

# Database

Persistenza dei dati

# Parte 1

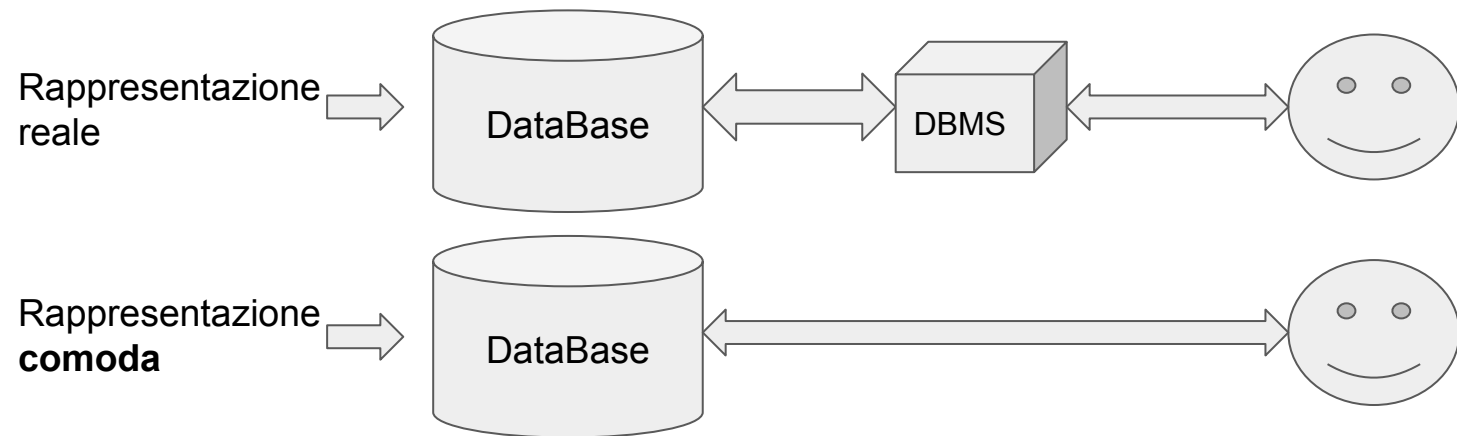


# Alcuni concetti



# Database e DBMS

- Sono delle collezioni di dati strutturati, organizzati e omogenei
- Fisicamente parlando questi dati si trovano su hardware non volatili
  - Di norma hard disk
- Per accedere e interagire con i dati vengono messi a disposizione software specifici chiamati DBMS (**DataBase Management System**)



# Operazione CRUD

Per poter usare un DB è necessario che i DBMS forniscono determinate operazioni:

- **Create/insert**
- **Read**
- **Update**
- **Delete**

Queste permettono di sfruttare a pieno un database e permettere la persistenza dei dati.

# I modelli logici

La struttura del database e i collegamenti dei dati devono seguire uno schema logico:

- Gerarchico
- Reticolare
- **Relazionale**
- Ad oggetti
- NoSQL
- Semantica

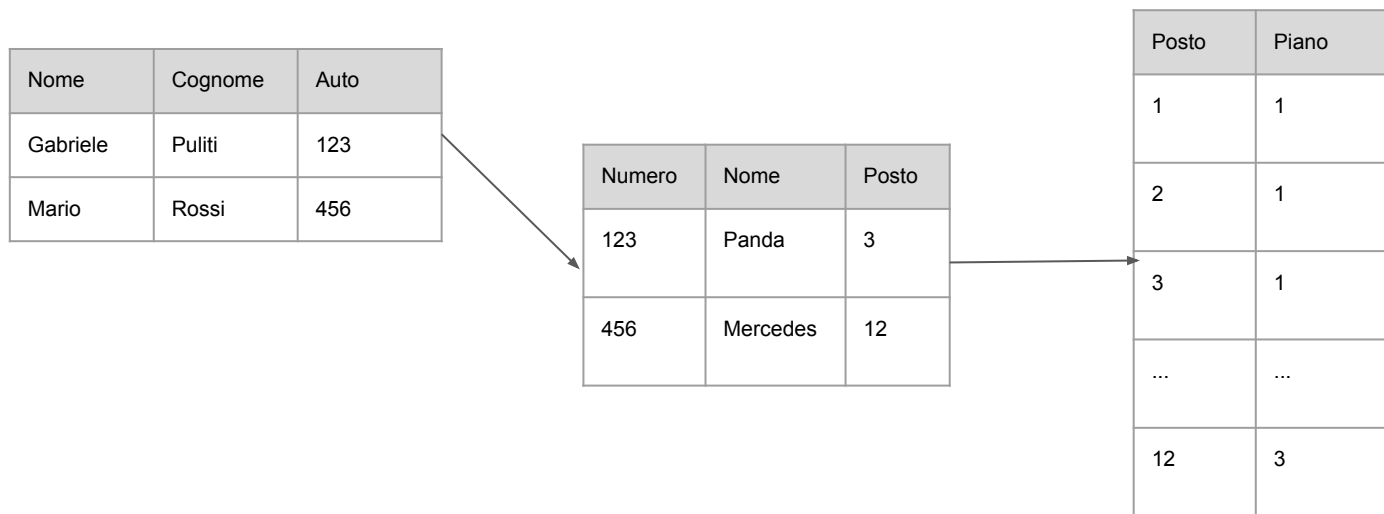
Curiosità #1 -> Il modello relazionale è stato inventato negli anni 70

Curiosità #2 -> NoSQL significa Not Only SQL e non NO SQL

# Modello relazionale

Gli elementi di questo modello sono tabelle di dati rappresentati come relazioni.

Consente di rappresentare l'informazione in modo reale e logico.



Il DBMS relativo a questo modello viene anche definito RDBMS, dove la R rappresenta Relational.

# La struttura del modello relazionale

Gli elementi fondamentali di questa struttura sono:

- attributi
- tipo e dominio di valori
- valore
- tupla/record

Nome	Cognome	Auto
Gabriele	Puliti	123
Mario	Rossi	456



# Relazioni

Tabella è la parola usata per indicare la relazione formata dagli elementi della matrice contenente informazioni comuni.

