SQL

Luca Anselma
anselma@di.unito.it

Programma della parte di SQL

- Definizione dei dati con SQL
- Interrogazioni:
  - di base
  - join
  - operatori aggregati e raggruppamenti
  - operatori insiemistici
  - query annidate semplici e correlate
  - common table expressions
- Inserimenti, modifiche, cancellazioni
- Vincoli di integrità generici
- Viste

SQL

- pronuncia "esse cu elle" o anche "sìquel"
- originariamente *Structured Query Language*, ora è un nome a sé
- linguaggio di riferimento per tutti i DBMS relazionali (una sorta di lingua franca)
- contiene sia il DDL che il DML
  - DDL: Data Definition Language
  - DML: Data Manipulation Language
- ne esistono varie versioni

Storia di SQL

- prima proposta SEQUEL (Structured English Query Language) da Chamberlin e Boyce in IBM (1974)
- nato per fornire un linguaggio più intuitivo rispetto all'algebra e al calcolo relazionale per utenti non informatici
- prime implementazioni in System R e in SQL/DS di IBM, ma non sfruttato
- « adottato » da Oracle che ha prodotto un DBMS di grande successo (1981)

Storia di SQL

- in un primo tempo standard de facto
- poi 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)

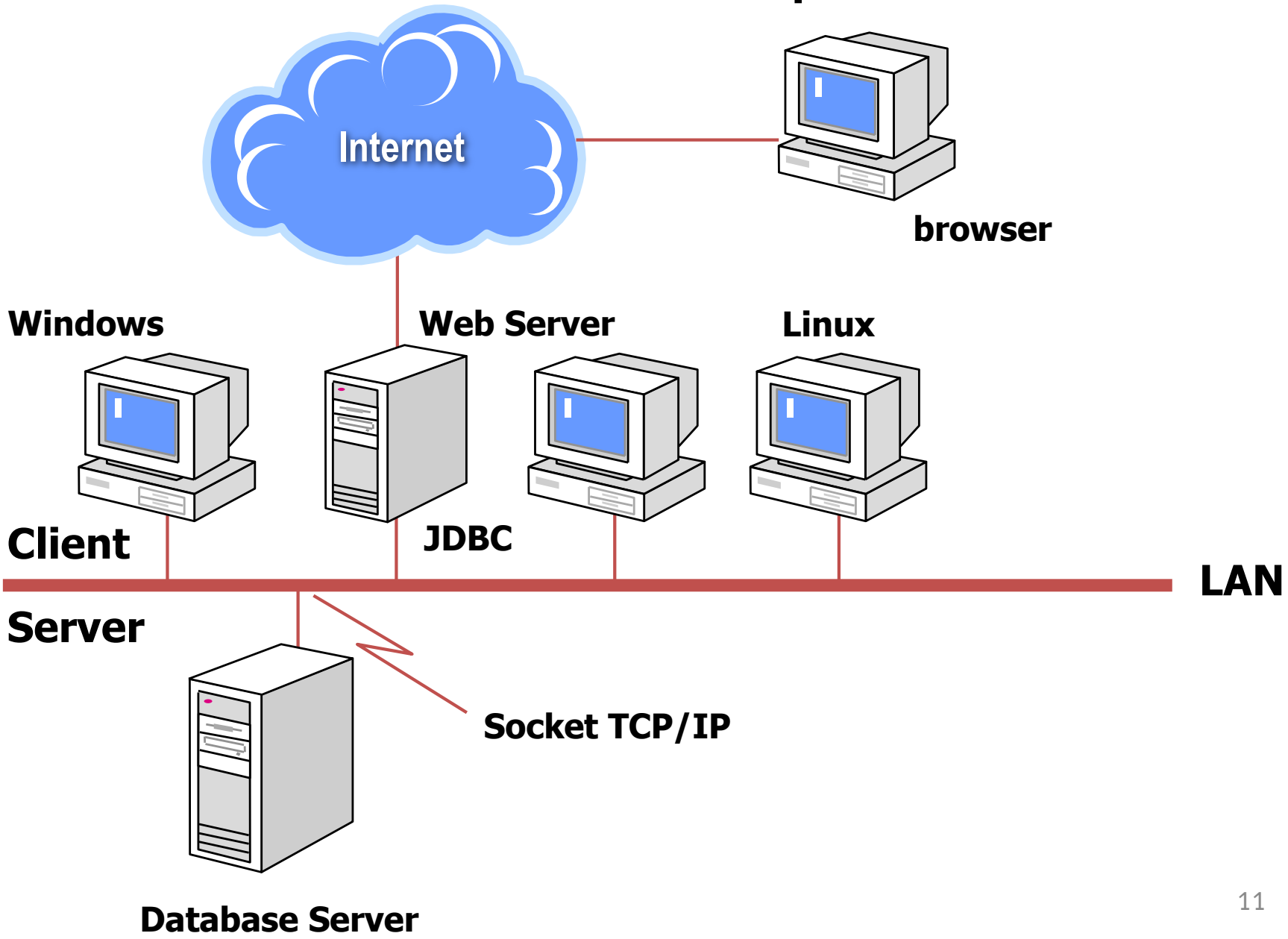
Storia di SQL

- in questo corso ci riferiamo a SQL-92 perché è il più diffuso
- qualche cenno ai costrutti più recenti
- differenze tra produttori di DBMS
- diversi livelli di implementazione e di supporto al linguaggio:
  - Entry SQL
  - Intermediate SQL
  - Full SQL (non ancora recepito dai DBMS in commercio)

