SQL

Francesca Gasparini
gasparini@disco.unimib.it

SQL - Introduzione

Structured Query Language

Il linguaggio SQL è un linguaggio per la definizione e la manipolazione dei dati in database relazionali, sviluppato originariamente presso il laboratorio IBM a San Jose' (California) e adottato nel sistema *System R.*

SQL - Introduzione

La standardizzazione è stata cruciale per la diffusione di SQL:

- 1983 : standard de facto
- 1986 : prima versione ufficiale, rivista nel 1989 (SQL-89),
- 1992 : seconda versione (SQL-92 o SQL-2),
- 1999 : terza versione (SQL-99 o SQL-3)

SQL-2

- È ricco e complesso e nessun sistema commerciale lo implementa in maniera completa
- Sono definiti 3 livelli di complessità che individuano dei sotto-standard che possono essere realizzati in un dato DBMS
  - *Entry* - molto simile a SQL - 89,
  - *Intermediate*,
  - *Full*

SQL-3

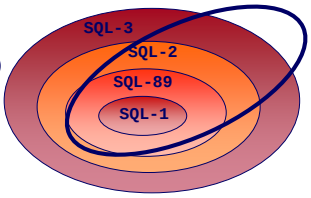
- Introduce il concetto di oggetto.
- Permette all'utente di definire e manipolare tipi di dati complessi.
- I sistemi commerciali spesso offrono funzionalità aggiuntive che non sono specificate nello standard e sono quindi dipendenti dal sistema usato

Implementazioni di SQL

Alcuni DBMS

- ORACLE
- DB2 (IBM)
- Access (Microsoft)
- MSSQL server (Microsoft)
- Informix
- Mysql
- Interbase (Borland / O.S.)
- FireBird (Open Source)
- ...

Un DBMS tipico



SQL - Generalità

- E' un linguaggio con varie funzionalità che contiene:
  - **DDL**: definizione di domini, tabelle, autorizzazioni, vincoli, procedure, ecc.
  - **DML**: linguaggio di query, modifica, comandi transizionali ne esistono varie versioni
- SQL è un linguaggio dichiarativo (non-procedurale)
  - Non specifica la sequenza di operazioni da compiere per ottenere il risultato come accade invece nell'algebra relazionale
  - le espressioni descrivono le proprietà del risultato piuttosto che la procedura per ottenerlo.

7

SQL - Generalità

- SQL è "relazionalmente completo"
  - Ogni espressione dell'algebra relazionale può essere tradotta in SQL
- Il modello dei dati di SQL è basato su tabelle anziché relazioni:
  - Possono essere presenti righe (tuple) duplicate
  - SQL adotta la logica a 3 valori introdotta con l'Algebra Relazionale

8

Notazione

Notazione per definire sintassi di SQL:

< *x* >	usate per isolare un termine *x*
[*x*]	indicano che termine *x* è opzionale
{ *x* }	indicano che termine *x* può essere ripetuto 0 volte o un numero arbitrario di volte
|	separa opzioni alternative Es: *x* | *y* indica che termini *x* e *y* sono alternativi

Le parentesi tonde ( ) appartengono a linguaggio SQL e non alla notazione sopra descritta

9

Definizione di schemi

Uno schema di base di dati è una collezione di oggetti:
- domini, tabelle, asserzioni, viste, privilegi

Uno schema ha un nome e un proprietario

uno schema è definito dalla sintassi:

CREATE SCHEMA
[*NomeSchema*]
[[authorization] *Autorizzazione*]
{ *DefinizioneElementoSchema* }

nome utente proprietario dello schema se omesso è utente che ha lanciato il comando

10

Definizione dei Dati : Le Tabelle

- una tabella è costituita da una collezione ordinata di attributi e da un insieme (eventualmente vuoto) di vincoli
- Istruzione CREATE TABLE:
  - Definisce uno schema di relazione e ne crea un'istanza vuota
  - Per ogni attributo va specificato il dominio, un eventuale valore di default ed eventuali vincoli
  - Possono essere espressi altri vincoli a livello di tabella