Nome	Cognome	Auto
Gabriele	Puliti	123
Mario	Rossi	456

Numero	Nome	Posto
123	Panda	3
456	Mercedes	12

Posto	Piano
1	1
2	1
3	1
...	...
12	3

Ogni attributo della tabella ha un nome **univoco**.

# Primary key / Chiave primaria

Il valore di una tupla associato a un determinato attributo può essere:

- **ripetibile** = che può essere usato sullo stesso attributo su più righe
- **univoco** = non può ripetersi nello stesso attributo su più righe
- **NULL** = valore nullo

Solo gli attributi che stabiliscono un **vincolo di unicità** possono formare una chiave.

Un attributo viene definito **chiave primaria** se e solo se ogni valore è unico nella tabella.

Con una chiave primaria è possibile fare riferimento univocamente ad una tupla.

# Foreign key / Vincolo di integrità referenziale

È possibile stabilire relazioni extra-tabellari utilizzando associazioni tramite le chiavi primarie.

In questo esempio abbiamo che la tabella Auto ha come **chiave primaria** l'attributo Numero e la tabella Persone utilizza questo valore per referenziare un suo record con quello della tabella Auto:

Nome	Cognome	Auto
Gabriele	Puliti	123
Mario	Rossi	456

Numero	Nome	Posto
123	Panda	3
456	Mercedes	12

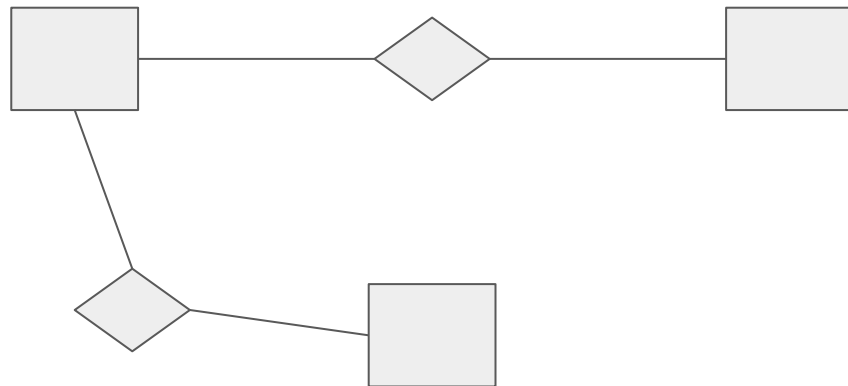
# Progettare un db



# Passi della progettazione

Per implementare un db è necessario attraversare diversi passi:

- Progettazione concettuale produce uno schema concettuale



# Passi della progettazione

Per implementare un db è necessario attraversare diversi passi:

- Progettazione concettuale produce uno schema concettuale
- Progettazione logica produce uno schema logico

attributo1	attributo2	attributo3
valore1	valore2	valore3
valore4	valore5	valore6

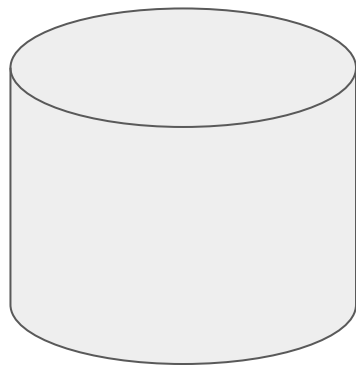
per facilità

nome\_tabella(attributo1,attributo2,attributo3)

# Passi della progettazione

Per implementare un db è necessario attraversare diversi passi:

- Progettazione concettuale produce uno schema concettuale
- Progettazione logica produce uno schema logico
- Progettazione fisica produce uno schema fisico



# Esempio: Parco macchine

L'esempio che abbiamo usato fino a ora rappresenta un modello relazionale associabile a un parco **macchine** in cui si vuole tenere memoria sia del **proprietario** sia del **posto occupato** dall'auto.

Applicazione schema logico:

proprietario(nome, cognome, auto)

auto(numero, nome, posto)

locazione(posto, piano)

proprietario(auto) -> auto(numero)

auto(posto) -> locazione(posto)



# Esercizi parte 1



# Esercizio 1: Attività commerciale

Un'attività commerciale ha necessità di tenere traccia dei prodotti che ha nei suoi magazzini:

- Un prodotto viene identificato da un numero crescente
- È necessario sapere in che magazzino il prodotto si trova
- Il magazzino ha un numero identificativo e un indirizzo

# Esercizio 1: Soluzione

Possibile soluzione:

magazzino(identificativo, indirizzo)

prodotto(numero, nome, magazzino)

prodotto(magazzino) -> magazzino(identificativo)

## Esercizio 2: Università

L'università vuole tenere traccia delle informazioni relative a:

- professori con un nome e cognome
- corsi tenuti dai professori in una determinata aula
- Studenti che seguono i corsi

## Esercizio 2: Soluzione

Una possibile soluzione:

professore(nome, cognome, corso)

corso(nome, aula)

studenti(nome, cognome, corso)

professore(corso) -> corso(nome)

studenti(corso) -> corso(nome)

# Esercizio 2: Problematiche

Ci potrebbero essere dei problemi:

- Più professori e studenti con lo stesso nome e cognome
- Un professore può tenere più corsi
- Più studenti possono seguire più corsi

# Parte 2



# Progettazione concettuale





# Progettazione concettuale

Rappresentare le specifiche della realtà di interesse.

Analizzare i requisiti forniti e produrre un modello concettuale dei dati.

Lo standard usato per produrre uno schema concettuale è il modello entità relazione, chiamato anche modello e-r.




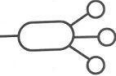




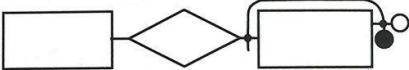
# Modello E-R

Fornisce una serie di costrutti utili a descrivere la realtà di interesse in una maniera facile da comprendere.




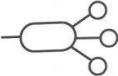


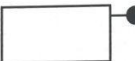

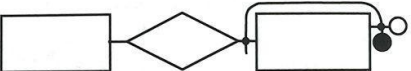
Prescinde dai criteri di organizzazione dei computer.

Ogni costrutto ha una rappresentazione grafica associata, questa rappresentazione consente di definire un diagramma che sarà lo schema concettuale utile per la progettazione concettuale.

