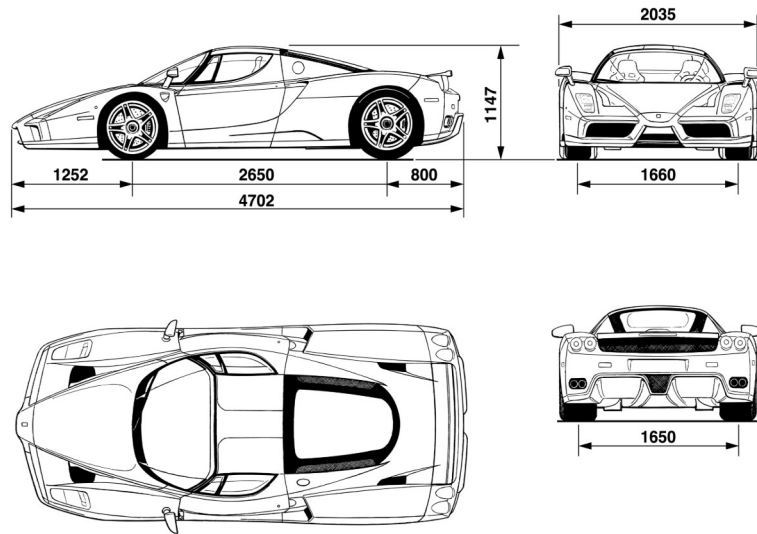




3.Relational Model



- 1. Database Fundamentals*
- 2. Database Design*
- 3. Relational Model*** 
- 4. Derivation Rules*
- 5. SQL Language Fundamentals*

Modelli di database

MODELLO	TIPO DI MODELLO	COSTRUTTI	DERIVAZIONE
concettuale	Entità-relazioni	E-R 	Algoritmi di derivazione
logico	Relazionale	Tabelle	SQL
fisico	DBMS Relazionale	Tabelle fisiche	



Modello Relazionale

- *E' un modello logico, basato sul concetto matematico di relazione*
- *Relazione = tabella*
 - **tabella** (concetto intuitivo)
 - **relazione** (concetto matematico, proprio dalla teoria degli insiemi, da non confondere con la relazione del modello concettuale)

Focus: il significato di relazione

Livello concettuale (modello ER)

- Relazione= legame tra entità

Livello logico (modello Relazionale)

Relazione = Tabella



Id	Name	SurName	Age
1	Jodie	Tucker	34
2	Jayden	Archer	56
3	Grace	Wheeler	18
4	Freddie	Humphries	56

Indipendenza fisica

Il modello relazionale è quello sui cui sono basati tutti gli **RDBMS**: quanto diremo vale quindi per tutti i database basati su tale modello **logico**



PostgreSQL



SQLite



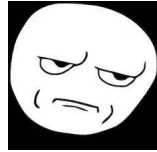
Microsoft®
SQL Server®



MySQL™

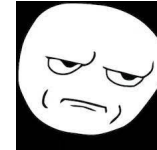
Terminologia

- Una base di dati relazionale è costituita da un insieme di relazioni .



Relazioni=tabelle

Una relazione è costituita da un insieme di tuple.



Tupla=riga

- Attributi, campi, colonne: intestazioni delle colonne.
 - Ex nome, cognome, età

Notazione per le tabelle

- Lo **schema logico** di una tabella lo possiamo rappresentare con la notazione:

NomeTabella(Attrib.1, Attrib.2,...,Attrib.N)

CORSI(Cod. Corso, Titolo, Docente)

Questa notazione è simile a quella che verrà utilizzata a **livello fisico** per creare la tabella corsi

SQL

```
create table corsi(cod_corso int, titolo char(20), docente char(20));
```


Documentazione

- *Un DB lo possiamo rappresentare con la notazione:
NomeDB={Tab.1, Tab.2,...,Tab.N}*
- *Ex Lo schema LOGICO del database "SCUOLA" è costituito da tre tabelle: CORSI, STUDENTI, ESAMI*

*SCUOLA={CORSI(Cod. Corso, Titolo, Docente),
STUDENTI(Matricola, Cognome, Nome),
ESAMI(Studiante, Voto, Corso)
}*

