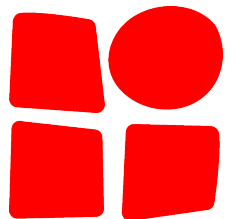
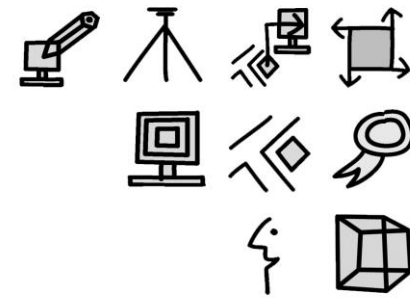


Banche dati

Introduzione alle basi di dati





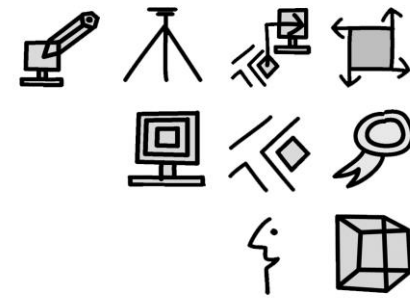
Marco Bernasocchi

- MSc in GIS @ UNIZH
- Director, Developer, Consultant, Teacher & foremost Geek
- Skier, Climber, Diver
- marco@opengis.ch



OPENGIS.ch
ANDROID · [Q]GIS · WEB



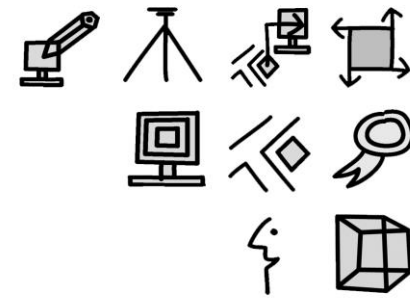


Sistemi informativi e basi di dati

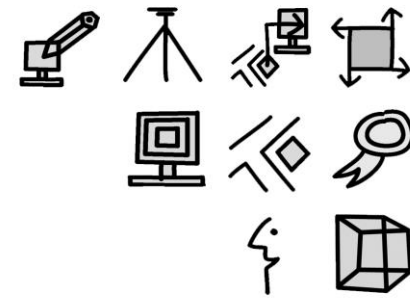
- Ogni organizzazione ha bisogno di memorizzare e mantenere informazioni specifiche. Per esempio:
 - Utenze telefoniche
 - Conti correnti bancari
 - Studenti iscritti a un corso di laurea
 - Quotazioni di azioni nei mercati telematici



Sistemi informativi e basi di dati



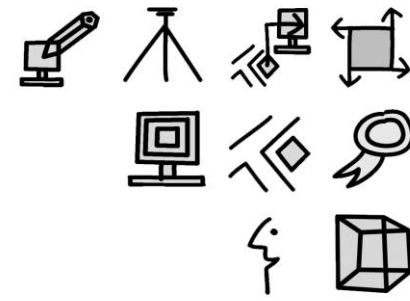
- I sistemi informativi organizzano e gestiscono le informazioni necessarie alle attività di un'organizzazione:
 - Inizialmente non erano automatizzati (per esempio, gli archivi bancari)
 - Informatica → gestione automatica dei dati → **basi di dati**
 - Informazione memorizzata in modo rigoroso



Informazione e dati

- Rappresentazione dell'**informazione**:
 - Basata su codifica (interpretata da programma)
 - **Dati** = elementi di informazione, che di per sé non hanno interpretazione
 - Mario Rossi → *nome e cognome*
 - 2334455 → *numero matricola*



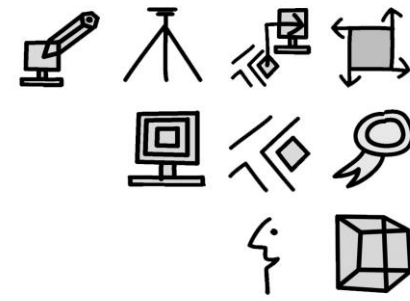


Dati e applicazioni

- I dati possono variare nel tempo (per esempio, importo conto corrente)
- Le *modalità* con cui i dati sono rappresentati in un sistema sono di solito stabili
- Le *operazioni* sui dati variano spesso (per esempio, ricerche)

separare i dati dalle applicazioni che operano su essi

Esempio: Rappresentazione astratta di utenze telefoniche



■ Utente

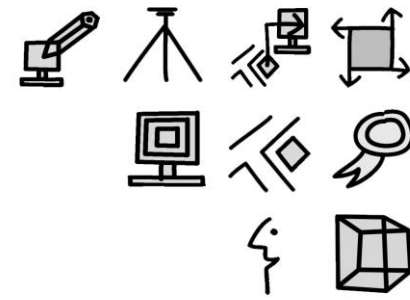
- Codice
- Cognome
- Nome
- Comune
- ListaTelefonate

■ Telefonata

- NumeroTelefonico
- OrarioInizio
- OrarioFine
- NumeroScatti



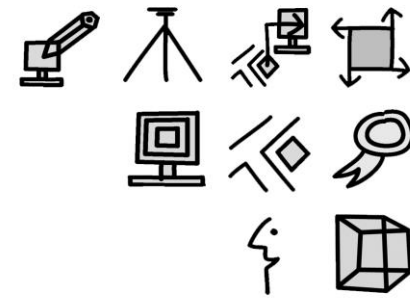
Differenze tra foglio elettronico e database (1)



- Foglio elettronico: software che simula un foglio di carta utilizzato per catalogare dati e creare grafici basati sui dati
- Database: insieme di dati correlati tra loro e logicamente strutturato



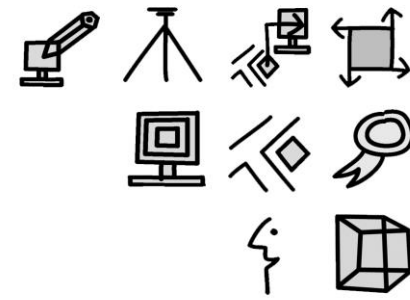
Differenze tra foglio elettronico e database (2)



- Foglio elettronico: piccole quantità di dati
- Database: grandi quantità di dati

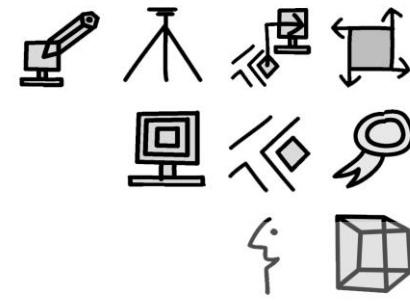


Differenze tra foglio elettronico e database (3)



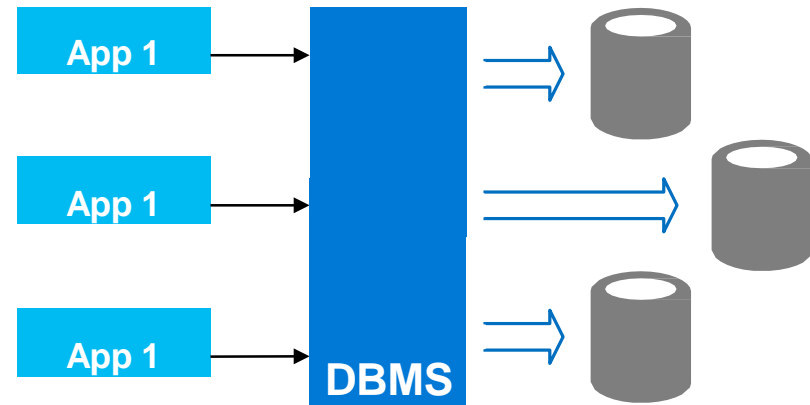
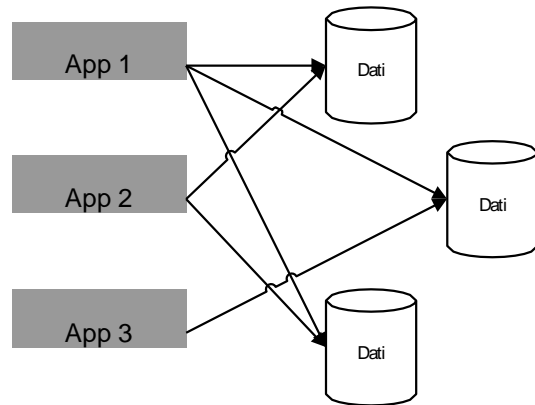
- Foglio elettronico: accesso effettuato da persone
- Database: accesso trasversale
 - Possibilità di accesso da più persone
 - Possibilità di accesso da più applicazioni

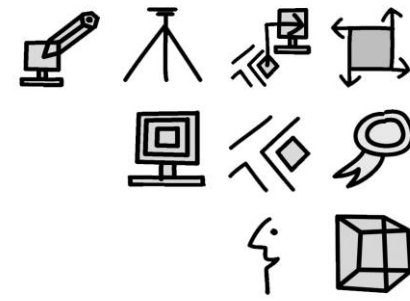




DBMS

- I DBMS sono strumenti software che hanno il compito di gestire le informazioni contenute in un database in maniera efficiente ed efficace.

