
# *memorizzazione*

*i formati dei dati*

*Alberto Ferrari – Big Data*

# *i formati dei dati digitali*

- *dpcm_3-12-2013*
- *«Il presente documento fornisce indicazioni iniziali sui formati dei documenti informatici che per le loro caratteristiche sono, al momento attuale, da ritenersi coerenti con le regole tecniche del documento informatico, del sistema di conservazione e del protocollo informatico.»*
- *« I formati descritti sono stati scelti tra quelli che possono maggiormente garantire i principi dell'interoperabilità tra i sistemi di conservazione e in base alla normativa vigente riguardante specifiche tipologie documentali.»*
- *« Il formato di un file è la convenzione usata per interpretare, leggere e modificare il file. »*

# *identificazione del formato di un file*

- *«**L'associazione** del documento informatico al suo **formato** può avvenire, attraverso varie modalità, tra cui le più impiegate sono:*
  - ***l'estensione**: una serie di lettere, unita al nome del file attraverso un punto*
    - *esempio [nome del file].docx identifica un formato testo di proprietà della Microsoft;*
  - *i **metadati** espliciti*
    - *l'indicazione "application/msword" inserita nei tipi MIME che indica un file testo realizzato con l'applicazione Word della Microsoft*
  - *il **magic number**:*
    - *i primi byte presenti nella sequenza binaria del file, ad esempio 0xffd8 identifica i file immagine di tipo .jpeg»*

# *formati più diffusi*

- testi/documenti (DOC, HTML, PDF, CSV,...)
- calcolo (XLS, ...)
- immagini (GIF, JPG, BMP, TIF, EPS, SVG, ...)
- suoni (MP3, WAV, ...)
- video (MPG, MPEG, AVI, WMV,...)
- eseguibili (EXE, ...)
- archiviazione e Compressione (ZIP, RAR, ...)

# *formato CSV*

- ***Comma-Separated Values*** (valori separati da virgole)
- è un file di ***testo***
- utilizza le ***virgole*** (o altri caratteri particolari) per ***separare*** i dati contenuti all'interno delle singole celle di una tabella
- è uno dei primi formati ad essersi diffusi per ***l'interscambio*** di dati
- è ancora oggi ***diffuso*** in molte applicazioni

# *esempio oscar miglior attrice 2000 - 2016*

"Year", "Age", "Name", "Movie"

2000, 25, "Hilary Swank", "Boys Don't Cry"

2001, 33, "Julia Roberts", "Erin Brockovich"

2002, 35, "Halle Berry", "Monster's Ball"

2003, 35, "Nicole Kidman", "The Hours"

2004, 28, "Charlize Theron", "Monster"

2005, 30, "Hilary Swank", "Million Dollar Baby"

2006, 29, "Reese Witherspoon", "Walk the Line"

2007, 61, "Helen Mirren", "The Queen"

2008, 32, "Marion Cotillard", "La Vie en rose"

2009, 33, "Kate Winslet", "The Reader"

2010, 45, "Sandra Bullock", "The Blind Side"

2011, 29, "Natalie Portman", "Black Swan"

2012, 62, "Meryl Streep", "The Iron Lady"

2013, 22, "Jennifer Lawrence", "Silver Linings Playbook"

2014, 44, "Cate Blanchett", "Blue Jasmine"

2015, 54, "Julianne Moore", "Still Alice"

2016, 26, "Brie Larson", "Room"

| "Year" | "Age" | "Name" | "Movie" |
|---|---|---|---|
| 2000 | 25 | "Hilary Swank" | "Boys Don't Cry" |
| 2001 | 33 | "Julia Roberts" | "Erin Brockovich" |
| 2002 | 35 | "Halle Berry" | "Monster's Ball" |
| 2003 | 35 | "Nicole Kidman" | "The Hours" |
| 2004 | 28 | "Charlize Theron" | "Monster" |
| 2005 | 30 | "Hilary Swank" | "Million Dollar Baby" |
| 2006 | 29 | "Reese Witherspoon" | "Walk the Line" |
| 2007 | 61 | "Helen Mirren" | "The Queen" |
| 2008 | 32 | "Marion Cotillard" | "La Vie en rose" |
| 2009 | 33 | "Kate Winslet" | "The Reader" |
| 2010 | 45 | "Sandra Bullock" | "The Blind Side" |
| 2011 | 29 | "Natalie Portman" | "Black Swan" |
| 2012 | 62 | "Meryl Streep" | "The Iron Lady" |
| 2013 | 22 | "Jennifer Lawrence" | "Silver Linings Playbook" |
| 2014 | 44 | "Cate Blanchett" | "Blue Jasmine" |
| 2015 | 54 | "Julianne Moore" | "Still Alice" |
| 2016 | 26 | "Brie Larson" | "Room" |

