BASI DI DATI

SQL

Polese G. Caruccio L. Breve B.

a.a. 2023/2024

Definizione di Schemi in SQL

- CREATE SCHEMA ***nome_schema***
  AUTHORIZATION ***nome_utente***
  - Crea uno schema ***nome_schema***, il cui proprietario è l'utente con account ***nome_utente***

- DROP SCHEMA ***nome_schema*** *drop-behaviour*
  - Elimina lo schema. L'opzione ***drop-behaviour*** può assumere i valori **CASCADE** o **RESTRICT**:
    - ✓ DROP SCHEMA **COMPANY** CASCADE
      - ❖ Lo schema del db COMPANY viene rimosso, con tutte le tabelle, domini ed altri elementi
    - ✓ DROP SCHEMA **COMPANY** RESTRICT
      - ❖ Lo schema viene eliminato solo se non contiene elementi

CREATE TABLE, esempio

```
CREATE TABLE Impiegato(
        Matricola CHAR(6) PRIMARY KEY,
        Nome CHAR(20) NOT NULL,
        Cognome CHAR(20) NOT NULL,
        Dipart CHAR(15),
        Stipendio NUMERIC(9) DEFAULT 0,
        FOREIGN KEY(Dipart) REFERENCES Dipartimento(NomeDip),
        UNIQUE(Cognome, Nome)
 )
```

Domini

- Domini elementari (predefiniti)
- Domini definiti dall'utente (semplici, ma riutilizzabili)

Elementari

- Carattere: singoli caratteri o stringhe, anche di lunghezza variabile
- Bit: singoli booleani o stringhe
- Numerici, esatti e approssimati
- Data, ora, intervalli di tempo
- Introdotti in SQL:1999:
  - Boolean
  - BLOB, CLOB (binary/character large object): per grandi immagini e testi

Domini in SQL2: Numeri e Stringhe

- Numerici
  - Interi (INTEGER o INT, SMALLINT)
  - Reali (FLOAT, REAL, DOUBLE PRECISION)
  - Numeri formattati (DECIMAL(i,j), DEC(i,j), NUMERIC(i,j))
    - ✓ i, detta precisione, indica il numero di cifre decimali
    - ✓ j, detta scala, indica il numero di cifre dopo la virgola
- Stringhe di caratteri
  - A lunghezza fissa (CHAR(n), CHARACTER(n))
  - A lunghezza variabile (VARCHAR(n) o CHAR VARYING(n))
    - ✓ Per default n, il numero massimo di caratteri, è 1
- Stringhe di bit
  - A lunghezza fissa (BIT(n))
  - A lunghezza variabile (BIT VARYING(n))

Domini in SQL2: Date e Orari

- DATE
  - Ha dieci posizioni, con componenti YEAR, MONTH e DAY. Formato YYYY-MM-DD
- TIME
  - Ha (almeno) otto posizioni con componenti HOUR, MINUTE e SECOND. Formato HH:MM:SS
- TIME(i)
  - i = precisione delle frazioni di secondo. Specifica i+1 posizioni aggiuntive per TIME, una per il separatore ed i per le frazioni di secondo
- TIME WITH TIME ZONE
  - Usa ulteriori 6 posizioni per lo spiazzamento dal GMT, con un range da +13:00 a –12:59

Definizione di nuovi domini

- Istruzione CREATE DOMAIN:
  - definisce un dominio (semplice), utilizzabile in definizioni di relazioni, anche con vincoli e valori di default

Esempio:

```
CREATE DOMAIN Voto
    AS SMALLINT DEFAULT NULL
    CHECK (value >= 18 AND value <= 30)
```

Vincoli intrarelazionali

- NOT NULL
- UNIQUE definisce chiavi
- PRIMARY KEY: chiave primaria (una sola, implica NOT NULL)
- CHECK, permette di esprimere vincoli generici (vedremo più avanti)

