Corso di Laurea in Ingegneria Informatica

Fondamenti di Informatica II

Modulo "*Basi di dati*"

a.a. 2014-2015

Docente: Gigliola Vaglini

Docente laboratorio: Francesco Pistolesi

Lezione 3

Structured Query Language (SQL): DD

SQL: "storia"

- prima proposta SEQUEL (1974);
- originariamente "Structured Query Language", ora "nome proprio„; prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto" poi standard ANSI-ISO(1986, ancora 1989 e infine 1992, SQL-2)
- SQL:1999 o SQL-3 non è ancora completamente adottato

Un linguaggio completo

- SQL permette di svolgere sia
  - Operazioni di definizione schemi mediante i costrutti CREATE,
  - Operazioni di interrogazione mediante il costrutto SELECT
  - Operazioni di modifica dati mediante i costrutti INSERT, DELETE, UPDATE e modifica schemi mediante i costrutti ALTER e DROP, sia

Definizione dei dati in SQL

- Istruzione CREATE TABLE:
  - definisce uno schema di relazione e ne crea un'istanza vuota
  - specifica attributi, domini e vincoli

Domini elementari

- Caratteri: 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

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

Definizione da utente di domini

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

CREATE DOMAIN, 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)

CREATE TABLE (1)

```
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)
)
```

UNIQUE e PRIMARY KEY

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

Definizioni alternative

Matricola CHAR(6) PRIMARY KEY

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

Chiavi su più attributi

```
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!

CREATE TABLE (1)

```
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)
)
```

Vincoli interrelazionali

- 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

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

CREATE TABLE (2)

```
CREATE TABLE Infrazioni(
   Codice CHAR(6) 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)
            ON DELETE SET NULL
            ON UPDATE CASCADE )
```

Auto

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

Structured Query Language
(SQL): DM

Operazioni di lettura (interrogazione)

- Si esaminano 1 o più relazioni e si produce una nuova relazione

Un primo esempio

- Sia data la tabella Persone
- Progettare una query che produca come risultato nome e reddito delle persone con meno di trenta anni

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

Costrutto SELECT

SELECT Attributo1 , Attributo2 , . . .
FROM Tabella1 , Tabella2 , . . .
[**WHERE** condizione]

SELECT indica quali attributi produrre in uscita
FROM indica quali tabelle utilizzare per produrre il risultato
WHERE permette di indicare le condizioni che devono essere rispettate dalle tuple. Più condizioni possono essere composte usando gli operatori logici **AND**, **OR**, **NOT**. Se non ci sono condizioni da imporre, la clausola WHERE può essere omessa.

Soluzione

SELECT Nome,Reddito
FROM Persone
WHERE Età < 30;

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30
Gigi	21

Ridenominazione

- I campi del risultato e le tabelle possono essere rinominate (temporaneamente) nella query utilizzando l'operatore **AS**
- Sarà soprattutto utile con query complesse (Più volte la stessa tabella o campi dallo stesso nome in tabelle diverse)

SELECT X. Att1 , Y. Att4
FROM Tabella **AS** X, Tabella **AS** Y
WHERE X. Att2 = Y. Att3

Esecuzione

SELECT Reddito
FROM Persone
WHERE Eta < 30

Reddito
15
21
30
21

SELECT DISTINCT Reddito
FROM Persone
WHERE Eta < 30

Reddito
15
21
30

Esecuzione del select

- SQL non elimina i duplicati nel risultato (operazione costosa), va chiesto esplicitamente
- I duplicati possono essere eliminati utilizzando il costrutto

  DISTINCT
- I duplicati vengono eliminati dal risultato (solo dopo che la query ha prodotto il risultato)

Abbreviazioni

Per visualizzare tutti gli attributi delle tabelle contenute nella clausola **FROM** si può usare il carattere *

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

SELECT *
FROM Persone
WHERE Eta < 30

Espressioni nella Target List

