Basi di Dati

Definizione Basi di dati

1.SistemiAle
Una base di dati è un insieme di dati permanenti, gestiti da un elaboratore, suddivisi in metadati che ne definiscono la struttura,scheda del DB e i dati effettivi conformi alla struttura

Definizione DBMS

Sistema centralizzato o distribuito che offre linguaggi per:

- definire lo schema
- scegliere le strutture dati
- memorizzare, recuperare e modificare il dato

Esempi definizione DB

- Schema vuoto

```
CREATE DATABASE EsempioEsami;
```

- Definizione schema:

```
CREATE TABLE Studenti (
        Nome char(8),
        Matricola int NOT NULL,
        Città char(10),
        AnnoNascita int,
        PRIMARY KEY (Matricola)
);

CREATE TABLE ProveEsami (
        Materia char(5),
        Matricola int,
        Data char(8),
        Voto int,
        Lode char(1),
        PRIMARY KEY (Materia, Matricola)
);

ecc...
```

- Inserzione di dati

```
INSERT INTO ProveEsami VALUES ('BD', 71523, '28.12.06', 30, 'S');
```

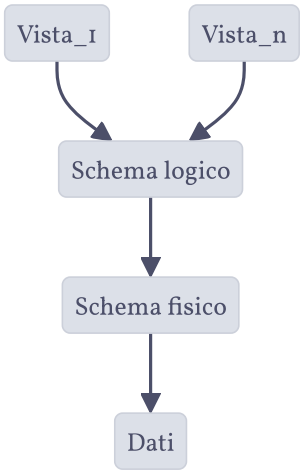
- Interrogazione

```
SELECT Matricola
FROM ProveEsami
WHERE Materia = 'BD' AND Voto = 30;
```

DDL

Tre diversi livelli di descrizione dei dati (schemi):

- Vista logica
- Logico
- Fisico



Livello Vista logica

Descrive come deve apparire la struttura della base di dati ad una certa applicazione (schema esterno o vista)
Esempio:

- InfCorsi(IdeC char(8), Titolo char(20), NumEsami int)

```
CREATE VIEW InfCorsi (IdeC, Titolo, NumEsami) AS
                SELECT IdeC,
                          Titolo,
                          COUNT(*)
                FROM Corsi NATURAL JOIN Esami
                GROUP BY IdeC, Titolo;
```

Livello Logico

Descrive la struttura degli insiemi di dati e delle relazioni
Esempio:

- Studenti(Matricola int, Nome char(20), Login char(8), AnnoNascita int, Reddito real)
- Corsi(IdeC char(8), Titolo char(20), Credito int)
- Esami(Matricola int, IdeC char(8), Voto int)

Livello fisico

Descrive lo schema fisico, ovvero come vanno organizzati fisicamente i dati e definisce strutture dati ausiliare per l'uso (es. indici)

- Studenti organizzata in modo sequenziale con indice

```
CREATE INDEX Indice ON Studenti(Matricola);
```

DML

- Utenti non programmatori
  - Interfaccia grafica per accedere ai dati
  - Linguaggio di interrogazione
- Utenti programmatori
  - Linguaggio convenzionale + librerie predefinite
  - Linguaggio esteso per marcare i comandi SQL, necessita di un pre-compilatore per interpretare le query
  - Linguaggio integrato disegnato ad-hoc per usare SQL, con comandi controllati staticamente dal traduttore ed eseguiti dal DBMS

DBMS: controllo dei dati

- Meccanismi offerti per garantire
  - Integrità
  - Sicurezza
    - Restrizione dell'accesso ai soli utenti autorizzati
    - Limitazione delle operazioni eseguibili
  - Affidabilità
    - Protezione da interferenze dovuto ad accessi concorrenti
    - Malfunzionamenti hw e sw

DBMS: transazioni

Una transazione è una sequenza di azioni di lettura e scrittura in memoria permanente e di elaborazioni di dati in memoria temporaneam con le seguenti proprietà:

- Atomicità ovvero terminando prematuramente vengono trattate come non fossero mai iniziate
- Serializzabilità se vengono eseguite concorrentemente più transazioni, l'effetto è quello di un'esecuzione seriale

- Persistenza Le modifiche al DB di una transazione terminata sono permanenti, non alterabili da eventuali malfunzionamenti

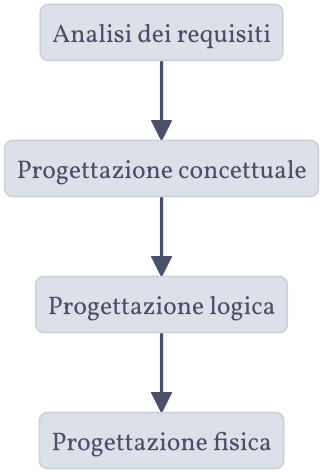
DBMS: controllo dei dati

Protezione da interferenze indesiderate tra accessi concorrenti ai dati e da malfunzionamenti hw o sw, questi ultimi gestiti con journal e copie di sicurezza

Modelli informatici

> Un **modello astratto** è la rappresentazione formale di idee e conoscenze a un fenomeno

- Aspetti di un modello
  - Il modello rappresenta certi fatti
  - Rappresentazione data con un linguaggio formale
  - Modello <-- Risultato processo di interpretazione



Modellazione concettuale

- Cosa si modella?
  - Conoscenza concreta (fatti)
  - Conoscenza astratta (Struttura e vincoli sulla conoscenza concreta)
  - Conoscenza procedurale (operazioni base/degli utenti)
  - Comunicazioni (Come si comunicherà con il sistema informatico)

Conoscenza concreta

- Fatti specifici che si vogliono rappresentare
  - Entità con le loro proprietà
    - Ciò di cui interessa rappresentare i fatti
    - Le proprietà sono fatti che interessano perché descrivono caratteristiche di determinate entità
      - **Classificazione delle proprietà**
      - atomica o strutturata
      - univoca / multivalore