UNIQUE e PRIMARY KEY

- Due forme:
  - Nella definizione di un attributo, se forma da solo la chiave
  - Come elemento separato

Chiavi su più attributi, attenzione

```
Nome            CHAR(20) NOT NULL,
Cognome         CHAR(20) NOT NULL,
UNIQUE (Cognome, Nome),
```

```
Nome            CHAR(20) NOT NULL UNIQUE,
Cognome         CHAR(20) NOT NULL UNIQUE,
```

- **Non è la stessa cosa!**

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Vigili

Matricola	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto

Prov	Numero	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

CREATE TABLE, esempio

```
CREATE TABLE Infrazioni(
   Codice CHAR(6) PRIMARY KEY,
   Data DATE NOT NULL,
   Vigile INTEGER REFERENCES Vigili(Matricola) ON UPDATE CASCADE ON
                        DELETE SET NULL,
   Provincia CHAR(2),
   Numero CHAR(6) ,
   FOREIGN KEY(Provincia, Numero) REFERENCES Auto(Provincia, Numero)
         ON UPDATE CASCADE ON DELETE CASCADE
 )
```

Modifiche degli schemi

ALTER DOMAIN

ALTER TABLE

DROP DOMAIN

DROP TABLE

...

Alter Table: aggiunta di un attributo

- Vogliamo aggiungere l'email di un impiegato nella tabella Impiegato:
  - ALTER TABLE IMPIEGATO
    - ✓ADD EMAIL VARCHAR(12);
- Il valore di EMAIL o si specifica di default o sarà null.
- Con la ALTER TABLE non è permessa la clausola NOT NULL

ALTER TABLE: Eliminazione attributo

- Quando si elimina una colonna occorre scegliere l'opzione CASCADE o RESTRICT
  - Con CASCADE vincoli e viste che referenziano la colonna sono eliminati dallo schema
  - Con RESTRICT il comando ha successo solo se nessun vincolo o vista referenzia la colonna.
- Esempio: rimuovere la colonna INDIRIZZO dalla tabella IMPIEGATO
  - ALTER TABLE IMPIEGATO DROP INDIRIZZO CASCADE;

ALTER TABLE: Modifica vincoli

- Modifica di una colonna eliminando una clausola di default o definendone una nuova
  Esempi:
  - ALTER TABLE IMPIEGATO ALTER DIPART DROP DEFAULT;
  - ALTER TABLE INFRAZIONE ALTER VIGILE SET DEFAULT “3334”;
- E’ possibile eliminare un vincolo solo se gli si è dato un nome nella CREATE TABLE tramite la keyword CONSTRAINT

SQL, operazioni sui dati

- Interrogazione:
  - SELECT
- Modifica:
  - INSERT, DELETE, UPDATE

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Selezione e proiezione

- Nome e reddito delle persone con meno di trenta anni

$$\mathbf{PROJ}_{\mathbf{Nome, Reddito}}(\mathbf{SEL}_{\mathbf{Eta < 30}}(\mathbf{Persone}))$$

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

Condizione “LIKE”

- Le persone che hanno un nome che inizia per 'A' e ha una 'd' come terza lettera

```
SELECT *
FROM Persone
WHERE Nome LIKE ‘A_d%’
```

Selezione, proiezione e join

- I padri di persone che guadagnano più di venti milioni

$$\mathbf{PROJ}_{\mathbf{Padre}}(\mathbf{Paternita}\ \mathbf{JOIN}_{\mathbf{Figlio = Nome}}\ \mathbf{SEL}_{\mathbf{Reddito > 20}}(\mathbf{Persone}))$$

```
SELECT DISTINCT Padre
FROM Persone, Paternita
WHERE Figlio = Nome AND Reddito > 20
```

Join naturale

- Padre e madre di ogni persona

Paternita JOIN Maternita

```
SELECT Paternita.Figlio, Padre, Madre
FROM Maternita, Paternita
WHERE Paternita.Figlio = Maternita.Figlio
```

