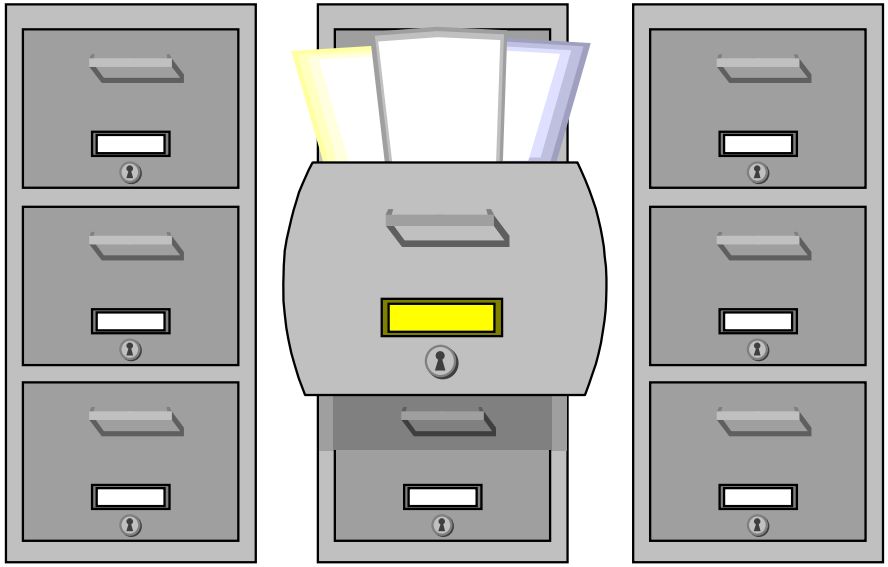
DATA BASE (1)



Problematica gestione dati:

∗ oggetti delle elaborazioni, difficili da gestire, memorizzare, reperire, modificare;

∗ talvolta ridondanti/incongruenti;

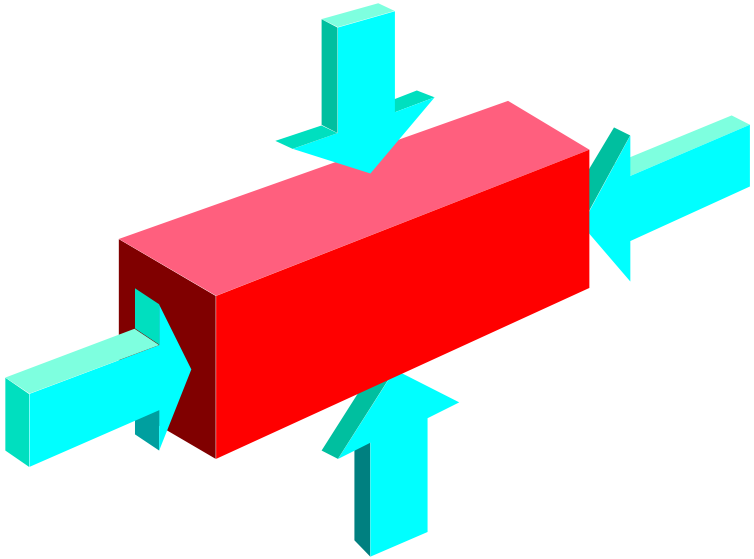
∗ non sufficientemente protetti;

∗ spesso comuni a più applicazioni/utilizzatori e quindi potenziali sorgenti di conflitto.

Terminologia:

- **BANCA DATI**: raccolta di dati, utilizzabili da una molteplicità di utenti;
- **DATA BASE**: tecnica di organizzazione dei dati in una Banca Dati;
- **DATA BASE MANAGEMENT SYSTEM** (**DBMS**): sistemi software per la gestione di banche dati secondo tecniche Data Base.