# Costrutti e Rappresentazioni

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

# Esempio: entità

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

Prendiamo l'esempio 2 e usiamo questa rappresentazione passo passo, iniziando con la rappresentazioni delle entità:

PROFESSORI

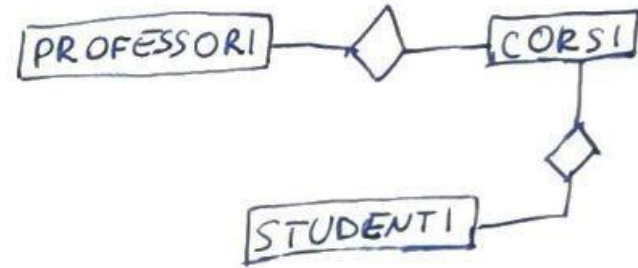
CORSI

STUDENTI




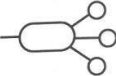




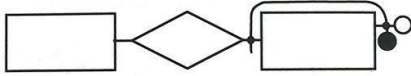
## Esempio: relazioni

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	
Identificatore esterno	

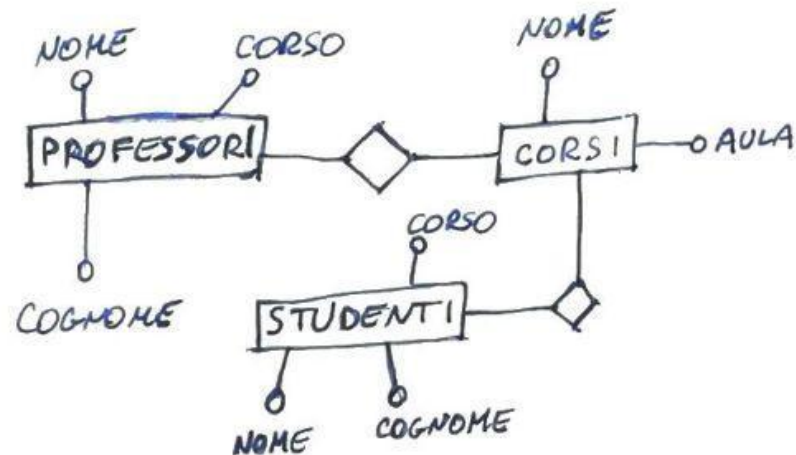
Inseriamo le relazioni tra le entità:






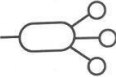



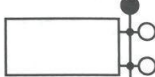
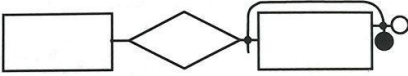
# Esempio: attributi

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

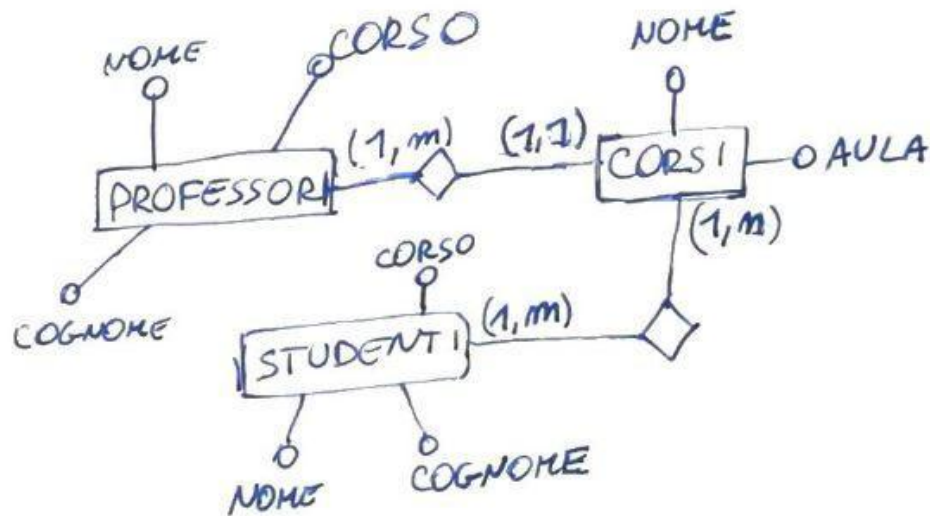
Inseriamo gli attributi delle entità:






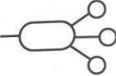




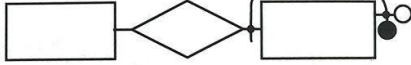
# Esempio: cardinalità

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

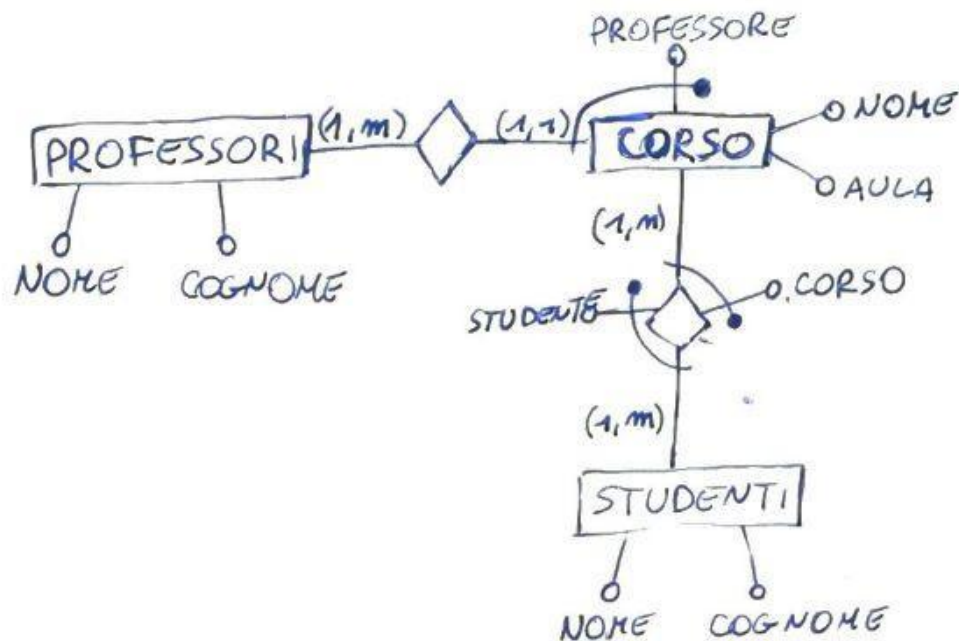
Inseriamo le cardinalità delle relazioni:



# Esempio: identificatori esterni




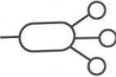




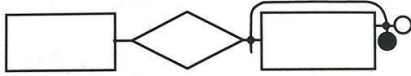
Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

Definiamo gli identificatori esterni e modifichiamo gli attributi della relazione:

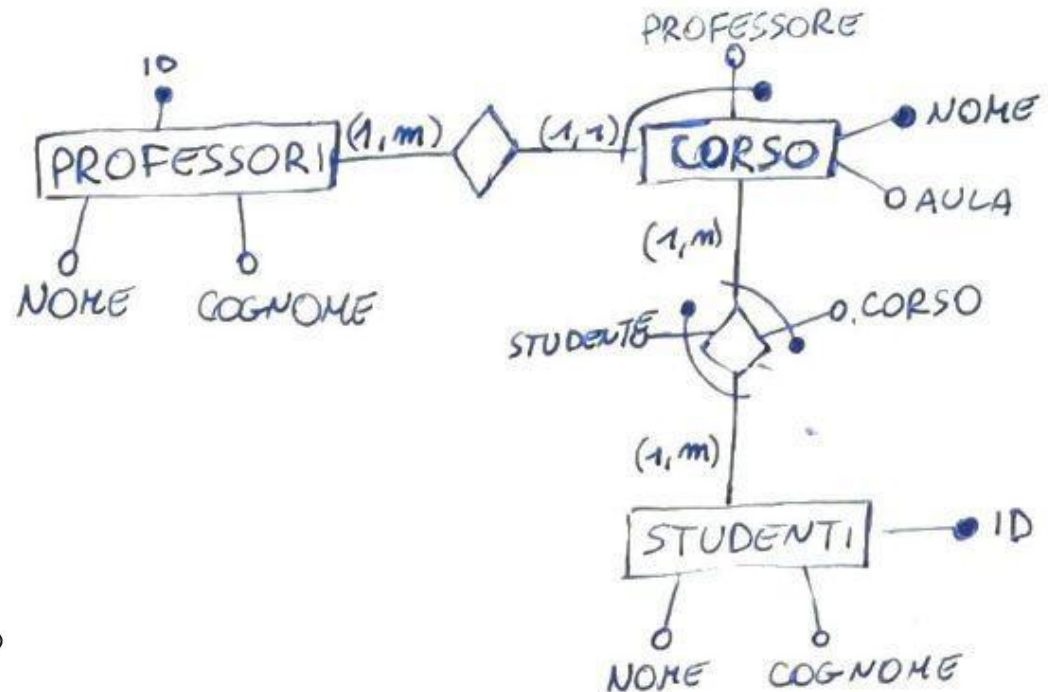




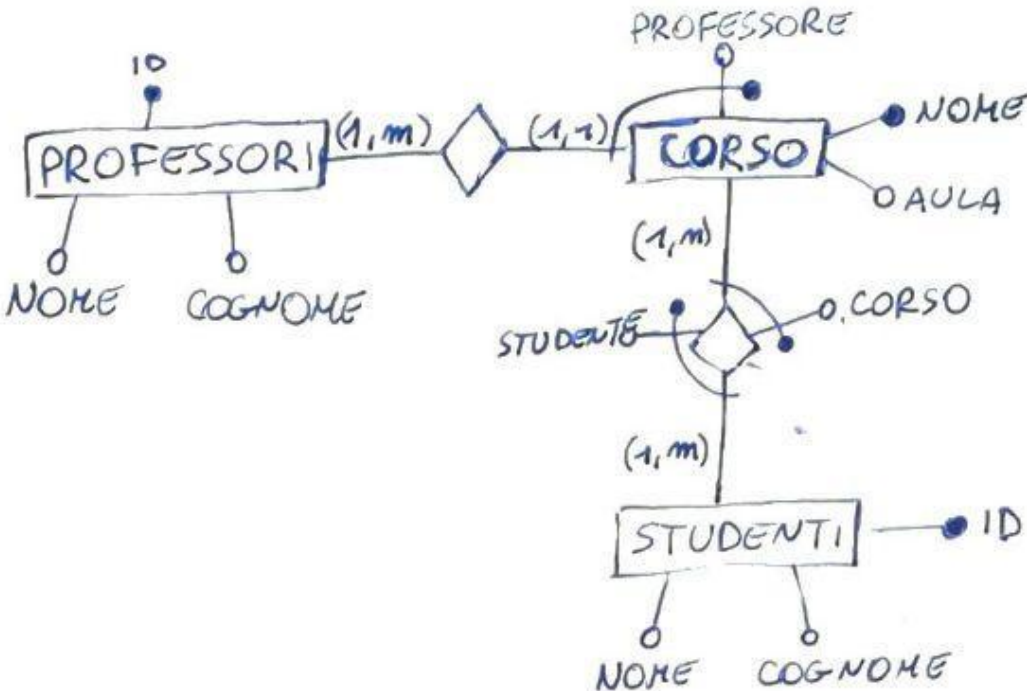
# Esempio: identificatori interni

Costrutti	Rappresentazione grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità di relazione	
Cardinalità di attributo	
Identificatore interno	 
Identificatore esterno	

Definiamo gli identificatori interni:



# Esempio: schema logico



Trasformiamo questo schema concettuale in uno schema logico che possiamo usare per costruire un reale database:

```
professori(id,nome,cognome)
corso(nome,aula,professore)
studenti(id,nome,cognome)
studenti_corso(studente,corso)
```

vincoli di integrità referenziali:

```
corso(professore)->professori(id)
studenti_corso(studente)->studenti(id)
studenti_corso(corso)->corso(nome)
```

# Progettazione concettuale: metodologia generale

- Analisi requisiti:
  - glossario termini
  - eliminare ambiguità
  - raggruppare requisiti in insiemi omogenei
- Passo base: individuare concetti più rilevanti e rappresentarli in uno schema
- Passo decomposizione: dividere schemi in parti, non sempre è necessario
- Passo iterativo: si ripete
  - raffinare concetti con attributi o relazioni
  - aggiungere concetti per specifiche non ancora descritte
- Passo integrazione: inglobare i sottoschemi in uno generale, solo se esistono
- Analisi qualità:
  - verifica della correttezza dello schema ed eventuale ristrutturazione
  - verifica della completezza ed eventuale ristrutturazione
  - verifica minimalità ed eventuale ristrutturazione
  - verifica leggibilità ed eventuale ristrutturazione