Valore Nullo

*Il modello relazionale (e quindi i DBMS basati su tale modello) ci mette a disposizione uno strumento detto valore **NULLO** (NULL) per tutti i casi in cui potrebbe:*

- essere **non disponibile** il valore
- essere **inesistente** il valore
- essere **sconosciuto** il valore

IL valore NULL indica l'assenza di informazione

*Il valore NULL **non** va racchiuso tra gli apici, esattamente come in Java*

Vincoli di Integrità

- ❑ *I vincoli sono delle regole sui valori dei dati e costituiscono una "barriera" a protezione dell'integrità della base di dati*
- ❑ *Base di dati integra=consistente*



- ❑ *Il DBMS impedisce del tutto l'inserimento di una tupla (riga) che non rispetti tutti i vincoli*
 - Il mancato rispetto dei vincoli viene opportunamente segnalato dal DBMS



Vincoli di Integrità

*E' possibile che i valori assunti dagli attributi di una relazione debbano soddisfare dei vincoli per avere un significato, per fornire l'informazione corretta e tutelare quindi **l'integrità** della base dei dati.*

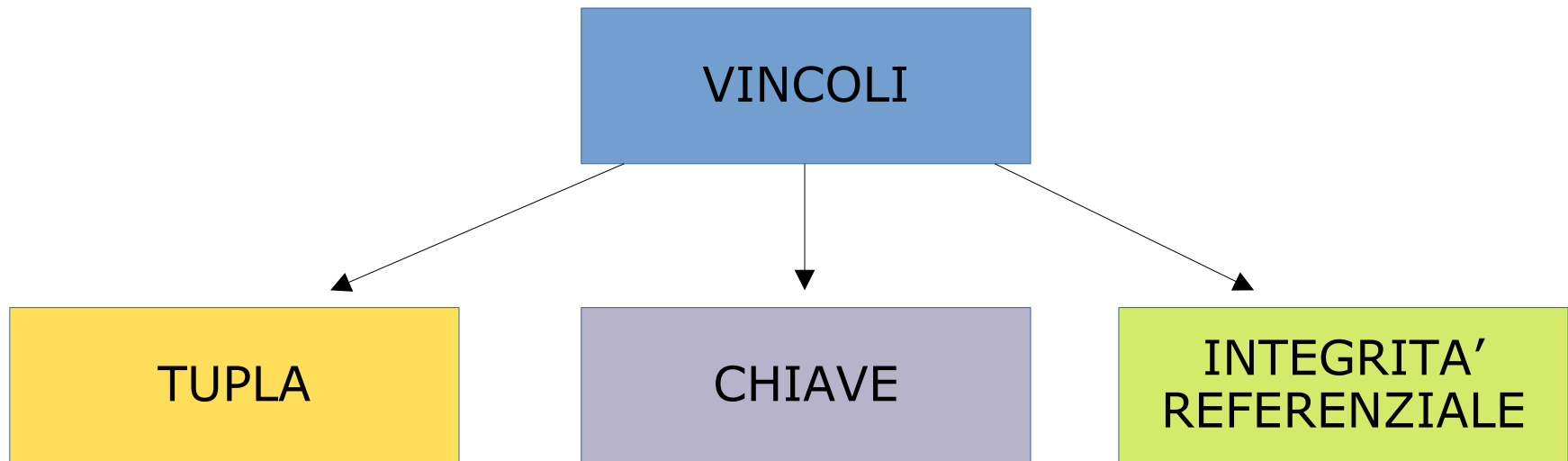
Esempio:

- il voto deve essere ≥ 18 e ≤ 30
- l'età deve essere > 0
- Inserire lo username è necessario
- due identificatori(chiavi) non devono essere uguali
- Se uno studente abbandona definitivamente l'università bisogna restituirgli il diploma e cancellare tutti gli esami che ha fatto (esercizio università)

Tipi di vincoli di Integrità

I vincoli di integrità sono di tre tipi:

- vincoli di tupla
- vincoli di chiave
- vincoli di integrità referenziale



Vincoli di tupla

I vincoli di tupla sono vincoli

- Sul **formato** di un attributo
 - `CHECK (length(targa) =7)`
- Sul **range** di valori ammissibili per quell'attributo
 - `CHECK (età >18)`
- Sull'**obbligo di esistenza** di quell'attributo
 - `NOT NULL`