```
select Reddito/2 as redditoSemestrale
  from Persone
  where Nome = 'Luigi'
```

- Si possono introdurre all'interno della clausola SELECT alcune espressioni (somma, moltiplicazione, sottrazione, divisione).

SELECT Matricola
FROM Impiegati
WHERE Eta > 40

Risultato?

- Se invece chiedo le matricole degli impiegati con età che potrebbe essere maggiore di 40 (o che non è sicuramente minore o uguale di 40)

I valori **NULL**

Matricola	Cognome	Filiale	Eta
1000	Verdi	NULL	56
5998	Neri	Milano	45
7309	Rossi	Roma	32
9553	Bruni	Milano	NULL

Elenco delle matricole degli impiegati con età superiore ai 40

SELECT Matricola
FROM Impiegati
WHERE Eta > 40
OR Eta **IS NULL**

Risultato?

Composizione di più tabelle

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

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

Persone

Nome	Eta	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

34

Esercizio

Elencare i padri di persone che guadagnano più di 20

```
select distinct padre
from persone, paternita
where figlio=nome and reddito>20
```

Esercizio

Elencare le persone che guadagnano più dei rispettivi padri; mostrarne nome, reddito e anche il 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 con clausola join

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

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

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

Persone

Nome	Eta	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

38

Soluzione con Join

- Elencare i nomi delle persone di cui sono noti il padre e la madre

```
select p.figlio, padre, madre
from maternita as m, paternita as p
where p.figlio = m.figlio
```

```
select p.figlio, padre, madre
from maternita as m join paternita as p on
                p.figlio = m.figlio
```

Commenti

- Non è detto che tutte le persone stiano anche nelle tabelle Paternita e Maternita
- Nel risultato ci stanno solo le persone di cui sono noti il padre e la madre
- Perché ON e non WHERE

– 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
```

Join esterno: "outer join"

- *Elencare ogni persona, suo padre e sua madre, dove il padre della persona è noto e la madre può essere nota o no.*

```
select p.figlio, padre, madre
from paternita p left outer join maternita m on p.figlio =
m.figlio
```

- outer e' opzionale

```
select p.figlio, padre, madre
from paternita p left join maternita m
        on p.figlio = m.figlio
```

join naturale

```
select p.figlio, padre, madre
from maternita m join paternita p on
p.figlio = m.figlio
```

Devo dire che attributo figlio voglio!

```
select madre, figlio, padre
from maternita natural join paternita
```

Differenze

```
select p.figlio, padre, madre
from maternita m join paternita p
   on m.figlio = p.figlio
```

```
select p.figlio, padre, madre
from maternita left join paternita
   on m.figlio = p.figlio
```

```
select p.figlio, padre, madre
from maternita full join paternita
   on m.figlio = p.figlio
```

– Che cosa si ottiene?

Operatori insiemistici

Schema del risultato

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

- quali nomi per gli attributi del risultato?
  - quelli del primo operando

Unione

- La select non permette di fare unioni; serve un costrutto esplicito:

```
select  …
union  [all]
select ...
```

- i duplicati vengono eliminati (a meno che si usi all); anche dai risultati delle select!
- N.B. Le due relazioni non devono avere necessariamente lo stesso schema (basta che i tipi degli attributi siano uguali)

Notazione posizionale: attenzione all'ordine degli attributi

```
select padre, figlio
from paternita
union
select figlio, madre
from maternita
```

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

Differenza

```
select Nome
from Persone
except
select Padre as Nome
from Paternita
```

- vedremo che si può esprimere in altro modo (select annidate)

Nota bene

- Intersezione e differenza non sono operatori primitivi, l'unione sì
- Qual è il motivo?

Intersezione

```
select Nome
from Persone
intersect
select Padre as Nome
from Paternita
```

– equivale a

```
select Nome
from Persone, Paternita
where Nome = Padre
```

Structured Query Language (SQL): DD

ancora

Modifiche degli schemi

ALTER DOMAIN

ALTER TABLE

Si possono aggiungere o rimuovere vincoli, modificare i valori di default.

