Revision of persistence data systems

A DBMS is a system that managed databases, providing reliability and persistence of data.
Database is a data structure that represents the information interest on a information system.

DBMS are better than file system because they can manage different problems like: integrity and consistency of data, atomic execution of the operations, concurrent access of data and security.

There are 3 main problems: **inconsistency and redundancy of data, difficulties in accessing data, data isolation.**

DBMS characteristics

- **Data dictionary**: contains the definition of data and the relationship between them
- **Data storage**: store data physically automatically, providing tools for management,
- **Data management:** avoid the programmer to manage the distinction between physical and logical format of data.
- **Security management**: it provides data security and privacy within the db
- **Multi-user access control**:it ensure multiple access of data simultaneously maintaining consistency.
- **Backup and restore management**: it allows to backup ad restore information to ensure security and integrity.
- **Data integrity management**: it contains the definition of rules to maintain the integrity.

Transactions

A transaction is a part of the execution of the program that access and/or updates certain amount of data in the database. It can be composed by several SQL commands. A transaction is completed if and only if all the operations are completed successfully.
When transaction is completed it will be committed and all operation will be performed on the final database, it must be consistent after a commit.
During transaction execution database could be not consistent.

To preserve integrity, DBMS must ensure **ACID**:

- **Atomicity**: either all operations of a transaction are reflected on db or none are
- **Consistency**: dbms ensures that none dbms integrity constrains are violated
- **Isolation**: each transaction must be unaware of others
- **Durability**: after a transaction has been successfully completed changes must be persist on database even in the event of system failure(blackout).

DBMS provides 2 types of transaction execution:

- **Serial**, execution one by one
- **Concurrent**, execution in parallel, better performance.

During a transaction the updates are visible only in the transaction, others don’t see those.
Isolation layers don’t allow the modification of data to maintain consistency.

Consistency issues
The Transactions Manager must ensure that the transactions must not interfere with others.
Problems:

- **Lost Update:** you read and edit something and other transaction edit that value before your modification.
- **Dirty Read:** you write something an other transaction read the edit but you rollback, the other transaction has a non valid value
- **Unrepeatable read:** you read something at start, you don't write, other transaction write but when you read you haven't the start data but the edit of the other transaction
- **Phantom row:** transactions add rows but the others can't see the new entries.

Transaction in SQL
Commit work: execute the commit
Rollback work: execute the rollback
Transaction are managed by DDL commands.
We can define save-points to rollback the transaction partially.
Savepoint label_name;
Rollback work to savepoint label_name;

we can set transaction as read-only, so it will not see commits made by the other transactions.
Set transaction readonly

Transaction failure
transaction can failure for different causes:

- Logical errors: internal error condition
- System errors: DBMS abort transaction due to an error condition
- System failure: power failure, hardware failure, system crash
- Media failure: hard-drive errors, broken or full

when the transaction fails it will be rollback.

DBMS and concurrency
A common technique used for data accessing is **locks.**
Locks ensure consistency and integrity.
DBMS have 2 types of locks:

- **Exclusive locks:** used to modify data
- **Share locks:** it allows multiple access to the resource, but they can only read, if this lock is acquired writes can't write because they must use exclusive lock on that resource.

All locks acquired are maintained until commit or rollback execution.
There are lock's levels every DBMS can managed and support them in different ways.

A page-level lock causes all rows that physically share the same page to be locked, similarly for table-levels locks.

Explicit locks
SQL provides basic mechanism to lock table, if you want lock some rows put at the end of query *FOR UPDATE* ; if you need to lock all table use *LOCK TABLE* command.
Normally when you select something with "*for update"* command is followed by update query.
If a table is already locked normally selects don't return until lock is removed. Locks are removed with commit or rollback commands. If you wont wait the locks you can put *NOWAIT* command, it will return an error if it encounter a lock.

DBMS has also implicit locks they are automatically applied by DBMS during queries of insert, update and delete.

A system is deadlocked when in one set of transactions each transaction awaits the end of another.

Deadlock can be fixed making partial or total rollbacks, normally the cheapest.