# *python – esempio lettura file CSV*

```
import csv                                              # modulo per gestione file csv
dati = []                                               # lista per i dati letti dal file csv
with open('dati_covid.csv',newline='') as f:            # in memoria il file ha nome f
    contenuto = csv.reader(f)                           # lettura di tutto il file
    for riga in contenuto:                              # per ogni riga
        dati.append(riga)                               # inserimento della registrazione nella l
ista
print('prima riga del file (significato colonne)',dati[0])
print('----------------------------------------')
print('prima registrazione (riga 1 del file)',dati[1])
print('----------------------------------------')
print('denominazione regione',dati[1][3],'totale_casi',dati[1][-2])
print()
print('----------------------------------------')
print('ultima registrazione ',dati[-1])
print('denominazione regione',dati[-1][3],'totale_casi',dati[-1][-2])
```

# *formato JSON*

- JSON (JavaScript Object Notation) popolare formato per lo scambio dei dati
- i dati sono rappresentati coppie proprietà/valori separate da virgola
- un oggetto JSON è racchiuso tra parentesi graffe

# *esempio JSON*


```
[
  {
    "data":"2020-02-24T18:00:00",
    "stato":"ITA",
    "ricoverati_con_sintomi":101,
    "terapia_intensiva":26,
    "totale_ospedalizzati":127,
    "isolamento_domiciliare":94,
    "totale_positivi":221,
    "variazione_totale_positivi":0,
    "nuovi_positivi":221,
    "dimessi_guariti":1,
    "deceduti":7,
    "casi_da_sospetto_diagnostico":null,
    "casi_da_screening":null,
    "totale_casi":229,
    "tamponi":4324,
    "casi_testati":null,
  },
   {
    "data":"2020-02-25T18:00:00",
    "stato":"ITA",
    "ricoverati_con_sintomi":114,
    …
```

# *python – esempio lettura file JSON*

```
import json
dati_covid = json.load(open("dpc-covid19-ita-andamento-nazionale.json"))
print('numero record',len(dati_covid))
riga = dati_covid[-1]
print('ultimo record')
print(riga)
```

# *database*

- insieme di dati strutturati
- omogeneo per contenuti e formato
- dati strutturati in modo da
  - razionalizzare la gestione delle informazioni
  - permettere l'aggiornamento delle informazioni
  - permettere lo svolgimento di ricerche anche complesse

# *DBMS*

- ***DataBase Management System***
- insieme di programmi che offrono a diverse tipologie di utenti tutti gli ***strumenti*** necessari per gestire grandi ***basi di dati***
- un DBMS permette di definire la ***struttura*** di tabelle di dati e offre strumenti per recuperare ***informazioni***
- un DBMS ***gestisce tutti i dettagli*** di basso livello necessari alla *memorizzazione, recupero e ricerca dell'informazione*