        - totale / parziale
    - Ogni entità ha un tipo che ne specifica la natura, identifica proprietà e dominio relativo
        - Es. **Antonio** ha tipo *Persona* con proprietà:
            - Nome: *string*
            - Indirizzo: *string*
- Collezioni di entità omogenee
    - Es. **Studenti**: insieme di tutti gli studenti nel dominio del corso
    - Spesso sono organizzate in una gerarchia di specializzazione/generalizzazione (si parla anche sottoclassi e superclassi)
        - Es. Nel DB la collezione *Utenti* può essere considerata una generalizzazione di *Studenti* e *Docenti*
        - Importanti caratteristiche di una gerarchia:
            - Ereditarietà (delle proprietà)
            - Inclusione (se la collezione C1 specializza C2, gli elementi di C1 sono un sottoinsieme degli elementi di C2)
- Associazioni fra entità
    - Un fatto che correla due o più entità, stabilendo un legame logico tra loro
        - Es. L'utente *Utente* **HA IN PRESTITO** una copia della "Divina commedia"
    - Un'associazione R(X,Y) fra due collezioni di entità X e Y è un insieme di istanze di associazione tra elementi di X e Y, che varia in generale nel tempo.Il prodotto cartesiano X*Y è detto dominio dell'associazione
    - Possiede le seguenti proprietà:
        - molteplicità o cardinalità
            - Un'associazione R(X,Y) è **univoca da X a Y** se per ogni elemento x di X esiste al più un elemento di Y che è associato ad X; se non vale, l'associazione è **multivalore da X a Y**
            - Cardinalità:
                - R(X,Y) è (1:N) se essa è **multivalore** da X a Y ed **univoca** da Y a X
                - R(X,Y) è (N:1) se essa è **univoca** da X a Y e **multivalore** da Y a X
                - R(X,Y) è (N:M) se essa è **multivalore** da X a Y e **multivalore** da Y a X
                - R(X,Y) è (1:1): se essa è **univoca** su da X a Y e **univoca** da Y a X
            - Es. Cardinalità:
                - *Frequenta*(Studenti, Corsi) ha molteplicità (N:M)
                - *Insegna*(Professori, Corsi) ha molteplicità (1:N)
                - *SuperatoDa*(Esami, Studenti) ha molteplicità (N:1)
                - *Dirige*(Professori, Dipartimenti) ha molteplicità (1:1)
        - totalità
            - Un'associazione R(X,Y) è **totale da X a Y** se per ogni elemento x di X esiste almeno un elemento di Y che è associato ad X; se non vale, l'associazione è **parziale da X a Y**

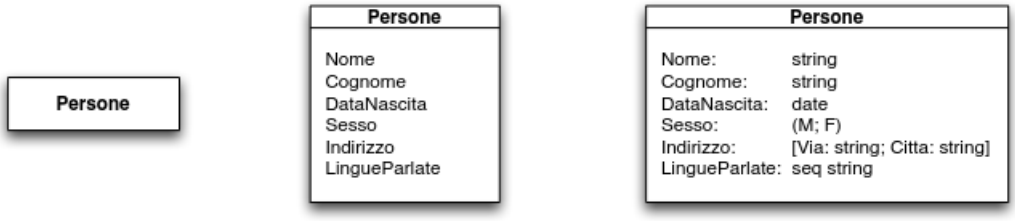
Conoscenza astratta

- Vincoli di integrità statici
  - Definiscono delle condizioni sui valori della conoscenza concreta che devono essere soddisfatte indipendentemente da come evolve l'universo del discorso
- Vincoli di integrità dinamici
  - Definiscono delle condizioni sul modo in cui la conoscenza concreta può evolvere nel tempo
- Fatti derivabili
  - Arrivano da altre fonti, come l'età di una persona, la quale si ricava per differenza fra l'anno attuale e il suo anno di nascita

Modello dei dati a oggetti

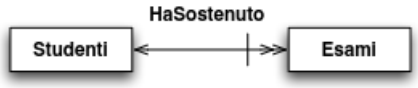
> Un modello dei dati è un insieme di meccanismi di astrazione per descrivere la struttura della conoscenza concreta (schema). Uno schema verrà dato usando una notazione grafica, variante dei cosiddetti diagrammi ER
> **TERMINOLOGIA**:
>
> - entità ==> oggetto
> - tipo entità ==> tipo oggetto
> - collezione ==> classe
> - associazione ==> associazione/relazione

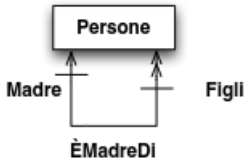
- Ad ogni entità del dominio corrisponde un oggetto, composto da stato(variabili/costanti) comportamento(metodi) e identità, capace di rispondere a richieste chiamate messaggi, restituendo valori memorizzati nello stato o calcolati tramite procedura locale
- **CLASSI**: insieme di oggetti dello stesso tipo, modificabile con operatori per includere o estrarre elementi dall'insieme, associabile a vincoli di integrità



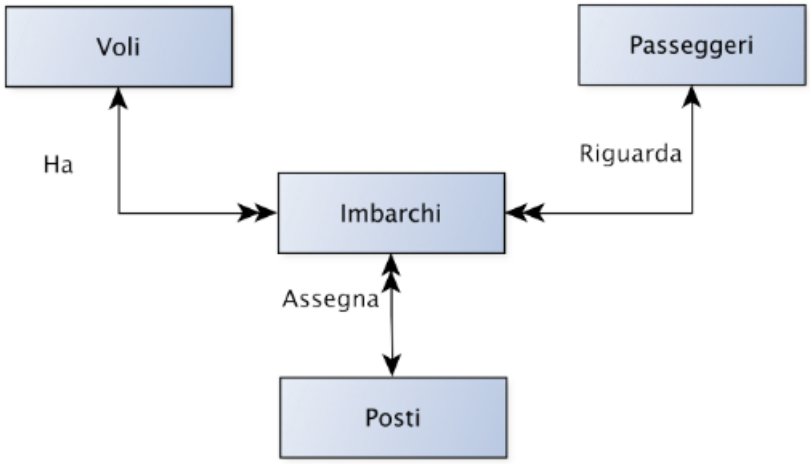
- Gli attributi(Nome, Cognome, ...) possono essere:
  - Primitivi (*int, real, bool, date, string*)
  - Non primitivi
- Le associazioni:
  - Possono avere proprietà
  - Possono essere **Ricorsive**

- Associazioni n-arie

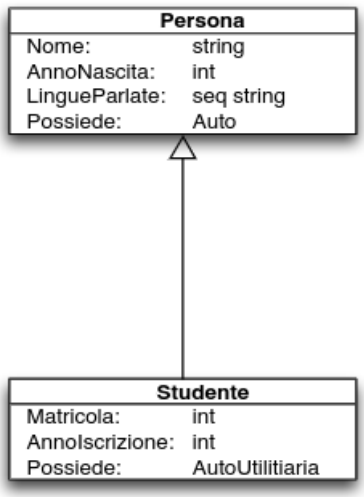




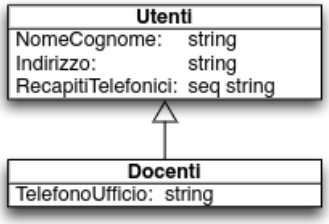
- Esempio di associazione:
  - Esempio: Si vuole rappresentare l'associazione tra Voli, Passeggeri e Posti. Per ogni volo, al momento dell'imbarco, viene assegnato un posto a ciascun passeggero.



