Lezione 3

SQL come DDL

SQL

- Sviluppato presso il Centro di Ricerca dell'IBM
- Nasce come SeQueL
- Diventa poi SQL: Structured Query Language
- Contiene:
  - DDL (Data Definition Language)
  - DML (Data Manipulation Language)
  - DCL (Data Control Language).
- I sistemi commerciali spesso offrono una serie di strumenti che estendono le funzionalità definite a livello di standard.

Lo standard SQL

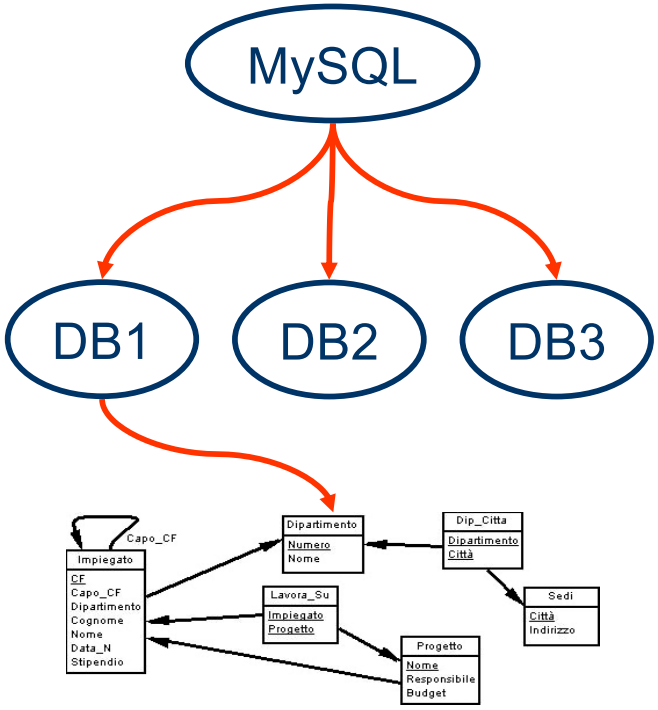
- È un linguaggio di tipo dichiarativo
- SQL Base
  - SQL-86 - costrutti base
  - SQL-89 – integrità referenziale
- SQL-2
  - SQL- 92: entry level, intermediate level, full
- SQL-3
  - SQL-99 – estensione ad oggetti, trigger, ..
  - SL-2003 – Estensione del modello a oggetti, SQL/XML, ….
- La sua forza sta nel fatto di essere uno standard
  - I sistemi commerciali sono classificati in base all'aderenza allo standard 92
    - Entry Level (SQL-89)
    - Intermediate Level
    - Full SQL (SQL-92)

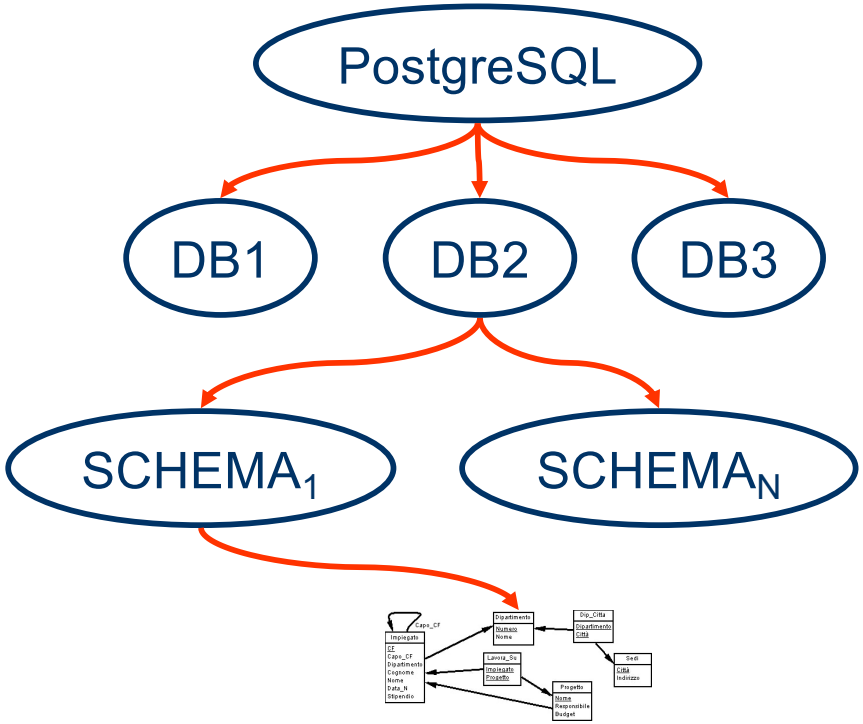
In SQL:

- TABLE ⇔ Tabella ⇔ Relazione
- ROW ⇔ Riga ⇔ Tupla
- COLUMN ⇔ Colonna ⇔Attributo
- SQL come DDL ha tre comandi fondamentali:
  - Create
    - Database, Schema, Table, Domain, Constraint, ….
  - Alter
    - Database, Schema, Table, Domain, Constraint, ….
  - Drop
    - Database, Schema, Table, Domain, Constraint, ….

Database vs Schema

- Nelle prime versioni tutte le tabelle erano considerate parte del medesimo schema. Oggi, le tabelle sono spesso raggruppate in SCHEMI (insieme di oggetti correlati)





Domini Elementari

- Specificare il tipo di un dato significa già imporre un vincolo
  - Vincoli di dominio
- Tutti i domini condividono il valore NULL

- I possibili domini
  - BIT
  - CHARACTER
  - NUMERICI ESATTI
  - NUMERICI APPROSSIMATI
  - DATA e ORA
  - INTERVALLI TEMPORALI

BIT

- BIT
  - Permette di rappresentare dati a due valori
    - in realtà a **TRE** valori: vero, falso, NULL
    - si usa per i flag si o no, vero o non vero

- Stringhe di BIT
  - bit [varying] [(lunghezza)]
    - bit(10) stringa di 10 bit
    - bit varying (10) stringa di al massimo 10 bit
  - varbit [(lunghezza)]
    - lunghezza deve avere valori tra 1 e 32767
    - stringhe lunghe fino 32767 bit
    - Se lunghezza non è specificata viene assunta uguale a 1