DATA BASE (2)

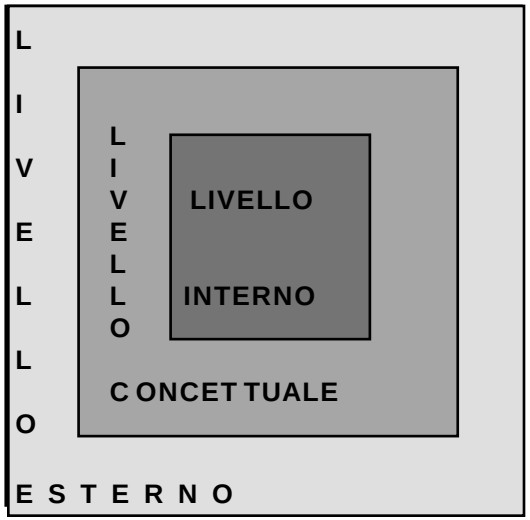


Obiettivi di un DBMS:

- **organizzare e gestire dati** (grandi volumi);
- **facilità di accesso ai dati**;
- **indipendenza dei dati dai programmi** e viceversa;
- **indipendenza dall'ambiente HW/SW** (trasparenza delle applicazioni da supporti e metodi di memorizzazione);
- **condivisione dei dati da più utenti**;
- **salvaguardia dell'integrità** (correttezza) dei dati;
- **protezione da accessi non autorizzati** (gestione della riservatezza);
- **eliminazione della ridondanza** dei dati.

DATA BASE (3)

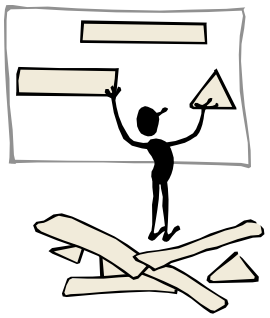
Schema a tre livelli di un DBMS:



- **LIVELLO INTERNO** (o fisico): organizzazione dei dati sui supporti magnetici;
- **LIVELLO CONCETTUALE** (o logico): schema globale dei dati;
- **LIVELLO ESTERNO** (sottoschemi): modalità di accesso ai dati da parte degli utenti, con procedure diverse e visione dei dati spesso parziale.

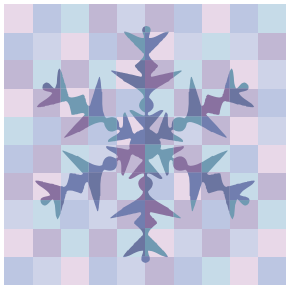
DATA BASE (4)

Modelli di database:



gerarchico *(in disuso; utilizzato come modello interno)*

reticolare *(in disuso; utilizzato come modello interno)*



relazionale *(attuale)*

DATA BASE (5)

Modello relazionale

Basato sulla teoria degli insiemi e su tabelle dette “relazioni”

Esempio: Tabella ≡ Relazione CLIENTI

colonna ≡ campo ≡ attributo

l’ordine delle righe e delle colonne non è significativo

CODICE	NOME	CITTÀ	ZONA
C01	Rossi	Genova	NW
C04	Bianchi	Ancona	CN
C02	Verdi	Roma	CE
C12	Finzi	Firenze	CN
C34	Carli	Torino	NW
C09	Fazio	Milano	NO

CHIAVE PRIMARIA

ATTRIBUTI
(valori dello stesso tipo per ogni colonna)

DATA BASE (6)

Modello relazionale

- **Descrizione dei dati indipendente dalla rappresentazione fisica** (nomi e valori degli attributi, ma non posizione).

- **Ordine delle colonne e delle righe non significativo**.

- **Chiave primaria** = uno o più attributi che identificano univocamente le righe.

- **Valori elementari** non ulteriormente scomponibili.

DATA BASE (7)

Chiave esterna

attributo(i) i cui valori corrispondono a chiavi primarie in altre tabelle
e consentono di mettere in relazione (stabilire associazioni tra) le tabelle

Tabella **CLIENTI**

CODICE	NOME	CITTÀ	ZONA
...	...	...	NW
...	...	...	CN

Tabella **ZONE**

ZONA	DESCRIZIONE
CN	Centro-Nord
NW	Nord-Ovest

DATA BASE (8)

Esempio di chiave esterna

Le tabelle non devono contenere **dipendenze funzionali** *e* **ridondanze** *che pongano problemi di aggiornamento.*

Bisogna ridurre o **normalizzare** *la tabella a relazioni semplici.*

CODICE	NOME	CITTÀ	ZONA	DESCRIZIONE	···
C01	Rossi	Genova	NW	Nord-Ovest	···
C34	Carli	Torino	NW	Nord-Ovest	···

Tabella non normalizzata

È molto meglio avere le due tabelle CLIENTI e ZONE separate, con la chiave esterna che ne consente l'associazione: ciò garantisce l"univocità" del dato, cioé la sua presenza una sola volta in un unico posto.