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

$$\mathbf{PROJ}_{\mathbf{Nome, Reddito, RP}}\,(\mathbf{SEL}_{\mathbf{Reddito > RP}}\,(\mathbf{REN}_{\mathbf{NP, EP, RP \leftarrow Nome, Eta, Reddito}}(\mathbf{Persone})\ \mathbf{JOIN}_{\mathbf{NP = Padre}}\,(\mathbf{Paternita\ JOIN}_{\mathbf{Figlio = Nome}}\ \mathbf{Persone})))$$

```
SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p, Paternita, Persone f
WHERE p.Nome = Padre AND Figlio = f.Nome AND f.Reddito > p.Reddito
```

SELECT, con ridenominazione del risultato

```
SELECT Figlio, f.Reddito AS Reddito, p.Reddito AS RedditoPadre
FROM Persone p, Paternita, Persone f
WHERE p.Nome = Padre
                    AND Figlio = f.Nome AND f.Reddito > p.Reddito
```

Join esplicito

- Padre e madre di ogni persona

```
SELECT Paternita.Figlio, Padre, Madre
FROM Maternita, Paternita
WHERE Paternita.Figlio = Maternita.Figlio
```

```
SELECT Madre, Paternita.Figlio, Padre
FROM Maternita JOIN Paternita ON Paternita.Figlio = Maternita.Figlio
```

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p, Paternita, Persone f
WHERE p.Nome = Padre AND
        Figlio = f.Nome AND
        f.Reddito > p.Reddito

SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p JOIN Paternita ON p.Nome = Padre
        JOIN Persone f ON Figlio = f.Nome WHERE f.Reddito > p.Reddito
```

Ulteriore estensione: join naturale (meno diffuso)

$$\mathbf{PROJ}_{\mathbf{Figlio, Padre, Madre}}(\mathbf{Paternita}\ \mathbf{JOIN}_{\mathbf{Figlio = Nome}}\ \mathbf{REN}_{\mathbf{Nome} \leftarrow \mathbf{Figlio}}(\mathbf{Maternita}))$$

Paternita JOIN Maternita

```
SELECT Madre, Paternita.Figlio, Padre
FROM Maternita JOIN Paternita ON Paternita.Figlio = Maternita.Figlio
```

```
SELECT Madre, Paternita.Figlio, Padre
FROM Maternita NATURAL JOIN Paternita
```

Join esterno: "outer join"

- Padre e, se nota, madre di ogni persona

```
SELECT Paternita.Figlio, Padre, Madre
FROM Paternita LEFT JOIN Maternita ON Paternita.Figlio = Maternita.Figlio
```

```
SELECT Paternita.Figlio, Padre, Madre
FROM Paternita LEFT OUTER JOIN Maternita
                    ON Paternita.Figlio = Maternita.Figlio
```

- outer è opzionale

Outer join

```
SELECT Paternita.Figlio, Padre, Madre
FROM Maternita JOIN Paternita ON Maternita.Figlio = Paternita. Figlio

SELECT Paternita.figlio, Padre, Madre
FROM Maternita LEFT OUTER JOIN Paternita
                    ON Maternita.Figlio = Paternita.Figlio

SELECT Paternita.Figlio, Padre, Madre
FROM Maternita FULL OUTER JOIN Paternita
                    ON Maternita.Figlio = Paternita.Figlio
```

Ordinamento del risultato

- Nome e reddito delle persone con meno di trenta anni in ordine alfabetico inverso

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
ORDER BY Nome DESC
```

- La clausola di default è ASC

Operatori aggregati: COUNT

- Il numero di figli di Franco

```
SELECT COUNT(*) AS NumFigliDiFranco
FROM Paternita
WHERE Padre = 'Franco’
```

- L’operatore aggregato (COUNT) viene applicato al risultato dell’interrogazione:

```
SELECT *
FROM Paternita
WHERE Padre = 'Franco'
```

