

Structured Query Language SQL

SQL

- originariamente "Structured Query Language", ora "nome proprio"
- linguaggio con varie funzionalità:
 - contiene sia il DDL sia il DML
- ne esistono varie versioni

SQL: "storia"

- prima proposta **SEQUEL** (1974);
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto"
- standard (1986, poi 1989 e infine 1992, 1999)

Definizione dei dati in SQL

- Istruzione CREATE TABLE:
 - definisce uno schema di relazione
 - specifica attributi, domini e vincoli

```
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)

Domini 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

Definizione di domini

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

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

UNIQUE e PRIMARY KEY

- due forme:
 - nella definizione di un attributo, se forma da solo la chiave
 - come elemento separato

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)  
)
```

PRIMARY KEY, alternative

Matricola CHAR(6) PRIMARY KEY

Matricola CHAR(6),
...,
PRIMARY KEY (Matricola)

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)  
)
```

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!

Vincoli interrelazionali

- CHECK
- REFERENCES e FOREIGN KEY permettono di definire vincoli di integrità referenziale
- di nuovo due sintassi
 - per singoli attributi
 - su più attributi
- E' possibile definire politiche di reazione alla violazione

Infrazioni

<u>Codice</u>	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

<u>Matricola</u>	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Infrazioni

<u>Codice</u>	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

<u>Prov</u>	<u>Numero</u>	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

CREATE TABLE, esempio

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

Modifiche degli schemi

```
ALTER DOMAIN  
ALTER TABLE  
DROP DOMAIN  
DROP TABLE  
...
```

Definizione degli indici

- è rilevante dal punto di vista delle prestazioni
- a livello fisico e non logico
- in passato era importante perché in alcuni sistemi era l'unico mezzo per definire chiavi
- CREATE INDEX

DDL, in pratica

- In molti sistemi si utilizzano strumenti diversi dal codice SQL per definire lo schema della base di dati

SQL, operazioni sui dati

- interrogazione:
 - SELECT
- modifica:
 - INSERT, DELETE, UPDATE

Istruzione SELECT (versione base)

```
SELECT ListaAttributi  
FROM ListaTabelle  
[ WHERE Condizione ]
```

- "target list"
- clausola FROM
- clausola WHERE

Selezione e proiezione

- Nome e reddito delle persone con meno di trenta anni

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

```
select nome, reddito  
from persone  
where eta < 30
```

SELECT, abbreviazioni

```
select nome, reddito  
from persone  
where eta < 30  
  
select p.nome as nome, p.reddito as reddito  
from persone p  
where p.eta < 30
```

Selezione, senza proiezione

- Nome, età e reddito delle persone con meno di trenta anni

$SEL_{Eta < 30}(Persone)$

```
select *  
from persone  
where eta < 30
```

Proiezione, senza selezione

- Nome e reddito di tutte le persone

$PROJ_{Nome, Reddito}(Persone)$

```
select nome, reddito  
from persone
```

Condizione complessa

```
select *  
from persone  
where reddito > 25  
and (eta < 30 or eta > 60)
```

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%'
```

Gestione dei valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

SEL `Età > 40 OR Età IS NULL` (Impiegati)

```
select *  
from impiegati  
where eta > 40 or eta is null
```

Selezione, proiezione e join

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:
 - selezioni, proiezioni, ridenominazioni
- con più relazioni nella FROM si realizzano join (e prodotti cartesiani)

SQL e algebra relazionale

- $R1(A1, A2) \text{ } R2(A3, A4)$

```
select R1.A1, R2.A4  
from   R1, R2  
where  R1.A2 = R2.A3
```

- prodotto cartesiano (FROM)
- selezione (WHERE)
- proiezione (SELECT)

SQL e algebra relazionale

$R1(A1, A2) \text{ } R2(A3, A4)$

```
select R1.A1, R2.A4  
from   R1, R2  
where  R1.A2 = R2.A3
```

$\text{PROJ}_{A1, A4} (\text{SEL}_{A2=A3} (R1 \text{ JOIN } R2))$

- possono essere necessarie ridenominazioni
 - nel prodotto cartesiano
 - nella target list

```
select X.A1 AS B1, ...  
from   R1 X, R2 Y, R1 Z  
where  X.A2 = Y.A3 AND ...
```

```
select X.A1 AS B1, Y.A4 AS B2  
from   R1 X, R2 Y, R1 Z  
where  X.A2 = Y.A3 AND Y.A4 = Z.A1
```

```
      RENB1,B2←A1,A4 (  
    PROJA1,A4 (SELA2 = A3 AND A4 = C1 (  
R1 JOIN R2 JOIN RENC1,C2 ← A1,A2 (R1))))
```

Proiezione, attenzione

- cognome e filiale di tutti gli impiegati

Matricola	Cognome	Filiale	Stipendio
7309	Neri	Napoli	55
5998	Neri	Milano	64
9553	Rossi	Roma	44
5698	Rossi	Roma	64

PROJ Cognome, Filiale (Impiegati)

select
cognome, filiale
from impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma
Rossi	Roma

select distinct
cognome, filiale
from impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

Maternità	Madre	Figlio	Persone		
			Nome	Età	Reddito
	Luisa	Maria	Andrea	27	21
	Luisa	Luigi	Aldo	25	15
	Anna	Olga	Maria	55	42
	Anna	Filippo	Anna	50	35
	Maria	Andrea	Filippo	26	30
	Maria	Aldo	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			

Selezione, proiezione e join

- I padri di persone che guadagnano più di venti milioni

```

PROJPadre(paternita
  JOINFiglio = Nome
  SELReddito > 20(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

```
PROJNome, Reddito, RP (SELReddito>RP
(RENNP,EP,RP ← Nome,Eta,Reddito (persone)
JOINNP=Padre
(paternita JOINFiglio =Nome 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

```
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 .reddito > p.reddito
```

Join esplicito

```
SELECT ...  
FROM Tabella { ... JOIN Tabella ON CondDiJoin }, ...  
[ WHERE AltraCondizione ]
```

- 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

$$\text{paternita} \underset{\text{Figlio} = \text{Nome}}{\text{JOIN}} \underset{\text{Nome} = \text{Figlio}}{\text{REN}} \text{maternita}$$

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 e' 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

```
select nome, reddito  
from persone  
where eta < 30  
order by nome
```

```
select nome, reddito  
from persone  
where eta < 30
```

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

```
select nome, reddito  
from persone  
where eta < 30  
order by nome
```

Persone

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30

Operatori aggregati

- Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple:

- conteggio, minimo, massimo, media, totale
- sintassi base (semplificata):

Funzione ([DISTINCT] *)
Funzione ([DISTINCT] Attributo)

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'
```

Paternità

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

NumFigliDiFranco

2

COUNT e valori nulli

```
select count(*) from persone
```

```
select count(reddito) from persone
```

```
select count(distinct reddito) from persone
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

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 valori nulli

```
select avg(reddito) as redditomedio
from persone
```

Persone	Nome	Età	Reddito
	Andrea	27	30
	Aldo	25	NULL
	Maria	55	36
	Anna	50	36

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), avg(reddito)
from persone
```

Operatori aggregati e raggruppamenti

- Le funzioni possono essere applicate a partizioni delle relazioni
- Clausola GROUP BY:

GROUP BY listaAttributi

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

Semantica di interrogazioni

- interrogazione senza `group by` e senza operatori aggregati
 - `select *`
`from paternita`
- si raggruppa e si applica l'operatore aggregato a ciascun gruppo

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 20

```
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 ]
```