Isolation layers
Every transaction has an isolation level, if it is not specified it sets up the default layer.
SQL define 4 standard layers:

- **Read uncommitted:** read data without shared locks, you can read uncommited data (lost update)
- **Read committed**: (default level) it used shared locks but it not ensure that data can change before end of transaction
- **Repeatable read:** it ensure that data don't change before transaction, but it is could have phantom rows
- **Serializable:** (max level) it prevents other users insert or update data till the end of the transaction.

To set isolation layer use the following command: *set transaction isolation level <mode>*

	Phantom	Urepeatable Read	Dirty Read	Lost Update
Serializable	NO	NO	NO	NO
Repeatable Read	YES	NO	NO	NO
Read Committed	YES	YES	NO	NO
Read Uncommitted	YES	YES	YES	NO

Edited data are save in a special portion of memory, this allowed also to have an old copy for rollback.
Each transaction can be isolated from the others, atomicity is implemented because if transaction failed you can restore the previous state.
To recover data after a failure is commonly used log-based recovery technique.
A log is stored in the storage and contains all the operations done by the DBMS.
Media failures can be recovered using log saved on a external media and backups.

Indexes
An Index is a set of pairs of the type (k,p) where k is the key and p is a pointer to the record(possibly unique) with the key k.
Indexes speed up research and reduce hard-disk access, it is slow.
Index contains less data than original table, in fact it contains only a portion of data.

Indexes grow with data if it become large it could be a problem, we must use a structure that ensure fast access of data, b-tree or b-tree+, hash indexes or bitmap indexes.

Btree can grow because of the insert algorithm build a tree where if a node is full it split and bring the middle element in the upper level and the remain elements will be put left and right of middle element.
A node can contains maximum 2*O+1 children; each intermediate node has at least O+1 pointers.

Bitmap
bitmap stores a value for each possible value of the key. If the bit is set, it means that the line to which the pointer refers contains the key. This structure is useful for complex where condition.

cust_id	marital status	Region	Gender	Monthly expense
101	Single	East	Man	2500
102	Married	Central	Woman	1200
103	Married	West	Woman	1345
104	Divorced	West	Man	2340
105	Single	Central	Woman	1560
106	Married	Central	Woman	3450

- ✓ Region, marital_status, gender are columns with low cardinality of values, so they are good to be indexed with bitmap indexes.
- ✓ Example of bitmap index on 'region' column

East	1	0	0	0	0	0
Central	0	1	0	0	1	1
West	0	0	1	1	0	0

- ✓ The execution of logical operations (AND,OR, NOT) between columns indexed with bitmap indexes is very fast.

You can do where bit by bit and this indexes are easily compressible.

Hash indexes
it works only for exact searches, it distributes the pointer using an hash function to the rows among buckets. A bucket is the unit of storage containing one or more records. An hash can have multiple records bind to that.

Create an index in SQL
Create index index_name ON table_name (attributes-list) we can specify that index has unique values for keys using unique keyword after create. Use drop to delete index.
The order of definition of attributes is relevant if we use multiple attributes index.
A multi-attribuite index A1,A2,...An is effective if you specify values of all attributes or for the first i <n.

Physical design
the basic unit of dbms' storage is **block** (8k), a set of contiguous block is called **extent**, the space where is allocated an object is called **segment**, the spaces where segments are allocated is called **tablespaces.**
Tablespaces are local levels, and they are contains in files (1 or more).

The objective of physical design is: implement the schema resulting from logical design to a DBMS that use certain physical resources.

Through the storage table it's possible to define storage settings of tables, if it isn't declared it will use default parameters.
Initial: it represents the space allocated at table creation
next:it represents the space to allocate in case of space is exhausted.
Maxextents: it represents the number of extensions (default:unlimited)
pctincrease: percentage increase in extension size.
Pctused:percentage under which lines are added to a block
pctfree:percentage of space in a block that must remain free for updates of contained lines

A **table cluster** is a group of table logically separated but physically integrated. It is useful for big joins. They share the same physical block following certain criteria, specified in the declaration of the cluster.
Cluster has sense to exist if there is one or more common columns.
IOT: index organized table, it use indexes based on primary keys, it ensure very fast access of data. Iot needs lower storage than classic implementation (TABLE+INDEX), data are stored in the index. Table updated are resolved in the in the update only.

