinate in una istruzione UNION devono avere lo stesso numero di campi che possono anche essere di tipo diverso.

I nomi dei campi saranno quelli assegnati nella prima delle istruzioni di union. Questo vale anche per l'uso di loro alias (... AS ...).

IL LINGUAGGIO SQL

Istruzioni INTERSECT e EXCEPT

Le istruzioni INTERSECT ed EXCEPT equivalgono rispettivamente agli operatori IN () e NOT IN () usati nell'istruzione WHERE.

INTERSECT ed EXCEPT non vengono riconosciute da Access che per questo scopo utilizza solo gli operatori IN () e NOT IN ().