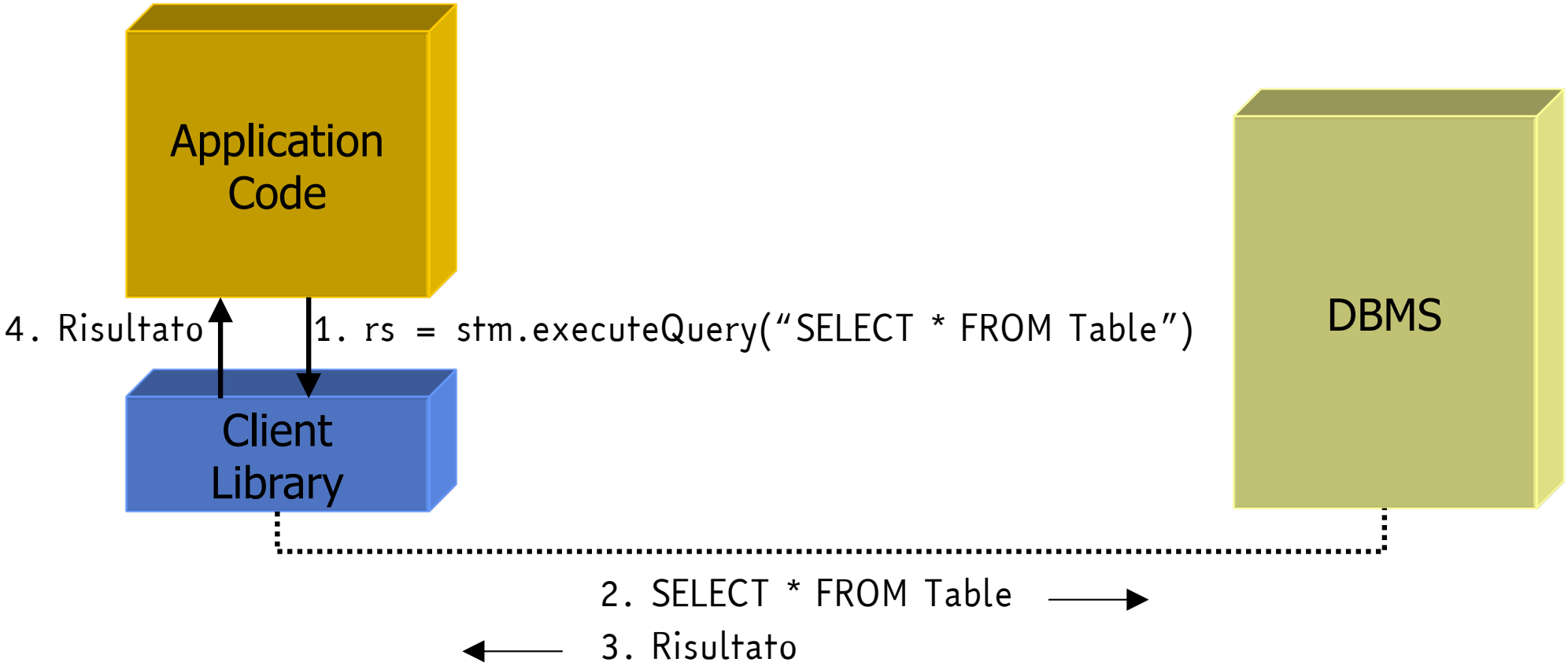
Architettura

- La maggior parte dei DBMS si basa sull'architettura client/server
- Una connessione è il frutto della cooperazione di due processi:
  - Il processo server accetta le connessioni dai client e ha il compito di interagire con i database su delega del client che ha fatto richiesta
  - Un client richiede di effettuare determinate operazioni su uno o più database. Un client può essere: un'applicazione grafica, una pagina web, un programma scritto da un utente.
    Ad es. *pgAdmin* e *psql* (per PostgreSQL) e *MySQLWorkbench (per MySQL)* sono utility che permettono di impartire comandi SQL interattivamente a un server e sono esempi di client
- Un server è in grado di accettare più connessioni contemporaneamente

Scenario tipo



Client Library



Esistono client library differenti (ad es., libpq (interfaccia con il C), JDBC (interfaccia con Java), ...) che isolano il codice applicativo dallo specifico codice di interazione con il DBMS

Interrogazione dei dati - SQL come DML

- Prime semplici query

Caratteristiche di SQL come linguaggio di interrogazione

- linguaggio dichiarativo:
  - l'utente definisce *cosa* vuole ottenere, il DBMS determina *come* ottenerlo
  - l'algebra relazionale, invece, è procedurale: l'interrogazione specifica i passi da compiere per estrarre le informazioni
  - una query SQL viene analizzata dal query optimizer del DBMS e trasformata in modo da essere eseguita in modo molto efficiente con un linguaggio procedurale interno
  - quando si scrivono query SQL, quindi, si possono in gran parte ignorare gli aspetti di efficienza

Sintassi

- Struttura essenziale (introdurremo le variazioni di volta in volta):

select *ListaAttributi* (target list)
[**from** *ListaTabelle*] (clausola "from")
[**where** *Condizione*]; (clausola "where")

Significato dell'interrogazione

- Restituisce una tabella:
  - considerando il prodotto cartesiano delle tabelle della clausola "*from*"
  - selezionando le righe che soddisfano la condizione della clausola "*where*" (opzionale)