Partitioning
tables can be partitioned to improve their management. Each partition become an object partially independent from the others. We can do query on whole table or only on a partition.
Partitions can be put in different tablespaces. Example: useful to divide orders in the years for an e-commerce
using partition by (fields)(partition name value less then (x)...) we can create partition based on a field of the table.
Query processing
When DBMS must execute a query it use the following pattern:

- Parsing and translation in relational algebra
- Optimization
- Evaluation (execution)

Parsing and translation: it translate the query into internal form, relational algebra.
Parser checks syntax errors, operators are applied in the right way, the referenced objects exists and user has the privileges to execute the query.
After parsing and translation the query is rewrote to simplify the query and get to a form that is easier to analyze and thus optimize.

- **View resolution**: merge the input query with the queries that define the referenced views.
- **Unnesting**: if the query includes sub-queries it tries to transform it into a form without them
- **Use of Constraints**: the constraints defined on the schema are used to simplify the query

Optimization process is that process that find the cheapest execution plan for the query.
It based the decisions on the cost of the query.
The is calculated estimating statistical information contained in the DBMS data dictionary. The cost depends by several parameters like table rows size, number of blocks in the table ecc., **cost-based optimization**.
it can use predefined rules **(rules-based optimization) .**
The optimizer has a strategy of enumeration of execution plans whose: enumerate all plans that potentially could be optimal. Don't generate access plans that are certainly not optimal.
The evaluation process execute the query using selected execution plan and the rewrote query. The output is the query result.
To estimate the cost of a query there are several ways like: cost of disk access, cpu time, time in a distributed system …
disk access has dominant cost, it's used measure the block transfers.
The algorithm have to consider the size of the buffer.

DBMS Security
data security classification:

- Authentication
- Authorization
- Access Control
- Auditing

GRANT CONNECT TO username IDENTIFIED BY psw give access to user to db

password can changed with alter user and user can be created using create user, drop user remove user; these commands can be executed only by DBA.
With authorization DBA can give the access to user in the database, it can allow modify and display information.
Authorization can be defined at different levels using grant command
GRANT typeOfAction ON typeOfResource(table/database) TO user

User that has CONNECT privilege can connect to DBMS, query the tables that he is authorized to, make changes on tables which he has authorization.
He can't create/drop/alter tables and indexes.
Users that have connect authorization can only work on tables provided by other users.

Users with RESOURCE authorization can edit the databases structure the limit is on the amount of space it can use (quota).

GRANT RESOURCE ON tablespace(quota) to user

the DBA privilege is the maximum privilege it can choose everything on DBMS.

In the to we can specify a list of users.
We can specify permissions on all sql commands. We can also give permissions on the column specifying in () the column i.e. ***grant select(indirizzo) on studenti to rossi;.*** The OPTION command grant user to give permissions of the selected command on that table, grant select on branch to u1 with grant option= authorized u1 to give sleect permissions on the branch table.

We can revoke permissions using revoke keyword
REVOKE privilage_list ON table FROM user_list

DBA can create ROLES to have a better management of permissions it can create using
CREATE ROLE name
and it can grant permissions putting in to clause the name of the role, then to apply to user DBA uses: grant role_name to user

Audit can monitor the activity of users.
Example: audit connect resource whenever successful

Sql injection
sql injection is an attack that allows the attacker to execute sql injection exploiting sql query vulnerabilities.
We can attack using special characters like ‘ or ; or – and other that are special commands of sql and they can broke the query executing something that system is not programmed to do.
We can protect system validating inputs, restricting database privileges, encrypting sensitive data.

Data encryption is the possibility to hide the real data using encryption techniques.
Data redaction is the process that hide data to other users following a policies defined in the DBMS. It allows also users to edit data and do everything all following policies rules.
It provide also reports about sensitive data. It use page region, pattern matching or manual way.
Data masking is used to obfuscate data, it used to hide data when you export data. It substitute the data with random chars or other fake data, typically used for exporting db for development for example. Data masking use encryption, character scrambling, nulling out deletion and shuffling techniques.

Data firewalls are filters that lock access to some resources from unauthorized users or again SQL injections.