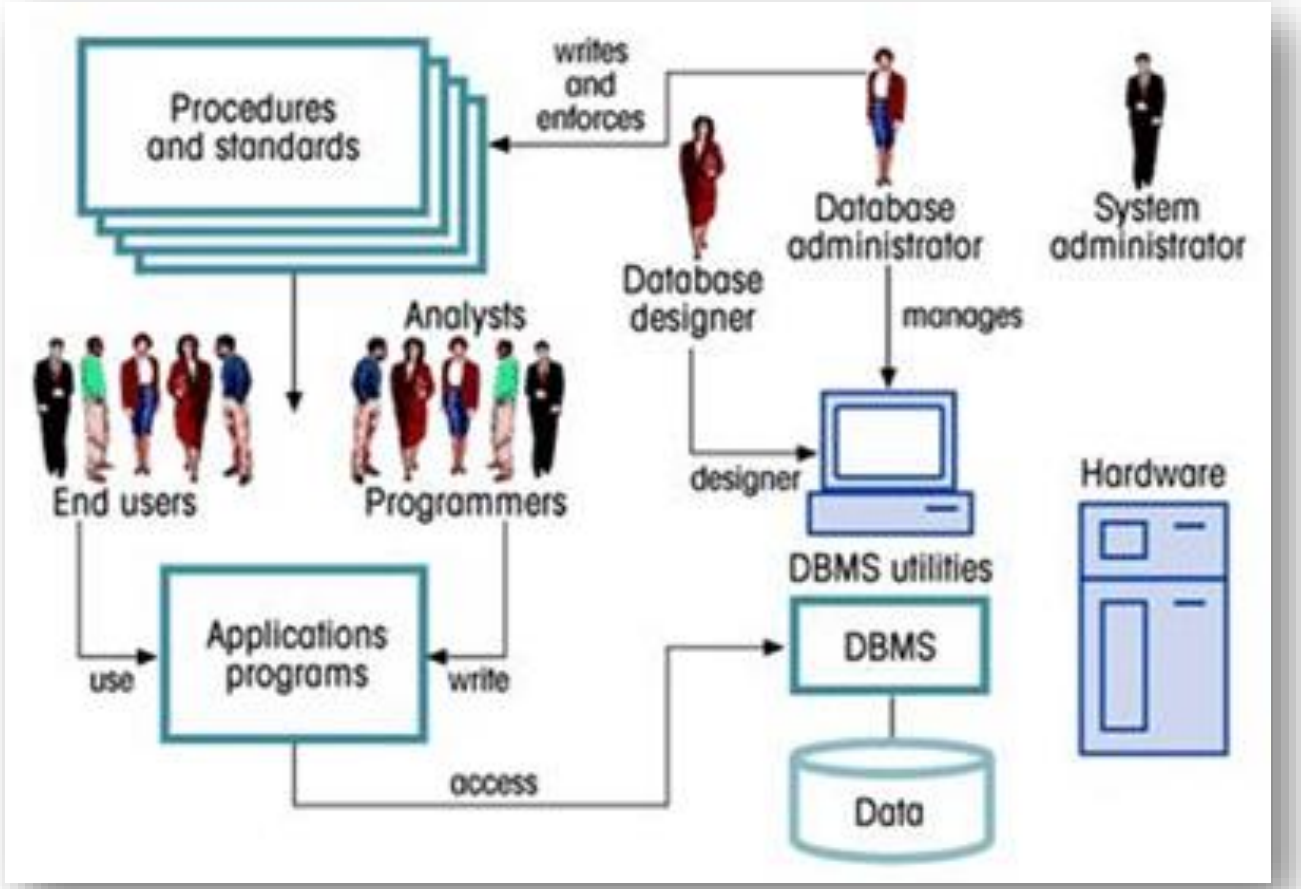
# *DBMS: accesso ai dati*

- ***interfaccia*** per accedere ai dati
  - permette di ***variare lo schema***
  - consente di ***visualizzare***, in forma tabellare, il ***contenuto*** di uno schema (*istanze*)
- attraverso un ***programma***
  - un software scritto in un linguaggio di programmazione si ***connette*** al server DBMS e, utilizzando il suo specifico protocollo di comunicazione, effettua le stesse operazioni descritte al punto precedente