11

Definizione dei Dati : Le Tabelle

CREATE TABLE *NomeTabella* (

NomeAttributo Dominio [*ValoreDiDefault*] [*Vincoli*]

{,*NomeAttributo Dominio* [*ValoreDiDefault*] [*Vincoli*] }

[*AltriVincoli*])

12

Definizione dei Dati : I Domini

I domini specificano i valori ammissibili per gli attributi di una relazione:

- Domini elementari (SQL ha 6 domini predefiniti)
  1. Bit SQL-2 poi eliminato e sostituito parzilmente da BOOLEAN in SQL-3
  2. Carattere
  3. Numerico Esatto
  4. Numerico Approssimato
  5. Data/Ora
  6. Intervallo Temporale
- Domini definiti dall'utente (semplici, ma riutilizzabili)

13

Domini Elementari : Il tipo Bit

Corrisponde ad attributi che possono assumere solo due valori (0,1). Attributi di questo tipo (flag) indicano se l'oggetto rappresentato possiede o meno una certa proprietà.

```
bit [varying][(lunghezza)]
bit
bit(lunghezza)
bit varying (lunghezza)
varbit (lunghezza)
```

Si definisce l'attributo Lavoratore nella relazione STUDENTI per indicare se se lo studente è o meno lavoratore

```
Lavoratore bit
```

14

Domini Elementari : Il tipo carattere

Rappresenta singoli caratteri alfanumerici oppure stringhe di lunghezza fissa o variabile

```
char [varying] [(lunghezza)]
char
char(lunghezza)
varchar(lunghezza)
char varying (lunghezza)
```

Si definisce l'attributo Nome della relazione IMPIEGATI come sequenza di caratteri di lunghezza massima 20

```
Nome char(20)
```

```
Paolo_Bianchi_______
```

```
Nome varchar(20)
```

```
Paolo_Bianchi
```

15

Domini Elementari : I Tipi Numerici Esatti

Rappresentano numeri interi o numeri decimali in virgola fissa (con un numero prefissato di decimali)

```
integer
smallint

numeric   numeric(precisione) numeric(precisione,scala)
decimal   decimal(precisione) decimal(precisione,scala)
```

Si definisce l'attributo Eta nella relazione IMPIEGATI

```
Eta decimal(2)
```

Rappresenta tutti i numeri fra -99 e +99

Si definisce l'attributo Cambio nella relazione PAGAMENTO per indicare il valore del cambio di una certa moneta preciso al centesimo

```
Cambio numeric(6,2)
```

Rappresenta tutti i numeri fra -9999,99 e +9999,99

16

Domini Elementari : I Tipi Numerici Esatti

INTEGER / SMALLINT rappresentano valori interi.

La precisione (numero totale di cifre) varia a seconda della specifica implementazione di SQL. SMALLINT richiede minore spazio di memorizzazione.

NUMERIC / DECIMAL rappresentano i valori decimali.

La differenza tra NUMERIC e DECIMAL è che il primo deve essere implementato esattamente con la precisione richiesta, mentre il secondo può avere una precisione maggiore.

17

Domini Elementari : Tipi numerici approssimati

Sono utili per rappresentare valori reali approssimati, ad esempio grandezze fisiche (rappresentazione in virgola mobile).

```
float              float(precisione)
double precision
real
```

Si definisce l'attributo Massa nella relazione ASTEROIDI

```
Massa real
```

0,17E16 = $1,7\ 10^{15}$

mantissa — esponente

18

Domini Elementari : Tipi numerici approssimati

REAL / DOUBLE PRECISION rappresentano valori a singola / doppia precisione in virgola mobile.

La precisione (lunghezza della mantissa) dipende dalla specifica implementazione di SQL.

FLOAT permette di richiedere la precisione che si desidera.