Trigger are procedures that can be automatically by DBMS after or before insert, update. They are useful to validate input or edit data if there is new particular entry.
Procedures are compiled and executed in the DBMS we can declare using
create procedure/trigger name(args) is begin ...code… end name
to delete use *drop procedure name* the args can be 3 type in it is input data, out it is output data it will be return, inout it is input and also output args it will be return at the end and read at the beginning of the procedure. Procedure are very useful to hide database to outside, in fact application have only call the procedure then all queries are made inside the DBMS, more secure.

When the number of rows returned by a SELECT statement is unpredictable, there must
be a way to avoid overflow of the program variables.
A Cursor is a structure that is acting like a kind of buffer for data returned by the query.
It is actually the address of the memory location of a query work area on the database server.
The program can FETCH rows from the cursor with multiple FETCHes
A cursor must be DECLAREd for a given query, OPENed, FETCHed and CLOSEd

Syntax: CURSOR cursor_name IS query;

Example:
```
DECLARE .....
        CURSOR c_students IS SELECT name, lastname FROM students;
 ..... BEGIN
END;
```

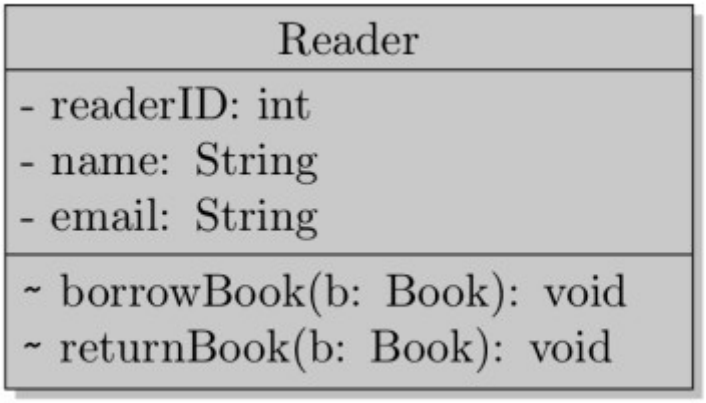
When the cursor is opened, the query is executed but the data are kept in a memory area on the server
The FETCH must be repeated till when the cursor is empty.
The CLOSE causes the memory area in the server to be freed.

Dynamic SQL is a programming technique for generating and running SQL statements at run time. It is mainly used to write the general-purpose and flexible programs where the SQL statements will be created and executed at run-time based on the requirement.
You need dynamic SQL to run:

- SQL whose text is unknown at compile time
- For example, a SELECT statement that includes an identifier that is unknown at compile time (such as a table name) or a WHERE clause in which the number of subclauses is unknown at compile time

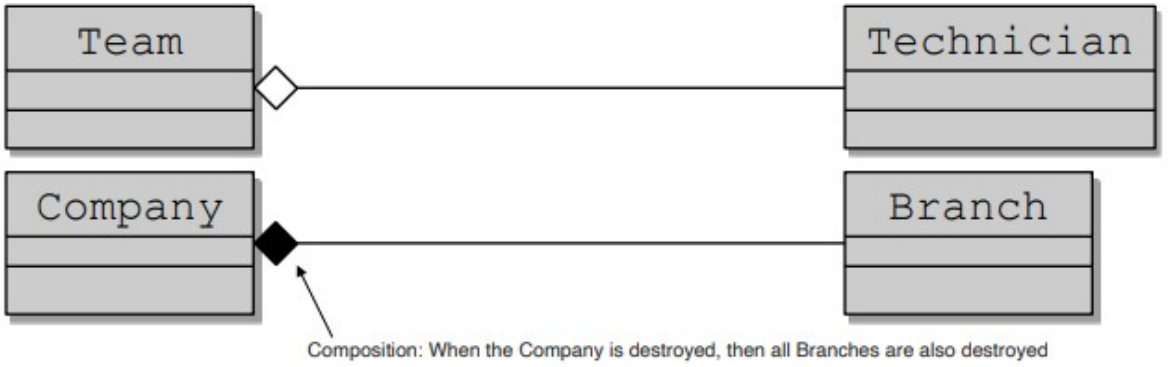
While Static SQL has the following advantage:
Successful compilation verifies that static SQL statements reference valid database objects and that the necessary privileges are in place to access those objects.