- Ereditarietà:
  - Permette di definire un tipo di oggetto a partire da un altro "per differenza", come aggiunta di attributi e/o ridefinizione di attributi esistenti
  - Normalmente si usa solo per definire sottotipi

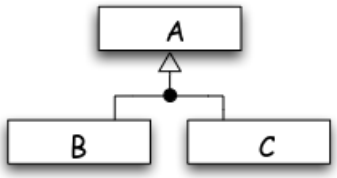


- Questo principio di eredità si manifesta appunto anche nelle definizioni delle classi



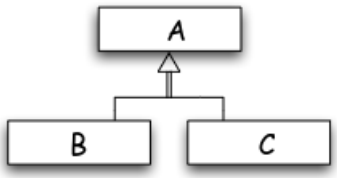
- Vincoli su sottoclassi:
  - **Vincolo di disgiunzione**

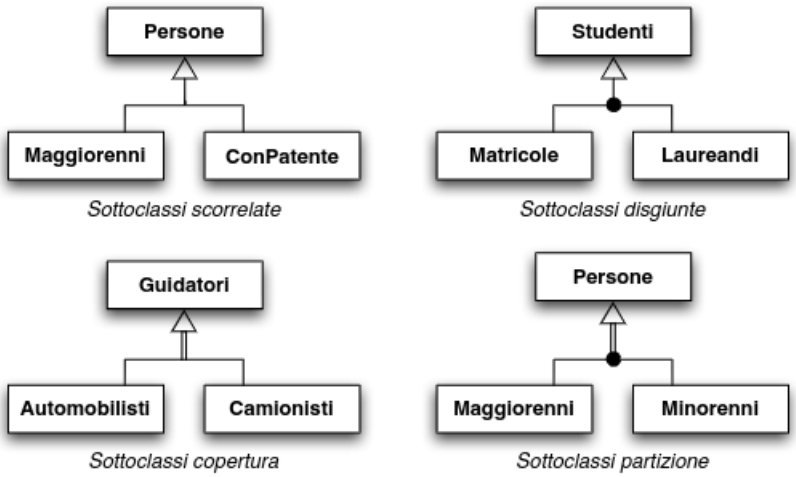
$B \cap C = \varnothing$



  - **Vincolo di copertura**

$B \cup C = A$



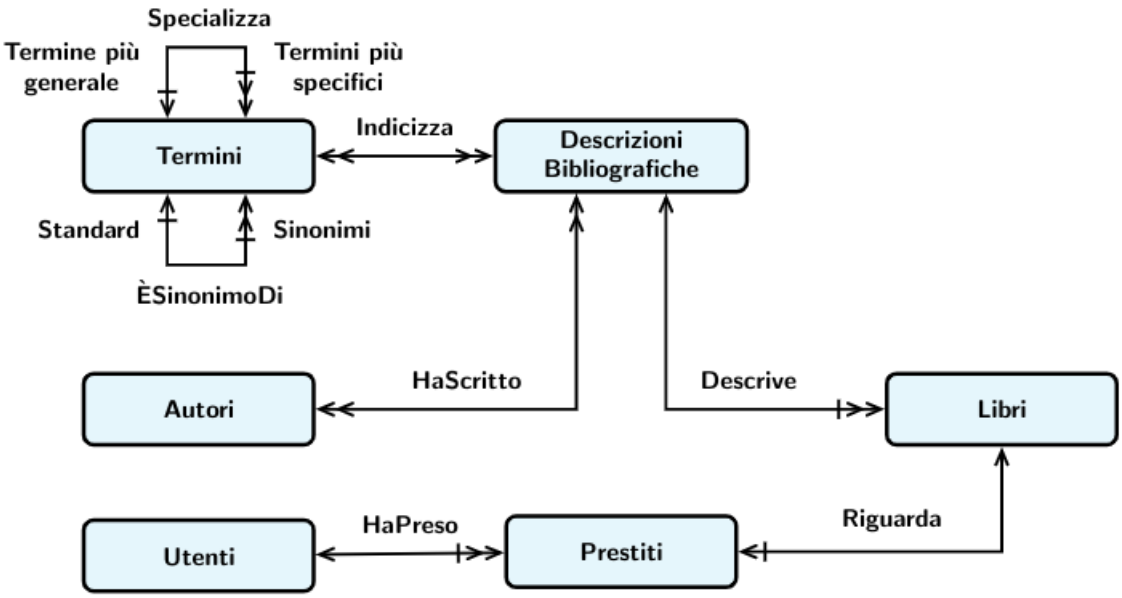


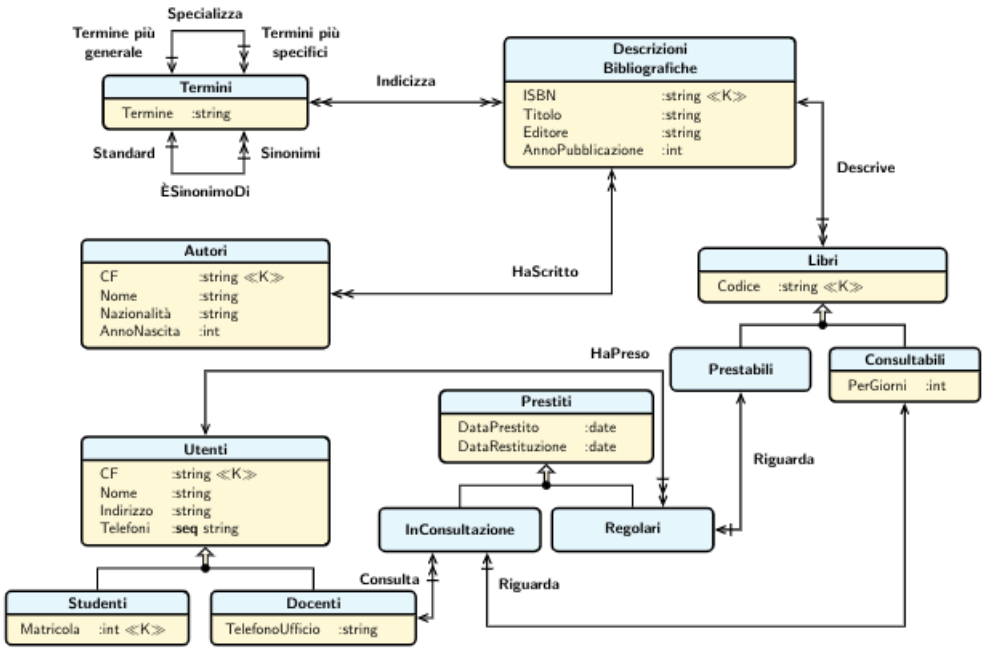
DESCRIZIONE DI UN CASO, ESEMPIO SLIDE 2.ModelliAle

Si vogliono modellare alcuni fatti riguardanti una biblioteca universitaria:

- le descrizioni bibliografiche dei libri, opere con un solo volume,
- i termini del thesaurus (parole chiave),
- le copie dei libri disponibili che corrispondono ad una descrizione bibliografica,
- gli autori dei libri,
- gli utenti della biblioteca,
- i prestiti in corso.