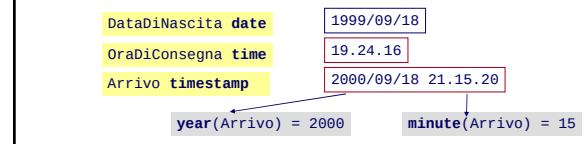
19

Domini Elementari : Data/Ora

Permettono di rappresentare istanti di tempo

```
date
time      time(precisione)
timestamp timestamp(precisione)
```

Ciascuno di questi domini è strutturato e decomponibile in un insieme di campi (anno, mese, giorno, ora, minuti, secondi)



20

Domini Elementari : Data/Ora

DATE rappresenta le date espresse come anno (4 cifre), mese (2 cifre), giorno (2 cifre)

DATE 'yyyy-mm-dd'

TIME [WITH TIME ZONE] rappresenta i tempi espressi come ora (2 cifre), minuto (2 cifre) e secondo (2 cifre)

TIME 'hh:mm:ss[.[nnnnnn]] [TimeZone]'

dove TimeZone è espresso nel formato {+|-} hh:mm

21

Domini Elementari : Intervalli temporali

Permette di rappresentare intervalli di tempo come durate di eventi

```
interval PrimaUnitàDiTempo
interval PrimaUnitàDiTempo to UltimaUnitàDitempo
```

Anzianità di servizio in anni e mesi (x anni e y mesi)

```
AnzianitaServizio interval year to month
```

Tempo di consegna in giorni ed ore (x giorni e y ore)

```
TempoConsegna interval day to hour
```

22

Domini Elementari : Intervalli temporali

INTERVAL rappresenta una durata temporale

Non è possibile costruire intervalli che comprendano mesi e giorni. È possibile definire una precisione per la prima e la seconda unità di tempo. Nel caso la seconda unità siano secondi, c'è un'interpretazione speciale.

Intervalli fino a 999 anni e 11 mesi

```
Interval year(3) to month
```

Intervalli fino a 9 secondi, approssimati a un millesimo di secondo

```
Interval second(1) to second(3)
```

23

BLOB e CLOB

Permettono di includere direttamente nel database oggetti molto grandi.
Binary Large Object (BLOB) e Character Large Object (CLOB)

Figure e documenti descrittivi

```
fotografia  BLOB(10M)
descrizione CLOB(100k)
```

Definiti solo in SQL-3, ma implementati in diversi DBMS commerciali

24

Domini definiti dagli utenti

Partendo dai domini predefiniti è possibile costrure nuovi domini tramite la primitiva **`create domain.`**

Un nuovo dominio è caratterizzato dalle seguenti informazioni: nome, dominio elementare, valore di default, insieme di vincoli (constraints)

```
CREATE DOMAIN NomeDominio as DominioElementare
                [ ValoreDefault ] [ Constraints ]
```

25

CREATE DOMAIN - Esempio

```
CREATE DOMAIN Voto AS SMALLINT
      DEFAULT 0
      NOT NULL
```

Il nuovo dominio Voto è definito come uno SMALLINT con valore di default e che non deve essere nullo.

la definizione di "nuovi domini" è utile perché permette di associare dei vincoli a un nome di dominio: questo è importante quando si deve ripetere la stessa definizione di attributo su diverse tabelle: ad esempio, modifiche alla definizione di Voto si ripercuotono in tutte le occorrenze di questo dominio nello schema del Database.

26

CREATE TABLE - Esempio

```
CREATE TABLE NomeTabella (
    NomeAttributo Dominio [ ValoreDiDefault ] [ Vincoli ]
    {,NomeAttributo Dominio [ValoreDiDefault ] [Vincoli ] }
    [ AltriVincoli ] )


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,
  UNIQUE (Cognome,Nome)
)
```

27

Valori di default per il dominio

Definiscono il valore che deve assumere l'attributo quando non viene specificato un valore durante l'inserimento di una tupla

DEFAULT < *ValoreGenerico* | user | null >

ValoreGenerico rappresenta un valore compatibile con il dominio, rappresentato come una costante o come un'espressione.

user è la login dell'utente che effettua il comando di aggiornamento della tabella.