Object Oriented Model
It used to represent the reality using objects, like OOP.
To represent entities we use classes diagrams.
The diagram used for this model is different from ER.
The behavior is optional, but the declaration of field is mandatory.

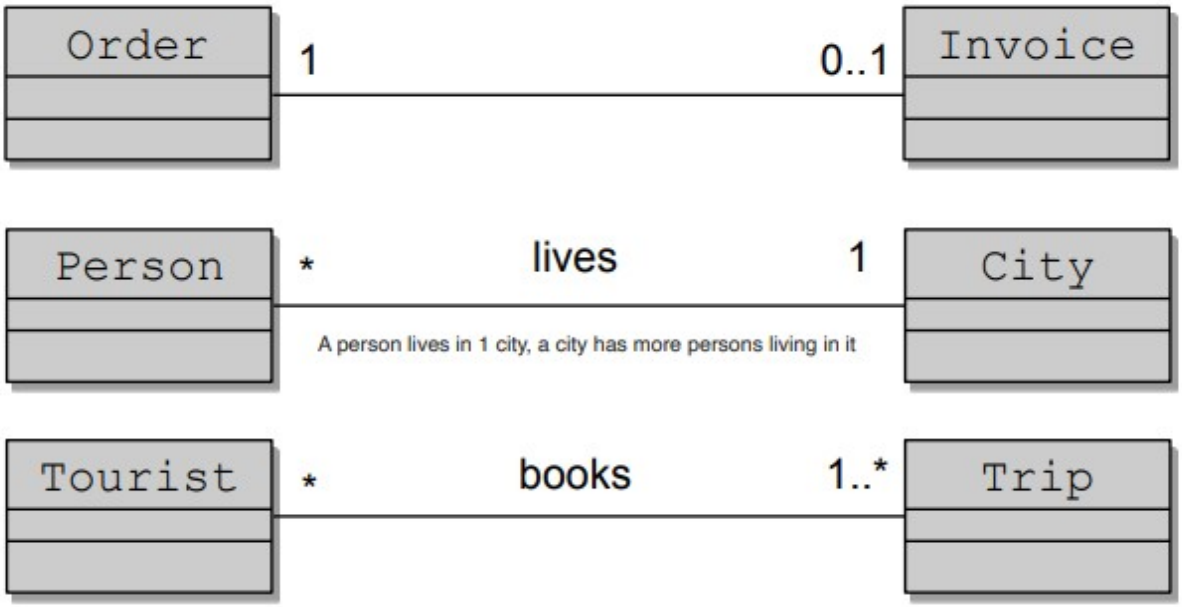


The association types are the same of ER (1:1;1:N;N:1;N:N)

To do aggregation we use the following notation



The first is independent from team, if we delete team, technician will exist anymore.
In the second if we delete a company branch will be deleted to.



There also here the normalization process.

Object
An object is an entity in the real world, which is distinguishable one from another.
The notion of object, in the object oriented data model, corresponds to that of entity, in the ER data model.
Class
A class represent a group of object that:
- share the same variables
- respond to the same messages
- use the same methods

The notion of class, in the object oriented data model, corresponds to that of entity set, in the entity-relationship data model.

Objects can be identified in different ways:
- **value based:** the data value identifies an entity
- **name base:** a name provided by the user identifies an entity
- **built-in:** the identification of the entities is inherent in the programming language and/or in the data model

An object can contains references to other objects.
We could delegate the object identifier by system, the disadvantages are:
- object identifiers are usually system specific and will need to be translated if you migrate data to a different system.
- Could be redundant, if the entity being modeled already has a natural unique attribute

The peculiarities of an object-oriented model are not well implemented in conventional databases (relational ones).
The incompatibility between the object oriented model and the relational model is called impedance mismatch.

ORM is a programming technique used to facilitate the integration of OOP with RDBMS systems.

There are 2 approaches:
- RDBMS → OO Language
- OO Language → RDBMS (Code first)

advantages:
- overcoming the incompatibility between the object project and relational model
- Reduce the amount of code
- change DBMS is easy and not require rewrite code in the persistence layer

Implementations:
- JAVA : JPA
- .NET: Entity Framework
- PHP: CakePHP

Object Oriented Database
are use to store complex data, also multimedia file and others.
OO relational DBMS need to ensure the functionality of RDBMs with respect to traditional data management.
We can query object data.
For each primitive type exists a fixed or predefined set of operations that can be performed on it, this limitations often limit the representation of real data.
There are 2 types of data:
- Simple types: primitive
- Complex types:

- Abstract data type(ADT): used to defined types (column types)
- Row type (table types)

ADTs are totally encapsulated!!! → the components of an ADT are accessible only through the available functions (methods).