Altri operatori aggregati

- SUM, AVG, MAX, MIN

- Media dei redditi dei figli di Franco

```
SELECT AVG(Reddito)
FROM Persone JOIN Paternita ON Nome = Figlio
WHERE Padre = ‘Franco’
```

Operatori aggregati e target list

- Un’interrogazione scorretta:

```
SELECT Nome, MAX(Reddito)
FROM Persone
```

- Di chi sarebbe il nome? La target list deve essere omogenea

```
SELECT MIN(Eta), AV(Reddito)
FROM Persone
```

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

```
SELECT Padre, COUNT(*)  AS NumFigli
FROM Paternita
GROUP BY Padre
```

Paternita

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Padre	NumFigli
Sergio	1
Luigi	2
Franco	2

Raggruppamenti e target list

- Scorretta

```
SELECT Padre, AVG(f.Reddito), p.Reddito
FROM Persone f JOIN Paternita ON Figlio = Nome
                                    JOIN Persone p ON Padre = p.Nome
GROUP BY Padre
```

- Corretta

```
SELECT Padre, AVG(f.Reddito), p.Reddito
FROM Persone f JOIN Paternita ON Figlio = Nome
                                    JOIN Persone p ON Padre = p.nome
GROUP BY Padre, p.Reddito
```

Condizioni sui gruppi

- I padri i cui figli hanno un reddito medio maggiore di 25

```
SELECT Padre, AVG(f.Reddito)
FROM Persone f JOIN Paternita ON Figlio = Nome
GROUP BY Padre
HAVING AVG(f.Reddito) > 25
```

WHERE o HAVING?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 25

```
SELECT Padre, AVG(f.Reddito)
FROM Persone f JOIN Paternita ON Figlio = Nome
WHERE Eta < 30
GROUP BY Padre
HAVING AVG(f.Reddito) > 25
```

Sintassi, riassumiamo

SelectSQL ::=

`select` ListaAttributiOEspressioni
`from` ListaTabelle
[`where` CondizioniSemplici]
[`group by` ListaAttributiDiRaggruppamento
[`having` CondizioniAggregate]]
[`order by` ListaAttributiDiOrdinamento]

Notazione posizionale, 2

```
SELECT Padre, Figlio
FROM Paternita
UNION
SELECT Figlio, Madre
FROM Maternita
```

```
SELECT Padre, Figlio
FROM Paternita
UNION
SELCT Madre, Figlio
FROM Maternita
```

Intersezione

```
SELECT Nome
FROM Impiegato
INTERSECT
SELECT Cognome AS Nome
FROM Impiegato
```

- Equivale a

```
SELECT I.Nome
FROM Impiegato I, Impiegato J
WHERE I.Nome = J.Cognome
```

Interrogazioni nidificate

- Le condizioni atomiche permettono anche
  - Il confronto fra un attributo (o più, vedremo poi) e il risultato di una sottointerrogazione
  - Quantificazioni esistenziali
- Nome e reddito del padre di Franco

```
SELECT Nome, Reddito
FROM Persone, Paternita
WHERE Nome = Padre
      AND Figlio = 'Franco'
```

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome = (SELECT Padre
              FROM Paternita
              WHERE Figlio = 'Franco')
```

- Nome e reddito dei padri di persone che guadagnano più di 20 milioni

```
SELECT DISTINCT P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre AND Figlio = F.Nome
                                    AND F.Reddito > 20
```

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome IN (SELECT Padre
                      FROM Paternita
                      WHERE Figlio = ANY (SELECT Nome
                                          FROM Persone
                                          WHERE Reddito > 20))
```

- Nome e reddito dei padri di persone che guadagnano più di 20 milioni

```
SELECT DISTINCT P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre AND Figlio = F.Nome
                     AND F.Reddito > 20
```