DATA BASE (9)

Linguaggi di un DBMS

DDL (data definition language)

Orientato alla definizione ed alla **descrizione dei dati** delle tabelle, in particolar modo per quanto riguarda:

- *nome della tabella;*
- *nome dei campi (colonne, attributi);*
- *lunghezza dei campi;*
- *tipi dei campi;*
- *chiave primaria;*
- *viste logiche (sottoinsiemi parziali o "calcolati" di tabelle);*
- *controlli sui valori dei dati;*
- *procedure automatiche da attivare su eventi specifici sui dati ("trigger").*

DATA BASE (10)

Linguaggi di un DBMS

DML (data manipulation language)

Comprende le operazioni di manipolazione dei dati, cioé:

- inserimento (**insert**) di nuove righe nelle tabelle;
- modifica (**update**) dei valori contenuti nelle righe;
- cancellazione (**delete**) di righe.

QL (query language)

Può anche essere visto come parte del DML. Si basa su tre operazioni fondamentali (che costituiscono la cosiddetta “algebra relazionale”):

- selezione di righe su condizione (**select**);
- fusione (**join**) di due tabelle intorno ad un attributo comune;
- estrazione (**project**) di colonne, e produzione di tabelle ridotte eliminando l’eventuale duplicazione di righe.

DATA BASE (11)

Esempi di algebra relazionale

a)**select from CLIENTI where ZONA="NW"**

CODICE	*NOME*	*CITTÀ*	*ZONA*
C01	Rossi	Genova	NW
C34	Carli	Torino	NW

b)**join CLIENTI, ZONE where CLIENTI.ZONA=ZONE.ZONA**

CODICE	*NOME*	*CITTA`*	*ZONA*	*DESCRIZIONE*
			↑	

4+2-1 colonne (una colonna in comune)

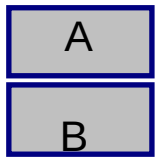
c)**project CITTÀ, ZONA from CLIENTI**

CITTÀ	ZONA
Genova	NW

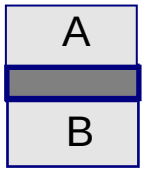
DATA BASE (12)

Esempi di algebra relazionale

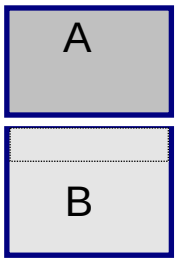
Altre operazioni (su tabelle omogenee, aventi cioè uguale struttura):

- **unione (A∪B):** 

(tutte le righe della tabella A e della tabella B)

- **intersezione (A∩B):** 

(solo le righe comuni alle tabelle A e B)

- **differenza (A-B):**


(le righe della tabella A che non sono nella tabella B)

DATA BASE (13)

- **DML** (Data Manipulation Language) = linguaggio autosufficiente/richiamabile/che richiama altri.

- **QBE** (Query By Example) = linguaggio di interrogazione guidata di dati, di facile e intuitivo utilizzo.

Altre parti di un DBMS:

- generatore automatico di prospetti di stampa (***report generator***)
- generatore automatico formati video per l'input dati (***form generator***)
- **linguaggio procedurale** (di terza generazione)

==> ***linguaggio procedurale di quarta generazione***.

DATA BASE (14)

INDICI E CHIAVI

Indice = tabella ordinata su una o più colonne

- ***univoco*** = ogni valore dell'indice corrisponde ad una sola riga nella tabella;
- altrimenti, si dice che l'indice ammette ***duplicati***.

Ogni riga della tabella indice contiene due attributi: il valore della chiave, e la corrispondente posizione nella tabella principale ==> permette la ***ricerca binaria*** *(max* $\boldsymbol{\log_2 N}$ *tentativi).*

Indice

NOME	Posizione
Bianchi	2
Carli	5
Fazio	6
Finzi	4
Rossi	1
Verdi	3
...	...

Tabella CLIENTI

...	NOME	...
...	Rossi	...
...	Bianchi	...
...	Verdi	...
...	Finzi	...
...	Carli	...
...	Fazio	...
...	...	...

DATA BASE (15)

IL LINGUAGGIO SQL

- SQL = Structured Query Language.

- Proposto da IBM (E.Codd) e successivamente standardizzato dall'ANSI nel 1992 (SQL-92);
  versioni attuali:
  ISO/IEC 9075:1992, "Information Technology --- Database Languages --- SQL"
  ANSI X3.135-1992, "Database Language SQL"
  versione in corso di standardizzazione: SQL-3

- Non solamente "query", ma tutte le funzionalità DDL, DML e QL.

