Laboratorio di Basi di Dati

Turni T3 e T4

a.a. 2018/2019

Ruggero Pensa - Fabiana Vernero

Argomenti

- Introduzione a SQL
- Definizione dei dati con SQL
- Interrogazioni di base
- Interrogazioni con vari tipi di join
- Interrogazioni con operatori aggregati
- Interrogazioni con raggruppamento
- Interrogazioni con operatori insiemistici
- Interrogazioni con query annidate
- Inserimenti, modifiche, cancellazioni
- Vincoli di integrità generici
- Viste

In questa lezione

- Introduzione a SQL
- Definizione dei dati con SQL

Introduzione a SQL

SQL - 1

- Pronuncia “esse cu elle” o anche “sìquel”:
  - Originariamente *Structured Query Language*, ora è un nome a sé.

SQL - 2

- Contiene sia il DDL che il DML:
  - DDL: Data Definition Language.
  - DML: Data Manipulation Language.
- In pratica:
  - SQL è il linguaggio per interrogare e gestire le basi di dati.
- Le interrogazioni si effettuano per mezzo di costrutti di programmazione detti **query.**

SQL - 3

- E' il linguaggio di riferimento per tutti i DBMS relazionali (una sorta di *lingua franca*).
- Ne esistono varie versioni!
  - Diversi standard e diverse implementazioni.

Storia di SQL - 1

- Prima proposta: linguaggio SEQUEL (1974), creato da IBM.
  - Prime implementazioni in SQL/DS di IBM e Oracle (1981).
  - In un primo tempo, dal 1983 circa, standard de facto.
- Nascono vari standard ANSI e ISO:
  - SQL Base (SQL-86, SQL-89)
  - SQL-92
  - SQL-3 (SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016)

Storia di SQL - 2

- In questo corso faremo riferimento a SQL-92 (SQL-2).
- Ci sono ancora differenze tra produttori di DBMS, con diversi livelli di implementazione e di supporto al linguaggio:
  - Entry SQL
  - Intermediate SQL
  - Full SQL (non ancora recepito dai DBMS in commercio)

Definizione di dati
SQL come DDL

Argomenti

- SQL come Data Definition Language:
  - Domini.
  - Creazione di tabelle.
  - Definizione dei vincoli.

Domini elementari: stringhe

- Caratteri
  - Lunghezza fissa: **`char(length)`** (abbreviazione di **`character`**).
  - Lunghezza variabile: **`varchar(max_length)`** (abbreviazione di **`character varying`**).
  - Se la lunghezza non è specificata, si assume sia 1.
- Esempi
  - **`char(100)`**: stringa di esattamente 100 caratteri.
  - **`varchar(20)`**: stringa variabile fino a max 20 caratteri.
  - **`char`**: un singolo carattere.

Domini elementari: numeri - 1

- Tipi numerici esatti
  - Numeri decimali: **`decimal(precisione,scala)`** o **`numeric(precisione,scala)`**
    - *precisione*: numero totale di cifre decimali (opzionale)
    - *scala*: numero di cifre decimali dopo la virgola (opzionale)
    - (differenza: per `decimal` la precisione specificata è quella minima garantita, per `numeric` è quella esatta)
  - Interi: **`smallint`**, **`integer`** (rappresentazione dipendente da implementazione)
- Esempi:
  - **`decimal(6,4)`**: decimali da -99,9999 a +99,9999
  - **`numeric(4)`**: decimali da -9999 a +9999
  - **`integer`**: interi a 32 bit (solitamente)

Domini elementari: numeri - 2

- Tipi numerici approssimati
  - Virgola mobile con mantissa ed esponente:

    `float(precisione)`, **`real`**, **`double precision`**
    - *precisione*: numero di cifre binarie per la mantissa (opzionale)
- Esempi:
  - **`real`** permette di rappresentare valori che possono essere nell'intervallo tra -3.40E+38 e -1.18E-38, tra 1.18E-38 e 3.40E+38 oppure 0

Domini elementari: date

- Istanti temporali
  - Per le date si usa **`date`**, che comprende i sottocampi **`year`**, **`month`**, **`day`**
  - Per gli orari si usa **`time(precisione)`**, che comprende i sottocampi **`hour`**, **`minute`**, **`second`**
  - Per specificare sia date che orari si usa **`timestamp(precisione)`**, che comprende **`year`**, **`month`**, **`day`**, **`hour`**, **`minute`**, **`second`**
    - *precisione*: numero di cifre per le frazioni di secondo (opzionale)
- Esempi
  - **`timestamp(2)`** per memorizzare eventi di log con precisione al centesimo di secondo