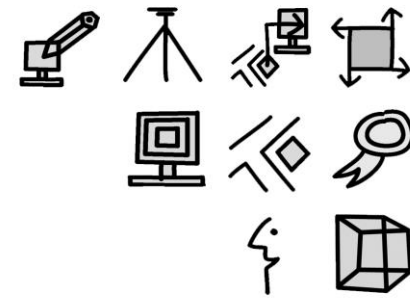




Basi di Dati (DB) e DBMS

- Basi di dati = collezione di dati per rappresentare informazioni di interesse:
 - *grandi,*
 - *condivise,*
 - *persistenti*





Basi di Dati (DB) e DBMS

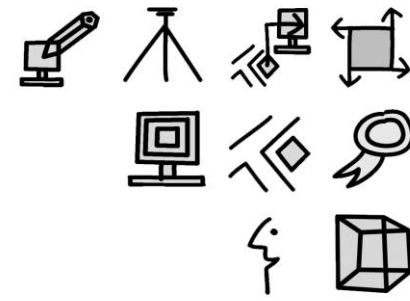
- DBMS

= Data Base Management System

= software in grado di gestire collezioni di dati

- Un DBMS deve essere: *affidabile, sicuro, efficiente, efficace*

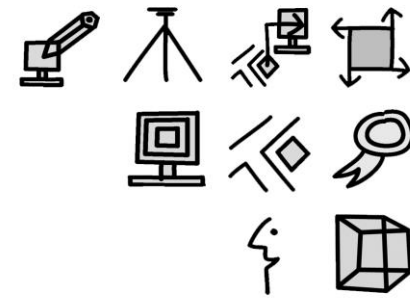




Caratteristiche dei DB e DBMS

- **Grandi** = miliardi di byte (gestione in memoria secondaria)
- **Condivisibili** = diverse applicazioni e utenti possono accedere a dati comuni
 - Evitare le *ridondanze*
 - Aggiornamenti agevoli
 - Evitare le *inconsistenze* dovute agli accessi concorrenti

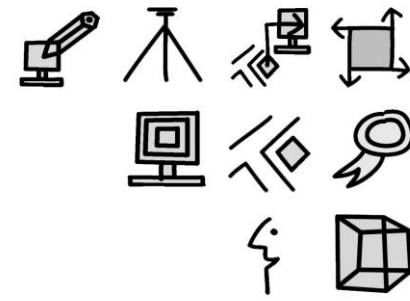




Caratteristiche dei DB e DBMS

- **Persistenza** = dati sempre disponibili, non “vivono” dentro una sola applicazione
- **Affidabilità** = protezione dei dati, in caso di guasto HW o SW capacità di ripristinare i dati (almeno parzialmente)
- **Privatezza** = abilitazioni diverse a seconda dell'utente
- **Efficienza** = tempi di risposta e occupazione spazio accettabili (dipende molto dalla tecnica di memorizzazione dei dati)
- **Efficacia** = facilitare l'attività di organizzazione



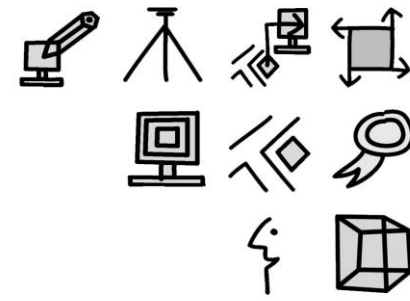


Basi di dati vs. file system

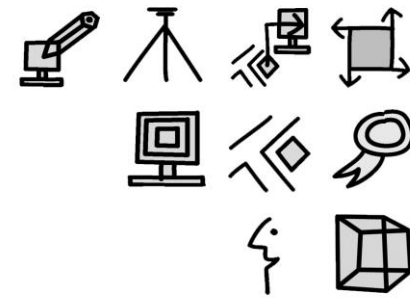
- Tecnicamente, i DB sono collezioni di file:
 - Si potrebbero usare direttamente i file per memorizzare i dati...
 - ...ma si perdono le buone proprietà garantite dai DBMS
 - Normalmente, file pensati per una specifica applicazione e non per servire più di una applicazione, magari in parallelo



Come organizzare i dati in un DB (Data Base)

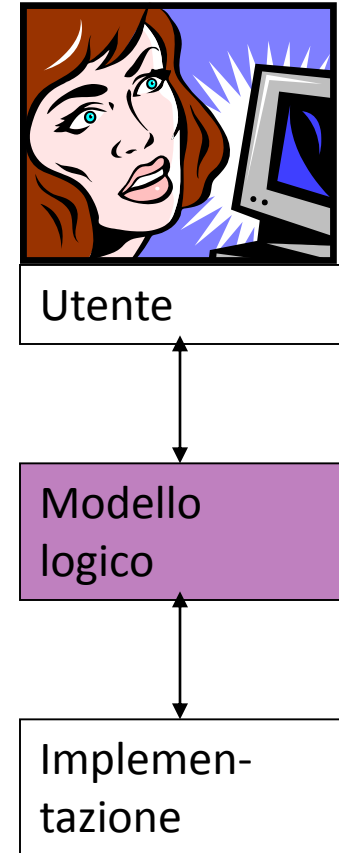


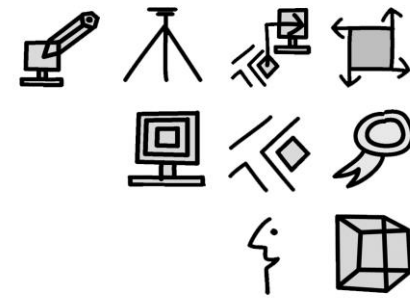
- Insieme di concetti per organizzare le informazioni di interesse e descriverne la struttura
- Meccanismi per strutturare tipi di dati complessi a partire da tipi semplici
 - Per esempio:
data = <giorno, mese, anno>



Modello logico dei dati

- Sottointende una specifica rappresentazione dei dati (tabelle, alberi, grafi, oggetti...)
- Descrive i dati a un livello intermedio, tra ciò che vede l'utente e il livello dell'implementazione
- Molte proposte in DBMS commerciali

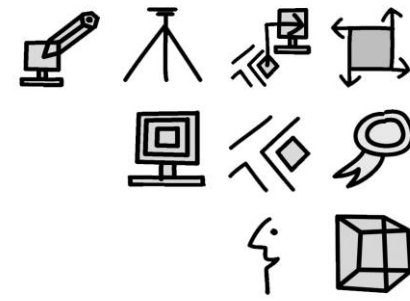




Modello concettuale

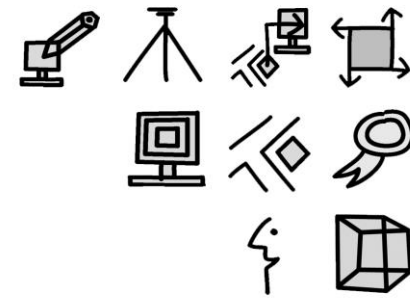
- Per la progettazione di un DB: analisi della realtà di interesse
- Modello astratto
- Indipendente dal modello logico
- Modello **concettuale** = rappresentazione dei **concetti**
- Modello **logico** = rappresentazione dei **dati**





Modello logico dei dati

- Modello **gerarchico** (anni '60) = struttura gerarchica (albero)
- Modello **reticolare** (inizio '70) = struttura a grafo
- Modello **relazionale** (fine '70) = struttura a tabelle
- Modello **orientato agli oggetti** (anni '80) = struttura a classi/oggetti

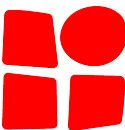


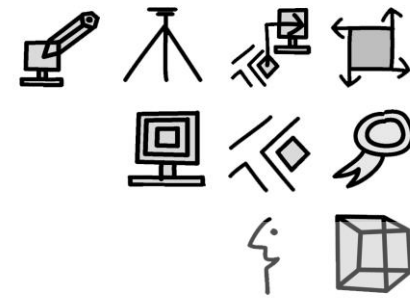
Modello logico dei dati

- Modello **gerarchico** (anni '60) = struttura gerarchica (albero)
- Modello **reticolare** (inizio '70) = struttura a grafo
- Modello **relazionale** (fine '70) = struttura a tabelle
- Modello **orientato agli oggetti** (anni '80) = struttura a classi/oggetti

Per esempio:
Microsoft
Access

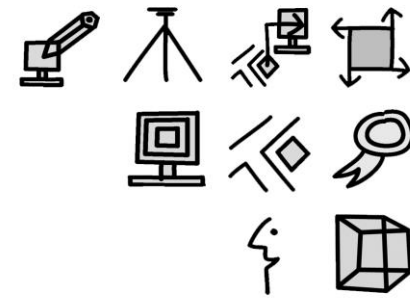
a





- «Il termine *Relational database management system* (RDBMS) indica un *database management system* basato sul **modello relazionale**»
- La struttura base del **modello relazionale** è composta da:
 - uno o più **attributi** (denominati campi);
 - un **tipo di dato** o **dominio** per ciascuno degli attributi;
 - un **valore** per ciascun attributo all'interno del dominio o tipo di dato consentito;
 - una **tupla** cioè l'insieme non ordinato di valori assunti dagli attributi.