DATA BASE (16)

IL LINGUAGGIO SQL

Comandi principali DDL

create table *tabella* (*attributo tipo*(*lunghezza*), ...);

```
create table PERSONALE(
  MATRICOLA char(5) not null,
  COGNOME char(30),
  NOME char(20),
  CODFISC char(16),
  ASSUNTO date,
  LIVELLO smallint,
  STIP_BASE integer);
```

DATA BASE (17)

IL LINGUAGGIO SQL

Tipi degli attributi:

- **char(n)** = stringa alfanumerica esattamente di “n” caratteri;
- **varchar(n)** = stringa alfanumerica di al massimo “n” caratteri;
- **bit** = valore “logico” (1=vero 0=falso);
- **tinyint, smallint, integer** = interi di varie dimensioni (8, 16, 32 bit, con segno);
- **decimal(m,n)** = intero a virgola fissa (“m” cifre totali, “n” dopo la virgola);
- **real** = numero floating-point a precisione singola (tipicamente 32 bit);
- **float** = numero floating-point a doppia recisione (tipicamente 64 bit);
- **datetime** = data e ora.

DATA BASE (18)

La parola chiave **not null** *indica l'obbligatorietà di un valore per l'attributo: ciò implica che l'inserimento successivo di una riga nella tabella non avente alcun valore per l'attributo sarà rifiutato.*

La possibilità (laddove previsto) di inserire valori nulli comporta però una complicazione nella valutazione delle espressioni di tipo logico basate su colonne del database.
Ad es.: (STIP_BASE>0) or (DATA_ASS>#01-01-1999#) = V/F/?

LOGICA A TRE VALORI

A	B	not B	A and B	A or B
?	?	?	?	?
?	vero	falso	?	vero
?	falso	vero	falso	forse

DATA BASE (19)

IL LINGUAGGIO SQL

Comandi principali DDL

- **alter table** *tabella*
  - **add** *attributo tipo;*
  - **drop** *attributo;*
  - **modify** *attributo tipo;*

```
alter table PERSONALE add (NASCITA date) before ASSUNTO;
```

- **create** [**unique**] **index** *indice* **on** *tabella (attributo, ...*);

```
create unique index I1 on PERSONALE(MATRICOLA);
create index I2 on PERSONALE(COGNOME, NOME);
```

DATA BASE (20)

Altri comandi DDL

- **drop** **table** *tabella;*
  (cancellazione di tabelle o indici)
  index *indice*;

```
drop table PERSONALE;
drop index I1;
```

Comandi per modifica dati (DML)

- **insert into** *tabella* **values** (*valore1, valore2, ...)*;

```
insert into PERSONALE values ('AB541', 'ROSSI', 'Ernesto', 'RSSRNS48M20R341E', '20/08/1948', 5, 1780000);
```

- **update** *tabella* **set** *attributo=valore* **where** *condizione*;

```
update PERSONALE set LIVELLO=6 where MATRICOLA='AB541';
```

- **delete from** *tabella* **where** *condizione*;

```
delete from PERSONALE where MATRICOLA='AB541';
```

DATA BASE (21)

Comando QL generalizzato di estrazione di dati

select *attributo1, ...* (***** = tutti) **from** *tabella* **where** *condizione*;

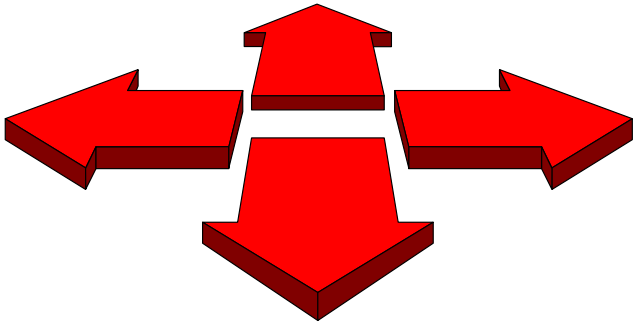
```
select NOME, NASCITA from PERSONALE where LIVELLO=5;
```

Clausole opzionali:

- **order by** (ordinamenti sui vari attributi)
- **group by** (risultati intermedi)
- **count**(*****) (numero totale di righe nella selezione)
- **count** (**distinct** *attributo*) (numero di valori distinti e non nulli dell'attributo nella selezione)
- **sum**(*attributo*)
- **sum**(**distinct** *attributo*)
- **min**(*attributo*)
- **max**(*attributo*)
- **avg**(*attributo*)

DATA BASE (22)

Comando SELECT



Operatori di selezione:

- where *attributo* **between** *valore_min* **and** *valore_max*
- *where attributo* **in** *(insieme_di_valori)*
- *where attributo* **like** *"modello"*

  (in "modello" è possibile utilizzare i caratteri jolly % = gruppo di caratteri e _ = singolo carattere; in Access sono però * e ?)