Carattere

- Carattere
  - Permette di rappresentare singoli caratteri o stringhe. Le stringhe possono essere a lunghezza fissa oppure variabile.
  - character [varying] [(lunghezza)] [character set *famiglia*]
  - char [varying] [(lunghezza)] [character set *famiglia*]
  - varchar [(lunghezza)] [character set *famiglia*]
    - la lunghezza deve avere valori tra 1 e 32767
    - se la lunghezza non si specifica viene assunta uguale a 1
    - character varying o varchar fissano stringhe a lunghezza variabile
  - character varying (10) character set Greek
    - Stringa fino 10 caratteri del set Greco

NUMERIC e DECIMAL

- Tipi numerici esatti:
  - permettono di rappresentare valori interi o valori in virgola fissa.
  - numeric [(Precisione[,Scala])]
  - decimal [(Precisione[,Scala])]
    - La precisione fissa il numero di cifre totali (max 127)
    - La scala quanti di queste sono decimali (dopo la virgola)
    - numeric(6,3) fissa valori comprresi nell'intervallo [-999.999,999.999]
  - ogni combinazione di precisione e scala produce un un tipo di dato diverso
  - la differenza tra NUMERIC e DECIMAL è che il primo deve essere implementato esattamente con la precisione fissata, mentre il secondo può avere una precisione maggiore

INTERI

- int
- smallint
  - Tipi di dati numerici esatti che utilizzano dati integer
  - Int
    - da -2^{31} (-2.147.483.648) a $2^{31}-1$ (2.147.483.647)
    - occupa 4 byte
  - smallint
    - da -2^{15} (-32.768) a $2^{15}-1$ (32.767)
    - occupa 2 byte

REALI

- Tipi numerici approssimati:
  - float [(n)]
    - Tipi di dati numerici approssimati da utilizzare con dati numerici a virgola mobile
    - n fissa il numero di bit della mantissa.
  - real
  - double precision
    - equivale a float(53)
    - la precisione di double è doppia rispetto a real

      - float da - $1{,}79E^{+308}$ a $-2{,}23E^{-308}$, 0 e da $2{,}23E^{-308}$ a $1{,}79E^{+308}$ (con n 1-24 occupa 4 byte, con n 25-53 occupa 8 byte)
      - real da - $3{,}40E^{+38}$ a $-1{,}18E^{-38}$, 0 e da $1{,}18E^{-38}$ a $3{,}40E^{+38}$ (4 byte)

Oracle data type

Oracle NUMBER Data Type

NUMBER Data Type:

The NUMBER data type stores zero, positive and negative fixed numbers.

Fixed-point number format:

NUMBER(p,s)

- Where p is the precision, of up to 20 base-100 digits, which is equivalent to 39 or 40 decimal digits depending on the position of the decimal point.
- s is the scale, the scale can range from -84 to 127.
- Positive scale is the number of significant digits to the right of the decimal point to and including the least significant digit.
- Negative scale is the number of significant digits to the left of the decimal point, to but not including the least significant digit.

Examples:

Actual Data	Format	Stored As
123.79	NUMBER	123.79
123.88	NUMBER(3)	124
123.89	NUMBER(3,2)	exceeds precision
123.89	NUMBER(4,2)	exceeds precision
123.89	NUMBER(5,2)	123.89
123.89	NUMBER(6,1)	123.9
123.89	NUMBER(6,-2)	100
.05678	NUMBER(4,5)	.05678
.00013	NUMBER(4,5)	.00013
.000127	NUMBER(4,5)	.00013
.0000012	NUMBER(2,7)	.0000012
.00000123	NUMBER(2,7)	.0000012
1.2e-4	NUMBER(2,5)	0.00012
1.2e-5	NUMBER(2,5)	0.00001

DATA e ORA

- Data e Ora:
  - date
    - formato YYYY-MM-DD
  - time [(fractional seconds scale)]
    - fractional specifica il numero di cifre per la parte frazionaria dei secondi (numero intero compreso tra 0 e 7)
    - da 8 posizioni minimo (hh:mm:ss) a 16 massimo (hh:mm:ss.nnnnnn)
  - timestamp
    - date+time
    - YYYY-MM-DD: hh:mm:ss

INTERVALLI TEMPORALI

- Intervalli temporali:
  - interval
    - rappresenta una durata temporale in riferimento a uno o più dei qualificatori
      - YEAR, MONTH, DAY, HOUR, MINUTE, SECOND
    - Se vengono specificati due qualificatori distinti, implicitamente vengono considerati anche tutti i qualificatori logicamente compresi tra essi

      INTERVAL ‘123-2’ YEAR(3) TO MONTH
      - (123 anni e 2 mesi)

      INTERVAL ‘4 5:12’ DAY TO MINUTE
      - (4 giorni, 5 ore e 12 minuti)

      INTERVAL ‘11:12:10’ HOUR TO SECOND
      - (11 ore, 12 minuti e 10 secondi)

Altri domini

- SQL-99 ha introdotto
  - boolean
    - precedentemente realizzato tramite il tipo bit (varbit), non implementato da molti sistemi
  - blob (binary long object)
    - per memorizzare immagini
  - cblob (character long object)
    - per memorizzare testi

Oracle data type

Oracle Built-in Data Types

Following table summarizes Oracle built-in data types.

Types	Description	Size
VARCHAR2(size [BYTE | CHAR])	Variable-length character string.	From 1 byte to 4KB.
NVARCHAR2(size)	Variable-length Unicode character string having maximum length size characters.	Maximum size is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify size for NVARCHAR2.
NUMBER [(p [, s])]	Number having precision p and scale s. Range of p : From 1 to 38. Ranges of s : From -84 to 127. Both precision and scale are in decimal digits.	A NUMBER value requires from 1 to 22 bytes.
FLOAT [(p)]	A FLOAT value is represented internally as NUMBER. Range of p : From 1 to 126 binary digits.	A FLOAT value requires from 1 to 22 bytes.
LONG	Character data of variable length up to 2 gigabytes, used for backward compatibility.	2^{31} -1 bytes
DATE	Valid date range : From January 1, 4712 BC, to December 31, 9999 AD. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter.	The size is fixed at 7 bytes.
BINARY_FLOAT	32-bit floating point number.	This data type requires 4 bytes.
BINARY_DOUBLE	64-bit floating point number.	This data type requires 8 bytes.
TIMESTAMP [(fractional_seconds_precision)]	This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It contains fractional seconds but does not have a time zone.	The size is 7 or 11 bytes, depending on the precision.
TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE	This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, and TIMEZONE_MINUTE. It has fractional seconds and an explicit time zone.	The size is fixed at 13 bytes.