  - dando in output i valori degli attributi elencati nella target list ("*select*")

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

S = Supplier (fornitore), P = Parts (parti), QTY = quantity
nut = dado, bolt = bullone, screw = vite, cam = camma, cog = ruota dentata

Forme di una query SQL

- Forma più semplice
  - mostra tutte le informazioni di tutti i fornitori

SQL:	*select ** *from S;*
calcolo su tuple con dichiarazioni di range:	*{ s.* | s(S) }*
algebra relazionale:	S

Forme di una query SQL

- Proiezione
  - mostra nome e città di tutti i fornitori

    SQL: *select SName, City*
    from S;

    c. r.: *{ s.SName, s.City | s(S) }*

    a. r.: $\pi_{SName,City}(S)$

Forme di una query SQL

- Selezione
  - mostra nome e città di tutti i fornitori che hanno status maggiore di 20

SQL:
```
select SName, City
from S
where Status > 20;
```

c. r.: $\{ s.SName, s.City \mid s(S) \mid s.Status > 20 \}$

a. r.: $\pi_{SName,City}(\sigma_{Status>20}(\pi_{SName,City,Status}(S)))$

oppure

$\pi_{SName,City}(\sigma_{Status>20}(S))$

Espressioni nella clausola select

- Nella clausola select possono comparire espressioni sul valore degli attributi di ciascuna riga selezionata
- Esempio: estrarre il peso in grammi delle viti

```
select PName, Color, Weight*1000
from P
where PName='Screw';
```

PName	Color	Weight*1000
Screw	Blue	17000
Screw	Red	14000

Definizione di dati – SQL come DDL

- 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
  - in Oracle "***varchar***" è riservato e occorre usare ***varchar2***
- Esempi
  - ***char(20)***: stringa di esattamente 20 caratteri
  - ***varchar(100)***: stringa variabile fino a max 100 caratteri
  - ***char***: un singolo carattere

Domini elementari - Numeri

- 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*** (almeno 16 bit, con segno), ***integer*** (almeno 32 bit, con segno)
- Esempi:
  - ***decimal(6,4)***: decimali da -99,9999 a +99,9999
  - ***numeric(4)***: decimali da -9999 a +9999