Domini elementari: intervalli - 1

- Intervalli temporali
  - Per esprimere la durata useremo
    `interval unità1 [to unità2]`
    - *unità1*: unità di tempo più grossolana
    - *unità2* (opzionale): unità di tempo più fine

Domini elementari: intervalli - 2

- Esempi:
  - **`interval year`**: esprime intervalli in anni
  - **`interval year to month`**: esprime intervalli misurati in anni e mesi (es. un anno e due mesi)
  - **`interval day to second(2)`**: esprime intervalli misurati in giorni, ore, minuti, secondi e centesimi di secondo (es. 3 giorni, 6 ore, 15 minuti e 10,45 secondi
  - **`interval year to minute`**: **non è permesso** (infatti non si può passare in modo preciso da **month** a **day** perché i mesi hanno durata diversa)

Altri domini (da SQL-3)

- **`boolean`**: valori booleani (true e false).
- **`bigint`**: interi «grossi»
- **`blob`**: binary large object (immagini, video, file di vario tipo)
  - In PostgreSQL **bytea** oppure **lo** (Large Object)
- **`clob`**: character large object (lunghi file di testo)
  - In PostgreSQL **text**

Domini personalizzati - 1

- È possibile creare domini personalizzati (le parti tra [] sono opzionali):

  `create domain` *`NomeDominio`* `as` *`TipoDato`*
  `[default` *`ValoreDefault`*`] [`*`Vincoli`*`]`

  - *`NomeDominio`*: nome del nuovo dominio
  - *`TipoDato`*: nome del tipo di base
  - **`default`** *`ValoreDefault`*: valore assegnato in automatico se non specificato esplicitamente (opzionale)
  - *`Vincoli`*: insieme di vincoli sui valori assunti dal dominio personalizzato (opzionale)
- I domini personalizzati sono comunque semplici (no array, no struct, no record).

Domini personalizzati - 2

- Voto di un esame

```
create domain Voto
as smallint default NULL
check (value >=18 AND value <= 30);
```

- Temperatura corporea (gradi Celsius)

```
create domain Temperatura
as decimal(3,1) default NULL
check (value >=35.0 AND value <= 42.0);
```

Domini personalizzati - 3

- Se non sono permessi domini «composti» a cosa serve definire nuovi domini?
  Per astrarre…

- Esempio: passaggio da Celsius a Fahrenheit
  - Basta cambiare una volta per tutte il dominio di Temperatura

```
create domain Temperatura
as decimal(4,1) default NULL
check (value >=95.0 AND value <= 107.6);
```

Definizione di tabelle

- Sintassi per la definizione e creazione di una tabella

```
create table NomeTabella (
  NomeAttributo1 Dominio1 [ValoreDefault1] [Vincoli1],
  NomeAttributo2 Dominio2 [ValoreDefault2] [Vincoli2],
  …
  NomeAttributoN DominioN [ValoreDefaultN] [VincoliN],
  [AltriVincoli]
);
```

- Esempio: *Dipartimento(Nome, Indirizzo, Citta)*

```
create table Dipartimento (
     Nome varchar(20) primary key,
     Indirizzo varchar(50),
     Citta varchar(20) not null
);
```

Valori di default

- Per ogni attributo può essere specificato un valore predefinito che verrà usato se, nell'inserimento di una riga, non viene specificato un valore per quell'attributo.
- Esempi:

```
NumeroFigli smallint default 0,
Email varchar(255) default 'guest@unito.it',
StatoCivile varchar(20) default 'libero'
```

- Se non si specifica esplicitamente un valore di default, viene usato **null**.

Definizione dei vincoli

- I vincoli servono a definire proprietà che devono essere verificate da ogni istanza della base di dati per garantirne l'**integrità** (vincoli di integrità).
- Si differenziano in:
  - vincoli *intrarelazionali* (relativi a una sola tabella)
  - vincoli *interrelazionali* (relativi a più tabelle)
- Si possono specificare:
  - Contestualmente alla definizione degli attributi.
  - Alla fine della definizione di tabella.
- Si possono usare vincoli predefiniti oppure si può specificare l'espressione logica che il vincolo deve verificare.