Oracle data type

INTERVAL YEAR [(year_precision)] TO MONTH	Stores a period of time in years and months, where year_precision is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2.	The size is fixed at 5 bytes.
INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]	Stores a period of time in days, hours, minutes, and seconds, where day_precision is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2.	The size is fixed at 11 bytes.
RAW(size)	Raw binary data of length size bytes.	Maximum size is 2000 bytes
LONG RAW	Raw binary data of variable.	Size up to 2 gigabytes.
ROWID	The unique address (base 64 string representing) of a row in its table.	
UROWID [(size)]	The logical address of a row (base 64 string representing) of an index-organized table.	The maximum size and default is 4000 bytes.
CHAR [(size [BYTE I CHAR])]	Fixed-length character data of length size bytes or characters.	Maximum size is 2000 bytes or characters. Default and minimum size is 1 byte.
NCHAR[(size)]	Fixed-length character data of length size characters. The number of bytes can be up to two times size for AL16UTF16 encoding and three times size for UTF8 encoding.	Maximum size is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum size is 1 character.
CLOB	A character large object containing single-byte or multibyte characters.	Maximum size is (4 gigabytes - 1) * (database block size).
NCLOB	A character large object containing Unicode characters.	Maximum size is (4 gigabytes - 1) * (database block size). Stores national character set data.
BLOB	A binary large object.	Maximum size is 4 gigabytes.
BFILE	Contains a locator to a large binary file stored outside the database.	Maximum size is 4 gigabytes.

Oracle data type

Oracle Character Data Types

The CHAR data type specifies a fixed-length character string. If you insert a value that is shorter than the column length, then Oracle blank-pads the value to column length and if the value is too long for the column, then Oracle returns an error. Following data types are used for character data :

Types	Description	Range in characters
NCHAR	The NCHAR data type is a Unicode-only data type. When you create a table with an NCHAR column, you define the column length in characters.	The maximum column size allowed is 2000 bytes.
NVARCHAR2	The NVARCHAR2 data type is a Unicode-only data type. When you create a table with an NVARCHAR2 column, you supply the maximum number of characters it can hold.	he maximum column size allowed is : 32767 bytes if MAX_STRING_SIZE = EXTENDED 4000 bytes if MAX_STRING_SIZE = STANDARD
VARCHAR2	The VARCHAR2 data type specifies a variable-length character string. When you create a VARCHAR2 column, you supply the maximum number of bytes or characters of data that it can hold. Oracle subsequently stores each value in the column exactly as you specify it, provided the value does not exceed the maximum length of the column.	You must specify a maximum length for a VARCHAR2 column. This maximum must be at least 1 byte, although the actual string stored is permitted to be a zero-length string (''). You can use the CHAR qualifier, for example VARCHAR2(10 CHAR), to give the maximum length in characters instead of bytes.
VARCHAR	Do not use the VARCHAR data type. Use the VARCHAR2 data type instead. Although the VARCHAR data type is currently synonymous with VARCHAR2.	

Identificatori utente

- Devono iniziare con una lettera
- Non più di 30 caratteri
- Possono contenere $, _, e #
- Se si usano altri caratteri vanno chiusi tra virgolette doppie (ad esempio "prova&vai")
- Non devono contenere reserved word (ad esempio prova_id)

- Altrimenti segnalazione di errore

  ORA-00904: invalid identifier

CREATE DOMAIN

- create domain nomedominio as dominio_preesistente [default] [vincoli]
  - per creare sottoinsiemi di valori utili per i controlli

- Esempio:
  - create domain voto as smallint default null
    check (value >=18 and value <= 30);

- il nuovo dominio ed il dominio di partenza sono compatibili
- i valori del nuovo dominio devono rispettare i vincoli indicati nella definizione

Creazione di Tabelle

```
create table nome (
    nattr dom [valoredef] [vincoli]
    {,nattr dom [valoredef] [vincoli]}
    [altri vincoli] );

create table dip (
    nome varchar(20),
    citta varchar(20));
```

Vincoli intrarelazionali

- not null (su singoli attributi)
- unique: permette di definire un insieme di attributi come superchiave
  - singolo attributo:
    - unique dopo la specifica del dominio
  - più attributi:
    - unique (Attributo,...,Attributo)
- primary key: definizione della chiave primaria
  - implica not null e unique
  - sintassi come per unique
- check (condizione)

Primary Key

- Indica che l'attributo (o gli attributi) rappresenta una chiave primaria per la tabella.
  - Può essere espresso una sola volta per tabella
- create table dip (
  nome varchar(20) primary key,
  citta varchar(20));
- create table dip (
  nome varchar(20),
  citta varchar(20),
  primary key(nome));

Dare nomi ai vincoli

- Si possono dare dei nomi ai vincoli:
  constraint nome_vincolo vincolo

- create table dip (
  nome varchar(20),
  citta varchar(20),
  constraint pk_dip primary key (nome));

- Per migliorare la comprensione delle diagnostiche del DBMS in presenza di errori

Primary Key

- Se la primary key è composta da due attributi:

- create table impiegato (
  nome varchar(20),
  cognome varchar(20),
  primary key (cognome, nome));

- create table impiegato (
  nome varchar(20),
  cognome varchar(20),
  constraint pk_imp primary key (cognome, nome));

Unique

- Indica nella tabella non possono esistere due righe con valori identici sugli attributi specificati
- create table impiegato (
  cognome varchar(20) unique,
  nome varchar(20));

- create table impiegato (
  cognome varchar(20),
  nome varchar(20),
  unique(cognome, nome));

- create table impiegato (
  cognome varchar(20) ,
  nome varchar(20),
  constraint unq_cog_nome unique(cognome, nome));

Nota

```
create table Impiegato
( ... nome character(20)       not null,
      cognome character(20) not null,
      unique (cognome, nome),
…. )
```

- è diverso da:

```
create table impiegato
( ... nome character(20)       not null unique,
      cognome character(20) not null unique,
….. )
```

Not Null

- indica che l'attributo non può assumere valore NULL.
- create table dip (
  nome varchar(20) not null,
  citta varchar(20))