Domini elementari - Numeri

- 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*** (che in Oracle non c'è ma è esprimibile come ***float(24)***) 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

- Intervalli temporali
  - Per esprimere la durata useremo
    ***interval unità1 [to** unità2]*
    - *unità1*: unità di tempo più grossolana
    - *unità2* (opzionale): unità di tempo più fine
    - ***Nota:*** Oracle permette di specificare la precisione anche per giorno e anno

Domini elementari - Intervalli

- Esempi:
  - ***interval year***: esprime durate in anni
  - ***interval year to month***: esprime durate misurate in anni e mesi (es. un anno e due mesi)
  - ***interval day to second(2)***: esprime durate misurate 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» (almeno 64 bit, con segno)
- ***blob***: binary large object (immagini, video, file di vario tipo) (in PostgreSQL ***bytea*** (oppure ***lo***))
- ***clob***: character large object (lunghi file di testo) (in PostgreSQL ***text***)

Domini personalizzati

- 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 - esempi

- 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 – utilità

- 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 predefiniti

- Vincolo **not null**
  - è 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 e non si è specificato un valore di default, si viola il vincolo di integrità e l'operazione è annullata
  - sintassi

    NomeAttributo Dominio ***not null***
  - se per l'attributo viene specificato un valore di default è possibile effettuare l'inserimento anche senza specificarne il valore
- Esempio

  Citta varchar(20) not null

Vincoli intrarelazionali predefiniti

- Vincolo **unique (di chiave univoca)**
  - è 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

- 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

- 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

- Vincolo **primary key (di chiave primaria)**
  - è 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

- 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

- 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

- 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

- Vincolo di **integrità referenziale** (di **chiave esterna**)
  - 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

- 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

- 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

- 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)

- 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 della tabella interna viene assegnato il valore nullo al posto del valore modificato nella tabella esterna
- ***set default***: all'attributo referente della tabella interna 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 della tabella interna viene assegnato il valore nullo al posto del valore cancellato nella tabella esterna
- ***set default***: all'attributo referente della tabella interna 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***)
- Inoltre in Oracle 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
- MySQL non supporta la sintassi inline all'attributo ma solo la sintassi *foreign key (A) references T(A)*

Assegnare nomi ai vincoli

- 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

- 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

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

Inserimento di dati

- Esempio:
  insert into S values ('S6', 'Alice', 40, 'Turin');

Cancellazione di dati

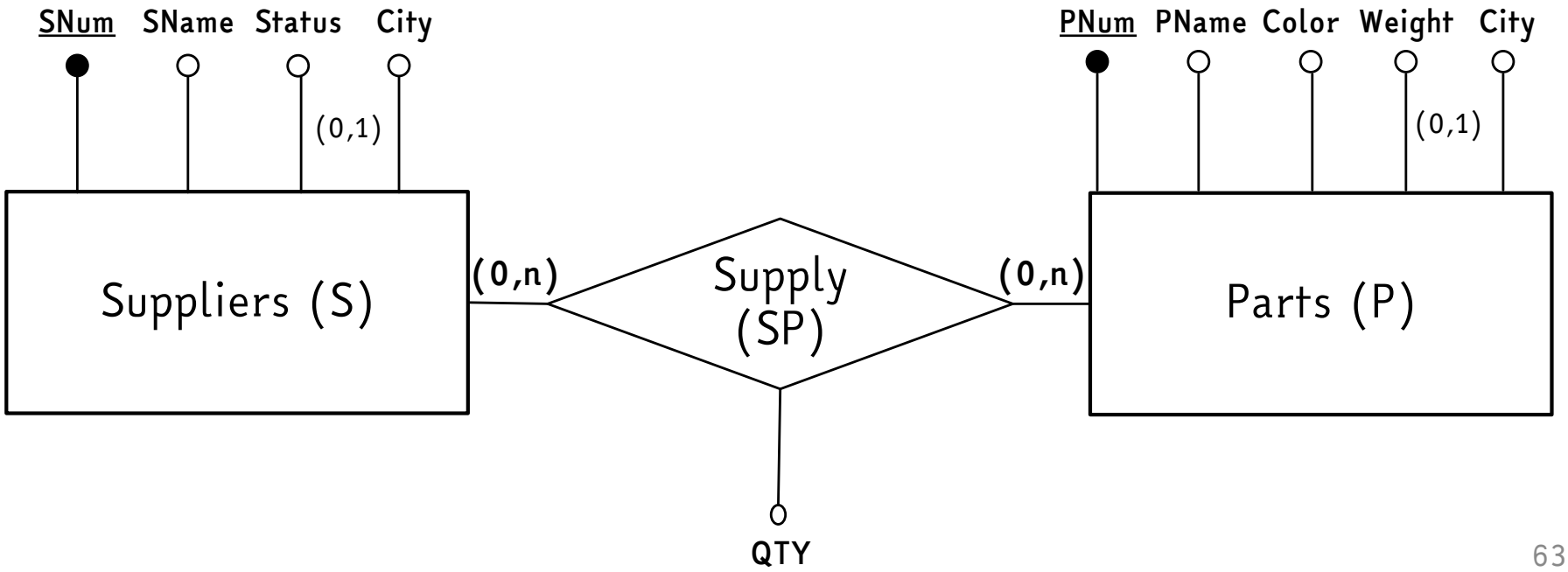
- Per cancellare condizionatamente delle righe da una tabella si usa il comando *delete*
- Sintassi