DATA BASE (23)

Esempi di istruzioni SQL

```
update PERSONALE
  set STIP_BASE = STIP_BASE*1,05
  where LIVELLO <= 3;
```

```
select *
  from PERSONALE
  where ASSUNTO > '31/12/1980'
  order by COGNOME, NOME;
```

```
select *
  from PERSONALE
  where ASSUNTO between '01/01/1980' and '31/12/1985'
    and PROV in ('MI', 'BA', 'FG)
     or COGNOME like 'ROS%';
```

DATA BASE (24)

Esempi di istruzioni SQL

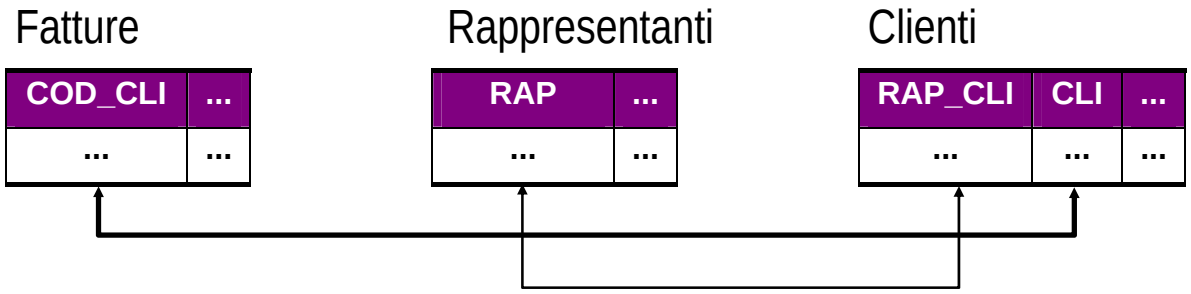
```
select COGNOME, NOME, DESCRIZIONE
  from PERSONALE, REPARTI
  where PERSONALE.REPARTO = REPARTI.COD_REP
    and STIP_BASE = (
            select max(STIP_BASE)
               from PERSONALE
               where FUNZIONE = 'IMPIEGATO'
                     )
  order by NASCITA;
```

```
delete
  from PERSONALE
  where (LIVELLO=5) and (DATA_ASS<#01-01-1980#);
```

DATA BASE (25)

Esempi di istruzioni SQL – il JOIN

```
select r.RAP, sum(f.IMPORTO)
  from ((FATTURE f join CLIENTI c on f.COD_CLI=c.CLI)
       join RAPPRESENTANTI r on c.RAP_CLI=r.RAP)
  where f.COD_CLI = c.CLI
    and c.RAP_CLI = r.RAP
  group by r.RAP
  order by 1;
```



DATA BASE (26)

VISTE LOGICHE

Tabelle virtuali = *finestre dinamiche* sui dati

create view *tabella* ***as select*** *attributi* **from** *tabella* **where** *condizione*;

```
create view CLIENTI_PUGLIA as
select CODICE, NOME, CITTÀ
   from CLIENTI
   where PROVINCIA in ('BA','BR','FG','LE','TA');
```

DATA BASE (27)

SICUREZZA

Comandi di gestione accessibilità:

- **grant** (per concedere permessi sugli “oggetti” del database)
- **revoke** (per negare accessi).

QBE: QUERY BY EXAMPLE

Le operazioni di Query (essenzialmente *select*) vengono richieste inserendo le condizioni nelle colonne degli attributi

Esempio:

	CODICE	ZONA	
	>300	**=NW**	

DATA BASE (28)

NORMALIZZAZIONE

Tre regole fondamentali:

a) ***ogni attributo ha un solo valore per riga*** (I forma normale);

b) ***... e ogni attributo non chiave dipende dall'intera chiave*** (II forma normale);

c) ***... e direttamente dalla chiave*** (III forma normale).