- create table dip (
  nome varchar(20),
  citta varchar(20),
  not null(nome))

- create table dip (
  nome varchar(20),
  citta varchar(20),
  constraint nn_nome not null(nome))

Nota

- I vincoli posso essere combinati

```
create table impiegato (
matricola  char(4),
cognome  varchar(20) not null,
nome       varchar(20),
constraint un_cog unique(cognome),
constraint pk_matr primary key(matricola));
```

Valore iniziale

- default
  - Si assegna un valore iniziale

stipendio integer default 1.000 not null

Nel caso di NULL sono equivalenti:

stipendio integer default null

stipendio integer null

Vincoli Intrarelazionali generici

- CHECK:
  - Utilizzato per specificare un vincolo di tupla generico

```
create table imp
(     matricola          char(4),
      cognome            varchar(20) not null,
      nome               varchar(20) not null,
      stipendio          integer default 1000 not null,
constraint un_cog        unique(cognome,nome),
constraint pk_matr       primary key(matricola),
constraint imp_stip_min check(stipendio > 800)
)
```

Operatori

- IN
  - appartenenza a un insieme

    check (sede in ('S02','S03'))

  - NOT IN per escludere valori

    check (sede not in ('Napoli','Roma','Milano'))

- between
  - Compreso in un intervallo

    check (stipendio between 1300 and 2000)

Vincoli Interrelazionali

- In SQL esiste il vincolo foreign key
  - detto anche vincolo di riferimento

- Crea il legame tra
  - l'attributo della tabella corrente (interna o referente)
  - e un attributo di un'altra tabella (esterna o riferita)
    - tipicamente la chiave primaria di questa
    - in alcuni casi basta che l'attributo della tabella esterna sia definito unique

Vincoli Integrità Referenziale

- Date le due tabelle

  dipartimento (nome, citta)

  impiegato(matricola, cognome)

- Dipartimento ⇔ tabella esterna o riferita
  - deve esistere all'atto della definizione del vincolo la tabella dipartimento

- Impiegato ⇔ tabella interna o referente
  - in essa viene definito il vincolo mediante l'aggiunta di un campo dello stesso tipo della chiave della tabella riferita

    impiegato(matricola, cognome, cod_dipartimento)

- La violazione può essere a livello di tabella
  - sulla tabella interna (per inserimento e modifica)
    - Inserimento di un dipartimento che non esiste
  - su quella esterna (cancellazione e modifica)
    - Eliminazione di un dipartimento a cui partecipano impiegati

Vincoli Integrità Referenziale

- Violazione sulla tabella referente:
  - L'operazione viene semplicemente impedita
- Violazione sulla tabella riferita:

create tabella_referente(
……..
chiave_referente varchar,
……..
constraint nome_vincolo foreign key (chiave_referente) references tabella_riferita(chiave_riferita) on [delete/update] [politica di reazione]);

  - no action operazione impedita
  - restrict come no action
  - cascade operazione ammessa ma si cancella attributo esterno
  - set null operazione ammessa ma si pone attributo esterno a null
  - set default operazione ammessa ma si pone attributo esterno a default

Vincoli Integrità Referenziale

- create table dipartimento (
  codedip char(4) primary key,
  nome varchar(20));

- create table imp(
  nome char(20),
  cognome char(20),
  cod_dipartimento char(4) references dipartimento(codedip)
  on delete set null on update cascade);

- create table imp (
  nome char(20),
  cognome char(20),
  dipartimento char(4),
  constraint fk_imp_dip foreign key (cod_dipartimento) references
  dipartimento(codedip) on delete set null on update cascade);

Indici

- Gli indici rappresentano uno strumento prezioso per velocizzare l'accesso ai dati.
- Per crearli
  create [unique] index nomidx on nometable (lista attributi)

  create unique index nomidx on anagrafica(cognome, nome)
  - L'ordine in cui sono dati gli attributi è importante
- unique indica che nella tabella non sono possibili righe che assumono lo stesso valore su (lista attributi) (che quindi è una chiave eventualmente non minimale)
- Per eliminarli.
  - drop index nomidx

Modifica di una tabella

alter table nome_tabella [specifica della modifica]

- Aggiungere un attributo
  - alter table n_t add attributo tipo_attributo

    alter table impiegato add sesso char(1)

- Rimuovere un attributo
  - alter table n_t drop attributo

    alter table impiegato drop sesso

Modifica di una tabella

- Modificare un attributo
  - alter table n_t modify attributo tipo_attributo

    alter table impiegato modify sesso bit

- Disattivare un attributo
  - alter table n_t set unused attributo

    alter table impiegato set unused sesso
  - la colonna non appare più nella struttura della tabella, né nelle query.
  - quando siamo sicuri della cancellazione, ad esempio quando la tabella non è eccessivamente utilizzata da altri utenti, possiamo anche procedere alla sua cancellazione

    alter table impiegato drop unused sesso

Modifica di una tabella

- Aggiungere un vincolo
  - alter table n_t add constraint vincolo

    alter table impiegato add
    constraint tipo_sesso check (sesso in (’M’,’m,’F’,’f’))

- Rimuovere un vincolo
  - alter table n_t drop constraint nome_vincolo

    alter table impiegato drop constraint tipo_sesso

- Per i vincoli senza nome
  - alter table n_t drop vincolo

    alter table impiegato drop unique(cognome,nome)

Modifica di una tabella

- Per rinominare una colonna

```
alter table n_t
rename column column_name to new_column_name
```

- Per rinominare un vincolo

```
alter table n_t rename constraint name to new_name
```

- L'intera tabella

```
alter table n_t rename new_name
```

Eliminare una tabella

drop table

- Se la tabella si trova in un vincolo di foreign key come riferita il dbms genera un errore e la tabella non viene eliminata

drop table cascade constraints

- La tabella viene eliminata dopo che tutti i vincoli di chiave esterna che fanno riferimento ad essa sono stati eliminati

Un controllo

drop table name if exists ……..

- Se esiste viene eliminata
- Se non esiste non scatta la condizione di errore

drop table name ……..