  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';

Esercizio 1.1

- Creare le tabelle (con vincoli) relative al seguente schema ER
  - Esempi di dati nella slide successiva
  - Impostare **cascade** in caso di cancellazione
  - Se usate un DB condiviso con altri utenti, usate come suffisso il vostro numero di matricola (es., ***S_123456)***



Esercizio 1.1

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

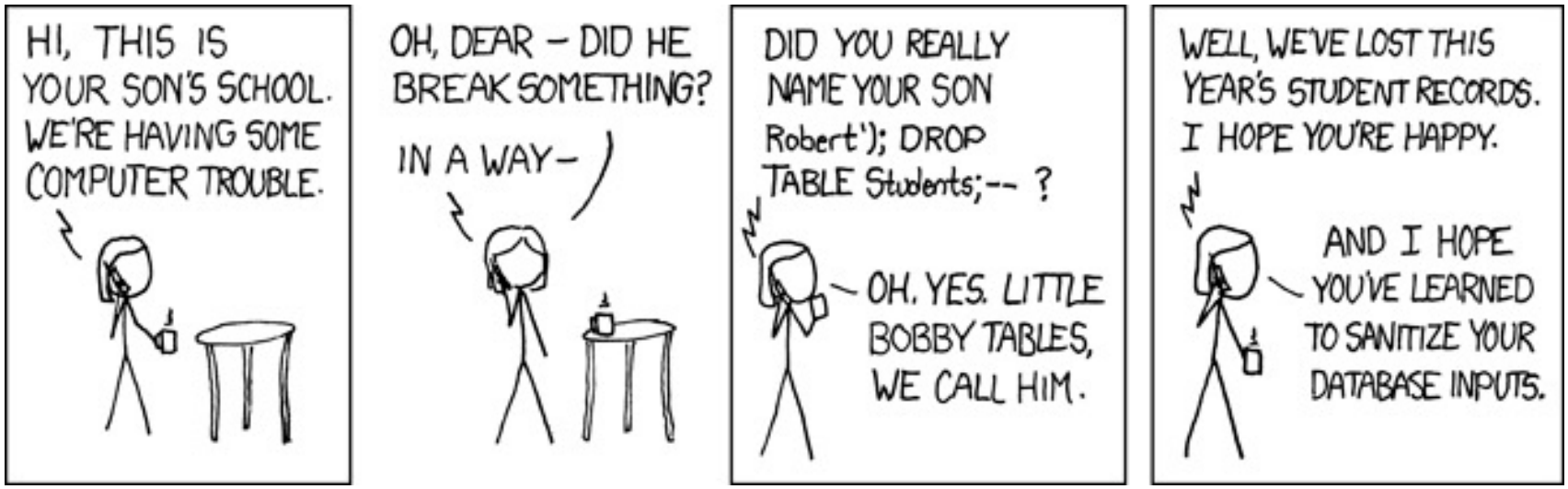
SP

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

nut = dado, bolt = bullone, screw = vite, cam = camma, cog = ruota dentata

Esercizio 1.1

- Effettuare inserimenti, modifiche, cancellazioni nella base di dati utilizzando l'interfaccia utente e osservare il comportamento del DMBS



https://xkcd.com/327/