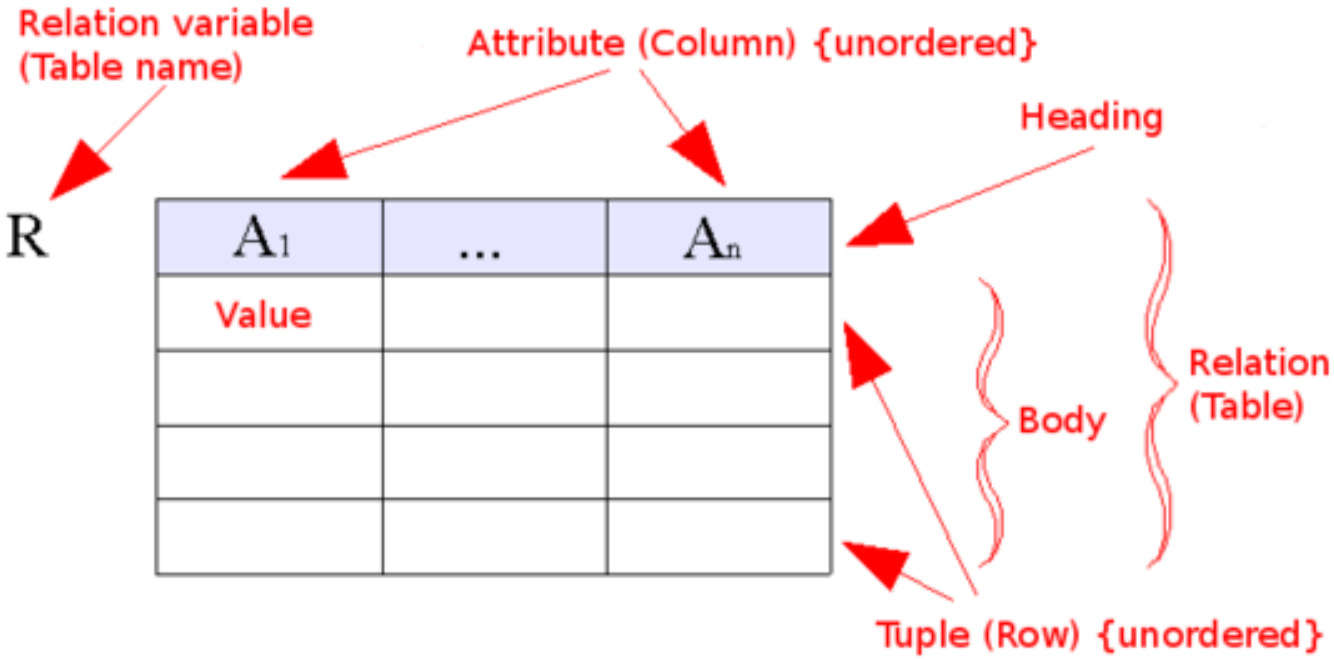
# *esempi di DBMS relazionali – RDBMS*

- ***Access***
  - per gestire quantità di informazioni limitate e tipicamente gestite da un singolo utente
- ***Oracle***
  - molto diffuso presso le aziende
- ***SQL Server***
  - il più diffuso in ambienti basati su Microsoft Windows (mentre Oracle è utilizzato prevalentemente su sistemi Unix)
- ***DB2***
  - database storico di IBM, diffuso in ambiente Mainframe, e interfacciato attraverso programmi COBOL o RPG.
- ***MySQL***
  - open source, gratuito, utilizzato spesso per il back end di applicazioni e siti Web

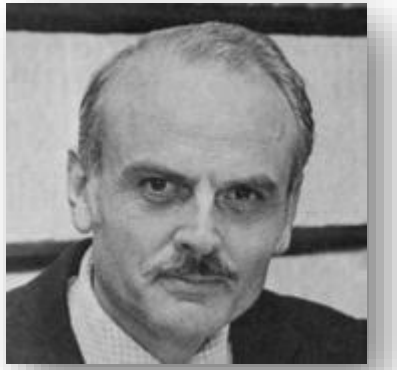
# *data base system environment*



# *modello relazionale*



# *storia*



- introdotto nel ***1970*** dal matematico inglese ***Edgar Frank Codd***
- lavora in IMB e pubblica *"A Relational Model of Data for Large Shared Data Banks" (un modello relazionale per i dati in grandi basi dati condivise)*
- prime ***implementazioni*** del modello intorno alla fine degli anni '70 (*ritardo dovuto alla* ***difficile*** *implementazione del modello matematico*)
- dagli anni ***'80*** ampia ***diffusione*** di DBMS relazionali anche per sistemi di piccole dimensioni