- Se non esiste scatta la condizione di errore

Creare un DB schema

create schema schema_name
[authorization owner_name]

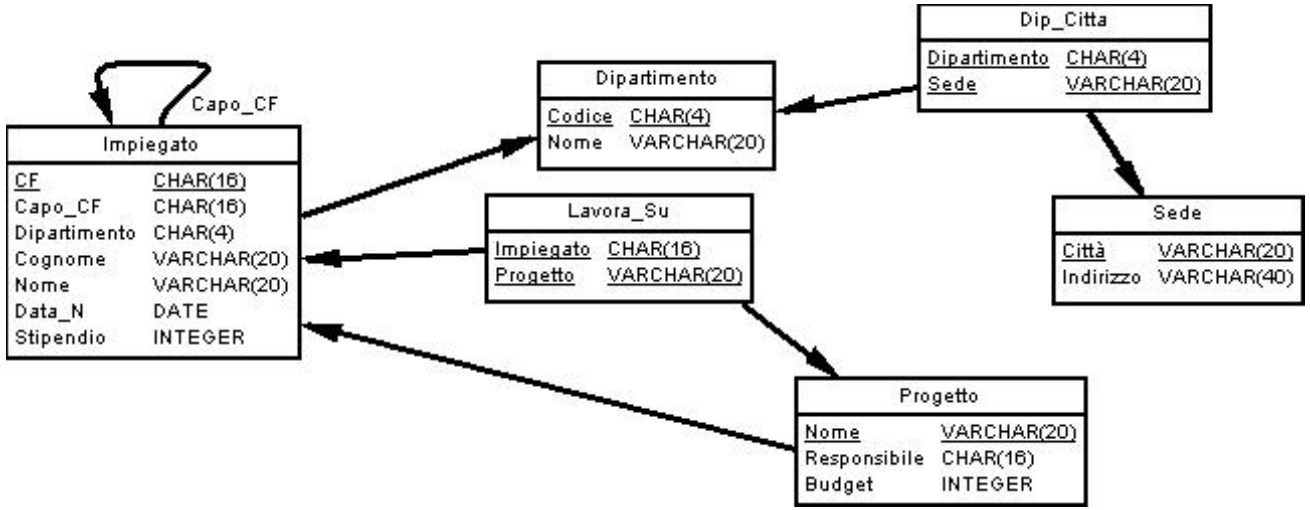
- Per creare un nuovo DB contenente:
  - Tabelle
  - Viste
  - Procedure
  - Trigger
  - Indici

- E per identificarne gli oggetti, ad esempio una tabella:
  schema_name.table_name

- Se owner_name non è specificato, il proprietario è l'utente corrente

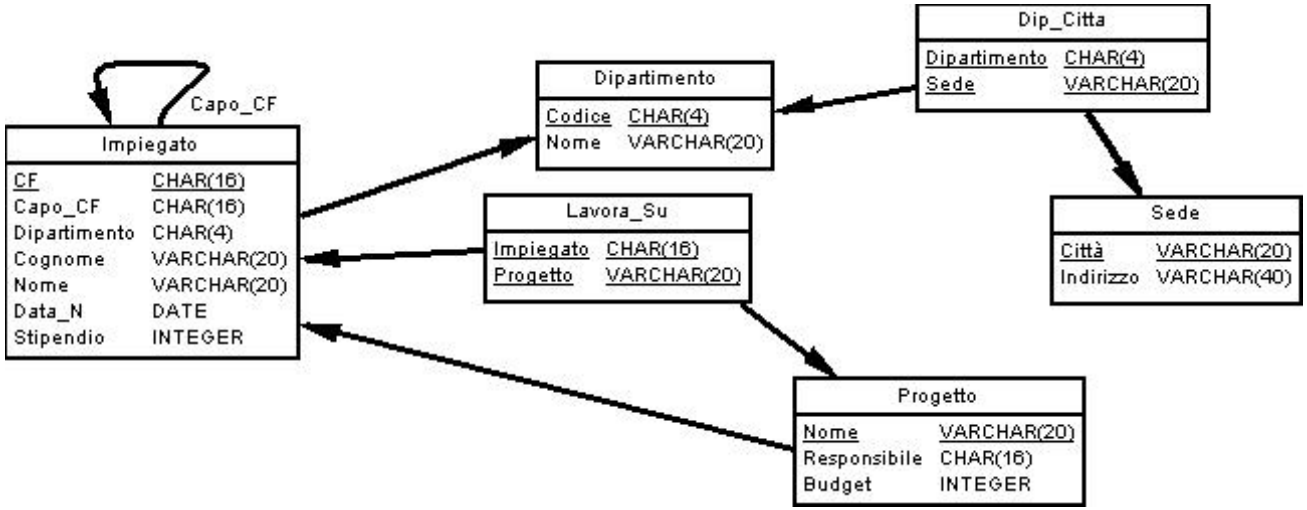
Un esercizio

- Provare in oracle a realizzare:



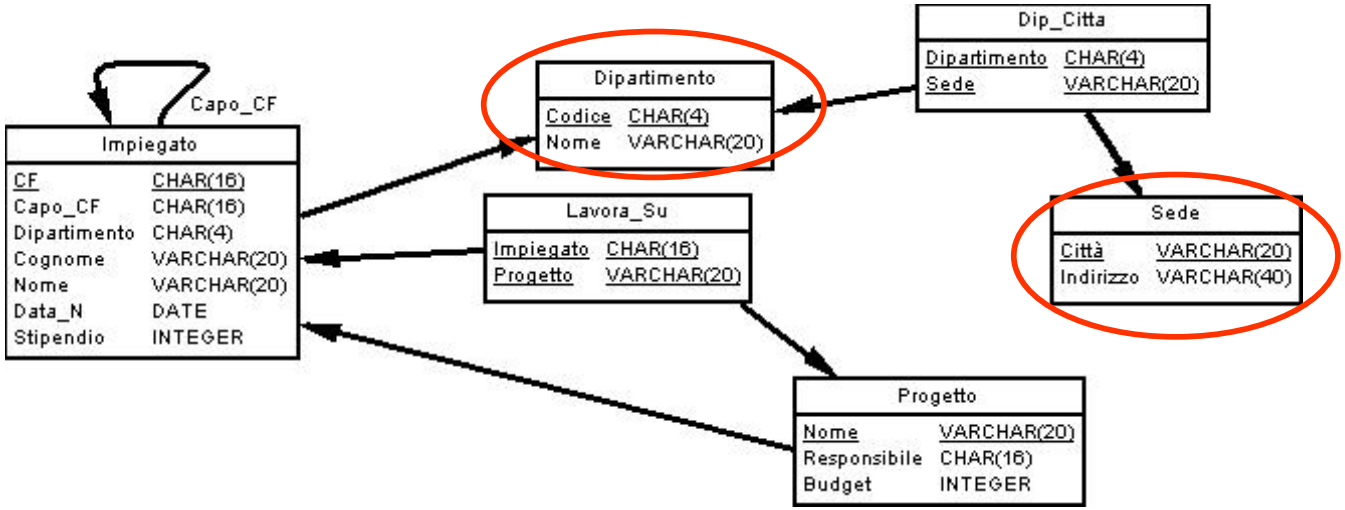
- Per prima cosa riempiamo tabella dei vincoli di chiave esterna