28

Definizione dei Dati : Il valore NULL

E' un valore polimorfico (che appartiene a tutti i domini) col significato di valore non noto:

1) il valore esiste in realtà ma è ignoto al database
   (es.: data di nascita)

2) il valore è inapplicabile
   (es.: numero patente per minorenni)

3) non si sa se il valore è inapplicabile o meno
   (es.: numero patente per un maggiorenne)

29

Definizione dei Dati : Vincoli (Constraints)

Un vincolo è una regola che specifica delle condizioni sui valori di un elemento dello schema del database.

Un vincolo può essere associato ad una tabella, ad un attributo, ad un dominio.

I vincoli possono essere di due tipi:

- Vincoli Intrarelazionali
  - Si applicano all'interno di una relazione
- Vincoli Interrelazionali
  - Si applicano tra relazioni diverse

30

Definizione dei Dati : Vincoli Intrarelazionali

Possono essere:

NOT NULL	(Il valore deve essere non nullo)
UNIQUE	(I valori devono essere non ripetuti)
PRIMARY KEY	(Chiave primaria)
CHECK	(Condizioni complesse)

31

Vincoli Intrarelazionali : Esempio

```
CREATE TABLE Impiegato (
   Matricola  CHAR(6)       PRIMARY KEY,
   Nome       CHAR(20)      NOT NULL,
   Cognome    CHAR(20)      NOT NULL,
   Stipendio  NUMERIC(9) DEFAULT 0,
   UNIQUE (Cognome, Nome)
)
```

Vincolo su più attributi

Vincolo su un attributo

32

Vincoli Intrarelazionali : UNIQUE

```
CREATE TABLE Impiegato (
   Matricola  CHAR(6)       PRIMARY KEY,
   Nome       CHAR(20)      NOT NULL,
   Cognome    CHAR(20)      NOT NULL,
   Stipendio  NUMERIC(9) DEFAULT 0,
   UNIQUE (Cognome, Nome)
)
```

L'attributo o la n-pla di attributi su cui è definito il vincolo UNIQUE non possono avere istanze uguali ripetute.

→l'attributo o attributi sono (super)chiave

eccezione: valore null può comparire su diverse righe senza violare il vincolo: si assume che i valori null siano tutti diversi fra loro.

33

Vincoli Intrarelazionali : PRIMARY KEY

```
CREATE TABLE Impiegato (
   Matricola  CHAR(6)       PRIMARY KEY,
   Nome       CHAR(20)      NOT NULL,
   Cognome    CHAR(20)      NOT NULL,
   Stipendio  NUMERIC(9) DEFAULT 0,
   UNIQUE (Cognome, Nome)
)
```

Il vincolo PRIMARY KEY può essere definito una sola volta all'interno della relazione.

34

Vincoli Intrarelazionali : PRIMARY KEY

```
CREATE TABLE Impiegato (
   Nome       CHAR(20)      NOT NULL,
   Cognome    CHAR(20)      NOT NULL,
   Stipendio  NUMERIC(9) DEFAULT 0,
   PRIMARY KEY (Cognome, Nome)
)
```

In alcune implementazioni di SQL potrebbe essere necessario specificare comunque anche il vincolo NOT NULL per tutti gli attributi coinvolti.

35

Vincoli Intrarelazionali : ATTENZIONE!