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome IN (SELECT Padre
               FROM Paternita, Persone
               WHERE Figlio = Nome
                     AND Reddito > 20)
```

- Nome e reddito dei padri di persone che guadagnano più di 20 milioni, **con indicazione del reddito del figlio**

```
SELECT DISTINCT P.Nome, P.Reddito, F.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre AND Figlio = F.Nome
      AND F.Reddito > 20

SELECT Nome, Reddito, ????
FROM Persone
WHERE Nome IN (SELECT Padre
               FROM Paternita
               WHERE Figlio = ANY (SELECT Nome
                                   FROM Persone
                                   WHERE Reddito > 20))
```

Interrogazioni nidificate, commenti, 3

- Regole di visibilità:
  - Non è possibile fare riferimenti a variabili definite in blocchi più interni
  - Se un nome di variabile è omesso, si assume riferimento alla variabile più “vicina”
- In un blocco si può fare riferimento a variabili definite in blocchi più esterni; la semantica base (prodotto cartesiano, selezione, proiezione) non funziona più, vedremo presto

Quantificazione esistenziale

- Ulteriore tipo di condizione
  - EXISTS (Sottoespressione)
- Le persone che hanno almeno un figlio

```
SELECT *
FROM Persone
WHERE EXISTS  (       SELECT *
                      FROM Paternita
                      WHERE Padre = Nome) OR
            EXISTS (  SELECT *
                      FROM Maternita
                      WHERE Madre = Nome)
```

- I padri i cui figli guadagnano tutti più di venti milioni

```
SELECT DISTINCT Padre
FROM Paternita Z
WHERE NOT EXISTS (
                        SELECT *
                        FROM Paternita W, Persone
                        WHERE W.Padre = Z.Padre
                              AND W.Figlio = Nome
                              AND Reddito <= 20)
```

Visibilità

- Scorretta:

```
SELECT *
FROM Impiegato
WHERE Dipart IN (SELECT Nome
                 FROM Dipartimento D1
                 WHERE Nome = 'Produzione') OR
      Dipart IN (SELECT Nome
                 FROM Dipartimento D2
                 WHERE D2.Citta = D1.Citta)
```

Disgiunzione e unione (ma non sempre)

```
SELECT * FROM Persone WHERE Reddito > 30
        UNION
SELECT F.*
FROM Persone F, Paternita, Persone P
WHERE F.Nome = Figlio AND Padre = P.Nome
        AND P.Reddito > 30

SELECT *
FROM Persone F
WHERE Reddito > 30 OR
        EXISTS (SELECT *
                FROM Paternita, Persone P
                WHERE F.Nome = Figlio AND Padre = P.Nome
                        AND P.Reddito > 30)
```

Differenza e nidificazione

```
SELECT Nome FROM Impiegato
EXCEPT
SELECT Cognome AS Nome FROM Impiegato

SELECT Nome
FROM Impiegato I
WHERE NOT EXISTS (SELECT *
                  FROM Impiegato
                  WHERE Cognome = I.Nome)
```

Massimo e nidificazione

- La persona (o le persone) con il reddito massimo

```
SELECT *
FROM Persone
WHERE Reddito = (SELECT MAX(Reddito) FROM Persone)
```

Operazioni di aggiornamento

- Operazioni di
  - Inserimento: **insert**
  - Eliminazione: **delete**
  - Modifica: **update**
- Di una o più ennuple di una relazione
- Sulla base di una condizione che può coinvolgere anche altre relazioni

Inserimento, esempi

```
INSERT INTO Persone VALUES (‘Mario’, 25, 52)

INSERT INTO Persone(Nome, Eta, Reddito) VALUES(‘Pino’, 25, 52)

INSERT INTO Persone(Nome, Reddito) VALUES(‘Lino’,55)

INSERT INTO Persone ( Nome )
        SELECT Padre
        FROM Paternita
        WHERE Padre NOT IN (SELECT Nome FROM Persone)
```

Eliminazione di ennuple