Tabella referente	Chiave esterna	Tabella riferita	Chiave
impiegato	dipartimento	dipartimento	codice
dip_citta	dipartimento	dipartimento	codice
lavora_su	impiegato	impiegato	cf
lavora_su	progetto	progetto	nome
progetto	responsabile	impiegato	cf
dip_citta	sede	sede	citta



- Possiamo procedere con la creazione delle tabelle seguendo un ordine ben preciso.

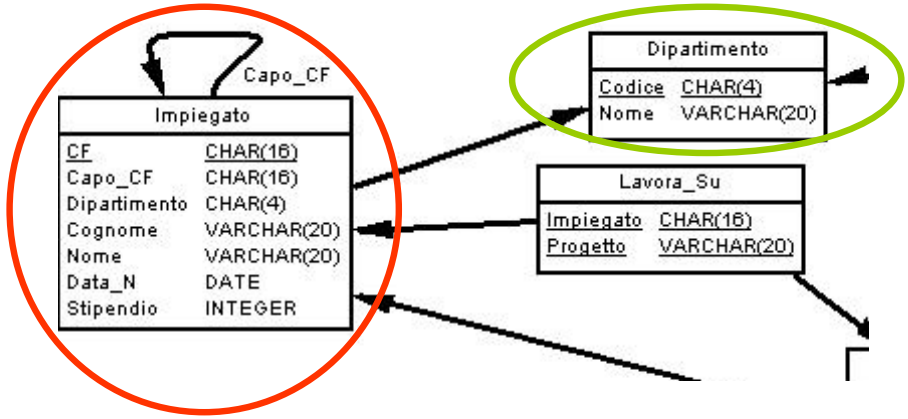
Prima le tabelle riferite



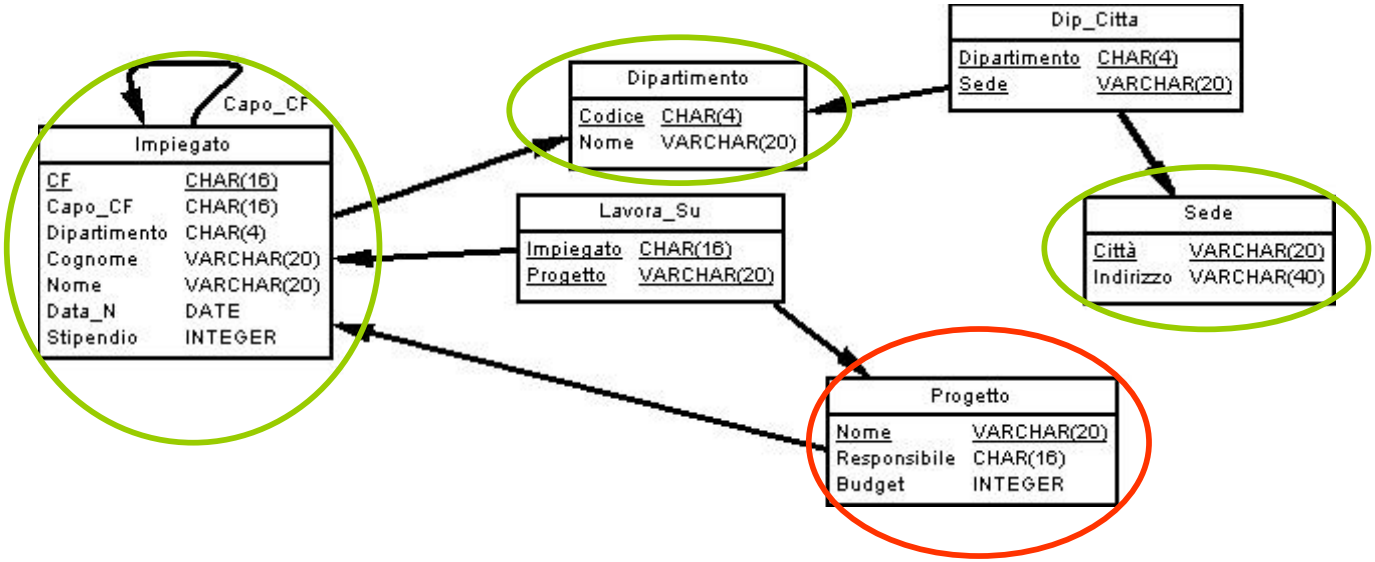

```
create table dipartimento (
codice    char(4) not null,
nome      varchar(20) default null,
constraint pk_dipartimento primary key (codice));

create table sede (
citta       varchar(20) not null,
indirizzo varchar(40) default null,
constraint pk_sede primary key (citta));
```

Dopo le tabelle referenti

```
create table impiegato (
cf char(16),
capo_cf char(16) default null,
dipartimento char(4) default null,
cognome  varchar(20) default null,
nome varchar(20),
data_n  date,
stipendio integer not null
      constraint ckc_stipendio_impiegat check (stipendio between '800' and '3000'),
constraint pk_impiegato primary key (cf),
constraint fk_capo_is_impiegato foreign key (capo_cf)
   references impiegato (cf)
   on delete no action on update no action,
constraint fk_impiegato_esiste_dipartimento foreign key (dipartimento)
   references dipartimento (codice)
   on delete restrict on update restrict );
```

