# An Brief Introduction to Data Storage

Jascha Schewtschenko

Institute of Cosmology and Gravitation, University of Portsmouth

May 10, 2018

# Outline

1. Data Storage / Unstructured Data (Files)
   - Distributed File Systems

2. Structured Data / Databases
   - Database types

3. SQL

# Data Storage / Unstructured Data (Files)

# Data Storage / Unstructured Data (Files)

- HPC system create/deal which a huge amount of data
- most of it is per se unstructured: Observational images/measurements, simulation snapshots
- this data is stored in form of files in either local or remote file systems
- these file systems are usually *distributed*/*clustered*, i.e. allow to be used by multiple hosts at the same time

# Clustered/Distributed File systems

- Goals (among others):
  - Access transparency files while distributed can be accessed in the same way as local files are accessed
  - Location transparency existence of consistent name space encompassing local as well as remote files
  - Concurrency transparency all clients have the same view of the state of the file system
  - Heterogeneity file service should be provided across different hardware and operating system platforms
  - Scalability file system should work well indep. of environment size
  - Replication transparency mask that files may be replicated across multiple servers (to support scalability/redundancy)
- *parallel file systems* are a type of clustered file system that spread data across multiple storage nodes, usually for redundancy or performance
- Examples: Lustre, G(eneral)P(arallel)FS, G(oogle)FS, H(a)D(oop)FS

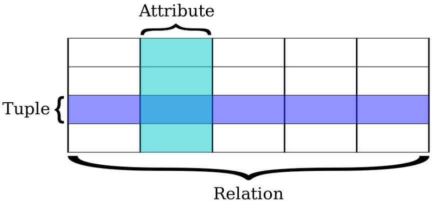
# Structured Data / Databases

# Structured Data / Databases

- Define the logical structure on data
- provides a mechanism for storage and retrieval of data
- Database-management system (DBMS) interacts with end-users, other applications, and the database itself to capture and analyse data
- A general-purpose DBMS allows the definition, creation, querying, update, and administration of databases.
- A database is **NOT** generally portable across different DBMSs, but different DBMSs can interoperate by using standards (e.g. SQL, ODBC)

# Relational Databases

- all data is represented in terms of tuples, grouped into relations (aka tables)



- virtually all relational database systems use SQL (→ SQL databases)

# Non-Relational Databases

- Anything **BUT** relational/SQL databases (→ NoSQL databases)
- NoSQL databases are often very fast, do not req uire fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally.
- Classes of NoSQL DBs and examples:

  Column Store stores data as columns rather than rows (e.g. Amazon DynamoDB, Bigtable, Cassandra, Druid, HBase, Hypertable)

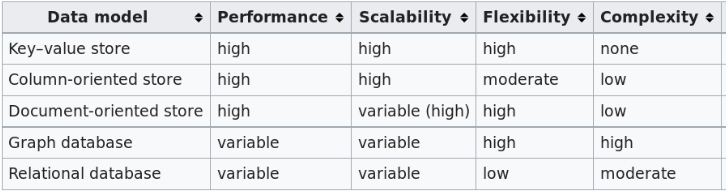
  Key-Value least complex NoSQL option; data stored in schema-less way that consists of indexed keys and values (e.g. Aerospike, Apache Ignite, ArangoDB, Couchbase, InfinityDB, Oracle NoSQL DB, OrientDB, Redis, Riak)

  Document Store key-value concept with added complexity; each document has its own data & unique key (e.g. ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB)

  Graph designed for data whose relations are well represented as a graph (e.g. AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic)

# Relational vs Non-Relational Databases

Each class of DB has certain advantages/disadvantages:



| Data model | Performance | Scalability | Flexibility | Complexity |
|---|---|---|---|---|
| Key–value store | high | high | high | none |
| Column-oriented store | high | high | moderate | low |
| Document-oriented store | high | variable (high) | high | low |
| Graph database | variable | variable | high | high |
| Relational database | variable | variable | low | moderate |

**Performance** rate at which a database management system (DBMS) supplies information to users

**Scalability** capability of a system to handle a growing amount of work in the same elapsed time when processing power is expanded to accommodate growth

**Flexibility** ability to deal with changing/variety of dataset types

**Complexity** complexity of DBMS/queries

# SQL

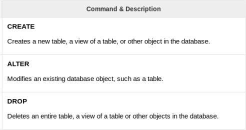
# SQL

- stands for *Structured Query Language*
- used to access and manipulate (relational) databases
- while ISO/ANSI standard since 80s, SQL implementations are partially incompatible between vendors (e.g. by omitting support for certain features of Standard SQL), called dialects:
    - MS SQL Server using T-SQL
    - Oracle using PL/SQL
    - MS Access using Jet SQL

# SQL - Commands / Statements

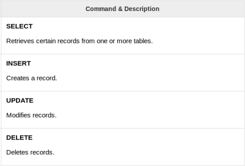
- The (standard) SQL commands can be classified into the following groups:

**DDL** (Data Definition Lang.)



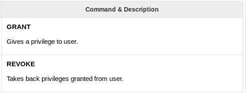
| Command & Description |
|---|
| **CREATE**<br>Creates a new table, a view of a table, or other object in the database. |
| **ALTER**<br>Modifies an existing database object, such as a table. |
| **DROP**<br>Deletes an entire table, a view of a table or other objects in the database. |

**DML** (Data Manipulation Lang.)



| Command & Description |
|---|
| **SELECT**<br>Retrieves certain records from one or more tables. |
| **INSERT**<br>Creates a record. |
| **UPDATE**<br>Modifies records. |
| **DELETE**<br>Deletes records. |

**DCL** (Data Control Lang.)



| Command & Description |
|---|
| **GRANT**<br>Gives a privilege to user. |
| **REVOKE**<br>Takes back privileges granted from user. |

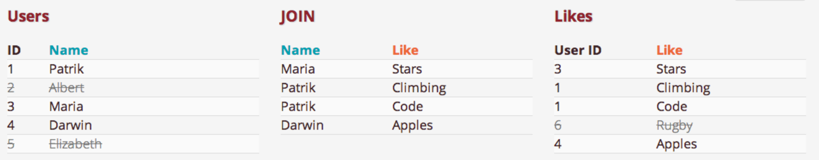
- These commands are used in SQL Statements, e.g. to retrieve data

```
SELECT first_name, last_name
FROM DISCnet_students WHERE affiliation=’ICG’
```

# SQL - Joining tables

- The SQL JOIN clause allows to combine tuples from different relations/tables into a new relation
- Example:

```
SELECT Users.name, Likes.like FROM Users
  JOIN Likes ON Users.id = Likes.user_id
```



**Users**

| ID | Name |
|---|---|
| 1 | Patrik |
| ~~2~~ | ~~Albert~~ |
| 3 | Maria |
| 4 | Darwin |
| ~~5~~ | ~~Elizabeth~~ |

**JOIN**

| Name | Like |
|---|---|
| Maria | Stars |
| Patrik | Climbing |
| Patrik | Code |
| Darwin | Apples |

**Likes**

| User ID | Like |
|---|---|
| 3 | Stars |
| 1 | Climbing |
| 1 | Code |
| ~~6~~ | ~~Rugby~~ |
| 4 | Apples |