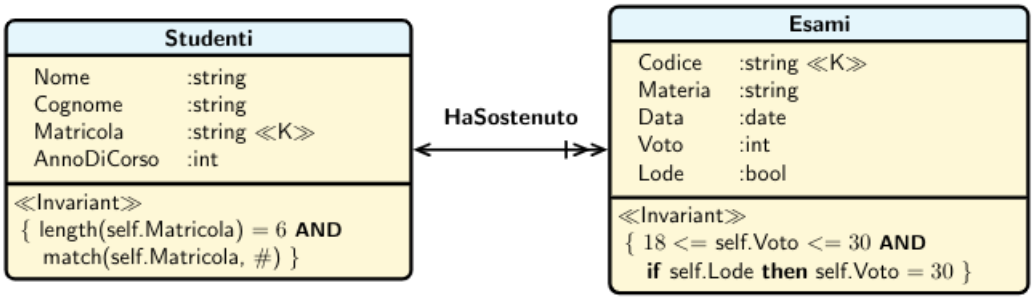
- Le Descrizioni Bibliografiche dei libri a volume unico, sia già disponibili che quelli ordinati ma non ancora consegnati alla biblioteca, sono caratterizzati dal Codice ISBN (International Standard Book Number) che le identifica, titolo del libro, autori, editore, anno di pubblicazione e termini che le descrivono.
- Degli autori dei libri interessano il codice fiscale, il nome e cognome, la nazionalità e la data di nascita.
- I libri, disponibili in una o più copie ognuna identificata da un Codice, una stringa con la loro collocazione e numero.
- Quando un utente prende un libro in prestito, si registrano i dati dell'utente, se non sono già presenti, la data del prestito e la data di restituzione. Di un utente interessano il codice fiscale, il nome, il cognome, l'indirizzo e i recapiti telefonici. Un utente può avere più libri in prestito. I dati su un prestito interessano fino al momento della restituzione del libro.
- Il thesaurus è un insieme di termini, e di associazioni fra di loro, che costituiscono il lessico specialistico da usare per descrivere il contenuto dei libri.
- Fra i termini del thesaurus interessano le seguenti relazioni, fra le tante possibili:
  - **Preferenza**, per rimandi da termini standard a termini non standard e viceversa. Per esempio:
    - Elaboratore **Standard** (**vedi**) Calcolatore;
    - Calcolatore **Sinonimi** (**UsatoPer**) Elaboratore, Calcolatrice, Stazione di lavoro.
  - **Gerarchia**, per mettere in evidenza il rapporto specificità-generalità tra due termini. Per esempio:
    - Felino **PiùSpecifico** Gatto Leone Tigre;
    - Gatto **PiùGenerale** Felino;
- Gli utenti possono essere studenti o docenti. Di uno studente interessa anche la matricola e di un docente anche il telefono dell'ufficio.
- Alcuni libri sono per la sola consultazione e possono essere presi in prestito per un numero prefissato di giorni solo dagli utenti che sono docenti.


Specializza
Termine più generale
Termini più specifici
Termini
Indicizza
Descrizioni Bibliografiche
Standard
Sinonimi
ÈSinonimoDi
Autori
HaScritto
Descrive
Libri
Utenti
HaPreso
Prestiti
Riguarda


Specializza
Termine più generale
Termini più specifici
Termini
Termine :string
Indicizza
Descrizioni Bibliografiche
ISBN :string ≪K≫
Titolo :string
Editore :string
AnnoPubblicazione :int
Standard
Sinonimi
ÈSinonimoDi
Descrive
Autori
CF :string ≪K≫
Nome :string
Nazionalità :string
AnnoNascita :int
HaScritto
Libri
Codice :string ≪K≫
HaPreso
Prestabili
Consultabili
PerGiorni :int
Prestiti
DataPrestito :date
DataRestituzione :date
Utenti
CF :string ≪K≫
Nome :string
Indirizzo :string
Telefoni :seq string
Riguarda
InConsultazione
Regolari
Consulta
Riguarda
Studenti
Matricola :int ≪K≫
Docenti
TelefonoUfficio :string

EXTRA

Descrittore di classe con vincoli



<<K>> chiave: sottoinsieme minimale di attributi che identifica l'oggetto

<<NOT NULL>> totalità

`self.Nome` = attributo `Nome` dell'oggetto stesso

In `Esami` potremmo usare `self.HaSostenuto.Matricola`

Progettazione Logica

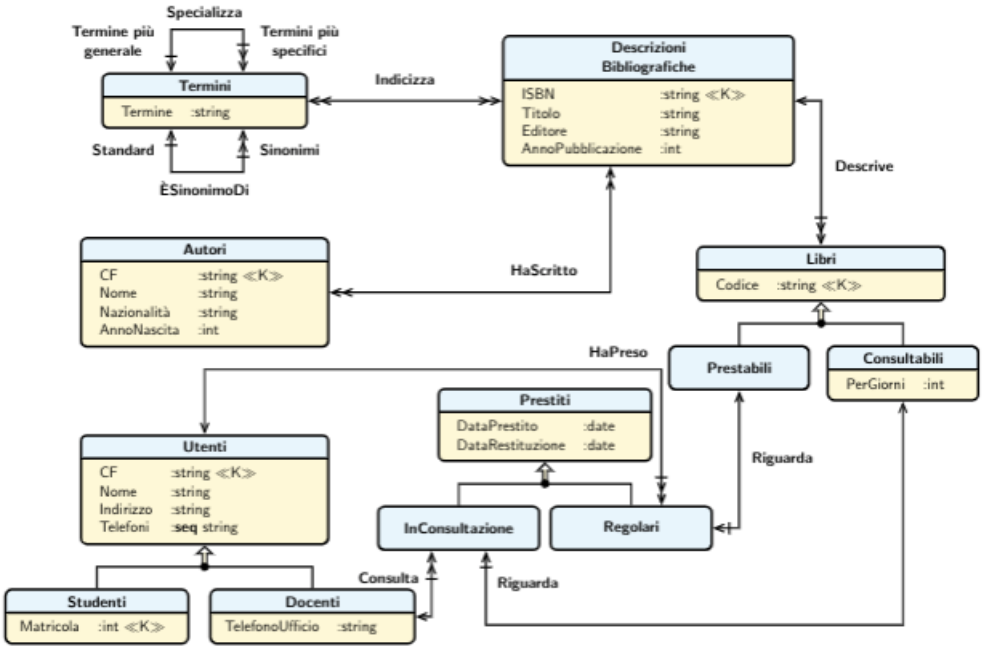
- Collezioni come relazioni
- Associazioni tramite chiavi
- Vincoli di integrità
  - Chiavi
    - Superchiave (sottoinsieme di attributi)
    - Chiave (superchiave minimale)
    - Chiave primaria (una delle chiavi, lunghezza minima)
      - Relazioni a classi radice ==> attributo univoco o artificiale
      - Relazioni a sottoclassi ==> chiave della superclasse
      - Relazioni per associazioni N:M ==> concatenazione delle chiavi esterne
    - Indicate con <<CK>>, valori non nulli
  - Chiavi esterne
    - Può essere nulla solo in caso di associazione parziale
  - Valori non nulli
    - Si possono imporre