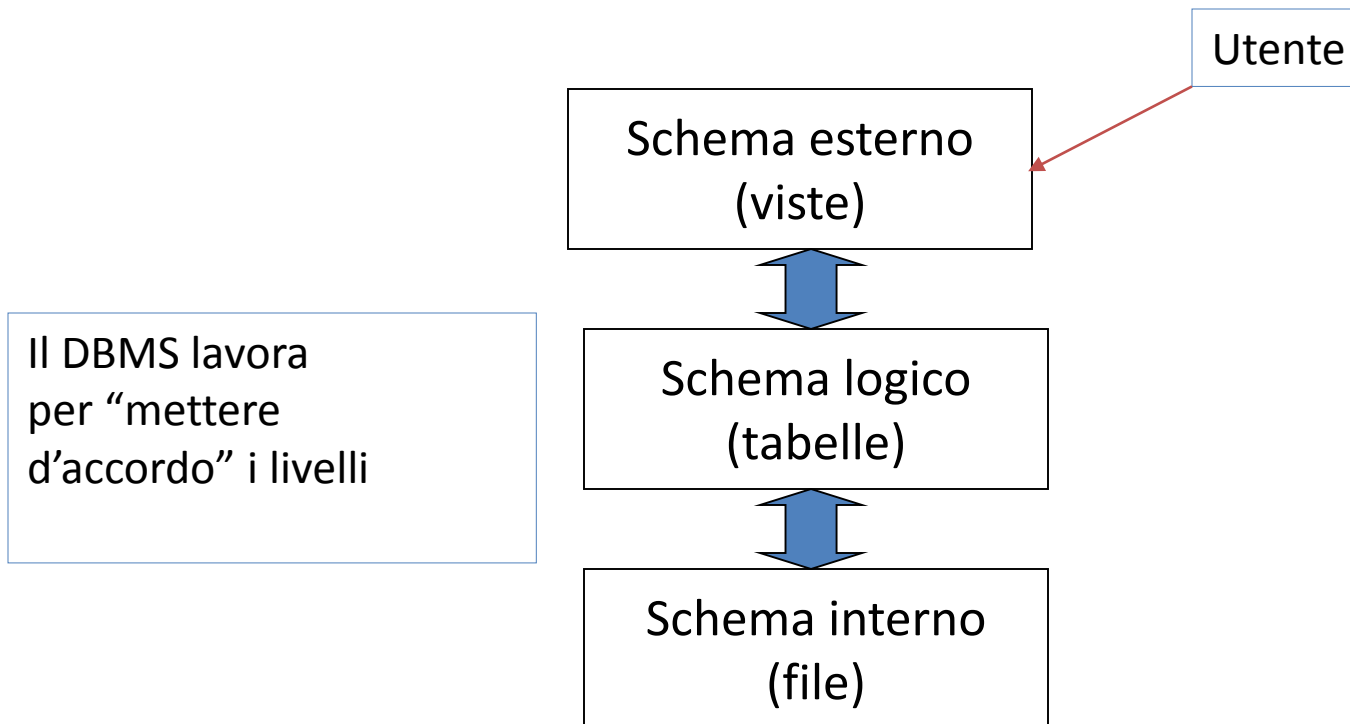
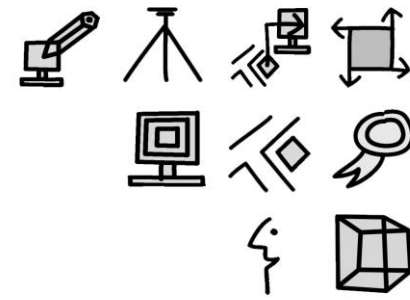
Livelli di astrazione nel DBMS

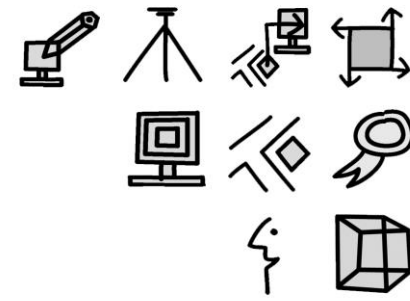
Architettura standard su 3 livelli:

- **Schema esterno:** descrizione di una porzione del DB (per vedere i dati da punti di vista diversi a seconda dell'utente)
- **Schema logico:** descrizione del DB tramite le strutture-dati del modello logico del DBMS (per esempio, le tabelle del modello relazionale)
- **Schema interno:** “mapping” tra schema logico e strutture fisiche di memorizzazione (file)



Livelli di astrazione del DBMS

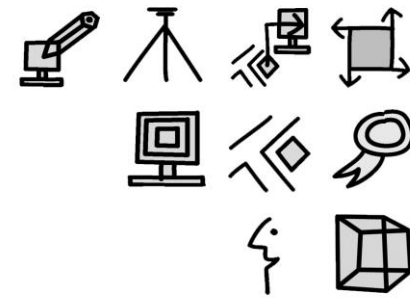




Indipendenza dei dati

- Garantita dai livelli di astrazione:
 - **Indipendenza fisica:** permette di interagire con DB in modo indipendente da struttura fisica in cui sono memorizzati i dati
 - Se i dati vengono riorganizzati fisicamente, le applicazioni funzionano lo stesso
 - **Indipendenza logica:** permette di accedere al DB in modo indipendente da struttura logica dei dati (per esempio, tabelle)
 - Modificare schema logico senza modificare le viste (livello esterno)
 - Estendere le viste senza alterare livello logico

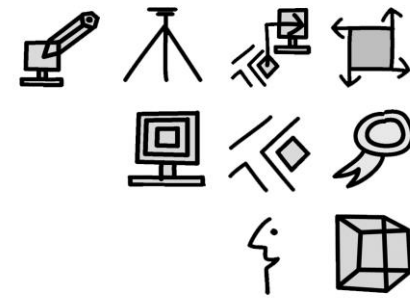




Linguaggi per DB

- **Data Definition Language (DDL)** = definisce schemi fisici, logici, esterni del DB (tratta anche le autorizzazioni di accesso)
- **Data Manipulation Language (DML)** = per formulare interrogazioni e aggiornamenti delle istanze del DB

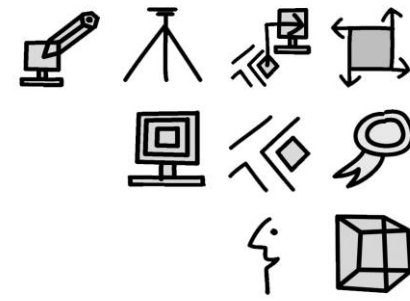




DDL e DML relazionali

- Due paradigmi:
 - Dichiarativo → **SQL** (Structured Query Language)
 - Procedurale → **algebra relazionale**
- Varie proposte commerciali (non esiste un vero “standard SQL”, sintassi un po’ diverse)

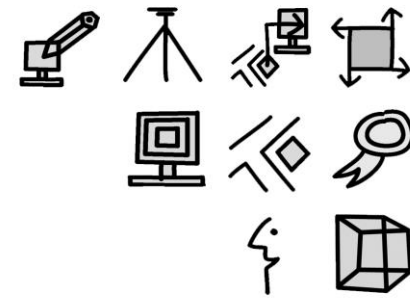




Accesso ai dati

- Mediante linguaggi testuali (per esempio, SQL)
- Tramite comandi speciali integrati nei linguaggi di programmazione
- Tramite interfacce “amichevoli” (per esempio, Wizards, in Access, ecc.)

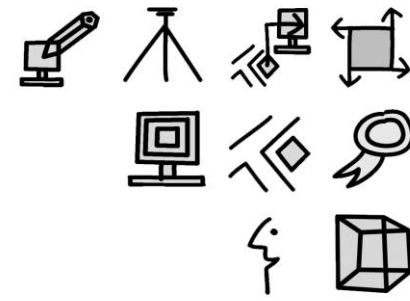




Utenti di un DB

- *Amministratori* del DB: progetta, controlla e amministra il DB
- *Progettisti e programmatori* di applicazioni: sviluppano i programmi che interagiscono con DBMS
- *Utenti*: usano il DB per trovare le informazioni di interesse (possono essere + o – esperti)

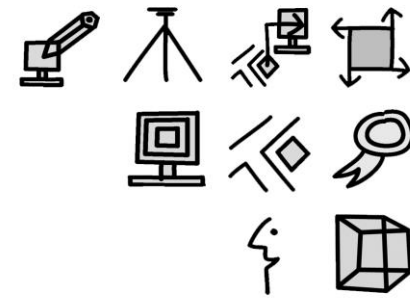




Vantaggi dei DBMS

- Permettono di considerare i dati come risorsa di un'organizzazione
 - Una risorsa comune: a disposizione di molteplici utenti e applicazioni
- Offrono un modello formale della realtà di interesse
 - Preciso, riutilizzabile