Functions can be summarized in 3 types:

1. **Constructor:** used to create new instance of an ADT, this method has the same name of the type.
2. **Observer**: getter, field(column) or columnType.field
3. **Mutator**: setter object.field='...'

Row type
we can create a table base on a particular type

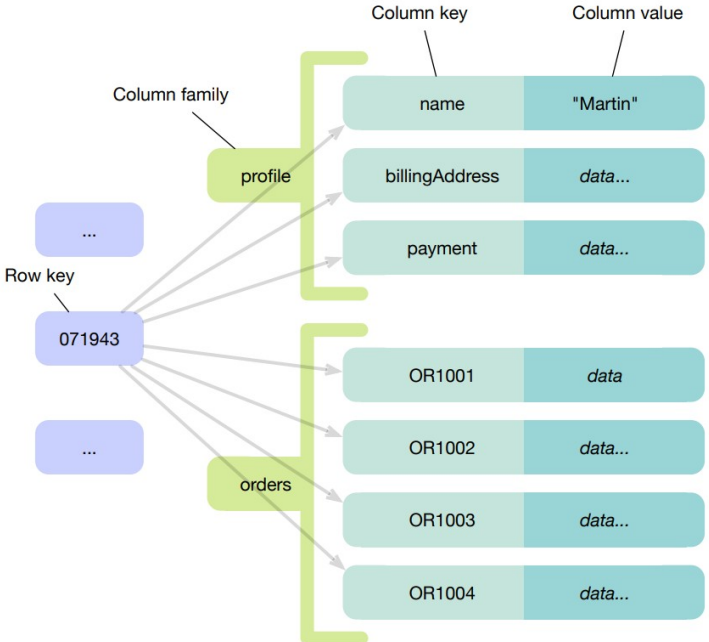
```
1 CREATE TABLE employees OF t_employee;
```

it will structured the table based on the type passed in the OF statement.

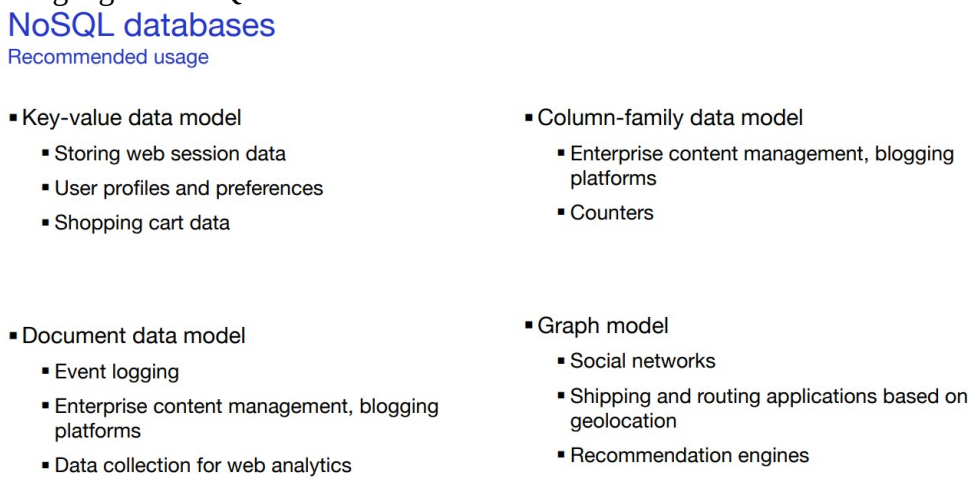
Row type hasn't encapsulation mechanism, we can execute query in standard way.
It's useful for relationships definition.
There isn't a primary key,but DBMS give to row ad unique id (OID: object identifier).
Ref type
it needs to create relationship between objects.

To create a relation we use **REF(type)** keyword
To manipulate nested table we MUST call in the table call selecting that becuase it is stored as an other table.