Schemi a oggetti -> Schemi relazionali

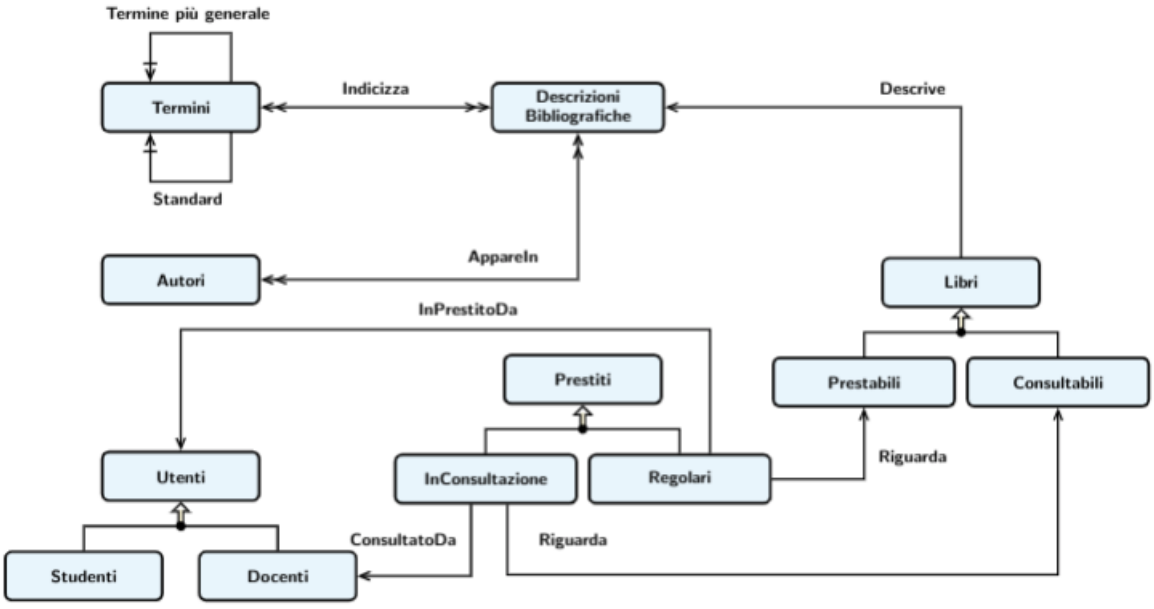
- Trasformazione per passi:
  - Associazioni molti a uno/uno a uno
  - Associazioni molti a molti
  - Gerarchie di inclusione
  - Identificazione chiavi primarie
  - Attributi multivalore
  - Attributi composti

Un esempio: BD per una Biblioteca

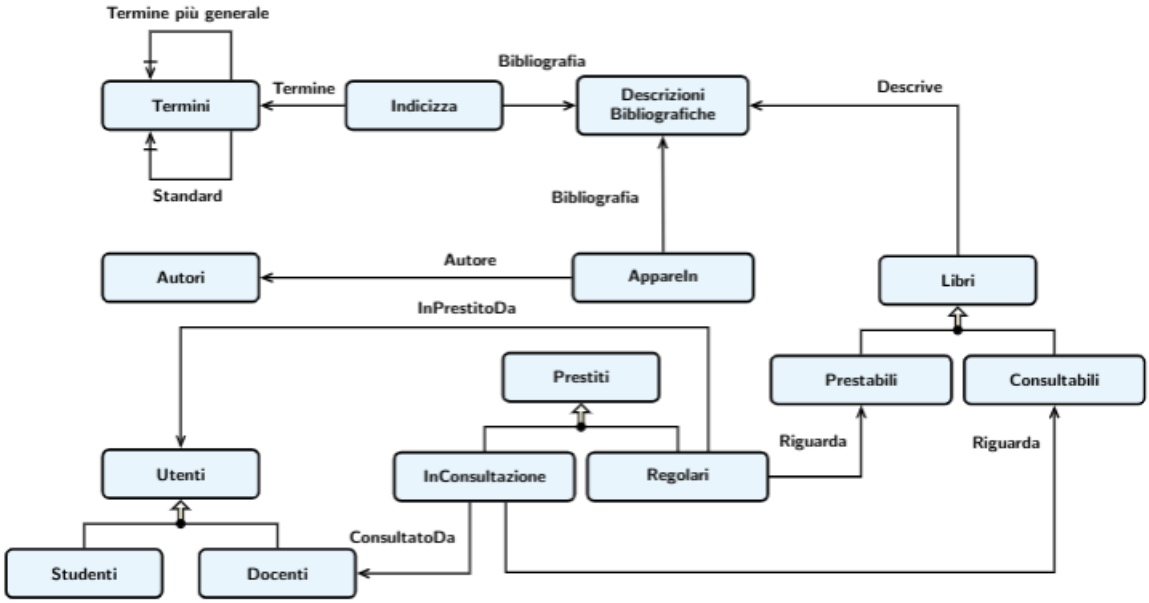
Modello concettuale



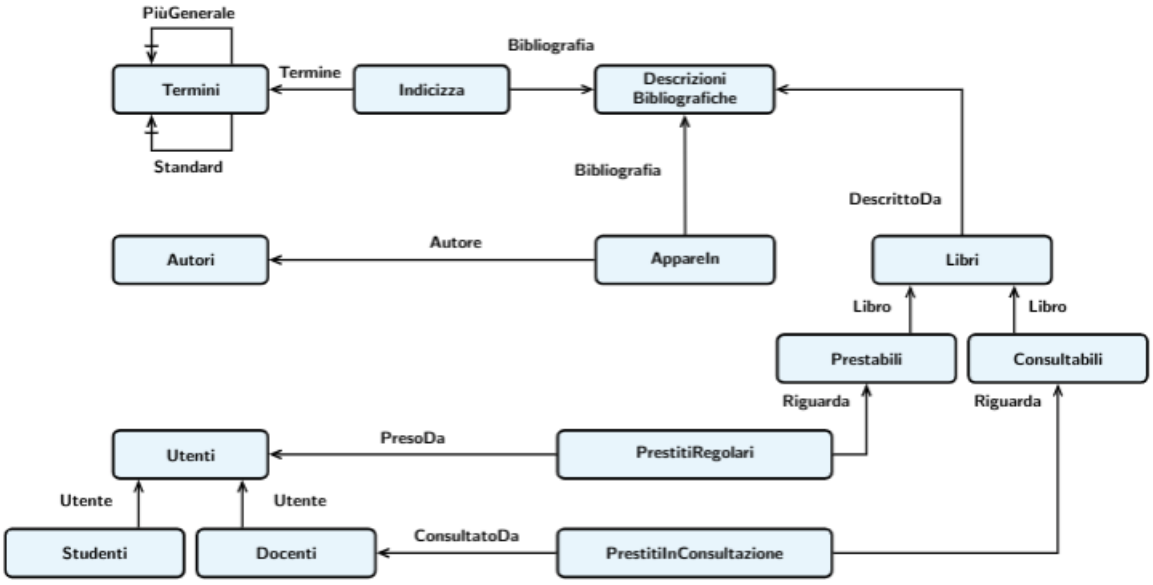
Schema Logico (Passo 1)