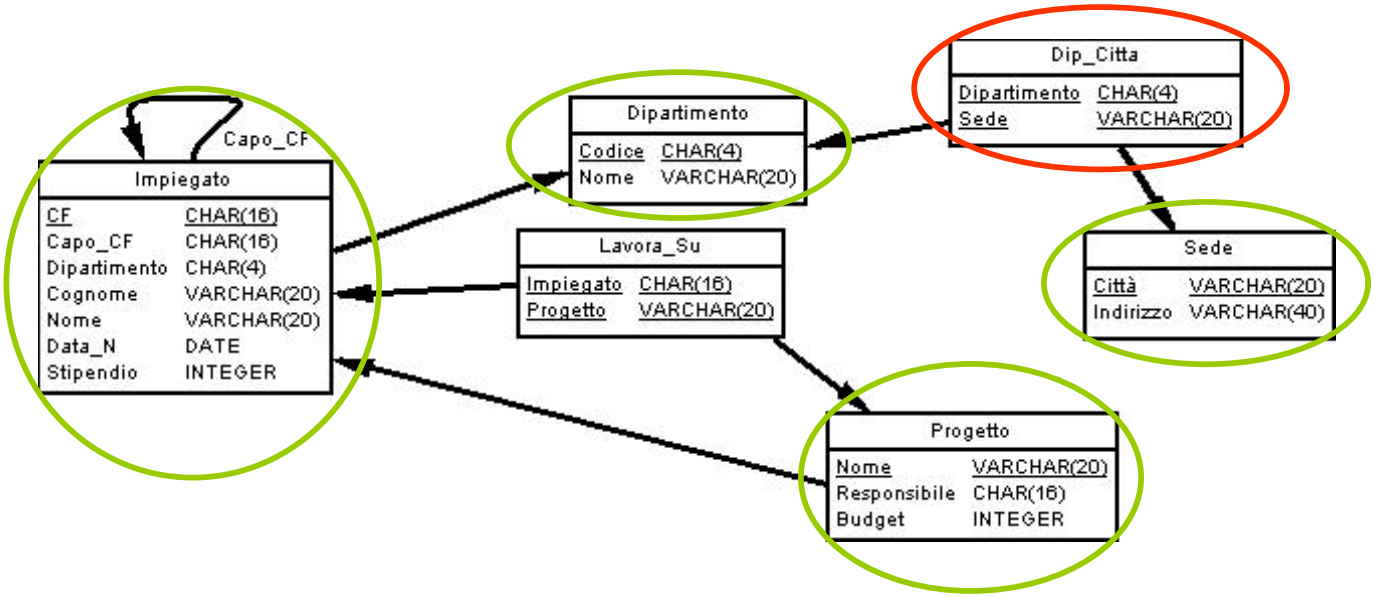


E così nell'ordine



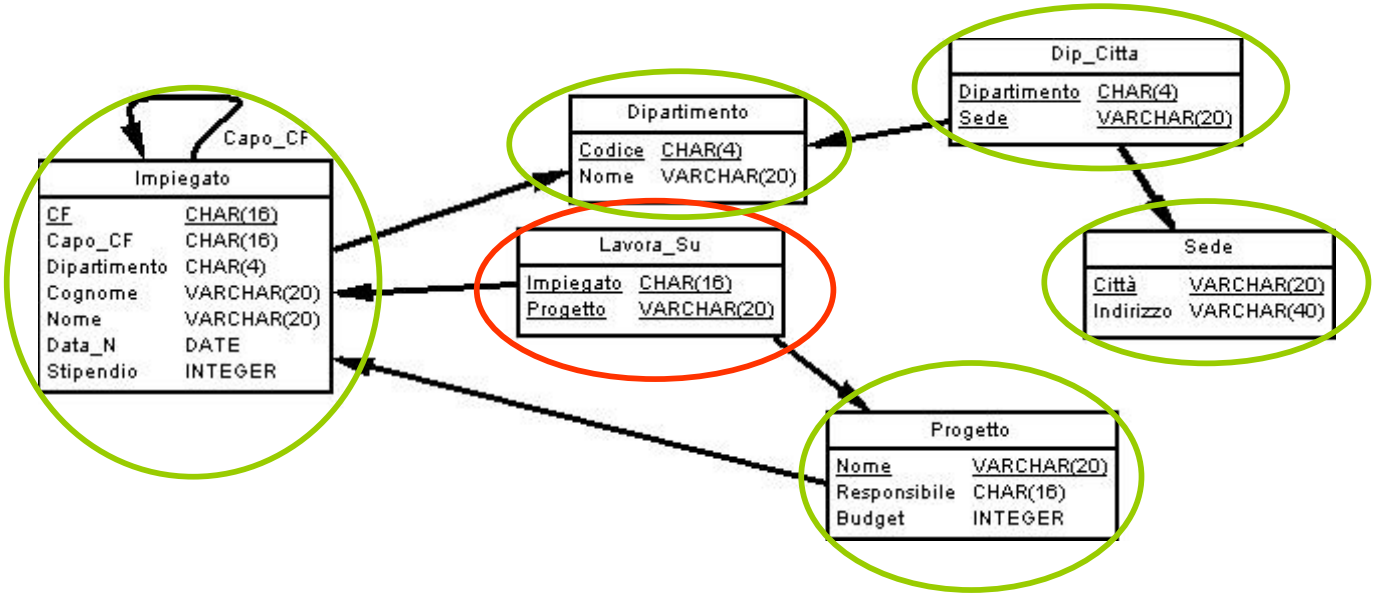

```
create table progetto (
nome varchar(20) not null,
responsabile char(16) default null,
budget integer default null,
constraint pk_progetto primary key (nome),
constraint fk_progetto_resp_impiegato foreign key (responsabile)
   references impiegato (cf)
   on delete restrict on update cascade);
```

Esempio




```
create table dip_citta (
dipartimento  char(4) not null,
sede  varchar(20) not null,
constraint pk_dip_citta primary key (dipartimento, sede),
constraint fk_dip_citta_dipartimento foreign key (dipartimento) references dipartimento (codice)
                on delete restrict on update restrict,
constraint fk_dip_citta_sede foreign key (sede) references sede (citta)
                on delete restrict on update restrict);
```

Esempio




```
create table lavora_su (
impiegato char(16) not null,
progetto varchar(20) not null,
constraint pk_lavora_su primary key (impiegato, progetto),
constraint fk_lavora_s_reference_impiegat foreign key (impiegato)
   references impiegato (cf) on delete restrict on update restrict,
constraint fk_lavora_s_reference_progetto foreign key (progetto)
   references progetto (nome) on delete restrict on update restrict
);
```

Come ovviare all'ordine seguito

- Creare le tabelle senza i vincoli di foreign key
  - In un ordine qualsiasi
- Aggiungerli con alter table add constraint …..
  - In un ordine qualsiasi