*Esprimono condizioni sui valori di una tupla **indipendentemente** dalle altre tuple della tabella e delle altre tabelle.*

- ❑ *Il DBMS impedisce del tutto l'inserimento di una tupla che non rispetti tutti i vincoli.*



Esempi:

ESAMI(Studente, Voto, Corso)

- *(Voto ≥ 18) AND (Voto ≤ 30)*

UTENTE(username,età, CF)

- *username NOT NULL*
- *età > 0*
- *Codice Fiscale di lunghezza 16*

Vincoli di Chiave

Un vincolo di chiave è un vincolo di unicità

*Definire un vincolo di chiave su uno o più attributi di una relazione significa imporre che il valore di quell'attributo/i debba essere unico all'interno di quella relazione: il vincolo di chiave è quindi un **vincolo di unicità**.*

Matricola	Cognome	Nome	Nascita	Corso
123	Bianchi	Pino	74	I.I.
124	Rossi	Pino	80	I.I.
125	Verdi	Lino	78	I.I.

Nel modello CONCETTUALE, non è obbligatorio individuare attributi identificatori o attributi ad esistenza obbligatoria. La loro esistenza dipende solo dalla realtà che sto modellando.

- Se nel modello CONCETTUALE ho individuato attributi unici e/o obbligatori,
 - Nel modello LOGICO questi attributi avranno i vincoli di unicità (u) e/o di esistenza (*)
 - Nel modello FISICO (SQL) verranno definiti con UNIQUE e/o NOT NULL

Focus : Vincolo di tupla VS vincolo di chiave

TUPLA

*Esprimono condizioni sui valori di una tupla **indipendentemente** dalle altre tuple della tabella e delle altre tabelle.*

- Sul **formato** di un attributo
- Sul **range** di valori ammissibili per quell'attributo
- Sull'**obbligo di esistenza** di quell'attributo

CHIAVE

*Essendo vincoli di unicità esprimono condizioni sui valori di una tupla il cui inserimento **dipende** dalle altre tuple della tabella e delle altre tabelle*

- Sono solo I vincoli di unicità

Vincoli e sviluppo applicazioni

I vincoli di tupla possono essere implementati non solo sul db ma anche lato App

Vantaggi:

- Meno occupazione di rete
- meno stress sul DBMS
- app più “responsive”

Svantaggi

- Sarà necessario scrivere più codice

I vincoli di chiave possono essere implementati solo lato db



Focus:Campo ID

*Nella pratica (livello fisico), in tutte le tabelle si prevederà un campo **id***

- di tipo intero
- positivo
- **non nullo**
- **unico**
- autoincrementato dal DBMS



*Un attributo di questo genere si chiamerà **chiave primaria** e garantirà sempre il rispetto del vincolo di unicità e di esistenza.*

Focus:Campo ID

- A livello FISICO , a causa dei limiti dei DBMS , posso avere una sola PRIMARY KEY per ogni tabella: questa sarà **sempre l'id**



- Spoiler: a livello fisico su MySQL

```
id int unsigned primary key auto_increment
```

- Spoiler: a livello fisico su Oracle

```
id NUMBER GENERATED BY DEFAULT ON NULL  
AS IDENTITY PRIMARY KEY
```

Un vincolo di Chiave primaria
implica quindi che i valori che essa
assume
siano:

- Unici
- Non nulli

Istanza logica

città

id	nome	kmq
3	Roma	120
5	Milano	80
6	Ladispoli	20
7	Fregene	20

abitante

id	nome	cognome	id_città
150	Rino	Rano	6
151	Nino	Rossi	3
152	Gino	Verdi	5
155	Lino	Rossi	3

Relazioni?
Attributi?
Tuple?