Schema Logico (Passo 2)



Schema Logico (Passo 3)



Schema delle relazioni (attributi, tipi, vincoli)

- Termini(Termine: string, PiùGenerale*: string, Standard*: string)
  - PK(Termine)
  - PiùGenerale FK(Termini), Standard FK(Termini)
- DescrizioneBib(ISBN: string, Titolo: string, Editore: string, Anno: int)
  - PK(ISBN)
- Indicizza (Termine*: string, Bibliografia*: string)
  - PK(Termine, Bibliografia)
  - Termine FK(Termini), Bibliografia FK(DescrizioniBib)
- Autori (CF: string, Nome: string, Nazionalita: string, DataNascita: date)
  - PK(CF)
- AppareIn(Autore*: int, Bibliografia*: string)
  - PK(Autore,Bibliografia)
  - Autore FK(Autori), Bibliografia FK(DescrizioniBib)

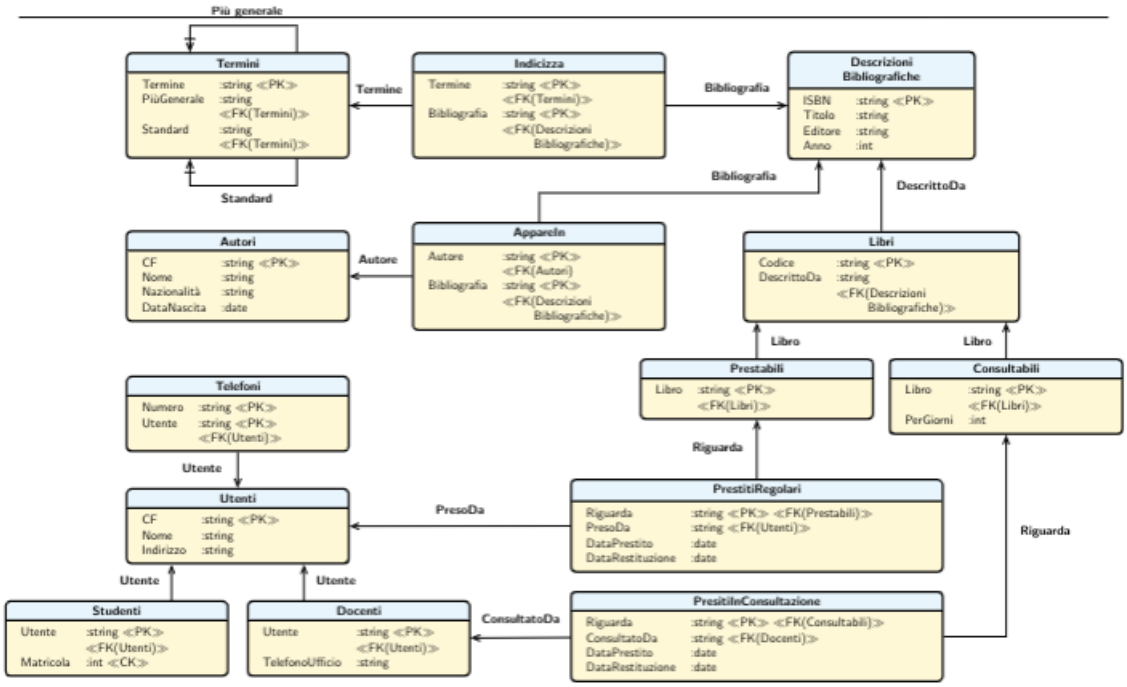
Schema delle relazioni (Cont.)

- Libri(Codice: string, DescrittoDa*: string)
  - PK(Codice)
  - DescrittoDa FK(DescrizioniBib)
- Consultabili(Libro*: string, PerGiorni: int)
  - PK(Libro)
  - Libro FK(Libri)
- Prestabili(Libro*: string)
  - PK(Libro)
  - Libro FK(Libri)
- Utenti (CF: int, Nome: string, Indirizzo: string)
  - PK(CF)
- Telefoni(Numero: string, Utente*: int)
  - PK(Numero,Utente)
  - Utente FK(Utenti)

Schema delle relazioni (Cont.)

- Studenti (Utente*: int, Matricola: string)
  - PK(Utente)
  - Utente FK(Utenti)
  - CK(Matricola)
- Docenti (Utente*: int, TelefonoUfficio: string)
  - PK(Utente)
  - Utente FK(Utenti)
- PrestitiRegolari(DataPrestito: date, DataRestituzione: date, PresoDa*: int, Riguarda*: string)
  - PK(Riguarda)
  - PresoDa FK(Utenti), Riguarda FK(Prestabili)
- PrestitiInConsultazione(DataPrestito: date, DataRestituzione: date, ConsultatoDa*: string, Riguarda*: string)
  - PK(Riguarda)
  - ConsultatoDa FK(Docenti), Riguarda FK(Consultabili)

Schema relazionale con attributi, tipi, vincoli



Algebra Relazionale

Operatori insiemistici ed espressioni regolari

- Prestare molta attenzione al tipo, se le operazioni vengono fatte tra tipi diversi, esse non sono possibili
- Operazioni:
  - UNION
  - INTERSECT
  - EXCEPT (Differenza)
- Esempi:

- SELECT Nome, Cognome, Matricola FROM Studenti WHERE Provincia='VE' UNION SELECT Nome, Cognome, Matricola FROM Studenti JOIN Esami ON Matricola=Candidato WHERE Voto>28
- SELECT Matricola FROM Studenti EXCEPT SELECT Tutor AS Matricola FROM Studenti
- Posso usare EXCEPT ALL, in questo modo i duplicati non vengono rimossi
- IS per verificare che sia il valore NULL, non voglio usare l'uguale, torna unknown!!
- SELECT Nome, Cognome, Matricola FROM Studenti JOIN Esami ON Matricola=Candidato WHERE Voto = 18 INTERSECT ALL SELECT Nome, Cognome, Matricola FROM Studenti JOIN Esami ON Matricola=Candidato WHERE Voto=30