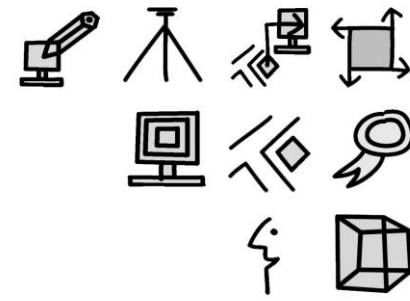




Vantaggi dei DBMS

- Controllo centralizzato dei dati
 - Riduzione di ridondanze e inconsistenze
- Indipendenza dei dati
 - Sviluppo di applicazioni flessibili e modificabili

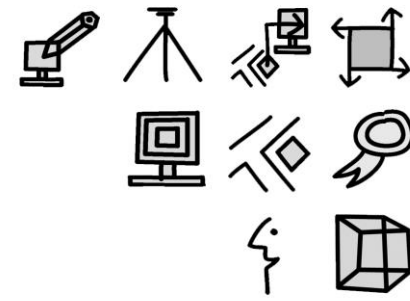




Svantaggi dei DBMS

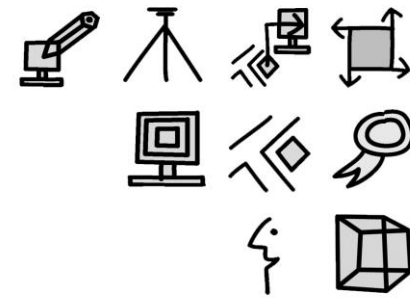
- Complessi, costosi, hanno specifici requisiti in termini di SW e HW
- Difficile separare servizi utili da quelli inutili
- Inadatti alla gestione di poche informazioni condivise da un numero basso di utenti





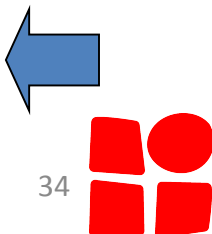
Transazione

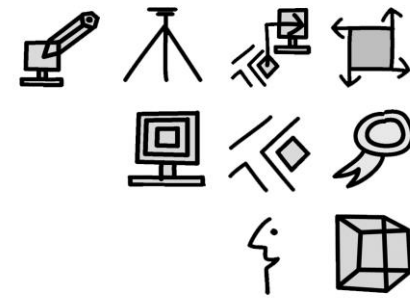
- Insieme di operazioni da considerare indivisibile ("atomico"), corretto anche in presenza di concorrenza e con effetti definitivi



Le transazioni sono ... atomiche

- La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente:
 - trasferimento di fondi da un conto A ad un conto B: o si fanno il prelevamento da A e il versamento su B o nessuno dei due

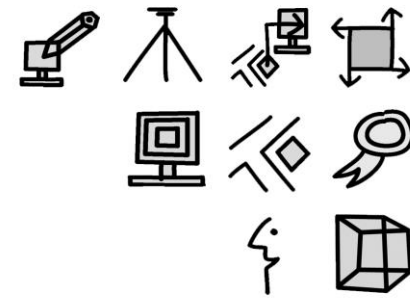




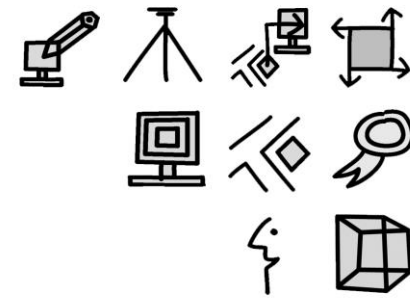
Le transazioni sono ... concorrenti

- L'effetto di transazioni concorrenti deve essere coerente (ad esempio "equivalente" all'esecuzione separata)
 - se due assegni emessi sullo stesso conto corrente vengono incassati contemporaneamente si deve evitare di trascurarne uno

I risultati delle transazioni sono permanenti



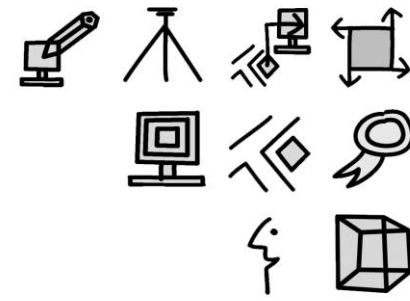
- La conclusione positiva di una transazione corrisponde ad un impegno (in inglese **commit**) a mantenere traccia del risultato in modo definitivo, anche in presenza di guasti e di esecuzione concorrente



Transazioni (per l'utente)

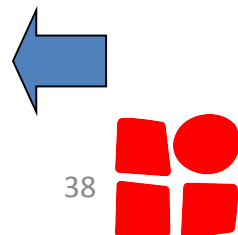
- Programmi che realizzano attività frequenti e predefinite, con poche eccezioni, previste a priori.
- Esempi:
 - versamento presso uno sportello bancario
 - emissione di certificato anagrafico
 - dichiarazione presso l'ufficio di stato civile
 - prenotazione aerea
- Le transazioni sono di solito realizzate in linguaggio ospite (tradizionale o ad hoc)





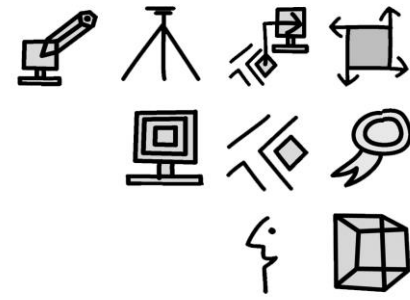
Transazioni, due accezioni

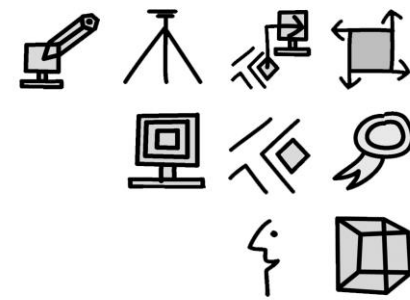
- Per l'utente:
 - programma a disposizione, da eseguire per realizzare una funzione di interesse
- Per il sistema:
 - sequenza indivisibile di operazioni



Tipi di DBMS

- Relational
- noSQL
- Object Oriented
- ...





DBMS stats

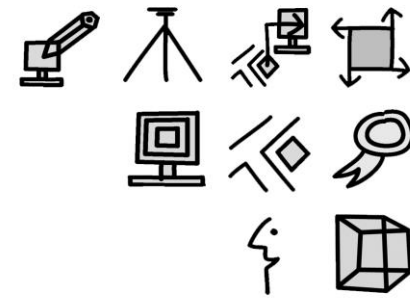
341 systems in ranking, March 2018

| Rank | | | DBMS | Database Model | Score | | |
|----------|----------|----------|------------------------|-------------------|----------|----------|----------|
| Mar 2018 | Feb 2018 | Mar 2017 | | | Mar 2018 | Feb 2018 | Mar 2017 |
| 1. | 1. | 1. | Oracle + | Relational DBMS | 1289.61 | -13.67 | -109.89 |
| 2. | 2. | 2. | MySQL + | Relational DBMS | 1228.87 | -23.60 | -147.21 |
| 3. | 3. | 3. | Microsoft SQL Server + | Relational DBMS | 1104.79 | -17.25 | -102.70 |
| 4. | 4. | 4. | PostgreSQL + | Relational DBMS | 399.35 | +10.97 | +41.71 |
| 5. | 5. | 5. | MongoDB + | Document store | 340.52 | +4.10 | +13.59 |
| 6. | 6. | 6. | DB2 + | Relational DBMS | 186.66 | -3.31 | +1.75 |
| 7. | 7. | 7. | Microsoft Access | Relational DBMS | 131.95 | +1.88 | -0.99 |
| 8. | 8. | ↑ 10. | Redis + | Key-value store | 131.22 | +4.21 | +18.22 |
| 9. | 9. | ↑ 11. | Elasticsearch + | Search engine | 128.54 | +3.23 | +22.32 |
| 10. | 10. | ↓ 8. | Cassandra + | Wide column store | 123.49 | +0.71 | -5.70 |
| 11. | 11. | ↓ 9. | SQLite + | Relational DBMS | 114.81 | -2.46 | -1.37 |
| 12. | 12. | 12. | Teradata | Relational DBMS | 72.46 | -0.53 | -1.07 |
| 13. | 13. | ↑ 17. | Splunk | Search engine | 65.67 | -1.60 | +11.58 |
| 14. | 14. | 14. | Solr | Search engine | 64.81 | +0.94 | +0.82 |
| 15. | ↑ 17. | ↑ 19. | MariaDB + | Relational DBMS | 63.10 | +1.45 | +16.22 |

<https://db-engines.com/en/ranking>



noSQL - evoluzione



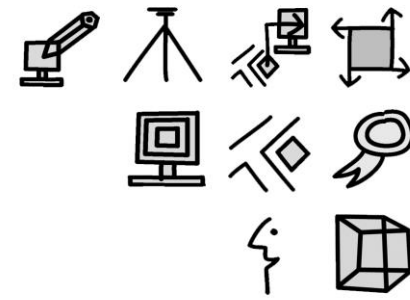
NoSQL è un movimento che promuove sistemi software dove la persistenza dei dati è caratterizzata fatto di **non utilizzare il modello relazionale**.