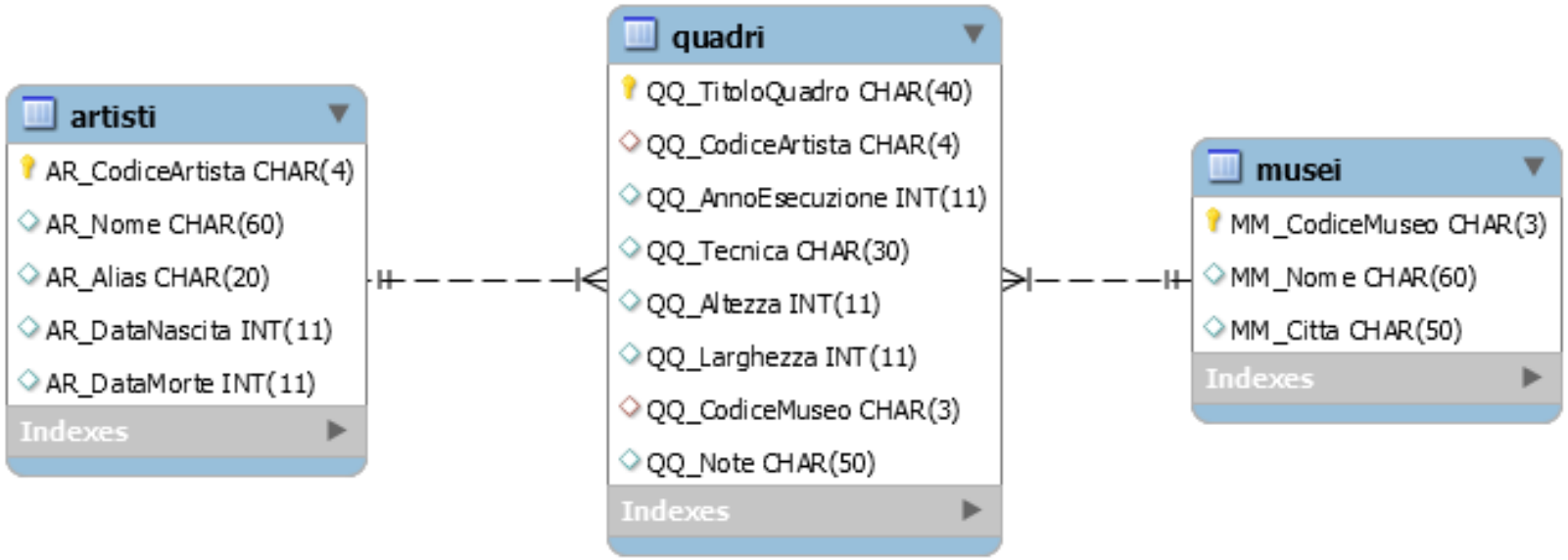
# *esempio di relazione*

- ***nome*** relazione **Studente**
- ***grado*** della relazione **3**
  (*Libretto, Nome, Data_Nascita*)
- ***cardinalità*** della relazione = **1200**
  *(numero di tuple = numero di studenti)*
- Libretto è ***campo chiave***

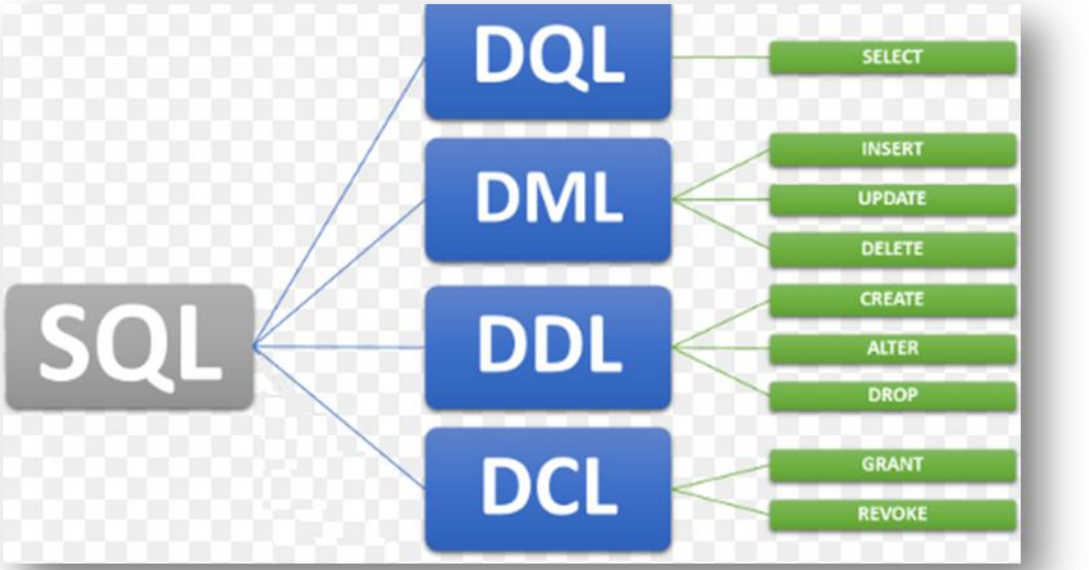
**Studente**

| Libretto | Nome | Data_Nascita |
|---|---|---|
| 34 | Verdi | 15/11/1990 |
| 123 | Rossi | 10/02/1990 |
| 210 | Bianchi | 27/04/1989 |
| 45 | Neri | 12/12/1988 |

# *esempio*
# *schema di un semplice database relazionale*



# *Structured Query Language (SQL)*



# *SQL*

- linguaggio di interrogazione per ***database relazionali*** progettato per
  - ***leggere*** *(recuperare informazioni)*
  - ***modificare***
  - ***gestire*** dati memorizzati in un sistema basato sul modello relazionale
  - creare e modificare ***schemi*** di database
  - creare e gestire strumenti di ***controllo*** ed ***accesso*** ai dati

# *DBMS: linguaggi (1)*

- ***DDL***
  *(Data Definition Language, linguaggio di definizione dei dati)*
  - per descrivere la ***struttura*** delle ***tabelle***
- ***DML***
  *(Data Manipulation Language,*
  *linguaggio per la manipolazione dei dati)*
  - per eseguire le operazioni di ***inserimento***, ***modifica*** e ***cancellazione*** dei ***dati***
- ***QL*** *(Query Language, linguaggio di interrogazione)*
  - per ***interrogare*** il database al fine di ***individuare i dati*** che corrispondono ai parametri di ***ricerca*** dell'utente

# *DBMS: linguaggi (2)*

- ***DMCL***
  *(Device Media Control Language, linguaggio per il controllo dei supporti di memorizzazione)*
  - per far corrispondere il modello logico definito con DDL al supporto fisico su cui scrivere i dati
- ***DCL***
  *(Data Control Language, linguaggio di controllo dei dati)*
  - per definire i ***vincoli*** sui dati *(permessi di accesso e i vincoli di integrità)*