NOSQL databases save data in different ways then relational, indeed they saves data without table division, in these type of database redundancy is a good thing.
These db was born near 2009/2010, they are often opensource, and schema free, you don't have to define schema before inserting of data.
Some names: mongoDB, redis, cassandra, dynamo db.
Some times application can use more database technologies based on what have you to save.
Key-value data model
this type save data associating data values to a key.
Column-family data model
this type associate data to key that reference a column family that has a key with associated data.



Document data model
It use JSON file with structured objects, you can query the JSON file.
It can also use XML file.
Graph data model
data are saved in nodes and they have relationships, that can be mono or bi directional.
The query's language isn't SQL.



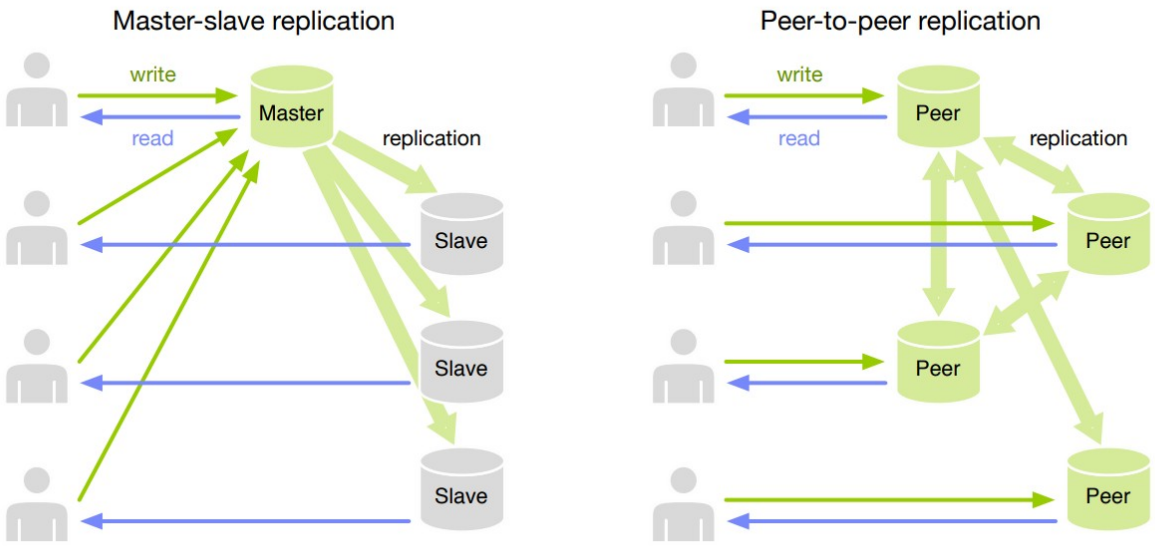
The biggest problem of this databases is that are designed for clustering, hence **distributed databases**.
Dimensions in which a database may need to scale:
Storage capacity
- Number of objects stored
  - Search engine: metadata of 2B pages of the world-wide-web
  - Social networks: user profiles of 1B users
  - Behavior tracking
  - Internet of Things

Read request throughput capacity
- Number of read requests per second
  - E-commerce
  - Online games: up to 100k reads per second

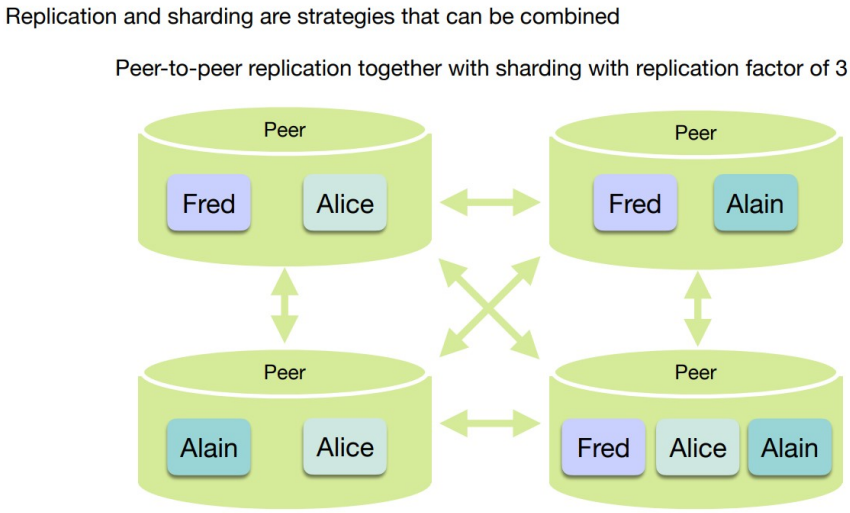
Write request throughput capacity
- Number of write requests per second
  - Online games: up to 100k writes per second
  - Internet of Things: up to 1M writes per second