  (Restituisce il minimo tra le due SELECT, le due query)
- SELECT DISTINCT Nome, Cognome, Matricola FROM Studenti JOIN Esami ON Matricola = Candidato JOIN Esami ON Matricola = Candidato WHERE Voto = 18 AND Voto = 30 (Serve per forza la doppia join, sto usando i valori dentro la doppia condizione, non posso farlo sempre sulla stessa tabella, corrisponde ai valori doppi)
- SELECT Nome, Cognome, Matricola FROM Studenti WHERE Nome LIKE 'A_%'
  - % = 0 o più caratteri possibili
  - _ = un carattere qualsiasi
  - A = lettera iniziale
  - LIKE = operatore per eseguire il controllo a stringa
- SELECT Nome, Cognome, Matricola FROM Studenti WHERE Nome LIKE 'A_%' SELECT Nome, Cognome, Matricola FROM Studenti WHERE Nome LIKE 'A%i' OR Nome LIKE 'A%a'

- Esempi di query
  - Studenti che vivono nella stessa provincia dello studente con matricola 71346, escluso lo studente stesso

```
SELECT *
FROM Studenti
WHERE Provincia = (SELECT Provincia
                                        FROM Studenti
                                        WHERE** Matricola='71346')
        AND** Matricola <> '71346'
```

```
SELECT altri.*
FROM Studenti s JOIN Studenti altri USING(Provincia)
WHERE s.Matricola = 71346 AND
        altri.Matricola <> '71346'
```

- Query studenti con almeno un voto > 27

```
SELECT *
FROM Studenti s
WHERE EXISTS(SELECT
```

```
                        FROM Esami e
                        WHERE e.voto > 27 AND e.Candidato = s.Matricola)
```

```
SELECT DISTINCT s.* --  distinct necessario per evitare ripetizioni
FROM Studenti s JOIN Esami e ON s.Matricola = e.Candidato
WHERE e.voto > 27
```

```
Exp Comp ANY (SottoSelect)
-- True quando un valore v restituito dalla sottoselect è in relazioni Comp
con Exp
-- False se tutti i valori sono != NULL e non esiste un valore v tale che
Exp Comp v sia vera
-- In particolare è false se sottoselect è vuota
-- È UNKNOWN se nella sottoselect ci sono valori NULL e per tutti i valori
diversi da NULL e per tutti i valori v != NULL ExpComp è false

Exp Comp ALL (SottoSelect)
-- È true se tutti i valori della sottoselect sono != NULL e per ogni v del
risultato della sottoselect Exp Comp v è vera
-- Se la sottoselect è vuota Exmp Comp rispetto ad essa è vera
-- È false se esiste un v risultato della sottoselect tale che Exp Comp v è
false
-- È UNKNOWN se nella sottoselect ci sono valori NULL e per tutti i valori v
!= NULL Exp Comp v è true
```

- Gli Studenti che hanno preso solo 30 == Studenti dove non esiste un voto diverso da 30

```
SELECT *
FROM Studenti s
WHERE NOT EXISTS (SELECT *
                                FROM Esami e
                                WHERE e.Candidato = s.Matricola AND e.voto
<> 30) -- diverso 30

-- Posso sostituire EXISTS con =ANY

SELECT *
FROM Studenti s
WHERE NOT(s.Matricola =ANY (SELECT *
                                FROM Esami e
                                WHERE e.voto <> 30)) -- diverso 30

SELECT *
FROM Studenti s
WHERE Matricola <> ALL(SELECT Candidato
                                            FROM Esami
```

```
                                        WHERE voti <> 30)
```

- GROUP BY

```
SELECT s.Matricola
FROM Studenti s JOIN Esami e ON s.Matricola = e.Candidato
GROUP BY s.Matricola
HAVING MIN(e.Voto) = 30
```

- Per ogni materia, trovare il nome della materia e il voto medio degli esami in quella materia

```
SELECT
FROM Esami
GROUP BY Materia
HAVING Count(*) > 3
```

```
SELECT s.Cognome, AVG(e.Voto)
FROM Studenti s JOIn Esami e ON s.Matricola = e.Candidato
GROUP BY s.Matricola, s.Cognome
HAVING YEAR(Data) > 2006; -- non è possibile, Data non fa parte del group by
                                                -- dove invece dovrebbe
essere stata salvata
```

```
SELECT COALESCE(Tutor, "Non ha tutor"), COUNT(*)
FROM Studenti
GROUP BY Tutor
```

```
CASE -- Un'espressione condizionale che può essere usata in qualsiasi
contesto deve un'espressione è valida.

CASE WHEN condizione THEN risultato
        [WHEN ...]
        [ELSE risultatoDefault]
END

condizione che restituisce un booleano
```

Modifica dei dati

```
INSERT INTO Tabella
VALUES (valoreA1, ..., valireAn),
           (valoreB1, ..., valireBn),
           ...
```

```
-- m può essere <del numero di attributi n (le restanti colonne o prendono
il valore di default o NULL)

INSERT INTO StNomeCognome AS
           SELECT Nome, COgnome, FROM Studenti
```

```
DELETE FROM Tabella
WHERE Condizione

-- Esempio
DELETE FROM Studenti
WHERE Matricola NOT IN (SELECT Candidato
                                          FROM Esami)
```

```
UPDATE Esami
SET voto = voto + 1
WHERE voto > 23 AND voto < 30
```

```
/*
1) Trovare il numero di voli internazionali che partono giovedì da Napoli
2) Cancellare gli aeroporti di cui non si conosce il numero di piste e i
voli che hanno partenze o arrivi da questi aeroporti
3) Restituire le città francesi da cui partono più di venti voli alla
settimana diretti in Italia
4) Restituire gli aeroporti italiani che hanno solo voli interni
        i) operatori insiemistici
        ii) sottoselect
*/

Aeroporti ( _Citt_,
                  Nazioni,
                  NumPiste)

Voli (_CodVolo_,
          _GiornoSett_,
          CittPart*,
          OraPart,
          CittArr*,
          OraArr,
          TipoAereo*)
          CittPart FK(Aeroporti) CittArr FK(Aeroporti) TipoAereoFK(Aerei)

Aerei (_TipoAereo_,
           NumPass,
           QtaMerci)

-- 1
```

```sql
SELECT COUNT(*)
FROM Voli v JOIN Aeroporti a ON v.CittArr = a.Citt
WHERE v.CittPArt = 'Napoli'
        AND a.Nazione <> 'Italia'
        AND v.GiornoSett = 'giovedì'

-- 2
-- Prima cancello voli, poi aeroporti, per evitare errori
DELETE FROM Voli
WHERE CittPart IN (SELECT Citt
                                    FROM Aeroporti
                                    WHERE NumPiste IS NULL)
          OR CittArr IN (SELECT Citt
                                    FROM Aeroporti
                                    WHERE NumPiste IS NULL)

DELETE FROM Aeroporti
WHERE NumPiste IS NULL

-- 3
SELECT v.Citt
FROM Voli v JOIN Aeroporti p ON v.CittPart = p.Citt JOIN Aeroporti a ON
v.CittArr = a.Citt
WHERE p.Nazione = 'Francia' AND a.Nazione = 'Italia'
GROUP BY v.CittPart HAVING COUNT(*) > 20

-- 4.i
SELECT Citt
FROM Aeroporti
WHERE Nazione = 'Italia'
EXCEPT
SELECT v.CittPart AS Citt -- altrimenti errore! RINOMINARE!!
FROM Voli v JOIN Aeroporti a ON v.CittArr = a.Citt
WHERE a.Nazione <> 'Italia'

-- 4.i
```

$$1)\gamma COUNT(*)(\sigma_{cittPart='Napoli' \wedge GiornoSett=Giovedi--CittArr=Citt}(Voli) \bigcup \sigma_{Nazione<>'Italia'}(Aeroporti))$$