<http://nosql-database.org/>

- In 2006
 - ▶ **Google** developed and uses **BigTable**
 - ▶ **amazon.com** developed and uses **Dynamo**
 - ▶ **NOKIA** uses **CouchDB**
- In 2009
 - ▶ **YAHOO!** developed and uses **Sherpa**
 - ▶ **LinkedIn** developed and uses **Voldemort**
 - ▶ **facebook** developed and used **Cassandra**, now uses **HBase**
 - ▶ **digg**, **reddit** and **twitter** use **Cassandra**
 - ▶ **The New York Times**, **theguardian**, **bitly**, **github**, **social coding**, and **Springer** use **MongoDB**
- In 2011
 - ▶ **ORACLE** introduced their own NoSQL

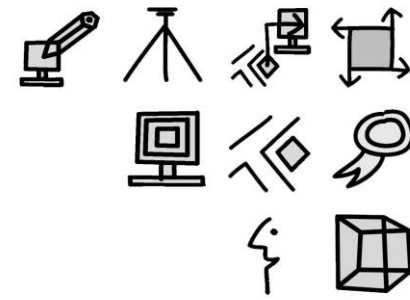


noSQL - Vantaggi



- Soluzione **semplice** (riduzione della complessità)
- Elevata capacità di comunicazione dati (*throughput*)
- **Scalabilità Orizzontale** (scaleout)
- Riduzione costi
- Elimina ORM
- Flessibilità modello dati (**schema-less**)
- Riduzione carico di lavoro DBA

noSQL - Database



Tipologia database noSQL

Key / Value



Column



Cassandra

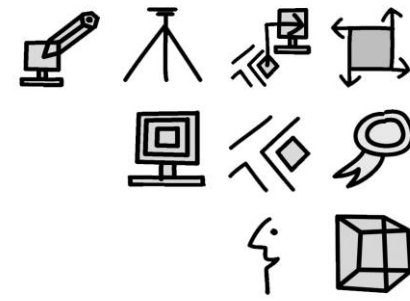


Graph



Document DB





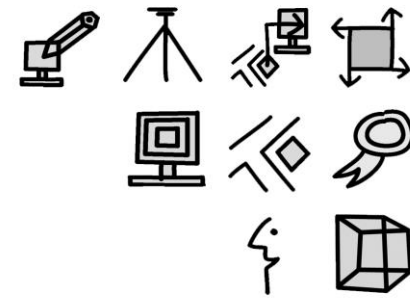
Document DB - MongoDB

MongoDB è un database noSQL open-source di uso generale, con le seguenti caratteristiche:

- **Modello di documento** dati con schemi dinamici
- Supporto di **indicizzazione** pieno e flessibile e ricche query
- Modulo **auto-sharding** per la scalabilità orizzontale
- **Replicazione integrata** per un'elevata disponibilità
- **Ricerca testuale**
- Sicurezza avanzata
- Framework per **aggregazioni** e MapReduce
- Ampio storage di memorizzazione con **GridFS**



MongoDB Trends



noSQL

Termine di ricerca

MongoDB

Software

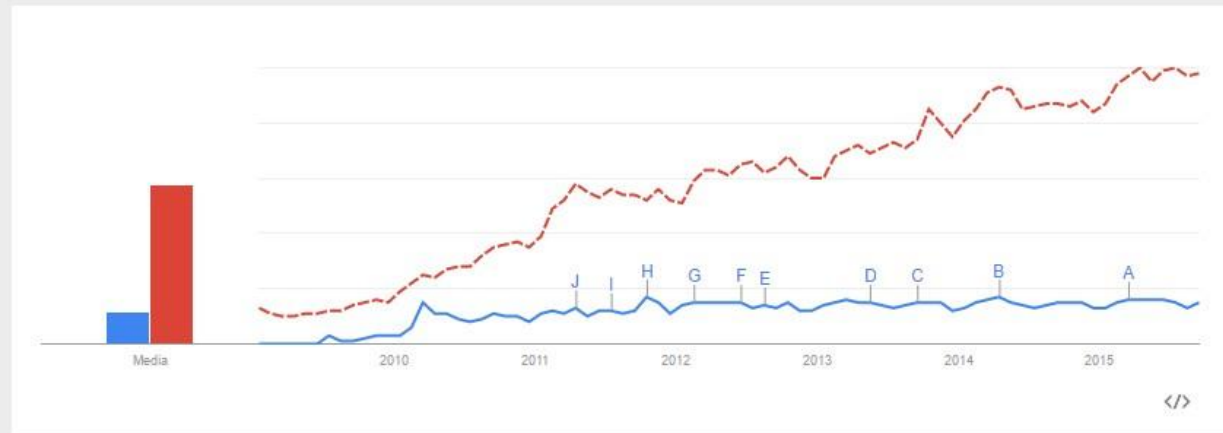
+ Aggiungi termine

Beta: la misurazione dell'interesse di ricerca per gli argomenti è una funzione beta che fornisce rapidamente misurazioni accurate dell'interesse di ricerca generale. Per misurare l'interesse di ricerca per una determinata query, seleziona l'opzione "termine di ricerca".

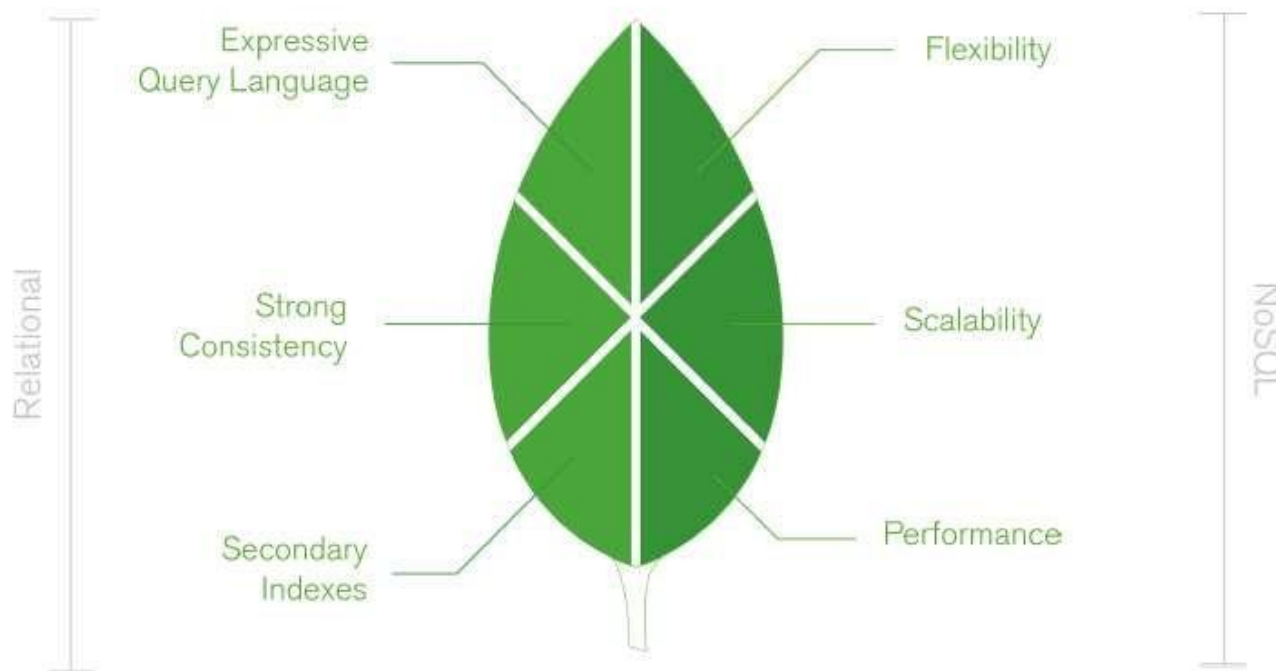
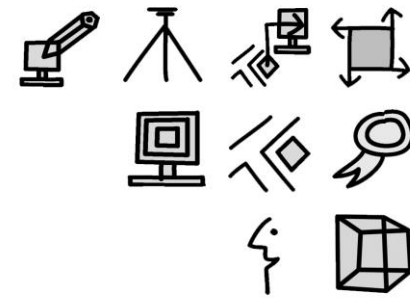
Interesse nel tempo