# Esercizi parte 2

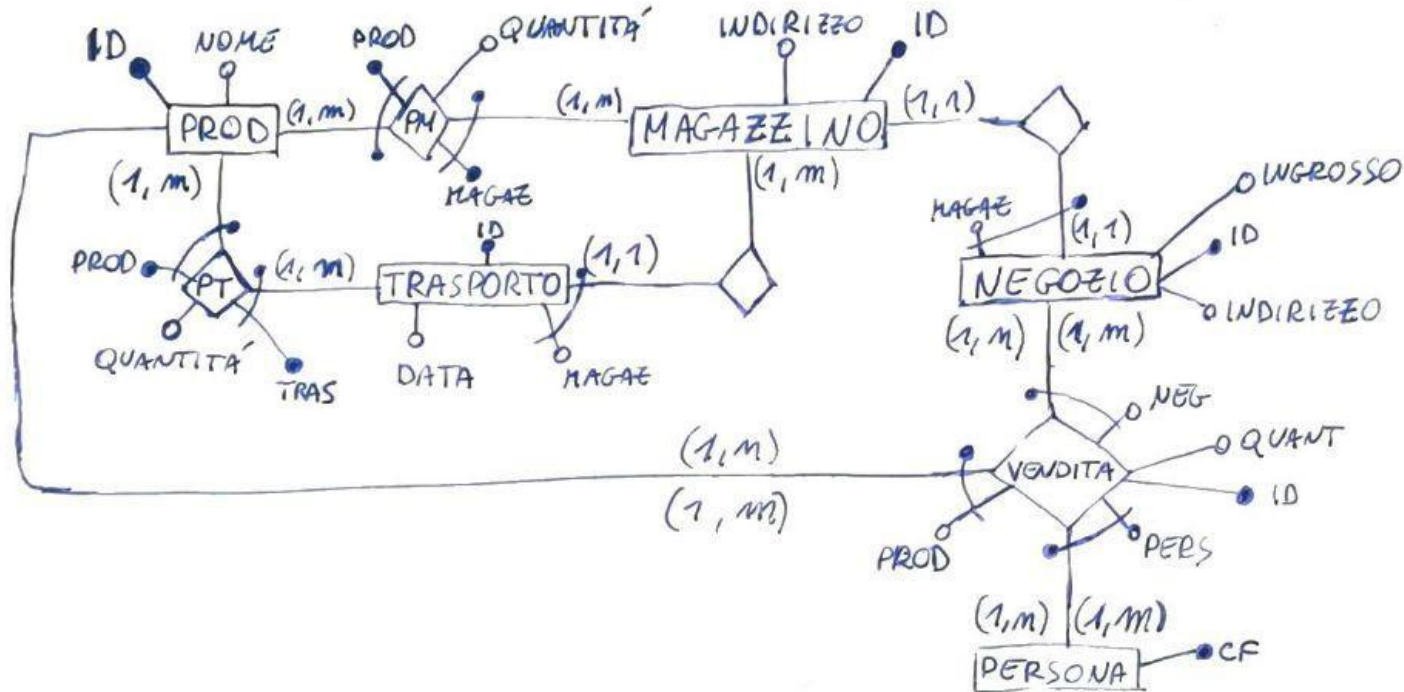


# Esercizio 3: Attività commerciale

Una attività commerciale ha necessità di tenere traccia dei prodotti che ha nei suoi magazzini:

- Un prodotto viene identificato da un numero crescente
- È necessario sapere in che magazzino il prodotto si trova
- Il magazzino ha un numero identificativo e un indirizzo
- Si possono trovare più prodotti su più magazzini diversi
- Si deve tenere traccia degli spostamenti dei prodotti tra un magazzino e l'altro
- Il magazzino fa parte di un negozio
- Il negozio vende a persone fisiche
- Teniamo traccia di tutti gli acquisti fatti dal negozio
- La persona fisica può anche acquistare all'ingrosso direttamente dal magazzino

# Soluzione esercizio 3



## Esercizio 4: Aeroporto

- Ogni aereo ha un numero identificativo, un aeroporto di partenza e un numero di passeggeri
- Ogni passeggero ha nome, cognome e un identificativo
- L'aeroporto ha un numero di gate e ogni aereo si ferma in uno specifico gate in orari e date stabilite prima di ripartire
- Sono presenti degli hangar dove gli aerei possono posteggiare
- Ogni hangar ha un numero e una posizione

## Esercizio 5: Biblioteca

I lettori che frequentano la biblioteca hanno una tessera su cui è scritto il nome e l'indirizzo ed effettuano richieste di prestito per i libri che sono catalogati nella biblioteca. I libri hanno un titolo, una lista di autori e possono esistere in diverse copie. Tutti i libri contenuti nella biblioteca sono identificati da un codice. A seguito di una richiesta, viene dapprima consultato l'archivio dei libri disponibili (cioè non in prestito). Se il libro è disponibile, si procede alla ricerca del volume negli scaffali; il testo viene poi classificato come in prestito. Acquisito il volume, viene consegnato al lettore, che procede alla consultazione. Terminata la consultazione, il libro viene restituito, reinserito in biblioteca e nuovamente classificato come disponibili. Per un prestito si tiene nota degli orari e delle date di acquisizione e di riconsegna.



# Esercizio 6: Campionato di calcio

Per ogni partita, descrivere il girone e la giornata in cui si è svolta, il numero progressivo nella giornata (per esempio prima partita, seconda partita ecc.), la data, con giorno, mese, anno, le squadre coinvolte nella partita, con nome, città della squadra e allenatore, e infine per ciascuna squadra se ha giocato in casa. Si vogliono conoscere i giocatori che giocano in ogni squadra con i loro nomi e cognomi, la loro data di nascita e il loro ruolo principale. Si vuole conoscere per ogni giornata, quanti punti ha ogni squadra. Si vogliono anche conoscere, per ogni partita, i giocatori che hanno giocato, i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita) e nome, cognome, città e regione di nascita dell'arbitro della partita. Distinguere anche le partite giocate in una città diversa da quella della squadra ospitante; per queste si vuole rappresentare la città in cui si svolgono, nonché il motivo della variazione di sede. Dei giocatori interessa anche la città di nascita.