DATA BASE (29)

NORMALIZZAZIONE

FORN	***PROD***	CITTÀ	DESCR	Q.TÀ
F1	P1	MILANO	DADO	30
F1	P2	MILANO	RUOTA	20
. .	. .	. .	. .	. .
F8	P9	TORINO	CHIODO	50

Anomalie:

- un nuovo fornitore non può “esistere” se non fornisce un prodotto;
- la cancellazione di una fornitura (per es. F8, P9) può implicare la sparizione di altri dati (per es. una città, un fornitore o un prodotto);
- se un fornitore cambia città, bisogna aggiornare tutte le righe relative.

CITTÀ *dipende da* **FORN**
DESCR *dipende da* **PROD**

DATA BASE (30)

NORMALIZZAZIONE

CLI	NOME	CITTÀ	AGENTE
C1	ALFA	MILANO	ROSSI
...	...	...	...

Anomalie:

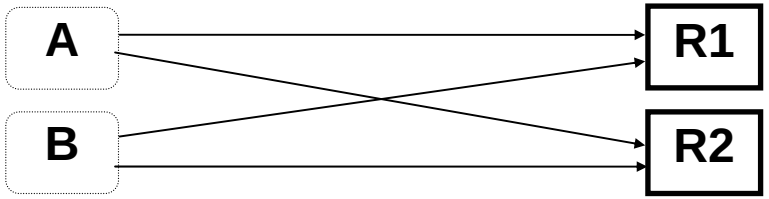
- un nuovo agente non può essere inserito, se non si acquisisce un cliente;
- la cancellazione di un cliente, può trascinarsi quella di un agente;
- la sostituzione di un agente, implica l'aggiornamento di tutte le righe relative.

AGENTE *dipende da* **CITTÀ** *che dipende da* **CLI**
(**dipendenza transitiva**)

DATA BASE (31)

INTEGRITÀ DEI DATI

1. ***Integrità semantica*** *(valori corretti, controllo di range, procedure di controllo coerenza eventuali dupicazioni, etc.), vincoli funzionali/contestuali, evolutivi.*
2. ***Integrità di contesa****:*
   - *gestione degli accessi e blocco risorse contese;*
   - *problema dead-lock che si può verificare se più operazioni accedono a più risorse.*



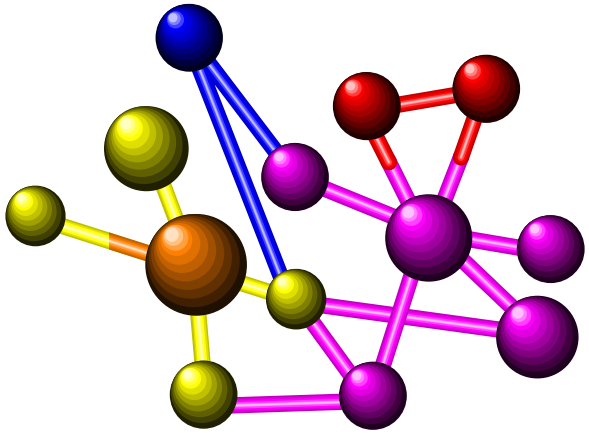
3. ***Integrità referenziale****:* i riferimenti tra tabelle (chiave esterna) devono essere corretti ed esistenti.

Ad esempio, se si cancella un cliente non vi devono essere operazioni che si riferiscono a quel cliente (neanche di tipo storico statistico); stesso discorso per fornitori, articoli, etc. Se ad un codice cliente si cambia il cliente reale, si può provocare una mescolanza tra operazioni del vecchio e del nuovo cliente.

4. ***Integrità di entità:*** un attributo che partecipa ad una chiave primaria non può avere valori nulli.

DATA BASE (32)

LIMITI DEI DATABASE RELAZIONALI



- **attributi elementari** (vietate matrici, record, etc.);
- **righe non *varianti***, cioé i dati non possono dipendere da altri attributi ==> proliferazione di tabelle e/o di attributi.
- **tabelle normalizzate** ==> alto numero di tabelle.

Tutti gli effetti tendono ad accrescere la ***complessità*** in termini di tabelle e corrispondenze tra esse.

Le operazioni su una molteplicità di tabelle comportano ***inefficienze*** ed incrementano il rischio di blocchi, specie su database distribuiti.