Proviamo

```
create table dipartimento (codice  char(4) not null, nome varchar(20) default null,
constraint pk_dipartimento primary key (codice));

create table sede (citta varchar(20) not null, indirizzo varchar(40) default null,
constraint pk_sede primary key (citta));

create table impiegato (cf char(16), capo_cf char(16) default null, dipartimento char(4) default null,
cognome  varchar(20) default null, nome varchar(20), data_n  date, stipendio integer not null,
constraint ckc_stipendio_impiegat check (stipendio between '800' and '3000'),
constraint pk_impiegato primary key (cf));

create table progetto (nome varchar(20) not null, responsabile char(16) default null, budget integer
default null,
constraint pk_progetto primary key (nome));

create table dip_citta (dipartimento  char(4) not null, sede  varchar(20) not null,
constraint pk_dip_citta primary key (dipartimento, sede));

create table lavora_su (impiegato char(16) not null, progetto varchar(20) not null,
constraint pk_lavora_su primary key (impiegato, progetto));
```

Aggiungiamo i vincoli

alter table impiegato add constraint fk_capo_is_impiegato foreign key (capo_cf) references impiegato (cf) on delete restrict on update restrict;

alter table impiegato add constraint fk_impiegato_esiste_dipartimento foreign key (dipartimento) references dipartimento (codice) on delete restrict on update restrict;

alter table progetto add constraint fk_progetto_resp_impiegato foreign key (responsabile) references impiegato (cf) on delete restrict on update cascade;

alter table dip_citta add constraint fk_dip_citta_dipartimento foreign key (dipartimento) references dipartimento (codice) on delete restrict on update restrict;

alter table dip_citta add constraint fk_dip_citta_sede foreign key (sede) references sede (citta) on delete restrict on update restrict;

alter table lavora_su add constraint fk_lavora_s_reference_impiegat foreign key (impiegato) references impiegato (cf) on delete restrict on update restrict;

alter table lavora_su add constraint fk_lavora_s_reference_progetto foreign key (progetto) references progetto (nome) on delete restrict on update restrict;

La leggibilità

```
create table impiegato
(               cf char(16),
                capo_cf char(16) null,
                dipartimento char(4) default null,
                cognome  varchar(20) default null,
                nome varchar(20),
                data_n  date,
                stipendio integer not null
        constraint ckc_stipendio_impiegat check (stipendio between '800' and '3000'),
        constraint pk_impiegato primary key (cf),
        constraint fk_capo_is_impiegato foreign key (capo_cf)
                        references impiegato (cf) on delete restrict on update restrict,
        constraint fk_impiegato_esiste_dipartimento foreign key (dipartimento)
                        references dipartimento (codice) on delete restrict on update restrict);
);
```

Sequenza di frasi SQL

- Le frasi devono essere terminate dal punto e virgola

```
Frase 1;
Frase 2;
Frase 3;
```

Un anticipazione

```
create table sede
(                   citta varchar(20) not null,
                    indirizzo varchar(40) null,
          constraint pk_sede primary key (citta)
);
```

- Per inserire valori nella tabella

insert into nometab [(lista attributi)] values (lista valori)

insert into sede values (‘Napoli’,’Via Roma’)
insert into sede (indirizzo) values (’Via Roma’)

- Per vedere il contenuto della tabella

select [(lista attributi)|*] from nometab

select * from SEDE fa vedere tutta la tabella
select citta from SEDE fa vedere solo la colonna città

Il controllo degli accessi

- Creazione degli utenti
  - Per fissare chi fa cosa
  - L'utente che crea il db può fare tutto
- Assegnazione di privilegi P

$$\mathcal{P} = \langle \mathcal{R}, \mathcal{U}_1, \mathcal{U}_2, \mathcal{A}, \mathcal{T} \rangle$$

dove:

- $\mathcal{R}$ rappresenta la risorsa su cui è concesso il privilegio;
- $\mathcal{U}_1$ rappresenta l'utente che concede il privilegio;
- $\mathcal{U}_2$ rappresenta l'utente che ottiene il privilegio;
- $\mathcal{A}$ rappresenta l'insieme delle azioni che sono permesse sulla risorsa;
- $\mathcal{T}$ rappresenta l'autorizzazione concessa all'utente che riceve il privilegio di trasmettere lo stesso privilegio ad altri utenti.

I privilegi

- create — per definire nuove istanze di risorse
- drop — per rimuovere istanze di risorse
- update — per modificare una risorsa
- insert — per inserire in una tabella o vista
- delete — per eliminare in una tabella o vista
- select — per la visualizzazione dei valori delle tabelle
- resource — per creare, modificare e rimuovere risorse da uno schema
- connect — per premettere a un utente di connettersi al dbms
- all privileges — per fare tutto

Creare un utente

- Definizione di username e password

create user nome_utente identified by password

create user angelo identified by aglo20123@USS

Assegnazione privilegi

- **Per assegnarli**

grant privilegio$_1$,...,privilegio$_n$ **on** nomerisorsa **to** username **[with grant option]**

  - L'opzione **with grant option** consente all'utente di concedere privilegi

- **Per revocarli**

revoke privilegio$_1$,...,privilegio$_n$ **on** nomerisorsa **from** username [⟨ **restrict** | **cascade** ⟩]

  - restrict impedisce il REVOKE se l'utente ha dato dei privilegi
  - cascade revoa in cascata tutti i privilegi dell'utente e di quelli dati da chi da lui è stato attivato

Si possono creare dei ruoli

CREATE ROLE nomeruolo;

GRANT privilegio1,...,privilegioN **ON** NomeRisorsa1 **TO** nomeruolo [**WITH GRANT OPTION**];

….,

GRANT privilegio1,...,privilegioN **ON** NomeRisorsaN **TO** nomeruolo [**WITH GRANT OPTION**];

- Per assegnarli a un utente

GRANT nomeruolo **TO** username

O assegnarli a una risorsa

- A un intero schema

grant privilegio1,...,privilegion **on** nomeschema.* **to** nomeruolo [**with grant option**]

- O a una singola risorsa

grant privilegio1,...,privilegion **on** nomeschema.nomeris **to** nomeruolo [**with grant option**]

- L'utente a cui è assegnata la risorsa ne acquisisce i privilegi