Vincoli intrarelazionali e interrelazionali predefiniti

- Vincoli intrarelazionali:
  - not null
  - unique
  - primary key
- Vincoli interrelazionali:
  - Integrità referenziale (chiave esterna)

Vincoli intrarelazionali predefiniti - 1

- Vincolo **not null**
  - E' un vincolo di tupla che indica che il valore nullo non è ammesso come valore per un determinato attributo, quindi rende l'attributo obbligatorio.
  - Se l'attributo non viene specificato in fase di inserimento, si viola il vincolo di integrità e l'operazione è annullata.
    - Se per l'attributo viene specificato un valore di default è possibile effettuare l'inserimento anche senza specificarne il valore.
  - Sintassi:

    NomeAttributo Dominio **not null**

- Esempio:

```
Citta varchar(20) not null
```

Vincoli intrarelazionali predefiniti - 2

- Vincolo **unique (di chiave univoca)**
  - E' un vincolo di tabella che indica che il valore di un attributo o le combinazioni di valori su un insieme di attributi sono una superchiave:

    righe diverse ⇔ valori diversi
  - Fa eccezione il valore nullo (che può comparire in più righe senza violare il vincolo).
  - Sintassi (vincolo su un solo attributo):

    NomeAttributo Dominio **unique**
- Esempio

```
Matricola varchar(6) unique
```

Vincoli intrarelazionali predefiniti - 3

- Vincolo **unique su insiemi di attributi**
  - Va specificato dopo la definizione degli attributi della tabella:
  - Sintassi

```
NomeAttributo1 Dominio1
NomeAttributo2 Dominio2
…
unique (NomeAttributo1, NomeAttributo2, …)
```

- Esempio

```
Nome varchar(255),
Cognome varchar(255),
unique (Nome,Cognome)
```

Vincoli intrarelazionali predefiniti - 4

- Vincolo **unique su insiemi di attributi:**

  ATTENZIONE!

```
Nome varchar(255) not null unique,
Cognome varchar(255) not null unique
```

  è diverso da:

```
Nome varchar(255) not null,
Cognome varchar(255) not null,
unique (Nome, Cognome)
```

  Perché? Qual è più restrittivo?

Vincoli intrarelazionali predefiniti - 5

- Vincolo **primary key (di chiave primaria)**:
  - E' un vincolo di tabella che indica che un attributo o un insieme di attributi sono la chiave primaria.
  - Gli attributi così definiti non possono assumere valore nullo.
  - Può esserci un solo vincolo primary key per ogni tabella.

Vincoli intrarelazionali predefiniti - 6

- Vincolo **primary key su un solo attributo**
  - Va specificato nella definizione dell'attributo.
  - Sintassi:

    NomeAttributo Dominio **primary key**
- Esempio:

```
Matricola varchar(6) primary key
```

(questo vincolo implica:

```
Matricola varchar(6) not null unique
```

ma non è equivalente: per ogni tabella può essere definito un solo vincolo primary key e inoltre i dati possono venire organizzati in modo diverso a livello fisico)

Vincoli intrarelazionali predefiniti - 7

- Vincolo **primary key su insiemi di attributi:**
  - Va specificato dopo la definizione degli attributi
  - Sintassi

```
NomeAttributo1 Dominio1,
NomeAttributo2 Dominio2,
…
primary key (NomeAttributo1, NomeAttributo2,…)
```

- Esempio:

```
Nome varchar(255),
Cognome varchar(255),
primary key (Nome,Cognome)
```

Vincoli interrelazionali predefiniti - 1

- Vincolo di **integrità referenziale** (di **chiave esterna**):
  - Crea un vincolo tra i valori di uno (o più) attributi della tabella (interna) su cui è definito e uno (o più) attributi di un'altra tabella (esterna).
  - Per ogni riga della tabella interna, il valore dell'attributo specificato nel vincolo, se diverso da *null*, deve essere presente tra i valori del corrispondente attributo della tabella esterna.
  - Sintassi con un solo attributo:
    - `NomeAttributo Dominio references NomeTabellaEsterna(NomeAttributoEsterno)`