# Parte 3



# Progettazione fisica - SQL



# Linguaggio standard

Diffuso perché è uno standard definito dagli istituti specializzati ANSI e ISO.

Prima versione dello standard SQL:86 e ultima versione standard SQL:2011.

Messo a disposizione come interfaccia di comunicazione con il db dai DBMS.

Il linguaggio quindi deve permettere di eseguire tutte le operazioni di tipo CRUD (vedi slide 4).

Inoltre deve essere necessario mantenere la struttura del modello relazionale (vedi slide 7).

# Tipologia di dati

- Caratteri
- Valori numerici esatti
- Valori numerici approssimati
- Istanti temporali
- Intervalli di tempo
- boolean
- blob e clob

# Tipologia di dati: Caratteri

Il dominio character permette di rappresentare singoli caratteri oppure stringhe.

La lunghezza delle stringhe di caratteri può essere fissa o variabile.

character

character (*lunghezza*)

character varying (*lunghezza massima*)

varchar (*lunghezza massima*)

# Tipologia di dati: Numerici esatti

Questo è il dominio che permette di rappresentare valori esatti, interi o con parte decimale di lunghezza prefissata.

`numeric` | `numeric (precisione)` | `numeric (precisione, decimali)`

`decimal` | `decimal (precisione)` | `decimal (precisione, decimali)`

`integer`

`smallint`

`bigint`

La differenza tra i primi due è nella precisione: valore esatto, requisito minimo.

# Tipologia di dati: Numerici approssimati

Questo è il dominio che permette di rappresentare valori reali approssimati, di solito utili per grandezze fisiche.

`float | float (precisione)`

`real`

`double precision`



# Tipologia di dati: Istanti temporali

Questo dominio permette di rappresentare istanti di tempo.

date

time | time (*precisione*) | time (*precisione*) with time zone

timestamp | timestamp (*precisione*) | timestamp (*precisione*) with time zone

Ogni campo è strutturato e decomponibile in un insieme di campi:

1. year, month, day
2. hour, minute, second
3. tutti i precedenti

Con with time zone vengono estratti ora locale e universale.

# Tipologia di dati: Intervalli temporali

Questo dominio permette di rappresentare intervalli di tempo, come ad esempio la durata di un evento.

```
interval year|month|day|hour|minute|second
```

```
interval year|month|day|hour|minute|second to year|month|day|hour|minute|second
```

Ogni Gli intervalli che indichiamo devono essere ordinati, quindi è corretto definire un intervallo di questo tipo:

```
interval year to day
```

Ma non di questo tipo:

```
interval month to year
```

# Tipologia di dati: Boolean

Questo dominio permette di rappresentare singoli valori booleani quindi vero o falso.

`boolean`

# Tipologia di dati: Blob e Clob

Questo dominio permette di rappresentare oggetti di grandi dimensioni, costituiti da una sequenza arbitraria di valori binari o di caratteri.

blob

clob

Per entrambi il sistema garantisce solo la memorizzazione del valore, ma non permette che il valore venga utilizzato come criterio di selezione.

# Creazione schema

Sql utilizza la definizione di schema per determinare le strutture che rappresentano collezione di oggetti: **tabelle**, domini, viste, etc.etc..

```
create schema NomeSchema {definizione}
```

Dentro le parentesi graffe ci sono le definizioni che implementano lo schema.

Per creare una tabella quindi possiamo usare il comando sopra:

```
create table NomeTabella {definizione}
```

# Creazione tabelle

La sintassi per creare gli attributi è la seguente:

```
create table NomeTabella {  
    NomeAttributo1 Dominio Vincoli,  
    NomeAttributo2 Dominio Vincoli  
}
```

Il dominio rappresenta un tipo di quelli elencati nelle slide precedenti.

È possibile per ogni attributo definire: valori di default, vincoli sull'attributo, chiavi primarie e vincoli interrelazionali.

# Esempio: Creazione tabelle #1

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

professori(**id**,nome,cognome)

corso(**nome**,aula,professore)

studenti(**id**,nome,cognome)


studenti\_corso(**studente**,**corso**)

vincoli di integrità referenziali:

corso(professore)->professori(id)

studenti\_corso(studente)->studenti(id)

studenti\_corso(corso)->corso(nome)



```
create table professori {  
    id bigint,  
    nome varchar(20),  
    cognome varchar(20)  
}
```

# Esempio: Creazione tabelle #1

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

professori(**id**,nome,cognome)  
corso(**nome**,aula,professore)  
studenti(**id**,nome,cognome)  
studenti\_corso(**studente**,**corso**)

vincoli di integrità referenziali:

corso(professore)->professori(id)  
studenti\_corso(studente)->studenti(id)  
studenti\_corso(corso)->corso(nome)

```
create table professori {  
  id bigint,  
  nome varchar(20),  
  cognome varchar(20)  
}
```

?

Chiave

?

Valori  
non  
nulli



# Vincoli intrarelazionali

Quando abbiamo fatto la progettazione concettuale abbiamo dovuto assumere alcuni aspetti: chiavi primarie, unicità degli attributi, valori non nulli. Questi sono definiti vincoli intrarelazionali e sono i seguenti:


- Not null = il valore nullo associato ad un attributo non è ammesso.
- Unique = applicato ad un attributo, o un insieme di attributi, che impone che non sono ammessi gli stessi valori per tuple/record diversi.
- Primary key = serve per specificare la chiave primaria della relazione.

NB: il vincolo primary key è implicitamente contenente il vincolo di not null e unique perchè un attributo che deve referenziare in modo unico una tupla/record non può essere nullo e soprattutto deve essere unico.