# *esempio on line python – mysql (w3schools)*

https://www.w3schools.com/python/python_mysql_getstarted.asp

- esempi python
  - creazione database
  - creazione tabelle
  - inserimento dati
  - ricerca dati

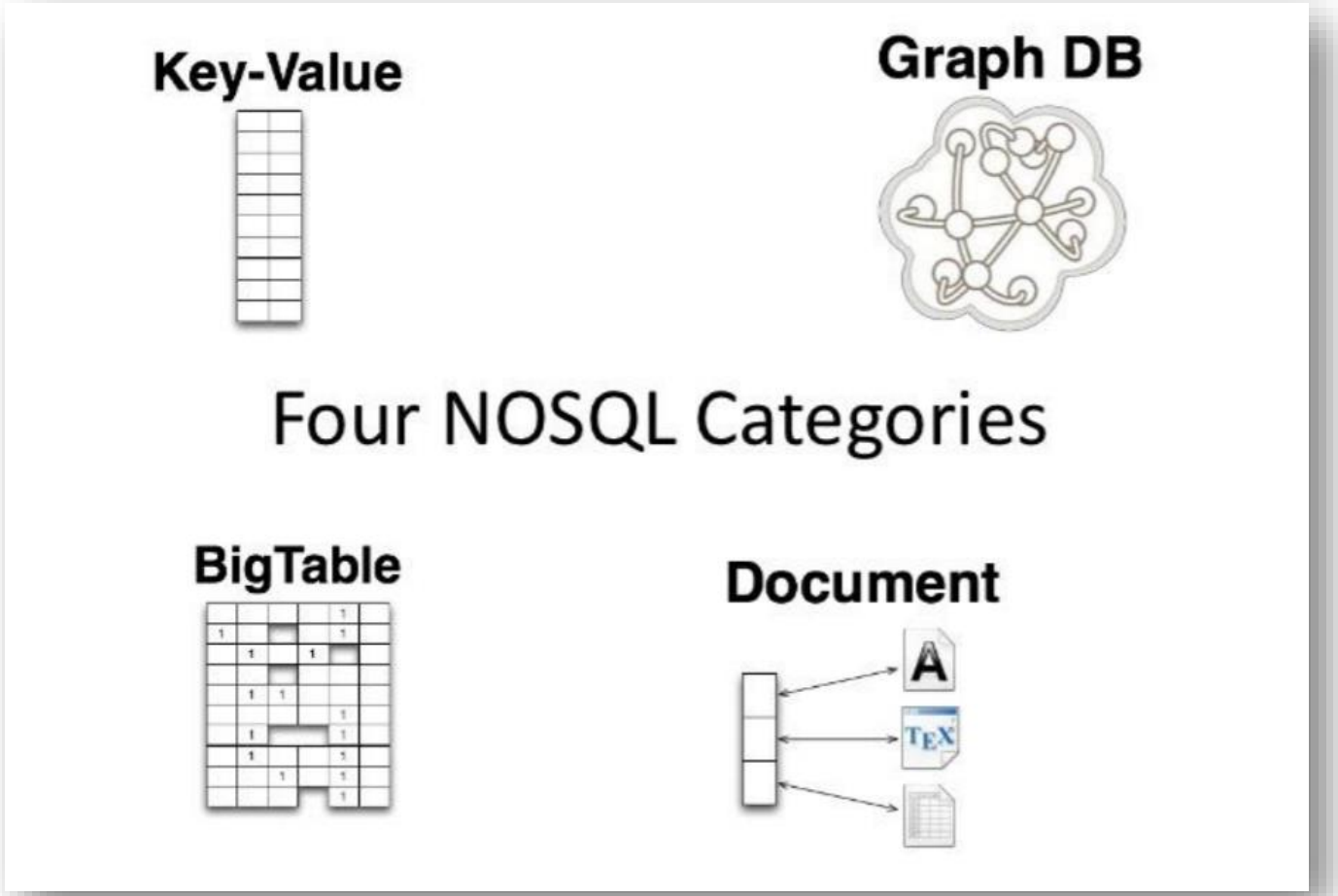

Not
Only
SQL

# *NoSQL*

- movimento che promuove sistemi software dove la persistenza dei dati è in caratterizzata dal fatto di ***non utilizzare il modello relazionale***
- ***schemeless***
  - gli archivi di dati non richiedono uno schema fisso

# *database NoSQL - tipologie*

- ***chiave-valore***
  - ogni singolo elemento viene salvato come un attributo (o chiave) assieme al suo valore
- orientato ai ***documenti***
  - ogni chiave è accoppiata con una struttura dati complessa detta documento
- a ***grafo***
  - utilizzati per conservare informazioni su reti e relazioni (es. connessioni all'interno di un social network)
- ***tabulare***
  - dati organizzati in colonne di tabelle