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

Nome	Cognome
Luca	Rossi
Giovanni	Bianchi
Emilio	Verdi
Emilio	Rossi

Non è la stessa cosa di

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

Nome	Cognome
Luca	Neri
Giovanni	Bianchi
Emilio	Verdi
Francesca	Rossi

36

Vincoli Intrarelazionali : CHECK

Introdotto in SQL-2, permette di definire vincoli più complessi di quelli standard.

```
CREATE DOMAIN Voto AS SMALLINT
        DEFAULT 0
        NOT NULL

CREATE DOMAIN Voto AS SMALLINT
        DEFAULT 0
        CHECK (Voto>=18 AND VOTO<=30)
```

37

Vincoli interrelazionali

Coinvolgono più relazioni / tabelle.

Possono essere definiti attraverso i costrutti sintattici:

REFERENCES	Permettono di definire vincoli di integrità
FOREIGN KEY	referenziale
CHECK	(Vincoli complessi)

Si hanno due sintassi
- per singoli attributi
- su più attributi

E' possibile definire politiche di reazione alle violazioni.

38

Vincoli interrelazionali : Integrità referenziale

Esprime un legame gerarchico (padre / figlio) fra tabelle.

Alcuni attributi della tabella figlio sono definiti FOREIGN KEY e si devono riferire (REFERENCES) ad alcuni attributi della tabella padre che costituiscono una chiave (devono essere UNIQUE e NOT NULL oppure PRIMARY KEY).

I valori contenuti nella FOREIGN KEY devono essere sempre presenti nella tabella padre.

39

Vincoli interrelazionali : Integrità referenziale

Si possono avere due sintassi:

•nella parte di definizione degli attributi con il costrutto sintattico REFERENCES:

AttrFiglio CHAR(3) REFERENCES *TabellaPadre*(*AttrPadre*)

•oppure dopo le definizioni degli attributi con i costrutti FOREIGN KEY e REFERENCES:

```
FOREIGN KEY (AttrFiglio)
                REFERENCES TabellaPadre(AttrPadre)
```

40

Vincoli interrelazionali : Integrità referenziale

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

41

Vincoli interrelazionali : Integrità referenziale

Se si omettono gli attributi destinazione, vengono assunti quelli della chiave primaria.

AttrFiglio CHAR(3) REFERENCES *TabellaPadre*

Quando si hanno più attributi da riferire, si utilizza sempre FOREIGN KEY:

```
FOREIGN KEY (AttrFiglio1 {,AttrFiglio2} )
        REFERENCES TabellaPadre(AttrPadre1 {,AttrPadre2})
```

42

Vincoli Interrelazionali

Definiamo tre tabelle con le informazioni degli esami sostenuti dagli studenti:

- Tabella Studente
- Tabella Esame
- Tabella Corso

43

Definizione Tabella Studente

```
CREATE TABLE Studente (
Matr        CHAR(6)          PRIMARY KEY,
Nome        VARCHAR(30)      NOT NULL,
Città       VARCHAR(20),
CDip        CHAR(3)
)
```

Matr	Nome	Città	CDip
34321	Luca	Mi	Inf
53524	Giovanni	To	Mat
64521	Emilio	Ge	Ing
73321	Francesca	Vr	Mat

44

Definizione Tabella Esame

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE             NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)
)
```

Matr	CodCorso	Data	Voto
34321	1	25/02/01	28
34321	2	14/12/01	30
64521	1	12/03/02	24
73321	4	18/01/02	18

45

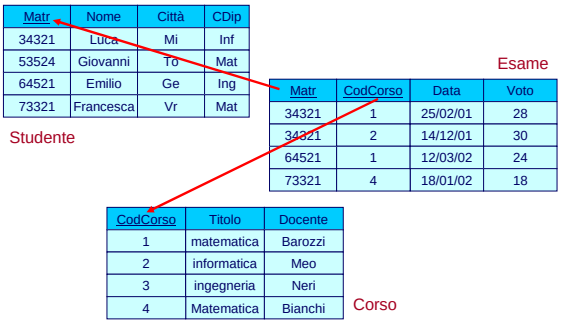
Definizione Tabella Corso