# Esempio: Vincoli intrarelazionali

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

```
professori(id,nome,cognome)
corso(nome,aula,professore)
studenti(id,nome,cognome)
studenti_corso(studente,corso)
```



```
create table professori {
    id bigint primary key,
    nome varchar(20) not null,
    cognome varchar(20) not null
}
```

vincoli di integrità referenziali:

```
corso(professore)->professori(id)
studenti_corso(studente)->studenti(id)
studenti_corso(corso)->corso(nome)
```

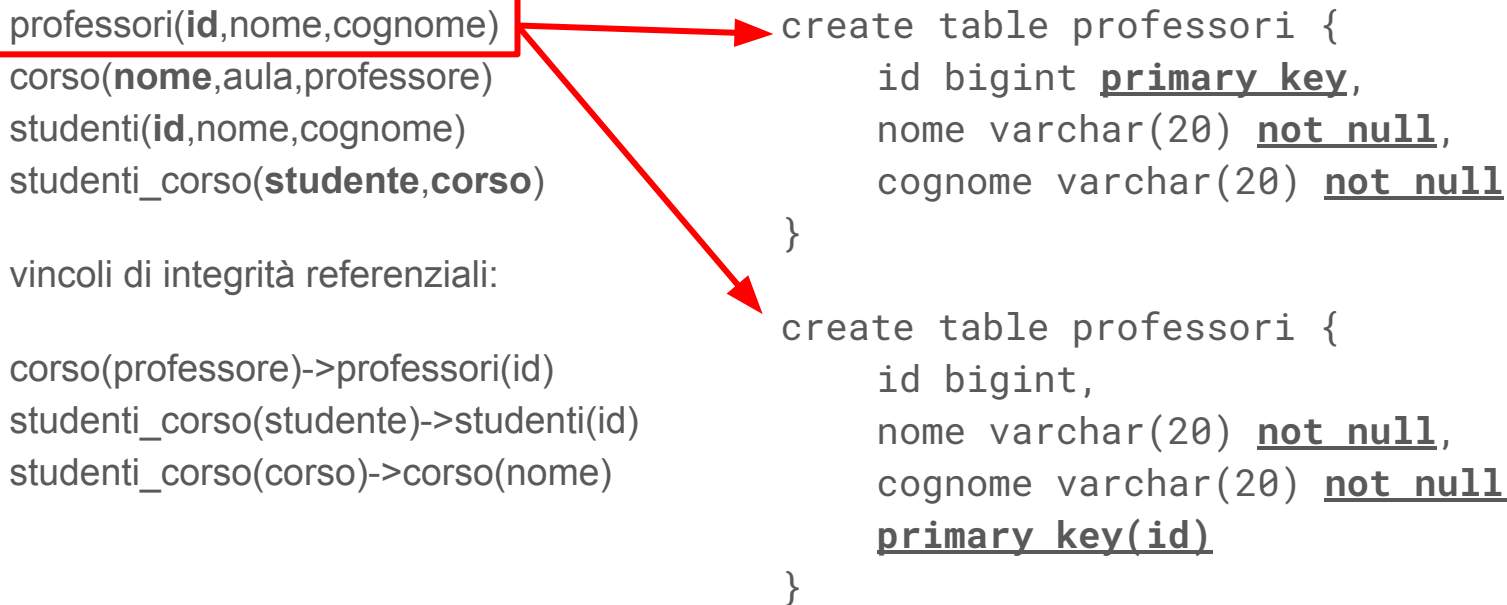
# Esempio: Vincoli intrarelazionali

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

professori(**id**,nome,cognome)  
corso(**nome**,aula,professore)  
studenti(**id**,nome,cognome)  
studenti\_corso(**studente**,**corso**)

vincoli di integrità referenziali:

corso(professore)->professori(id)  
studenti\_corso(studente)->studenti(id)  
studenti\_corso(corso)->corso(nome)



```
create table professori {  
    id bigint primary key,  
    nome varchar(20) not null,  
    cognome varchar(20) not null  
}
```

```
create table professori {  
    id bigint,  
    nome varchar(20) not null,  
    cognome varchar(20) not null,  
    primary key(id)  
}
```

## Esempio: Creazione tabelle #2

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

```
professori(id,nome,cognome)  
corso(nome,aula,professore)  
studenti(id,nome,cognome)  
studenti_corso(studente,corso)
```

vincoli di integrità referenziali:

```
corso(professore)->professori(id)  
studenti_corso(studente)->studenti(id)  
studenti_corso(corso)->corso(nome)
```

```
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint
```

?

}

# Vincoli interrelazionali

Dobbiamo avere un modo per poter indicare i collegamenti tra tabelle quindi introduciamo i vincoli:

- Foreign key, usato su un insieme di attributi
- References, usato per un solo attributo

Entrambi hanno bisogno di definire la tabella e l'attributo, o attributi, a cui vogliamo fare riferimento.

Gli attributi con questi vincoli non potranno più avere valori diversi da quelli contenuti nell'attributo della tabella che abbiamo indicato.

# Esempio: Vincoli interrelazionali

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

```
professori(id,nome,cognome)  
corso(nome,aula,professore)  
studenti(id,nome,cognome)  
studenti_corso(studente,corso)
```

vincoli di integrità referenziali:

```
corso(professore)->professori(id)  
studenti_corso(studente)->studenti(id)  
studenti_corso(corso)->corso(nome)
```

```
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint  
        references professori(id)  
}
```

```
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint,  
        foreign key (professore)  
        references professori(id)  
}
```

# Vincoli interrelazionali #2: modifiche

Come si ripercuotono le **modifiche** negli attributi legati da vincoli interrelazionali?

È possibile definire uno dei seguenti modi:

- cascade = il nuovo attributo viene riportato in tutte le tabelle esterne
- set null = l'attributo referente viene settato a null al posto del nuovo valore
- set default = nel caso in cui sia stato definito un valore di default viene impostato quello al posto del nuovo valore
- no action = l'azione di modifica non viene consentita

# Vincoli interrelazionali #3: cancellazione

Come si ripercuotono le **cancellazioni** negli attributi legati da vincoli interrelazionali?