Key-Value
Graph DB
Four NOSQL Categories
BigTable
Document

| | Database SQL | Database NoSQL |
|---|---|---|
| Tipologia | Un database per tutte le applicazioni | Diversi modelli di database, ad es. database orientati ai documenti, database a grafo, database chiave-valore e database a colonne |
| Archiviazione dei dati | I singoli dati (ad es. “titolo del libro”) vengono archiviati in righe all’interno di una tabella e associati a determinati attributi (ad es. “autore”, “anno di pubblicazione”, ecc.). I set di dati vengono archiviati in tabelle separate e riassemblati dal sistema in caso di query di ricerca complesse. | I database NoSQL non utilizzano tabelle, ma a seconda del tipo fanno ricorso a documenti completi, chiavi-valori, grafi o colonne. |
| Schema | Il tipo e la struttura dei dati vengono definiti in anticipo. Per archiviare nuove informazioni è necessario modificare l’intero database (e, a questo scopo, passare alla modalità offline). | Flessibile. I nuovi set di dati possono essere aggiunti immediatamente. I dati strutturati, semi-strutturati e non strutturati possono essere archiviati insieme, non è necessaria alcuna conversione preliminare. |
| Scalabilità | Scalabilità verticale. Un solo server deve assicurare le prestazioni dell’intero sistema di banca dati; questo determina un calo dell’efficienza in caso di grandi volumi di dati. | Scalabilità orizzontale. Ciascun amministratore può aggiungere nuovi commodity server e cloud server; il database NoSQL invia i dati automaticamente a tutti i server. |

# *prestazioni*

- database SQL
  - per elevati volumi di dati utilizzano ***indici*** di indirizzamento
  - per aumentare le prestazioni è necessario ***ottimizzare*** le query, ***modificare*** la struttura e gli indici
- database NoSQL
  - utilizzano ***cloud server*** e ***hardware cluster***
  - offrono prestazioni nettamente superiori

*Cloud Server*
*server virtuale che utilizza una porzione o un sottoinsieme del server fisico che lo ospita*

*Cluster*
*insieme di computer connessi tra loro tramite una rete telematica*

# *NoSQL database system*


mongo
DB

# *mongoDB («humongous» enorme)*

- sistema di gestione di database (DBMS) ***non relazionale***
  - ***open source***
  - orientato ai ***documenti***
  - supporta diverse tipologie di dati
- i dati sono archiviati in strutture definite ***collezioni***
  - le collezioni contengono insiemi di documenti
  - collezioni analoghe alle tabelle dei database relazionali

# *tabelle vs collezioni*

## Relational

| Customer ID | First Name | Last Name | City |
|---|---|---|---|
| 0 | John | Doe | New York |
| 1 | Mark | Smith | San Francisco |
| 2 | Jay | Black | Newark |
| 3 | Meagan | White | London |
| 4 | Edward | Daniels | Boston |

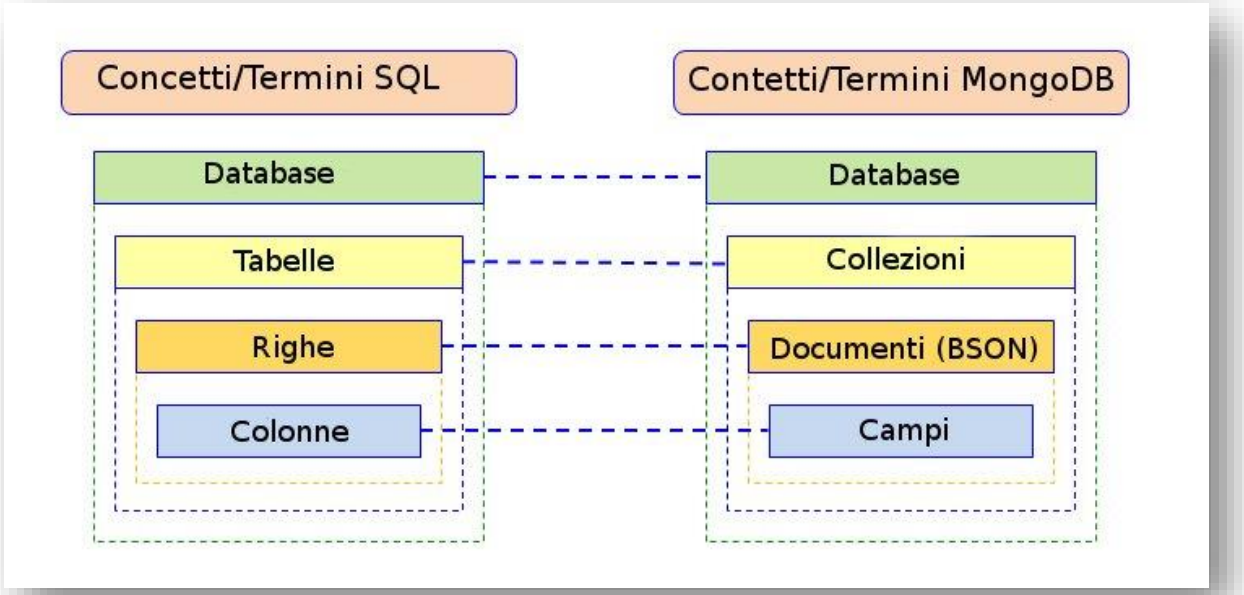
| Phone Number | Type | DNC | Customer ID |
|---|---|---|---|
| 1-212-555-1212 | home | T | 0 |
| 1-212-555-1213 | home | T | 0 |
| 1-212-555-1214 | cell | F | 0 |
| 1-212-777-1212 | home | T | 1 |
| 1-212-777-1213 | cell | (null) | 1 |
| 1-212-888-1212 | home | F | 2 |

