Riassunto Basi di Dati - SQL

Politecnico di Milano

Anna Mettifogo

December 14, 2024

Contents

1 Introduzione e Query Semplici

1.1 Definizione degli Schemi

Uno schema è una raccolta di oggetti come tabelle, domini e indici. Esempio di definizione di uno schema:

```
CREATE SCHEMA GestioneAffitti;
```

1.1.1 Tabelle nello schema

CLIENTE (CodiceCliente, Cognome, Nome, DataNascita)
APPARTAMENTO (CodAppartamento, Via, Città, Civico, Locali, Metratura, Piano)

1.2 Query Semplici

Le query semplici sono interrogazioni basate su condizioni dirette e restituiscono righe da una o più tabelle.

1.2.1 Clausola SELECT

Esempio:

```
SELECT Nome, Citt
FROM Studente
WHERE Citta = 'Roma';
```

1.2.2 Alias e DISTINCT

```
SELECT Nome AS NomeStudente FROM Studente;
SELECT DISTINCT Citta FROM Studente;
```

1.2.3 Operatori di confronto

Operatori: `=`, `<>`, `>`, `<`, `BETWEEN`, `LIKE`. Esempio LIKE:

```
SELECT *
FROM Studente
WHERE Nome LIKE '_a%';
```

1.3 Query di Modifica

Le query di modifica includono comandi come **INSERT**, **DELETE** e **UPDATE**.

INSERT:

```
INSERT INTO Studente (Matr, Nome, Citta, CL)
VALUES ('456', 'Giorgio', 'Bologna', 'Inf');
```

DELETE:

```
DELETE FROM Studente WHERE Matr = '456';
```

UPDATE:

```
UPDATE Studente
SET Citta = 'Milano'
WHERE Nome = 'Giorgio';
```

2 Query Complesse

2.1 Interrogazioni con Join

2.1.1 Esempi di Join

Estrarre i guidatori con le loro macchine (**Inner Join**):

```
SELECT FirstName, Surname, Driver.DriverID, CarRegNo, Make, Model
FROM Driver JOIN Automobile
ON Driver.DriverID = Automobile.DriverID;
```

Estrarre i guidatori con o senza macchine (**Left Join**):

```
SELECT FirstName, Surname, Driver.DriverID, CarRegNo, Make, Model
FROM Driver LEFT JOIN Automobile
ON Driver.DriverID = Automobile.DriverID;
```

Estrarre tutte le automobili con o senza guidatori (**Right Join**):

```
SELECT FirstName, Surname, Driver.DriverID, CarRegNo, Make, Model
FROM Driver RIGHT JOIN Automobile
ON Driver.DriverID = Automobile.DriverID;
```

2.2 Ordinamento dei Risultati

Ordinare i risultati per una colonna specifica (**ORDER BY**):

```
SELECT *
FROM Ordine
WHERE Importo > 100000
ORDER BY Data ASC;
```

Ordinamento combinato (**ORDER BY multiplo**):

```
SELECT *
FROM Ordine
WHERE Importo > 100000
ORDER BY CodCli ASC, Data DESC;
```

2.3 Funzioni Aggregate

Calcolare il numero totale di ordini:

```
SELECT COUNT(*)
FROM Ordine;
```

Estrarre il massimo importo tra gli ordini:

```
SELECT MAX(Importo) AS MaxImp
FROM Ordine;
```

2.4 Query con Raggruppamento

Raggruppare per cliente e calcolare il totale degli importi:

```
SELECT CodCli, SUM(Importo) AS SommaImp
FROM Ordine
GROUP BY CodCli;
```

2.5 Interrogazioni Binarie

Unire due query con l'operatore **UNION**:

```
SELECT CodOrd
FROM Ordine
WHERE Importo > 500
UNION
SELECT CodOrd
FROM Dettaglio
WHERE Qta > 1000;
```

3 Query Nidificate e Viste

3.1 Tipologie di Query Nidificate

1. **Confronto di un Attributo con una Query:**

```
AttrExpr Operator < any | all > SelectSQL
```

   - **any:** Vero se almeno una riga soddisfa la condizione.
   - **all:** Vero se tutte le righe soddisfano la condizione.

   Esempio:

```
SELECT CodOrd
FROM Ordine
WHERE Importo > ANY (SELECT Importo FROM Ordine);
```

2. **Operatori `in` e `not in`:**

   - **`in`:** Vero se almeno una riga coincide.
   - **`not in`:** Vero se nessuna riga coincide.

3. **Operatori `exists` e `not exists`:**

   - **`exists`:** Vero se la query ha risultati.
   - **`not exists`:** Vero se la query non ha risultati.