È possibile definire uno dei seguenti modi:

- cascade = tutte le righe della tabella interna corrispondenti alla riga cancellata 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 default al posto del valore cancellato nella tabella esterna
- no action = l'azione di modifica non viene consentita



# Esempio: Vincoli interrelazionali #2

Riprendiamo l'esempio creato nella slide 32 in cui abbiamo estratto dallo schema queste tabelle:

```
professori(id,nome,cognome)  
corso(nome,aula,professore)  
studenti(id,nome,cognome)  
studenti_corso(studente,corso)
```

vincoli di integrità referenziali:

```
corso(professore)->professori(id)  
studenti_corso(studente)->studenti(id)  
studenti_corso(corso)->corso(nome)
```

```
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint  
        references professori(id)  
        on update cascade  
        on delete set null  
}
```

# Aggiornare tabelle

La sintassi per alterare la composizione della tabella e i suoi attributi è la seguente:

```
alter table NomeTabella Modifica
```

La modifica alla tabella può comportare:

- alter column
- add constraint
- drop constraint
- add column
- drop column

# Eliminare tabelle

Non è comune, ma se risulta essere necessario è ovviamente possibile eliminare le tabelle create con il seguente comando:

```
drop table NomeTabella Opzioni
```

La tabella non può essere eliminata se possiede delle righe o se è presente in qualche definizione di tabella.

Le opzioni disponibili sono:

- `restrict` = specifica che la cancellazione non deve essere eseguita in presenza di oggetti non vuoti
- `cascade` = viene forzata la cancellazione di tutti gli oggetti

# Esercizi parte 3



# Esercizio 7

Completare l'esempio delle slide precedenti:

professori(id,nome,cognome)  
corso(**nome**,aula,professore)  
studenti(id,nome,cognome)  
studenti\_corso(**studente**,**corso**)

vincoli di integrità referenziali:

corso(professore)->professori(id)  
studenti\_corso(studente)->studenti(id)  
studenti\_corso(corso)->corso(nome)

```
create table professori {  
    id bigint primary key,  
    nome varchar(20) not null,  
    cognome varchar(20) not null  
}
```

```
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint  
        references professori(id)  
}
```

# Esercizio 8

A partire dagli esercizi della parte 2:

- Attività commerciale
- Aeroporto
- Biblioteca
- Campionato di calcio

Eseguire la trasformazione della progettazione logica in progettazione fisica.

# Parte 4



# Interrogazione db - SQL





# Interrogazioni semplici: clausole

Le operazioni di interrogazione in SQL vengono specificate per mezzo delle clausole `select` e `from`:

```
select ListaAttributi  
from NomeTabella
```

Con `select` selezioniamo la lista di attributi che vogliamo restituire, mentre con `from` definiamo la tabella a cui ci stiamo riferendo. Se vogliamo inserire un filtro in questa selezione possiamo usare la clausola `where`:

```
select ListaAttributi  
from NomeTabella  
where Condizione
```

# Interrogazioni su più tabelle

Possiamo fare selezioni anche su più tabelle nel caso in cui queste siano collegate. Prendiamo come esempio queste due tabelle:

```
create table professori {  
    id bigint primary key,  
    nome varchar(20) not null,  
    cognome varchar(20) not null  
}  
  
create table corso {  
    nome varchar(50) primary key,  
    aula integer not null,  
    professore bigint  
        references professori(id)  
}
```

Se volessi in un unico risultato tutti gli attributi di corso e gli attributi del professore associato ad ogni corso allora l'interrogazione è questa:

```
SELECT corso.nome, professori.nome,  
       professori.cognome  
FROM corso, professori  
WHERE corso.professore = professori.id
```

Il risultato sarà una tabella con tutti i nome dei corsi e nome e cognome dei professori che la tengono.

# Interrogazioni su più tabelle

La clausola **where** ammette come argomento un'espressione booleana costruita combinando predicati semplici con gli operatori

- and
- or
- not

Ogni predicato utilizza gli operatori classici =, <, >, <= e >= per confrontare da un lato un'espressione costruita a partire dai valori degli attributi per la riga, e dall'altro lato un valore costante o un'altra espressione.

# Esempio 1

1. Recuperare tutti i corsi che si tengono nell'aula 12:

```
SELECT nome  
FROM corso  
WHERE aula = 12
```

2. Tutti i corsi del professore Roberto Rossi che non si tengono in aula 2:

```
SELECT C.nome  
FROM corso C, professori P  
WHERE P.nome = 'Roberto' AND P.cognome = 'Rossi' AND c.aula <> 2
```

## Esempio 2

1. Recuperare il nome del corso seguito dallo studente Mario Rossi o da Anna Verdi tenuto dalla professoressa Anna Bianchi:

```
SELECT C.nome  
FROM corso C, professori P, studenti S, studenti_corso SC  
WHERE  
    P.id = C.professore AND  
    S.id = SC.studente AND C.nome = SC.corso AND  
    P.nome = 'Anna' AND P.cognome = 'Bianchi' AND  
    ((S.nome = 'Mario' AND S.cognome = 'Rossi') OR  
     (S.nome = 'Anna' AND S.cognome = 'Verdi'))
```

# Esercizi parte 4



## Esercizio 9: Attività commerciale

1. Tutti i prodotti venduti che appartenevano al magazzino numero 5
2. I prodotti acquistati da Mario Rossi
3. I prodotti acquistati da Mario Rossi nel negozio 4
4. Tutte le persone che hanno acquistato il prodotto 12
5. La lista dei prodotti nel magazzino 3
6. Tutti gli acquisti fatti nel negozio 2
7. I magazzini che contengono il prodotto 9

# Esercizio 10: Aeroporto

1. Tutti gli aerei che sono partiti dall'aeroporto di malpensa
2. Tutti gli aerei che contengono più di 100 persone
3. L'aeroporto di partenza degli aerei presi da Mario Rossi
4. Il gate in cui l'aereo di Mario Rossi si è fermato il 21 dicembre
5. L'aereo o gli aerei contenuti nell'hangar 13
6. Tutti gli aerei che si sono fermati nel gate 4
7. Tutte le persone che sono scese nel gate 4



# Esercizio 11: Biblioteca

1. Tutti i libri che ha in prestito da Mario Rossi
2. Il nome e indirizzo di tutte le tessere che hanno preso in prestito Guerra e Pace
3. I nomi dei libri non in prestito con la posizione degli scaffali (se esiste come attributo)
4. Tutti i libri presi in prestito tra il giorno 7 gennaio e 14 gennaio
5. Tutti i libri che sono stati riconsegnati il 12/05/21 alle 15:30
6. Tutti i libri dell'autore Dario Fo
7. La lista di tutte le persone che non hanno riconsegnato ancora i libri in prestito