```
CREATE TABLE Corso (
CodCorso    CHAR(6)          PRIMARY KEY,
Titolo      VARCHAR(30)      NOT NULL,
Docente     VARCHAR(20)
)
```

CodCorso	Titolo	Docente
1	matematica	Barozzi
2	informatica	Meo
3	ingegneria	Neri
4	Matematica	Bianchi

46

Vincoli interrelazionali : Integrità referenziale



47

Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE             NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)
)
```

48

Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE        NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)

FOREIGN KEY (Matr) REFERENCES Studente

)
```


Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE        NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)

FOREIGN KEY (Matr) REFERENCES Studente

FOREIGN KEY (CodCorso) REFERENCES Corso

)
```


Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6)     REFERENCES Studente,
CodCorso    CHAR(6)     REFERENCES Corso,
Data        DATE        NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)
)
```


Vincoli interrelazionali : Il problema delle violazioni

- Per tutti gli altri vincoli visti fino ad ora, a seguito di una violazione, il comando di aggiornamento viene rifiutato segnalando l'errore all'utente.
- per i vincoli di integrità referenziale invece SQL permette di scegliere delle reazioni da adottare in caso di violazioni.

Vincoli interrelazionali : Il problema delle violazioni

- si possono introdurre violazioni operando sulle righe della tabella padre (tabella esterna) o sulle righe della tabella figlio (tabella interna)
- modifiche sulla tabella interna (figlio):
  - inserimento di una nuova riga
  - modifica della foreign key

  non vengono proposte reazioni, solo il rifiuto dell'operazione
- modifiche sulla tabella esterna (padre):
  - cancellazione di una riga
  - modifica dell'attributo riferito

  vengono proposte diverse reazioni

Vincoli interrelazionali : Il problema dell'aggiornamento

Studente

Matr	Nome	Città	CDip
34321	Luca	Mi	Inf
53524	Giovanni	To	Mat
64521	Emilio	Ge	Ing
73321	Francesca	Vr	Mat

Esame

Matr	CodCorso	Data	Voto
34321	1	25/02/01	28
34321	2	14/12/01	30
64521	1	12/03/02	24
73321	4	18/01/02	18

Se eliminiamo da Studente la tupla con matricola 34321, cosa succede alla tabella Esame?

Problema degli orfani!

Vincoli interrelazionali : Reazioni alle violazioni

Le reazioni operano sulla tabella figlio (**es Esami**), in seguito a modifiche alla tabella padre (**es Studente**)

Reazioni previste:

CASCADE : propaga la modifica
SET NULL: annulla l'attributo che fa riferimento
SET DEFAULT: assegna il valore di default all'attributo
NO ACTION : impedisce che la modifica possa avvenire

Le violazioni possono essere introdotte:

1. da modifica (update) dell'attributo cui si fa riferimento
2. da cancellazioni di tuple

55

Vincoli interrelazionali : Cancellazioni

Cosa succede degli esami se si cancella uno studente?

CASCADE
si cancellano anche gli esami dello studente
SET NULL
si pone a null la matricola dei relativi esami
SET DEFAULT
si pone al valore di default la matricola dei relativi esami
NO ACTION
si impedisce la cancellazione dello studente

56

Vincoli interrelazionali : Cancellazioni

Cosa succede degli esami se si modifica la matricola di uno studente?

CASCADE
si modifica la matricola degli esami dello studente
SET NULL
si pone a null la matricola dei relativi esami
SET DEFAULT
si pone al valore di default la matricola dei relativi esami
NO ACTION
si impedisce la modifica della matricola dello studente

57

Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE        NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)
FOREIGN KEY (Matr) REFERENCES Studente
        ON DELETE CASCADE
        ON UPDATE CASCADE
)
```

58

Vincoli interrelazionali : Esempio

```
CREATE TABLE Esame (
Matr        CHAR(6),
CodCorso    CHAR(6),
Data        DATE        NOT NULL,
Voto        Voto,
PRIMARY KEY (Matr, CodCorso)
FOREIGN KEY (Matr) REFERENCES Studente
        ON DELETE CASCADE
        ON UPDATE CASCADE
FOREIGN KEY (CodCorso) REFERENCES Corso
        ON DELETE NO ACTION
        ON UPDATE CASCADE
)
```