   Esempio:

```
SELECT CodCli
FROM Ordine O
WHERE EXISTS (
    SELECT *
    FROM Ordine O1
    WHERE O1.CodCli = O.CodCli AND O1.Data = O.Data
);
```

3.2 Uso di Operatori con Query Nidificate

3.2.1 `any` e `all`

Esempio:

```
SELECT CodOrd
FROM Ordine
WHERE Importo >= ALL (SELECT Importo FROM Ordine);
```

3.2.2 Operatori `in` e `not in`

Esempio:

```
SELECT CodCli
FROM Cliente
WHERE CodCli IN (SELECT CodCli FROM Ordine WHERE Importo > 10000);
```

3.2.3 Query Nidificate Complesse

Esempio con costruttore di tupla:

```
SELECT *
FROM Persona P
WHERE (Nome, Cognome) NOT IN (
    SELECT Nome, Cognome
    FROM Persona P1
    WHERE P1.CodFisc <> P.CodFisc
);
```

3.3 Viste

3.3.1 Definizione e Uso delle Viste

Sintassi:

```
CREATE VIEW NomeVista [(ListaAttributi)] AS SelectSQL;
```

Esempio di Vista:

```
CREATE VIEW OrdiniPrincipali AS
SELECT *
FROM Ordine
WHERE Importo > 10000;
```

3.3.2 Composizione delle Viste

Query che utilizza una vista:

```
SELECT CodCli
FROM OrdiniPrincipali;
```

Equivalente:

```
SELECT CodCli
FROM Ordine
WHERE Importo > 10000;
```

3.3.3 Modifiche Tramite Viste

Esempio:

```
UPDATE OrdiniPrincipali
SET Importo = Importo * 1.05
WHERE CodCli = '45';
```

3.4 Controllo dell'Accesso

3.4.1 Privilegi e Autorizzazioni

Tipi di privilegi: `insert`, `update`, `delete`, `select`, `references`, `usage`.

3.4.2 Concessione e Revoca

Concessione:

```
GRANT SELECT ON Ordine TO User1;
```

Revoca:

```
REVOKE SELECT ON Ordine FROM User1;
```

4 Transazioni e Trigger

4.1 Transazioni

4.1.1 Definizione

Una transazione è un'unità elementare di esecuzione incapsulata tra i comandi:

```
BEGIN TRANSACTION;
END TRANSACTION;
```

4.1.2 Proprietà ACID

- **Atomicità:** La transazione è eseguita completamente o annullata.
- **Consistenza:** I vincoli della base di dati sono sempre rispettati.
- **Isolamento:** Nessuna interferenza tra transazioni concorrenti.
- **Persistenza:** Le modifiche di una transazione confermata non possono essere perse.

4.1.3 Esempio di Transazione

Transazione semplice:

```
BEGIN TRANSACTION;
UPDATE Account SET Balance = Balance + 10 WHERE AccNum = 12202;
UPDATE Account SET Balance = Balance - 10 WHERE AccNum = 42177;
COMMIT WORK;
END TRANSACTION;
```

4.2 Vincoli e Controllo della Qualità dei Dati

4.2.1 Definizione

I vincoli garantiscono la correttezza dei dati attraverso regole definite nello schema delle tabelle o come asserzioni separate.

4.2.2 Esempio di Vincoli

Vincoli generici in una tabella:

```
CREATE TABLE Magazzino (
  CodProd CHAR(2) PRIMARY KEY,
  QtaDisp INTEGER NOT NULL CHECK (QtaDisp > 0),
  QtaRiord INTEGER NOT NULL CHECK (QtaRiord > 10),
  CHECK (QtaDisp > QtaRiord)
);
```

4.2.3 Asserzioni

Vincoli inter-relazionali definiti esternamente alle tabelle:

```
CREATE ASSERTION SempreUnImpiegato CHECK (
  1 <= (SELECT COUNT(*) FROM Impiegato)
);
```

4.2.4 Vincoli Differiti

- **Immediate:** Verifica immediata, annulla l'ultima modifica in caso di violazione.
- **Deferred:** Verifica alla fine della transazione, annulla tutte le modifiche in caso di violazione.

4.3 Procedure

4.3.1 Definizione

Le procedure sono moduli predefiniti per la manipolazione dei dati. Possono contenere logica complessa e sono memorizzate sul server.

4.3.2 Esempio di Procedura

Procedura per gestire il prelievo da un magazzino:

```
CREATE PROCEDURE Prelievo (Prod INTEGER, Quant INTEGER) AS
BEGIN
  DECLARE Q1, Q2 INTEGER;
  SELECT QtaDisp, QtaRiord INTO Q1, Q2 FROM Magazzino WHERE CodProd
      = Prod;
  IF Q1 < Quant THEN RAISE EXCEPTION 'Quantita insufficiente';
  UPDATE Magazzino SET QtaDisp = QtaDisp - Quant WHERE CodProd =
     Prod;
  IF Q1 - Quant < Q2 THEN
    INSERT INTO Riordino VALUES (Prod, SYSDATE, Q2);
  END IF;
END;
```

4.4 Trigger

4.4.1 Definizione

I trigger sono regole che scattano automaticamente in risposta a eventi specifici (INSERT, UPDATE, DELETE).

4.4.2 Paradigma Evento-Condizione-Azione (ECA)

- **Evento:** Modifica che attiva il trigger.
- **Condizione:** Predicato SQL che determina se eseguire il trigger.

- **Azione:** Operazioni da eseguire.

4.4.3 Esempio di Trigger

Gestione automatica del riordino:

```
CREATE TRIGGER GestioneRiordino
AFTER UPDATE OF QtaDisp ON Magazzino
FOR EACH ROW
WHEN (NEW.QtaDisp < NEW.QtaRiord)
INSERT INTO Riordino VALUES (NEW.CodProd, SYSDATE, NEW.QtaRiord);
```

4.4.4 Tipologie di Trigger

1. **FOR EACH ROW:** Scatta per ogni riga modificata.
   - Variabili: `NEW`, `OLD`.
2. **FOR EACH STATEMENT:** Scatta una sola volta per ogni comando.
   - Variabili: `NEW_TABLE`, `OLD_TABLE`.