☒ Intestazioni notizie ☐ Previsione

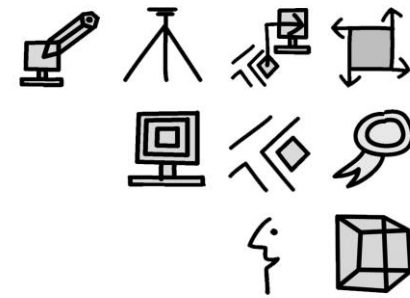
Google Trends



MongoDB



Formato Documenti

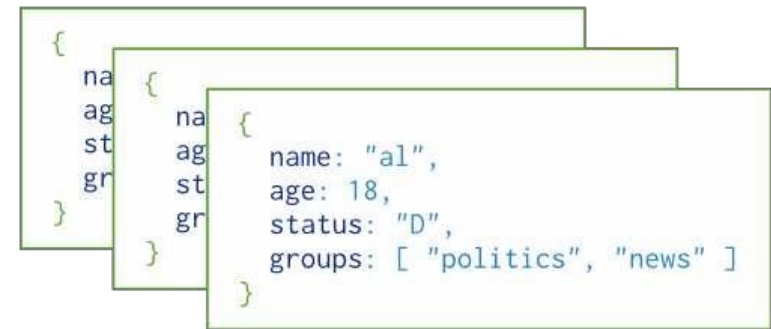


MongoDB archivia i dati sotto forma di documenti in formato JSON-like

nome :valore

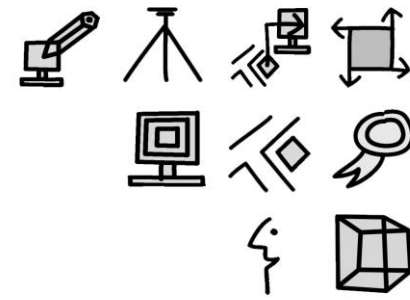
```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

← field: value
← field: value
← field: value
← field: value

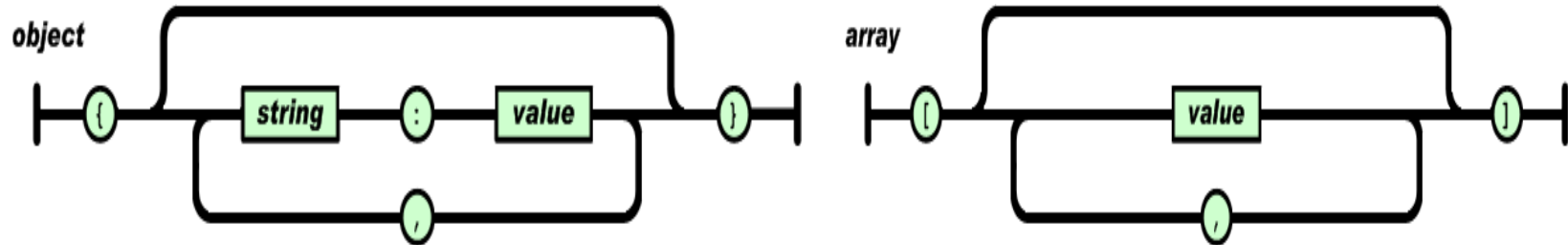


Collection

JSON



JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati.

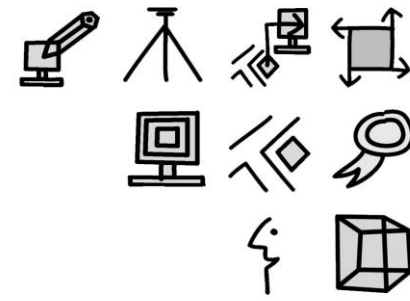


JSON è basato su due strutture:

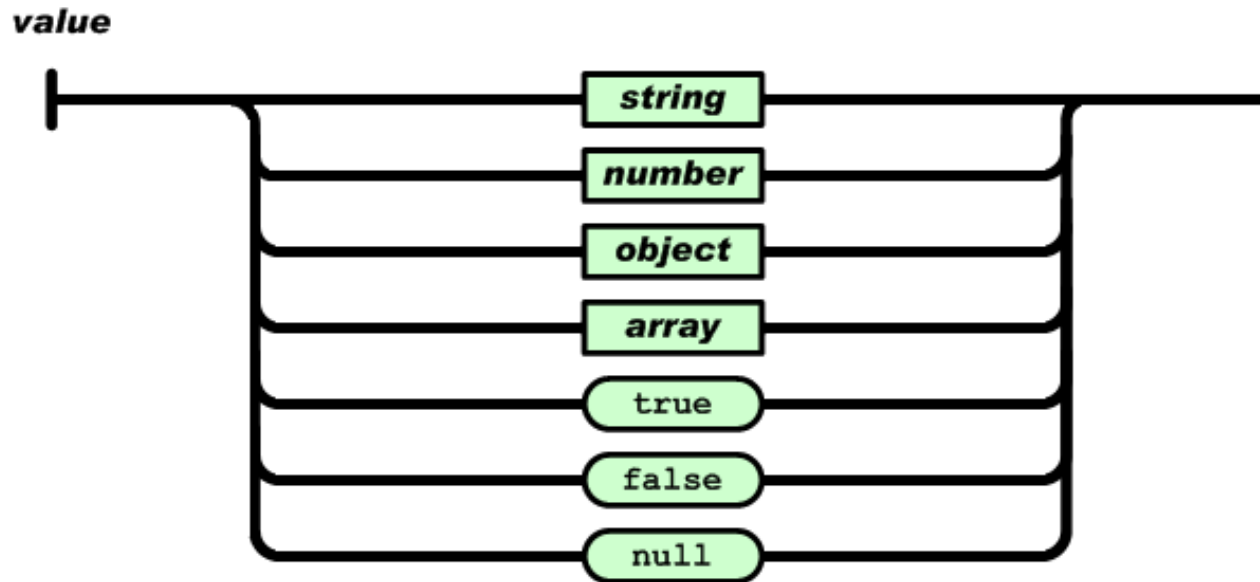
- Un insieme di **coppie nome/valore**. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- Un **elenco ordinato di valori**. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Fonte: <http://www.json.org/json-it.html>

JSON

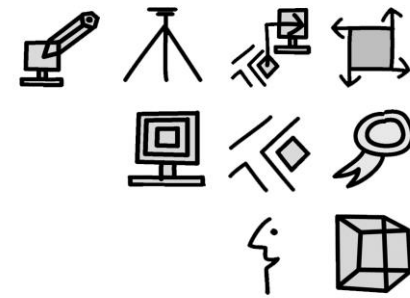


JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati.



Fonte: <http://www.json.org/json-it.html>

CRUD



Collection
↓
`db.users.insert(`
Document
↓
`{`
 `name: "sue",`
 `age: 26,`
 `status: "A",`
 `groups: ["news", "sports"]`
`}`
`)`

Document

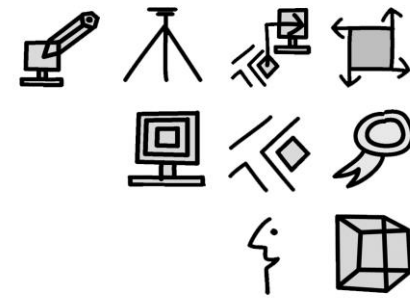
```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

insert →

Collection

| |
|--------------------------------|
| { name: "al", age: 18, ... } |
| { name: "lee", age: 28, ... } |
| { name: "jan", age: 21, ... } |
| { name: "kai", age: 38, ... } |
| { name: "sam", age: 18, ... } |
| { name: "mel", age: 38, ... } |
| { name: "ryan", age: 31, ... } |
| { name: "sue", age: 26, ... } |

users



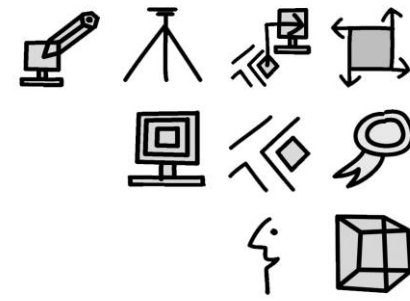
Modello relazionale

- Basato sul concetto di **relazione**
- Relazione = rappresentazione di un'entità complessa tramite **attributi**

Relazione → tabella:

- Colonna = attributo(/campo)
- Riga = valore degli attributi di un individuo appartenente all'entità





Esempio (modello relazionale)

■ Dati relativi a docenti

nomi relazioni

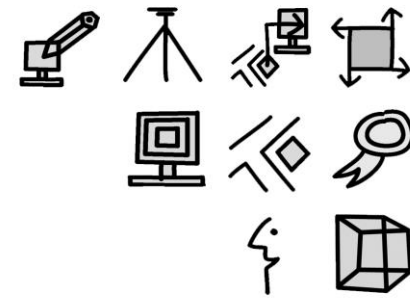
Docenza

attributi

individui

| corso | docente |
|--------------|----------------|
| Informatica | Bianchi |
| Economia | Rossi |
| Architettura | Verdi |





Esempio (modello relazionale)

■ Dati relativi a corsi

nomi relazioni

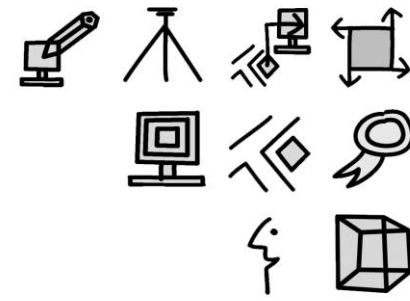
OrganizzCorsidi**Laurea**

attributi

| cdl | materia | anno |
|-------------|--------------------|-------------|
| Informatica | Basi di dati | 2 |
| Matematica | Analisi I | 1 |
| Lettere | Latino | 1 |
| Informatica | Programma zione | 1 |

individui





Esempio (modello relazionale)