59

Vincoli interrelazionali : Esempio

Una istanza scorretta

Studente

Matr	Nome	Città	CDip
123			
456			
789			
120			

Esame

Matr	CodCorso	Data	Voto
123	1	25/02/01	28
123	2	14/12/01	30
123	1	12/03/02	24
120	4	18/01/02	25
456	1	12/03/02	NULL
555	4	18/01/02	23

Viola la chiave
Viola il NULL
Viola integrità referenziale

60

Vincoli interrelazionali : Esempio

Una istanza corretta

Studente

Matr	Nome	Città	CDip
123			
456			
789			
120			

Esame

Matr	CodCorso	Data	Voto
123	1	25/02/01	28
123	2	14/12/01	30
120	4	18/01/02	25

61

Vincoli con nomi

A fini diagnostici (e di documentazione) è spesso utile sapere quale vincolo è stato violato a seguito di un'azione sul DB.

A tale scopo è possibile associare dei nomi ai vincoli, ad esempio:

```
Stipendio INTEGER CONSTRAINT StipendioPositivo
CHECK (Stipendio > 0),

CONSTRAINT ForeignKeySedi
FOREIGN KEY (Sede) REFERENCES Sedi
```

62

Modifiche degli schemi

Necessarie per garantire l'evoluzione della base di dati a fronte di nuove esigenze.

Ci sono due comandi SQL appositi:

ALTER: modifica oggetti persistenti

DROP: cancella oggetti dallo schema

63

Modifiche degli schemi : ALTER

Si applica su domini e tabelle:

```
ALTER DOMAIN NomeDominio <
      SET     DEFAULT ValoreDefault |
      DROP    DEFAULT |
      ADD     CONSTRAINT DefVincolo |
      DROP    CONSTRAINT NomeVincolo >
```

64

Modifiche degli schemi : ALTER

Si applica su domini e tabelle:

```
ALTER TABLE NomeTabella <
      ALTER COLUMN NomeAttributo  <
            SET DEFAULT NuovoDefault |
            DROP DEFAULT > |
      DROP COLUMN NomeAttributo |
      ADD COLUMN DefAttributo |
      DROP CONSTRAINT NomeVincolo
      ADD CONSTRAINT DefVincolo >
```

65

Modifiche degli schemi : DROP

Cancella oggetti DDL, si applica su domini, tabelle, indici, view, asserzioni, procedure,...

```
DROP < schema, domain, table, view, ...> NomeElemento
      [ RESTRICT | CASCADE ]
```

Opzioni:

RESTRICT Impedisce drop se gli oggetti comprendono istanze non vuote

CASCADE Applica drop agli oggetti collegati. Potenziale pericolosa reazione a catena

66

Cataloghi relazionali

Il catalogo contiene il dizionario dei dati (data dictionary), ovvero la descrizione della struttura dei dati contenuti nel database.

Lo standard SQL-2 organizza il catalogo su due livelli

Definition_Schema (composto da tabelle, non vincolante)

Information_Schema (composto da viste, vincolante)

67

ESERCIZI

Esercizio

Definire un attributo che permetta di rappresentare stringhe di lunghezza massima pari a 256 caratteri, su cui non sono ammessi valori nulli e con valore di default "sconosciuto".

```
CREATE DOMAIN NomeDominio as DominioElementare
        [ ValoreDefault ] [ Constraints ]
```

Soluzione:

```
CREATE DOMAIN String as character varying (256)
default 'sconosciuto' not null
```

69

Esercizio

Si consideri il seguente schema relazionale:

PRODOTTO(cod-prod,descrizione,prezzo-unitario)
FATTURA(cod-fatt,cliente,cod-prod,quantità)