Si possono aggiungere ed eliminare attributi e vincoli sullo schema di una tabella.

DROP DOMAIN

DROP TABLE

Si possono rimuovere componenti: schemi, tabelle, domini, viste.

Structured Query Language (SQL): DM

Operazioni di aggiornamento

- inserimento: insert
- eliminazione: delete
- modifica: update

- di una o più ennuple di una relazione

Inserimento

- insert into Tabella [(Attributi)] values (Valori)

- Si possono inserire righe all'interno della tabella composte dai valori elencati nella clausola VALUES
- Si possono inserire insiemi di righe estratti dalla base di dati

Un primo esempio

- Sia data la tabella Persone

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

Inserimento, commenti

- l'ordinamento degli attributi (se presenti) e dei valori è significativo
- le due liste debbono avere lo stesso numero di elementi
- se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione, secondo l'ordine con cui sono stati definiti
- se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un valore nullo (che deve essere permesso) o un valore di default

Esempi di inserimento

- INSERT INTO Persone(Nome, Età, Reddito) VALUES('Mario',25,52)
- INSERT INTO Persone VALUES ('Mario',52, 25)

- INSERT INTO Persone(Nome, Reddito) VALUES('Lino',55)
- INSERT INTO Persone (Nome)
  SELECT Padre
  FROM Paternita'

Eliminazione

- DELETE FROM Tabella [WHERE Condizione]

- la condizione può coinvolgere anche altre relazioni

Esempi

- DELETE FROM Persone
  WHERE Eta' < 35
- DELETE FROM Paternità

Modifica

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

Eliminazione, commenti

- elimina le ennuple che soddisfano la condizione
- può causare (se i vincoli di integrità referenziale sono definiti con politiche di reazione cascade) eliminazioni in altre relazioni
- se la where viene omessa, si intende where true, cioe' tutte le ennuple vengono eliminate
- N.B. delete from r non è la stessa cosa di drop r

Esempi

- UPDATE Persone SET Reddito = 45
  WHERE Nome = 'Piero'
- UPDATE Persone
  SET Reddito = Reddito * 1.1
  WHERE Eta' < 30

Modifica, commenti

- Aggiorna uno o più attributi di ennuple che soddisfano la condizione
- Se la where viene omessa, si intende where true, cioè tutte le ennuple vengono modificate
- Il valore a cui viene posto l'attributo può essere il risultato di un'espressione che si riferisce anche al valore corrente dell'attributo

Soluzione

- **Create table AUTORE**
  (Nome character(20),
  Cognome character(20),
  DataNascita date,
  Nazionalità character(20),
  primary key(Nome, Cognome))
- **Create table LIBRO**
  (TitoloLibro character(30) primary key,
  NomeAutore character(20),
  CognomeAutore character(20),
  Lingua character(20),
  foreign key (NomeAutore, CognomeAutore)
  references AUTORE(Nome, Cognome)
  on delete cascade
  on update set NULL)

Esercizio 1a

Dare le definizioni SQL delle tabelle

AUTORE (Nome, Cognome, DataNascita, Nazionalità)
LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)

Per il vincolo foreign key specificare una politica di cascade sulle cancellazioni e di set null sulle modifiche.

Esercizio 1b

- Spiegare quale è l'effetto dell'esecuzione dei seguenti comandi di aggiornamento:

```
delete from AUTORE
      where Cognome = 'Rossi'
update LIBRO
set NomeAutore= 'Umberto'
where CognomeAutore = 'Eco'
```

Soluzione

- Il comando cancella dalla tabella AUTORE tutte le tuple con Cognome = 'Rossi'. A causa della politica cascade, anche tutte le tuple di LIBRO con CognomeAutore = 'Rossi' vengono eliminate.
- Il comando causa una violazione a meno che la tabella AUTORE contenga la tupla "Umberto Eco".