- Dati relativi a corsi e docenti (**Dbcorsi**)

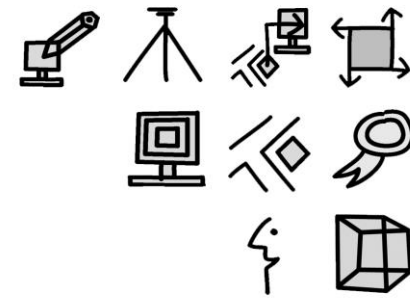
Docenza

| corso | docente |
|--------------|----------------|
| Informatica | Bianchi |
| Economia | Rossi |
| Architettura | Verdi |

OrganizzCorsi di Laurea

| cdl | materia | anno |
|-------------|----------------|-------------|
| Informatica | Basi di dati | 2 |
| Matematica | Analisi I | 1 |
| Lettere | Latino | 1 |
| Informatica | Programmazione | 1 |

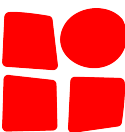
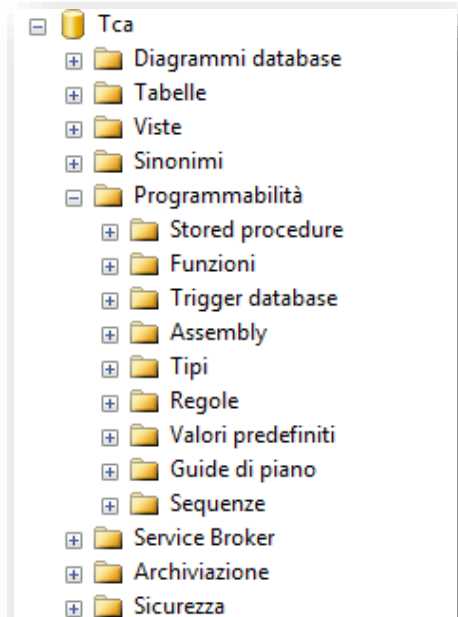
RDBMS - Elementi

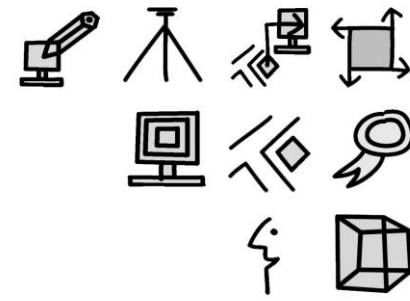


- Tabelle
 - ✓ Chiavi primarie
 - ✓ Chiavi esterne
 - ✓ Tipo di dati
 - ✓ Valori

| | Nome colonna | Tipo di dati | Consenti va... |
|---|----------------|---------------|-------------------------------------|
| 🔑 | IDCollaudatore | int | <input type="checkbox"/> |
| | Collaudatore | nvarchar(200) | <input checked="" type="checkbox"/> |
| | Indirizzo | nvarchar(100) | <input checked="" type="checkbox"/> |
| | IDComune | int | <input checked="" type="checkbox"/> |
| | NumIscriz | int | <input checked="" type="checkbox"/> |
| | DataIscriz | datetime | <input checked="" type="checkbox"/> |
| | DataColl | datetime | <input checked="" type="checkbox"/> |
| | Bloccato | bit | <input checked="" type="checkbox"/> |
| | Motivazione | ntext | <input checked="" type="checkbox"/> |
| | Note | ntext | <input checked="" type="checkbox"/> |

- Viste (Query)
- Procedure
- ...

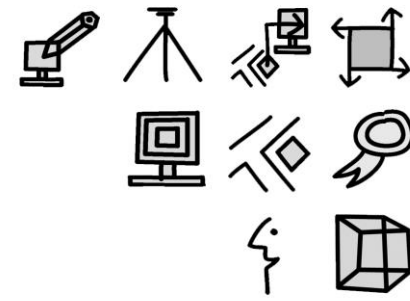




Normalizzazione

« La **normalizzazione** è un procedimento volto all'eliminazione della ridondanza informativa e del rischio di incoerenza dal database »

- Prima Forma Normale → Elimina gruppi ripetitivi
- Seconda Forma Normale → Elimina dati ridondanti
- Terza Forma Normale → Elimina campi non dipendenti dalla chiave

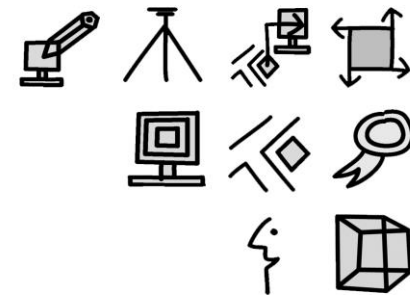


Prima forma normale

Si dice che una base dati è in 1NF (*prima forma normale*) se ogni relazione contenuta nella base dati è in 1NF, ovvero se e solo se:

1. tutte le righe della relazione hanno lo stesso numero di attributi
2. non presenta gruppi di attributi che si ripetono (ossia ciascun attributo è definito su un dominio con valori atomici)
3. tutti i valori di un attributo sono dello stesso tipo (appartengono allo stesso dominio)
4. esiste una chiave primaria (ossia esiste un insieme di attributi, che identifica in modo univoco ogni tupla della relazione)
5. l'ordine delle righe è irrilevante (non è portatore di informazioni)

Fonte: Wikipedia [[https://it.wikipedia.org/wiki/Normalizzazione_\(informatica\)](https://it.wikipedia.org/wiki/Normalizzazione_(informatica))]



Prima forma normale

| prot# | Impresa | Impr_comune | Collaudatore1 | Collaudatore2 | Collaudatore3 |
|-------|--------------|-------------|---------------|---------------|---------------|
| 1022 | Edil Srl | Siracusa | 908A | 486A | 695A |
| 4123 | Costruz. Srl | Catania | 862A | 1602A | 1929A |

Tabella **NON** Normalizzata

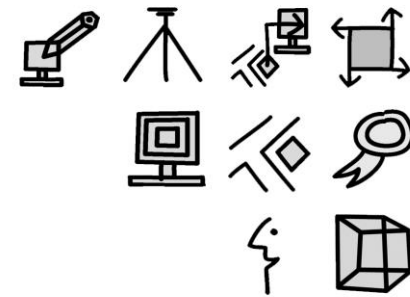


| prot# | Impresa | Impr_comune | Collaudatore |
|-------|--------------|-------------|--------------|
| 1022 | Edil Srl | Siracusa | 908A |
| 1022 | Edil Srl | Siracusa | 486A |
| 1022 | Edil Srl | Siracusa | 695A |
| 4123 | Costruz. Srl | Catania | 862A |
| 4123 | Costruz. Srl | Catania | 1602A |
| 4123 | Costruz. Srl | Catania | 1929A |

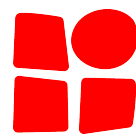
Tabella 1NF:
elimina gruppi ripetitivi



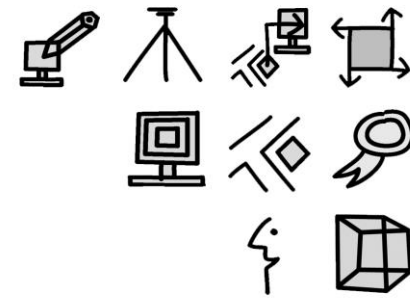
Seconda forma normale



Si dice che una base dati è in **2NF** (*seconda forma normale*), quando è in **1NF** e per ogni relazione tutti gli attributi non chiave dipendono funzionalmente dall'intera chiave composta (ovvero la relazione non ha attributi che dipendono funzionalmente da una parte della chiave).



Seconda forma normale



| prot# | Impresa | Impr_com | Collaud |
|-------|--------------|----------|---------|
| 1022 | Edil Srl | Siracusa | 908A |
| 1022 | Edil Srl | Siracusa | 486A |
| 1022 | Edil Srl | Siracusa | 695A |
| 4123 | Costruz. Srl | Catania | 862A |
| 4123 | Costruz. Srl | Catania | 1602A |
| 4123 | Costruz. Srl | Catania | 1929A |

Tabelle 1NF

Richieste



| prot# | Impresa | Impr_Comune |
|-------|--------------|-------------|
| 1022 | Edil Srl | Siracusa |
| 4123 | Costruz. Srl | Catania |

Tabelle 2NF:
elimina dati ridondanti

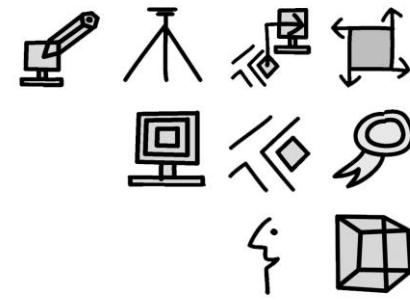
Segnalazioni



| prot# | Collaudatore# |
|-------|---------------|
| 1022 | 908A |
| 1022 | 486A |
| 1022 | 695A |
| 4123 | 862A |
| 4123 | 1602A |
| 4123 | 1929A |



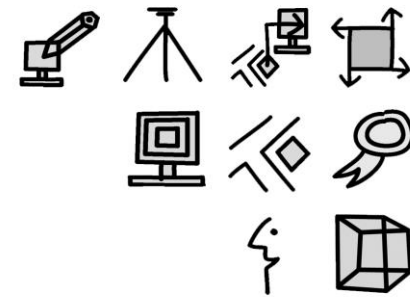
Terza forma normale



Si dice che una base dati è in 3NF (*terza forma normale*) se è in 2NF e tutti gli attributi non-chiave dipendono dalla chiave soltanto, ossia non esistono attributi che dipendono da altri attributi non-chiave.