```
DELETE FROM Tabella
    [ WHERE Condizione ]
```

ESEMPI

```
DELETE FROM Persone
    WHERE Eta < 35

DELETE FROM Paternita
    WHERE Figlio NOT IN (SELECT Nome FROM Persone)

DELETE FROM Paternita
```

Modifica di ennuple

```
UPDATE NomeTabella
SET  Attributo = < Espressione |
                   SELECT … |
                   NULL |
                   DEFAULT >
        [ WHERE Condizione ]
```

Aggiornamento, esempi

```
UPDATE Persone SET Reddito = 45 WHERE Nome = ‘Piero’

UPDATE Persone SET Reddito = Reddito * 1.1
                   WHERE Eta < 30
```

Vincoli di integrità generici: check

- Specifica di vincoli di ennupla (e anche vincoli più complessi)

  CHECK（Condizione）

ESEMPIO

```
CREATE TABLE Impiegato(
        Matricola CHARACTER(6),
        Cognome CHARACTER(20),
        Nome CHARACTER(20),
        Sesso CHARACTER NOT NULL CHECK (sesso IN (‘M’,‘F’)),
        Stipendio INTEGER,
        Superiore CHARACTER(6),
         CHECK (Stipendio <= (SELECT Stipendio
                              FROM Impiegato J
                              WHERE Superiore = J.Matricola))
)
```

Vincoli di integrità generici: asserzioni

- Specifica vincoli a livello di schema

```
CREATE ASSERTION NomeAss CHECK (Condizione)

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

Viste

```
CREATE VIEW NomeVista [ ( ListaAttributi ) ] AS SelectSQL
[ WITH [ LOCAL | CASCADED] CHECK OPTION]

CREATE VIEW ImpiegatiAmmin(Matricola, Nome, Cognome, Stipendio) AS
                SELECT Matricola, Nome, Cognome, Stipendio
                FROM Impiegato
                WHERE Dipart = ‘Amministrazione’ AND Stipendio > 10
```

Viste, esempio

```
CREATE VIEW ImpiegatiAmminPoveri AS
    SELECT *
    FROM ImpiegatiAmmin
    WHERE Stipendio < 50
    WITH CHECK OPTION
```

- CHECK OPTION permette modifiche, ma solo a condizione che la ennupla continui ad appartenere alla vista (non posso modificare lo stipendio portandolo a 60)

Un’interrogazione non standard

- La nidificazione nella HAVING non è ammessa

```
SELECT Dipart
FROM Impiegato
GROUP BY Dipart
HAVING SUM(Stipendio) >= ALL(SELECT SUM(Stipendio)
                             FROM Impiegato
                             GROUP BY Dipart)
```

Soluzione con le viste

```
CREATE VIEW BudgetStipendi(Dip, TotaleStipendi) AS
   SELECT Dipart, SUM(Stipendio)
   FROM Impiegato
   GROUP BY Dipart

SELECT Dip
FROM BudgetStipendi
WHERE TotaleStipendi = (SELECT MAX(TotaleStipendi)
                        FROM BudgetStipendi)
```

Ancora sulle viste

- Interrogazione scorretta

```
SELECT AVG(COUNT(DISTINCT Ufficio))
FROM Impiegato
GROUP BY Dipart
```

- Con una vista

```
CREATE VIEW DipartUffici(NomeDip, NroUffici) AS
   SELECT Dipart, COUNT(DISTINCT Ufficio)
   FROM Impiegato
   GROUP BY Dipart;
SELECT AVG(NroUffici)
FROM DipartUffici
```

Una transazione in SQL

```
BEGIN TRANSACTION;
UPDATE ContoCorrente
  SET Saldo = Saldo – 10
  WHERE NumeroConto = 12345 ;
UPDATE ContoCorrente
  SET Saldo = Saldo + 10
  WHERE NumeroConto = 55555 ;
COMMIT WORK;
```