Vincoli interrelazionali predefiniti - 2

- Sintassi con più attributi
  - ```
    NomeAttributo1 Dominio1,
    NomeAttributo2 Dominio2,
    …
    foreign key (NomeAttributo1, NomeAttributo2, …)
     references
    NomeTabellaEsterna(NomeAttributoEsterno1,
     NomeAttributoEsterno2,…)
    ```
- **ATTENZIONE**: l'ordine è importante (`NomeAttributo1` viene mappato su `NomeAttributoEsterno1`, `NomeAttributo2` su `NomeAttributoEsterno2`, ecc.)
    ```

Base di dati «Ricoveri»

PAZIENTI

COD	Cognome	Nome	Residenza
A102	Necchi	Luca	TO
B372	Rossigni	Piero	NO
B543	Missoni	Nadia	TO
B444	Missoni	Luigi	VC
S555	Rossetti	Gino	AT

MEDICI

MATR	Cognome	Nome	Residenza	Reparto
203	Neri	Piero	AL	A
574	Bisi	Mario	MI	B
461	Bargio	Sergio	TO	B
530	Belli	Nicola	TO	C
405	Mizzi	Nicola	AT	R
501	Monti	Mario	VC	A

REPARTI

COD	Nome	Primario
A	Chirurgia	203
B	Pediatria	574
C	Medicina	530
L	Lab-Analisi	530
R	Radiologia	405

RICOVERI

PAZ	Inizio	Fine	Reparto
A102	2/05/94	9/05/94	A
A102	2/12/94	2/01/95	A
S555	5/10/94	3/12/94	B
B444	1/12/94	2/01/95	B
S555	06/09/95	01/11/95	A

Esempio di vincoli di integrità referenziale

- Tabella Ricoveri
  - L'attributo `PAZ` della tabella (interna) `Ricoveri` si riferisce all'attributo `COD` della tabella (esterna) `Pazienti`.
  - L'attributo `Reparto` della tabella (interna) `Ricoveri` si riferisce all'attributo `COD` della tabella (esterna) `Reparti`.
  - Nella definizione della tabella `Ricoveri`, quindi, imponiamo questi vincoli:

```
PAZ varchar(4) references Pazienti(COD),
…
Reparto char references Reparti(COD)
```

  - oppure:

```
PAZ varchar(4),
Reparto char,
foreign key (PAZ) references Pazienti(COD),
foreign key (Reparto) references Reparti(COD)
```

Violazione del vincolo di integrità referenziale - 1

- In generale, quando un vincolo viene violato, il DBMS rifiuta l'operazione che causerebbe la violazione e segnala un errore.
- Con i vincoli di integrità referenziale si possono specificare altre reazioni da adottare in caso di violazione.

Violazione del vincolo di integrità referenziale - 2

- Casi in cui può avvenire una violazione di un vincolo di integrità referenziale:
  - Cambiamento della tabella interna:
    1. Inserimento di una nuova riga nella tabella interna
       Es. inserisco la riga `(A202, 1/1/15,30/1/15, D)` nella tabella *Ricoveri* (inserisco un nuovo ricovero di un paziente inesistente).
    2. Modifica, nella tabella interna, di un valore dell'attributo referente.
       Es. modifico nella tabella *Ricoveri* la riga `(A102, 2/5/94, 9/5/94, A)` in `(A202, 2/5/94, 9/5/94, A)` (facendo riferimento a un paziente inesistente).

Violazione del vincolo di integrità referenziale - 3

- Cambiamento della tabella esterna:
  3. Modifica, nella tabella esterna, di un valore dell'attributo riferito.
     Es. modifico nella tabella *Pazienti* il codice `A102` in `A202` (in questo modo la tabella *Ricoveri* vìola il vincolo di integrità referenziale)
  4. Cancellazione di una riga nella tabella esterna.
     Es. cancello dalla tabella *Pazienti* la riga relativa al paziente `A102` (in questo modo la tabella *Ricoveri* vìola il vincolo di integrità referenziale).

Violazione del vincolo di integrità referenziale - 4

- Quando cambia la tabella interna (casi 1 e 2), l'operazione viene semplicemente rifiutata.
- Quando cambia la tabella esterna (casi 3 e 4), si può specificare quali variazioni apportare alla tabella interna.
- L'asimmetria deriva dal fatto che dal punto di vista applicativo la tabella esterna ricopre il ruolo di tabella principale (**master**) e quella interna di secondaria (**slave**) che deve adeguarsi alle variazioni.

Reazioni in seguito a modifica (caso 3)

- **cascade**: il nuovo valore dell'attributo della tabella esterna viene riportato in tutte le corrispondenti righe della tabella interna.
- **set null**: all'attributo referente viene assegnato il valore nullo al posto del valore modificato nella tabella esterna.
- **set default**: all'attributo referente viene assegnato il valore di default al posto del valore modificato nella tabella esterna.
- **no action**: nessuna reazione (e quindi la modifica non viene consentita).

Reazioni in seguito a cancellazione (caso 4)

- **cascade**: tutte le righe della tabella interna corrispondenti alla riga cancellata nella tabella esterna vengono cancellate.
- **set null**: all'attributo referente viene assegnato il valore nullo al posto del valore cancellato nella tabella esterna.
- **set default**: all'attributo referente viene assegnato il valore di default al posto del valore cancellato nella tabella esterna.
- **no action**: nessuna reazione (e quindi la cancellazione non viene consentita).

Sintassi

- Per la modifica, subito dopo il vincolo:
  `on update` *Reazione*
- Per la cancellazione, subito dopo il vincolo:
  `on delete` *Reazione*
  - *Reazione* assume un valore tra `cascade`, `set null`, `set default`, `no action`
- Esempi. Nella tabella *Ricoveri*:

```
PAZ varchar(4) references Pazienti(COD) on update
cascade,
Reparto char,
foreign key (Reparto) references Reparti(COD) on
delete set null
```

- `on update` e `on delete` possono essere combinati.

Creazione tabella Ricoveri

```
create table Ricoveri (
      PAZ varchar(4),
      Inizio date,
      Fine date,
      Reparto char,
      primary key (PAZ, Inizio),
      foreign key (PAZ) references Pazienti(COD)
            on update cascade
            on delete no action,
      foreign key (Reparto) references Reparti(COD)
            on update cascade
            on delete set null
);
```

Supporto dei DBMS ai vincoli di integrità referenziale

- In Oracle:
  - Non è possibile definire una clausola `on update` (dà supporto solo alla `on delete`).
  - Nella `on delete` non sono ammesse le reazioni `set default` e `no action` (per `no action` basta omettere la clausola `on delete`)
- PostgreSQL non ha queste limitazioni
- In MySQL (InnoDB)
  - Non è permesso `set default` e sono presenti altre limitazioni.
  - Non è supportata la sintassi inline all'attributo ma solo la sintassi *`foreign key (A) references T(A)`*.

Assegnare nomi ai vincoli - 1

- In alcune occasioni può essere utile assegnare un nome a un vincolo.
- Si utilizza la seguente sintassi dopo tutte le definizioni di attributi:

```
constraint NomeVincolo DefinizioneVincolo
```

Assegnare nomi ai vincoli - 2

- Esempi:

```
create table Ricoveri (
      PAZ varchar(4) not null,
      Inizio date not null,
      Fine date,
      Reparto char not null,
      constraint pk_Ricoveri primary
      key(PAZ,Inizio),
      constraint fk_RicPaz foreign key (PAZ)
      references  Pazienti(COD),
      constraint fk_RicRep foreign key (Reparto)
      references Reparti(COD)
);
```

Modifiche alle definizioni

- SQL permette di modificare la definizione di tabelle, domini, vincoli precedentemente introdotti
  - `alter` (modifica), `drop` (cancellazione), `add` (aggiunta)

- Qual è il risultato dei seguenti comandi SQL?
  - `alter domain Temperatura drop default;`
  - `alter domain Temperatura set default 37.0;`
  - `drop table Ricoveri;`
  - `alter table Ricoveri drop constraint fk_RicPaz;`
  - `alter table Ricoveri add constraint fk_RicPaz foreign key (PAZ) references Pazienti(COD);`
  - `alter table Ricoveri add column Referto varchar(50);`
  - `alter table Ricoveri drop column Referto;`

Inserimento dei dati - 1

- Per inserire nuove righe in una tabella si usa il comando `insert`.
- Sintassi:
  - `insert into Tabella(`$Attributo_1$`, …,` $Attributo_n$`) values (`$ValoreAttributo_1$`, …,` $ValoreAttributo_n$`);`
- Inserisce nella tabella singole righe assegnando $Attributo_1 = ValoreAttributo_1$, ecc.
- Gli attributi omessi assumono il valore di default o null
  - Se l'attributo non è *nullable* e non ha default, il DBMS segnala l'errore e annulla l'inserimento.

Inserimento dei dati - 2

- Se si specificano i valori per tutte le colonne, la lista di attributi può essere omessa.
  - Esempio:

```
insert into S values ('S6', 'Alice', 40, 'Turin');
```

Cancellazione di dati

- Per cancellare condizionatamente delle righe da una tabella si usa il comando `delete`.
- Sintassi:

  `delete from` *`Tabella`* `where` *`Condizione`*`;`

  - Cancella tutte le righe in *Tabella* per cui *Condizione* è vera.
  - *Condizione* può essere anche un predicato con una sottointerrogazione.
  - Esempio:
    - `delete from P where City='London';`