Terza forma normale



Richieste



| prot# | ID_Imp |
|-------|--------|
| 1022 | 1 |
| 4123 | 2 |

| ID_Imp | Rag_Soc | Comune |
|--------|--------------|----------|
| 1 | Edil Srl | Siracusa |
| 2 | Costruz. Srl | Catania |



Imprese

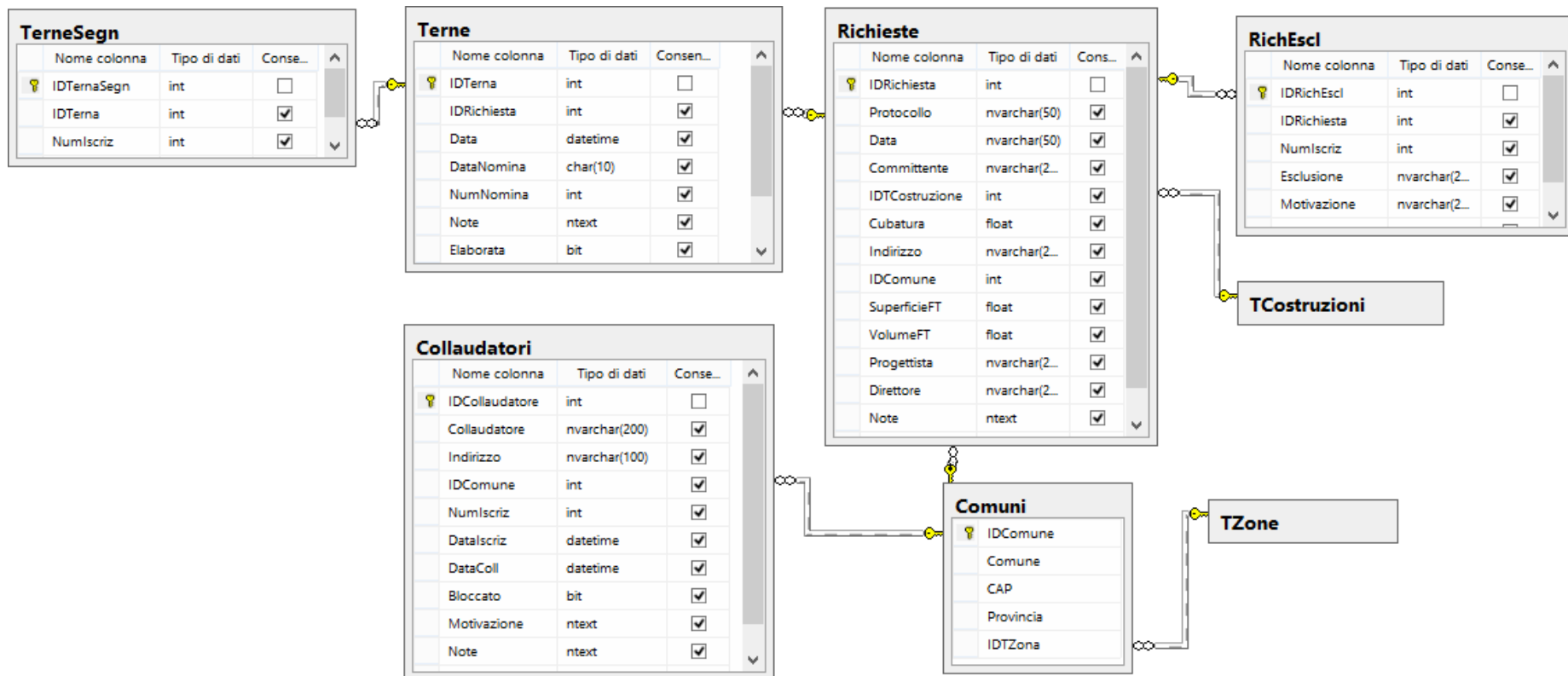
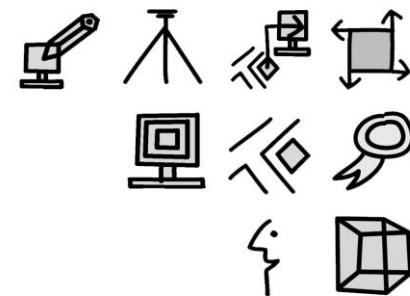
Segnalazioni

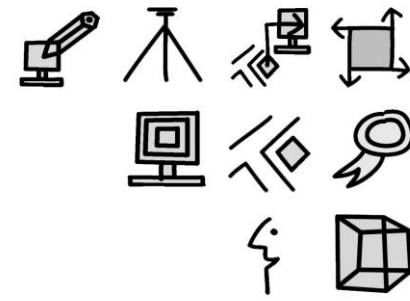


| prot# | Collaudatore# |
|-------|---------------|
| 1022 | 908A |
| 1022 | 486A |
| 1022 | 695A |
| 4123 | 862A |
| 4123 | 1602A |
| 4123 | 1929A |

Tabelle 3NF:
elimina dati non dipendenti
dalla chiave

Schema - Relazioni

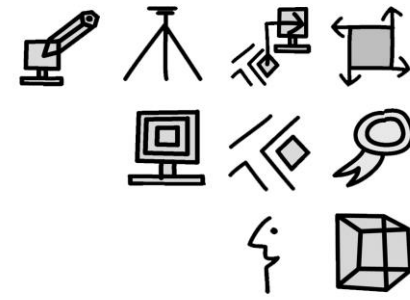




RDBMS - Vantaggi

- **Maturità** della soluzione
- **Consistenza** dei dati
- **Integrità** dei dati
- Uso efficiente dello **spazio** (*se normalizzato*)
- Linguaggio di **interrogazione** (*query*)
- Diffusione e facilità nel trovare supporto tecnico

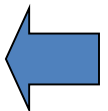
Criticità

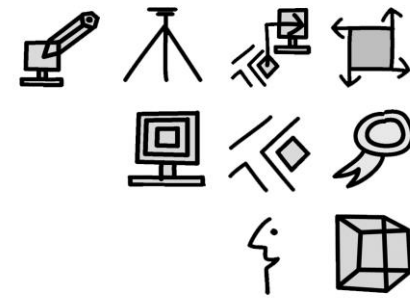


- **Scalabilità verticale** (scaling up)
- Soluzione centralizzata
 - Singolo punto di failure
 - Replica con elevata consistenza dei dati
- **Soluzione rigida** (non flessibile)
- Difficoltà di gestione **dati semi-strutturati** e **non-strutturati**
- **Consumo risorse** per operazioni di join (causa normalizzazione)

RDBMS Prodotti software - esempi

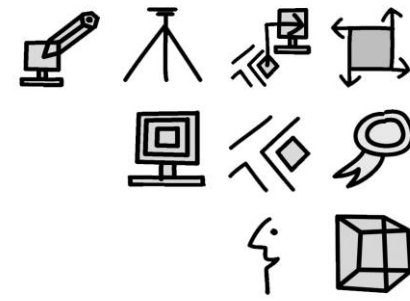
- PostgreSQL – PostGIS (opensource – server)
- MariaDB (MySQL) (opensource – server)
- SQLite (opensource – File based)
- GeoPackage (extends SQLite)
- Access (Filebased)
- DB2 (Server)
- Oracle (Server)
- Informix (Server)
- Sybase (Server)
- SQLServer (Server)





Schema di una relazione

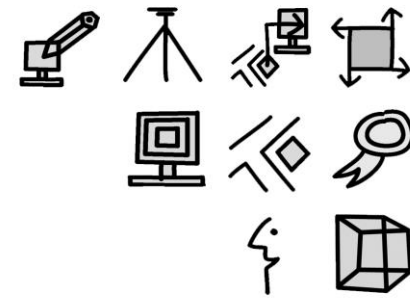
- Schema = definizione della struttura della relazione
- È l'intestazione della relazione:
$$\text{NomeRelazione}(\text{Attr}_1, \dots, \text{Attr}_n)$$
- Non varia nel tempo (modulo ristrutturazione del DB)
- Per esempio: in Dbcorsi
$$\text{Docenza}(\text{corso}, \text{docente})$$



Istanza di una relazione

- Istanza = dati che descrivono gli individui appartenenti alla relazione (sono le righe della tabella)
- Varia nel tempo (aggiunta, modifica, eliminazione dei dati riguardanti gli individui)





Schema e istanza di un DB

- **Schema** = insieme degli schemi delle relazioni (struttura)
- **Istanza** (o **stato**) = valori dei dati nelle tabelle (righe)

Domande?

Grazie per l'attenzione

