Linguaggio SQL

- DDL/DML: ulteriori costrutti

Database di esempio

S

SNum	SName	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

PNum	PName	Color	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SP

SNum	PNum	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Funzioni e operatori su stringhe

- Concatenazione: ||
- Esempio:

```
SELECT Nome || ' ' || Cognome
FROM Clienti;
```

- Minuscole: lower, Maiuscole: upper
- Esempio:

```
SELECT *
FROM Clienti
WHERE lower(Nome) like '%p%';
```

Manipolazione delle date

- C'è una grande differenza nel formato delle date tra i vari DBMS nelle varie configurazioni
- Controllare la documentazione dello specifico DBMS
- Esempio PostgreSQL:

Example	Description
1999-01-08	ISO 8601; January 8 in any mode (recommended format)
January 8, 1999	unambiguous in any `datestyle` input mode
1/8/1999	January 8 in `MDY` mode; August 1 in `DMY` mode
1/18/1999	January 18 in `MDY` mode; rejected in other modes
01/02/03	January 2, 2003 in `MDY` mode; February 1, 2003 in `DMY` mode; February 3, 2001 in `YMD` mode
1999-Jan-08	January 8 in any mode
Jan-08-1999	January 8 in any mode
08-Jan-1999	January 8 in any mode
99-Jan-08	January 8 in `YMD` mode, else error
08-Jan-99	January 8, except error in `YMD` mode
Jan-08-99	January 8, except error in `YMD` mode
19990108	ISO 8601; January 8, 1999 in any mode
990108	ISO 8601; January 8, 1999 in any mode
1999.008	year and day of year
J2451187	Julian date
January 8, 99 BC	year 99 BC

Manipolazione delle date

- Inserimento:

insert into MiaTabella values ('1999-01-08');

insert into MiaTabella values ('08-01-1999');

insert into MiaTabella values ('08-Jan-1999');

- Attenzione a possibili ambiguità tra giorno, mese (e anno)

- Esempi:

*select * from MiaTabella where extract(year from DataNascita)=1999*

*select * from MiaTabella where extract(hour from OrarioUscita)>18*

Chiavi surrogate

- Spesso si usano valori interi autoincrementanti
- Il caso più frequente è quello di chiave primarie surrogate, cioè identificatori unici non derivati dai dati applicativi
- In SQL si crea una "sequenza", che viene usata per generare i valori
- Esempio:

```
CREATE SEQUENCE MiaSequenza
INCREMENT BY 1 START 100;


CREATE TABLE MiaTabella (
    Identificatore integer DEFAULT nextval('MiaSequenza')
    PRIMARY KEY,
    Attributo1 varchar(20) );
```

Chiavi surrogate

- Una notazione più sintetica di PostgreSQL usa il tipo SERIAL
- Esempio:

```
CREATE TABLE MiaTabella (
  Identificatore SERIAL PRIMARY KEY,
  Attributo1 varchar(20) );
```

- Quando si inserisce una riga non si specifica il valore della chiave surrogata:

```
insert into MiaTabella(Attributo1) values ('Val1');
insert into MiaTabella(Attributo1) values ('Val2');
```

- Possono comparire buchi nella numerazione ad esempio dopo un rollback

Chiavi surrogate

- In PostgreSQL SERIAL è un alias per
  integer NOT NULL DEFAULT nextval('sequenza')
  su una sequenza creata con
  CREATE SEQUENCE tablename_colname_seq;
- In Oracle si deve usare il costrutto SEQUENCE
- In MySQL si può usare SERIAL (che è un alias per BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE)
- È utile conoscere gli alias se si usano i valori degli attributi autoincrementati nelle chiavi esterne

Window functions

- Le window functions sono simili alle funzioni aggregate: compiono calcoli su insiemi di righe
- Mentre le funzioni aggregate fanno sì che ogni gruppo di righe diventi una sola riga di output, le window functions mantengono le righe separate
- Esempio, per ogni fornitura riportiamo la quantità totale fornita per ogni fornitore:

```
SELECT *, sum(Qty) OVER (PARTITION BY SNum)
FROM SP;
```

Costrutto CASE

- Il costrutto CASE permette di esprimere espressioni condizionali simili al costrutto switch del C e di Java
- Esempio, mostriamo lo status come giudizio

```
SELECT SNum, SName, CASE WHEN Status<=10 THEN 'Basso'
                         WHEN Status>10 AND Status <30 THEN 'Medio'
                         ELSE 'Alto'
                         END
FROM S;
```

Indici

- Gli indici permettono al DBMS di recuperare le righe in modo più veloce
- Consistono nella creazione e manutenzione di specifiche strutture dati (B-alberi, Hash, ...)
- L'incremento delle prestazioni in lettura si paga con l'overhead della gestione della struttura dati, che viene mantenuta automaticamente aggiornata
- Esempio:

  CREATE INDEX MioIndice ON S(City);
- Per approfondire:

  https://www.postgresql.org/docs/10/static/indexes.html

EXPLAIN

- È possibile visualizzare il piano che il DBMS userà per l'esecuzione di una query
- Esempio:

  *EXPLAIN SELECT * FROM S;*
- Viene mostrato l'albero di esecuzione con le varie strategie di accesso (scan sequenziale, index scan, bitmap index scan,...), il costo stimato di inizio esecuzione e di esecuzione totale, il numero di tuple nel risultato e la loro dimensione in byte:

Seq Scan on s (cost=0.00..1.05 rows=5 width=144)

EXPLAIN

- Per avere risultati più precisi si può usare EXPLAIN ANALYZE, che esegue effettivamente la query
- Esempio:

  *EXPLAIN ANALYZE SELECT * FROM S;*
- Vengono mostrate costi e tempi effettivi, e non solo stimati:

Seq Scan on s (cost=0.00..1.05 rows=5 width=144) (actual time=0.084..0.085 rows=5 loops=1), Planning time: 6.764 ms, Execution time: 0.146 ms

- Per approfondire:

  https://www.postgresql.org/docs/10/static/using-explain.html