DDL

- SQL non è solo un linguaggio di interrogazione , ma anche un linguaggio per la definizione di basi di dati
  - Creazione della BD e della struttura logica delle tabelle
    - **CREATE SCHEMA** Nome **AUTHORIZATION** Utente
    - **CREATE TABLE/ VIEW**, con vincoli
  - vincoli di integrità
- Uno schema può essere creato con:

- Uno schema può essere creato con:

```
CREATE SCHEMA Università AUTHORIZATION rossi
```

- Uno schema può essere eliminato mediante un comando

```
DROP SCHEMA Nome [CASCADE | RESTRICT ]
```

- Esempio

```
DROP SCHEMA Università CASCADE
```

- Uno schema può contenere varie tabelle delle quali esistono più tipi:
  - Tabelle base
    - I metadati appartengono allo schema
    - I dati sono fisicamente memorizzati
  - Viste
    - I metadati sono presenti nello schema
    - I dati non sono fisicamente memorizzati
- ## Creazione di una tabella
  - Una tabella (base), creata con il comando **CREATE TABLE**, è un insieme di colonne/attributi per ciascuna delle quali va specificato:
    - nome
    - tipo di dato, che può essere
      - predefinito
      - definito dall'utente (dominio)
        costruito con il comando **CREATE DOMAIN**; e.g.

```
CREATE DOMAIN Voto AS SMALLINT
                  CHECK (VALUE <= 30 AND VALUE >= 18)
```

- Tipi di dati atomici:

- tipi interi:
  - **INTEGER, SMALLINT** ...
- valori decimali:
  - **NUMERIC(p,s)**
- virgola mobile:
  - **REAL**
- stringhe di bit:
  - **BIT(x), BIT VARYING(x)**
- booleani:
  - **BOOLEAN**

Tipi di Dato Predefiniti

- stringhe di caratteri:
  - **CHAR(x)** (o **CHARACTER(x)**)
  - **VARCHAR(x)** (o **CHAR VARYING(x)** o **CHARACTER VARYING(x)**)
- date e ore:
  - **DATE, TIME, TIMESTAMP**
- intervalli temporali:
  - **INTERVAL** {**YEAR, MONTH, DAY, HOUR, MINUTE, SECOND**}

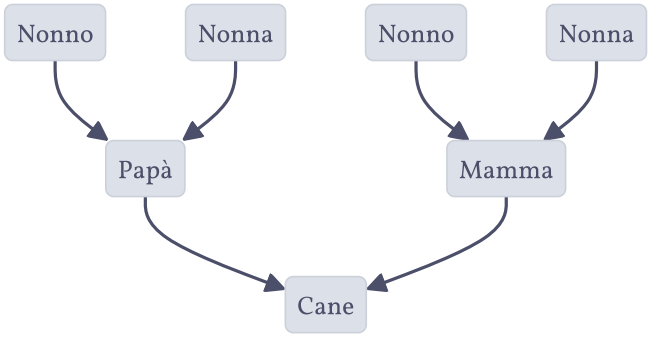
Viste ed Interrogazioni difficili

Viste e interrogazioni difficili

Cani(Cod, Nome, Razza*, Madre*, Padre*, AnnoNasc, AnnoMorte, Istruttore*)

Madre FK(Cani), Padre FK(Cani), Razza FK(Razze), Istruttore FK(Istruttori)

Dare il nome di ogni cane che ha entrambi i genitori e i loro genitori della sua stessa razza




```
create view StessaRazza (Idf, Padre, Madre, Razza)
as select f.Cod, p.Cod, m.Cod, f.Razza
        from Cani f join Cani m on f.Madre = m.Cod join Cani p on f.Padre =
p.Cod
        where f.Razza = m.Razza and f.Razza = p.Razza

-- devo estrapolare il nome dalla lista sopra creata
select
from StessaRazza c join StessaRazza m on c.Madre = m.Idf join StessaRazza p
on c.Padre = p.Idf join Cani ca on c.Idf = ca.Cod
```

Costrutto WITH

- **WITH** fornisce un modo alternativo per scrivere sottoquery da utilizzare in query più complesse.

```
WITH ProvinceMedia AS (
        SELECT s.Provincia, AVG(e.Voto) AS Media
        FROM Studenti s JOIN Esami e ON s.Matricola=e.Candidato
        GROUP BY s.Provincia
    )
SELECT Provincia, Media
FROM ProvinceMedia
WHERE Media = (SELECT MAX(Media) FROM ProvinceMedia);
```

Può essere pensata come una tabella temporanea che esiste SOLO per questa query.

Associazioni simmetriche (cont.)

- Se inserisco **tutte** le ennuple ...

- Ridondanza
- "Difficile" ottenere la lista dei fratelli senza ripetizioni. Es. la query

```
SELECT p1.*, p2.*
FROM   Persone p1 JOIN Fratelli f ON p1.Id = f.Id1
       JOIN Persone p2 ON f.Id2 = p2.Id
```

restituisce

```
+----+---------+---------+----+---------+---------+
| Id | Nome    | Cognome | Id | Nome    | Cognome |
+----+---------+---------+----+---------+---------+
| 13 | Giorgio | Conte   | 21 | Paolo   | Conte   |
| 21 | Paolo   | Conte   | 13 | Giorgio | Conte   |
+----+---------+---------+----+---------+---------+
```


Associazioni simmetriche (cont.)

- Devo modificare la query

```
SELECT p1.*, p2.*
FROM   Persone p1 JOIN Fratelli f ON p1.Id = f.Id1
       JOIN Persone p2 ON f.Id2 = p2.Id
WHERE  f.Id1 < f.Id2
```

-
- Facendo il maggiore, salvo solo una delle due coppie

Esercizi fatti a lezione, query

```
-- trovare ordini, pizze e nome cliente senza ripetizioni in ordine
descrescente
select o.nomecliente, count(*) as numeroordini, count(distinct o.codpizza)
as numeroPizze
    from ordini o
group by o.nomecliente
order by numeroordini desc
```

```
select pi.codpizza, pi.nome, pi.tempoprep
    from pizze pi natural join ricette r
where not exists(select *
                 from ricette r1 natural join ingredienti i
```

```
                    where i.nome like 'Funghi%' and pi.codpizza =
r1.codpizza)
group by pi.codpizza, pi.nome, pi.tempoprep
having count(*) >= 4
```