Vincoli di Integrità Referenziale

I valori di **id_città** della tabella **Abitanti** devono esistere come chiave primaria nel campo **id** della tabella **Città**

città

id	nome	kmq
3	Roma	120
5	Milano	80
6	Ladispoli	20
7	Fregene	20

abitante

id	nome	cognome	id_città
150	Rino	Rano	6
151	Nino	Rossi	3
152	Gino	Verdi	5
155	Lino	Rossi	3

Istanza logica

Studenti

Matricola	Nome	Cognome
150	Mario	Rossi
151	Nino	Bianchi

Corsi

CODICE	NOME
3	FISICA1
5	CANTO

Esami

Matricola_studente	Codice_corso	Voto
150	5	18
151	5	30
151	3	20

Relazioni?
Attributi?
Tuple?

Relazioni tra tuple

E' possibile stabilire corrispondenze tra righe di tabelle differenti mediante valori comuni

Studenti

Matricola	Nome	Cognome
150	Mario	Rossi
151	Nino	Bianchi

Corsi

CODICE	NOME
3	FISICA1
5	CANTO

Esami

Matricola_studente	Codice_corso	Voto
150	5	18
151	5	30
151	3	20

Relazioni?
Attributi?
Tuple?

Vincoli di Integrità Referenziale

I valori di **matricola_studente** della tabella **Esami** devono esistere come chiave primaria nel campo **Matricola** della tabella **Studenti**

- I valori di **codice_corso** della tabella **Esami** devono esistere come chiave primaria nel campo **Codice** della tabella **Corsi**

- Un vincolo di Integrità referenziale (foreign key) fra un insieme di attributi **X** di una tabella **T1** e un'altra Tabella **T2** è soddisfatto se
- i valori su **X** di ciascuna tupla dell'istanza di **T1** compaiono come valori della chiave (primaria) dell'istanza di **T2**.

Focus: Vincoli di integrità referenziale

La colonna `id_città` è detta **Foreign Key**

Nella pratica tutte le tabelle del DB avranno una colonna `ID` (Primary Key) e questa colonna verrà utilizzata per relazionare le righe di tabelle differenti mediante l'uguaglianza

Tab1.PrimaryKey = Tab2.ForeignKey
(ex Città.**id** = Abitanti.**id_città**)

Spoiler: dal logico al fisico

ttà

id	nome	kmq
3	Roma	120
5	Milano	80
6	Ladispoli	20

id	nome	cognome	id_città
150	Rino	Rano	6
151	Nino	Rossi	3
152	Gino	Verdi	5

```
table città(  
  id int unsigned primary key auto_increment,  
  nome varchar(20),  
  kmq int)  
  
create table abitante(  
  id int unsigned primary key auto_increment,  
  nome varchar(20),  
  cognome varchar(20),  
  id_città int unsigned,  
  foreign key(id_città) REFERENCES città(id);
```

Riepilogo

- 1) Quali costrutti mette a disposizione il modello relazionale per rappresentare i dati?
- 2) Cos'è una relazione
 - a livello logico
 - a livello concettuale
- 3) Da cosa è costituito uno schema relazionale?
- 4) Cosa rappresenta il valore `null`?
- 5) A cosa servono i vincoli?
- 6) Quali sono i tipi di vincolo?
- 7) Come si comporta un DBMS quando tento di inserire una tupla che non rispetti i vincoli

Operatori logici

*A **AND** B è vero se e solo se A e B sono veri, altrimenti è falso.*

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

*A **OR** B è vero se A oppure B sono veri, è falso solo se A e B sono entrambi falsi.*

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

***NOT**(A) è vero se A è falso e falso se A è vero.*

A	X
0	1
1	0

Operatori logici

- **A AND B** è vero se e solo se A e B sono veri, altrimenti è falso.
- **A OR B** è vero se A oppure B sono veri, è falso solo se A e B sono entrambi falsi.
- **NOT(A)** è vero se A è falso e falso se A è vero.