## MongoDB

```
{   customer_id : 1,
    first_name : "Mark",
    last_name : "Smith",
    city : "San Francisco",
    phones: [   {
         number : "1-212-777-1212",
      dnc : true,
         type : "home"
    },
    {
         number : "1-212-777-1213",
         type : "cell"
    }]
}
```

# *struttura*

- ***documenti*** sono organizzati in ***campi***
  - i campi sono l'analogo delle *colonne* in un database relazionale
  - i valori nei campi possono essere di ***tipi differenti*** inclusi altri documenti
- un ***record*** è un documento
  - una struttura dati composta da coppie campo-valore
  - necessaria una chiave primaria (identificatore univoco)
- i documenti sono rappresentati da file BSON (Binary JSON) estensione dei file JSON

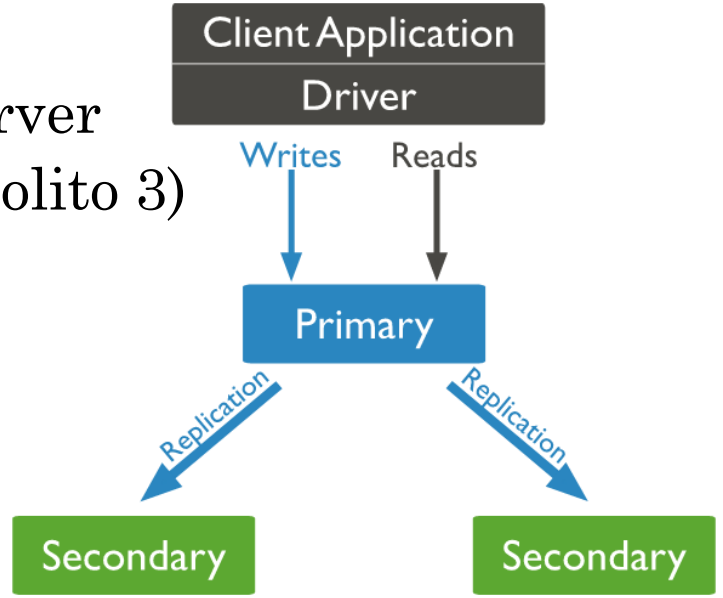


# *scalabilità*

- ***scalabilità***
  - capacità di funzionare con un carico di lavoro maggiore
- ***database relazionali***
  - ridimensionamento dei dati di tipo ***verticale***
  - ***unico server***
  - per gestire una mole maggiore di dati si aumenta lo spazio di archiviazione e si utilizza una CPU più potente
- ***mongoDB***
  - ridimensionamento ***orizzontale*** (sharding)
  - distribuzione dei dati su ***più macchine***
  - per gestire una mole maggiore di dati si aggiungono nuove macchine
-

# *disponibilità dei dati*

- ***replica***
  - dati ***sincronizzati*** tra più server
  - più ***copie*** dei dati su diversi server di database
  - protezione dal malfunzionamento di un singolo server
  - un set di repliche è formato da due o più nodi (di solito 3)
  - un nodo primario e più nodi secondari



# *MQL - MongoDB Query Language*

- i documenti sono rappresentati da file BSON (Binary JSON) estensione dei file JSON

# *esempio on line python – mongodb (w3schools)*

- https://www.w3schools.com/python/python_mongodb_getstarted.asp
  - creazione database
  - creazione collezione
  - inserimento dati
  - ricerca dati