Implementare in SQL lo schema relazionale dell'esercizio con gli opportuni vincoli di integrità

70

Esercizio

```
CREATE TABLE PRODOTTO (
cod-prod char(3) PRIMARY KEY,
descrizione char(25),
prezzo-unitario  integer CHECK (prezzo-unitario > 0)
)

CREATE TABLE FATTURA (
cod-fatt char(3), PRIMARY KEY
cliente char(25),
cod-prod char(3) REFERENCES Prodotto (cod-prod),
quantità smallint CHECK (quantità > 0),
)
```

71

Esercizio

```
CREATE TABLE PRODOTTO (
...
prezzo-unitario  integer CHECK (prezzo-unitario > 0)
)

CREATE TABLE FATTURA (
...
quantità smallint CHECK (quantità > 0),
...
)
```

Questi vincoli non sono direttamente deducibili dallo schema ma sono utili come vincoli di controllo.

PRODOTTO(cod-prod,descrizione,prezzo-unitario)
FATTURA(cod-fatt,cliente,cod-prod,quantità)

72

Esercizio

Dare le definizioni SQL delle tre tabelle

FONDISTA(Nome, Nazione, Età)
GAREGGIA(NomeFondista, NomeGara, Piazzamento)
GARA(Nome, Luogo, Nazione, Lunghezza)

rappresentando in particolare i vincoli di foreign key della tabella GAREGGIA.

73

Esercizio

FONDISTA(Nome, Nazione, Età)

```
CREATE TABLE FONDISTA (
Nome character(20) primary key,
Nazione character(30),
Età smallint
)
```

GARA(Nome, Luogo, Nazione, Lunghezza)

```
CREATE TABLE GARA(
Nome character(20) primary key,
Luogo character(20),
Nazione character(20),
Lunghezza integer
)
```

74

Esercizio

GAREGGIA(NomeFondista, NomeGara, Piazzamento)

```
CREATE TABLE GAREGGIA
(
NomeFondista character(20) references FONDISTA(Nome),
NomeGara character(20),
Piazzamento smallint,
primary key (NomeFondista, NomeGara),
foreign key (NomeGara) references GARA(Nome)
)
```

75

Esercizio

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 sulla cancellazione e di set null sulle modifiche.

76

Esercizio

AUTORE (Nome, Cognome, DataNascita, Nazionalità)

```
CREATE TABLE AUTORE (
Nome character(20),
Cognome character(20),
DataNascita date,
Nazionalità character(20),
primary key (Nome, Cognome)
)
```

77

Esercizio

LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)

```
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
)
```

78

Esercizio

Dare le definizioni SQL delle tabelle

CLIENTE (CodCli ,Indirizzo, Piva)
ORDINE (CodOrd, CodCli, Data, Importo**)**
DETTAGLIO (CodOrd,CodProd,Qta)
PRODOTTO (CodProd, Nome, Prezzo)

Per il vincolo foreign key specificare una politica di:

- default su cancellazione e modifiche di CodCli
- cascade su cancellazione e modifiche di CodOrd
- no action su cancellazione e modifiche di CodProd

79

Esercizio

```
CREATE TABLE Cliente
( CodCli char(6) primary key,
Indirizzo char(50) ,
PIva char(12) unique )

CREATE TABLE Ordine
( CodOrd char(6) primary key,
CodCli char(6) not null default='999999',
Data date,
Importo decimal,
foreign key CodCli references Cliente
on delete set default
on update set default)
```

80

Esercizio

```
CREATE TABLE DETTAGLIO (
CodOrd char(6),
CodProd char(6),
Qta smallint,
primary key (CodOrd,CodProd)
foreign key CodOrd references Ordine
on delete cascade
on update cascade
foreign key CodProd references Prodotto
on delete no action
on update no action)

CREATE TABLE PRODOTTO (
CodProd char(6) primary key,
Nome char(20),
Prezzo smallint )
```

81