Operatore **AND**

A	B	$A \wedge B$
F	F	F
F	V	F
V	F	F
V	V	V

Operatore **OR**

A	B	$A \vee B$
F	F	F
F	V	V
V	F	V
V	V	V

Operatore **NOT**

A	$\neg A$
F	V
V	F

Focus: operatori logici

- *Date due condizioni unite da un operatore logico, affinché il risultato finale sia verificato*
 - AND si devono verificare entrambe le condizioni
 - OR: basta che si verifichi almeno una condizione

Operatore AND

A	B	$A \wedge B$
F	F	F
F	V	F
V	F	F
V	V	V

Operatore OR

A	B	$A \vee B$
F	F	F
F	V	V
V	F	V
V	V	V

Operatore NOT

A	$\neg A$
F	V
V	F

Precedenze

1) NOT

2) AND (prodotto logico)

3) OR (somma logica)

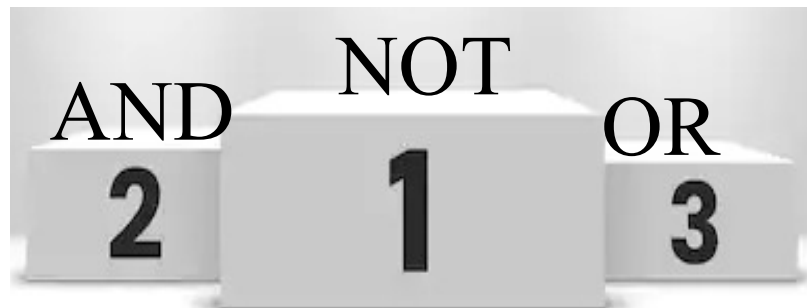
In matematica il prodotto ha la precedenza sulla somma.

Per cambiare le precedenze si utilizzano le parentesi esattamente come in matematica

Ex

$$5+2*3 = 5+(2*3) = 5+6=11$$

$$(5+2)*3= 7*3 = 21$$



Operatori logici

Data la relazione

PERSONE (nome, cognome, età, città)

Scrivere, mediante gli operatori logici, le condizioni di estrazione delle tuple

- 1) Di maggiorenni (soluz: `Persone.età ≥ 18`)
- 2) Di persone tra 18 e 75 anni estremi inclusi
- 3) Di maggiorenni che abitano a Roma
- 4) Di romani o di milanesi
- 5) Di minorenni che sono romani o milanesi
- 6) Di maggiorenni romani oppure di milanesi (a prescindere dalla loro età)
- 7) Di maggiorenni romani oppure di minorenni milanesi
- 8) Di romani di cui non è nota l'età
- 9) Di persone di cui è nota l'età

Operatore AND

A	B	$A \wedge B$
F	F	F
F	V	F
V	F	F
V	V	V

Operatore OR

A	B	$A \vee B$
F	F	F
F	V	V
V	F	V
V	V	V

Operatore NOT

A	$\neg A$
F	V
V	F



Operatori Logici

- 1) `Persone.eta >= 18`
- 2) `Persone.eta >= 18 AND Persone.eta <= 75`
- 3) `Persone.eta >= 18 AND Persone.città="Roma"`
- 4) `Persone.città="Roma" OR Persone.città="Milano"`
- 5) `Persone.eta < 18 AND (Persone.città="Roma" OR Persone.città="Milano")`
- 6) `(Persone.eta >= 18 AND Persone.città="Roma") OR Persone.città="Milano"`
- 7) `(Persone.eta >= 18 AND Persone.città="Roma") OR (Persone.eta < 18 AND Persone.città="Milano")`
- 8) `Persone.città="Roma" AND Persone.eta IS null`
- 9) `Persone.eta IS NOT null`

Note:

- Le parentesi al punto 5 sono obbligatorie
- Le parentesi dei punti 6 e 7 non sono obbligatorie, servono solo ad aumentare la leggibilità

Operatori logici

Operatore **AND**

A	B	$A \wedge B$
F	F	F
F	V	F
V	F	F
V	V	V

Operatore **OR**

A	B	$A \vee B$
F	F	F
F	V	V
V	F	V
V	V	V

Operatore **NOT**

A	$\neg A$
F	V
V	F

- Elencare le triplette di valori di A,B,C per I quali l'espressione

A or B and C

assume il valore `true`

1)Soluzione (A=V,B=F,C=F) ; (A=V,B=F,C=V) ...

- Scrivere codice (in un linguaggio a piacimento) per testare questa espressione (HINT: utilizzare Scanner (Java) o input (Python) ,tre variabili booleane A,B,C e visualizzare il risultato a schermo)
- Ripetere lo stesso esercizio con l'espressione
(A or B) and C
- Confrontare I risultati ottenuti: quale delle due espressioni è true più spesso?



Esercizio