Distribution principles
- Sharding: The storage space is subdivided into several zones. Each zone is assigned to a machine
- Replication: for each object a copy is created, each copy is stored on a different machine

▪ When data is replicated, there are two popular replication models: master-slave or peer-to-peer



Shardering manage the storage using rules and load-balancing.
Shardering can be combined with replication:



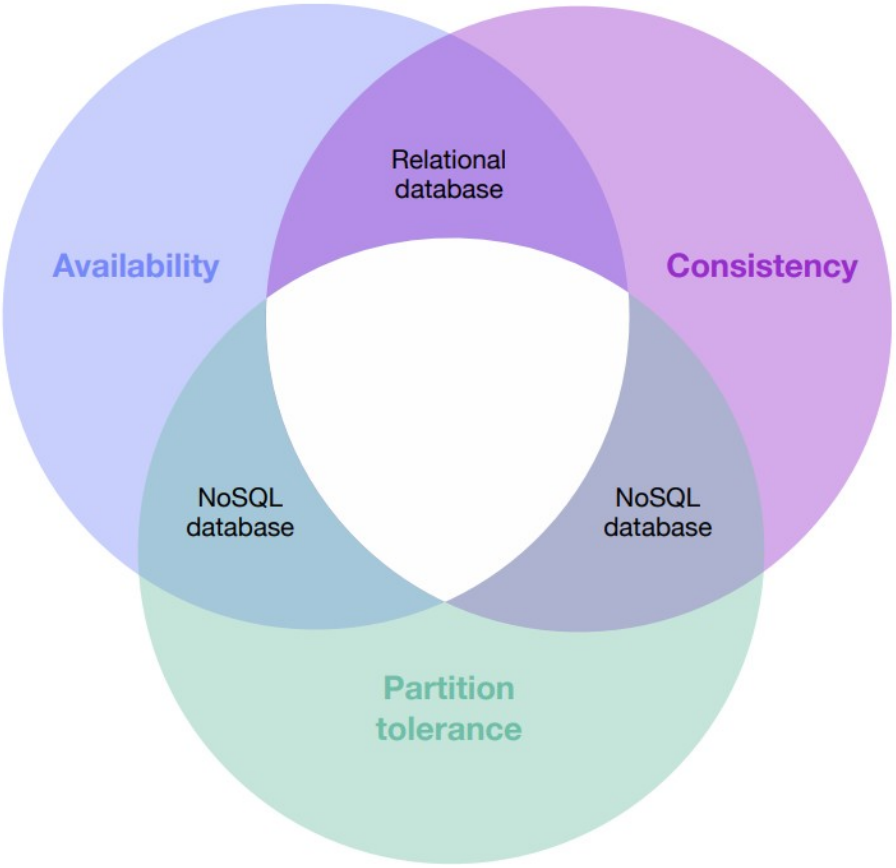
The Cap Theorem
In a distributed system, managing consistency (C), availability (A) and partition toleration (P) is important.
The CAP theorem states that between the goals of Consistency (C), Availability (A) and tolerance to network Partitions (P), only two can be achieved.

- **Consistency:** Strong consistency in the sense of ACID guarantees typically provided by relational databases. To all observers it appears as if there was a single database and updates appear to happen in the same sequence.
- **Availability:** The database remains available to applications for reading and writing at all times.
- **Partition tolerance:** In a distributed system, “partitions” occur when the network connection
  between nodes is interrupted, when a node crashes, but also when a node is taken down for maintenance.

The replication of data creates new consistency problems.
As long as the replicas are connected, all is well.

When they become disconnected (network partition) problems can occur.
Several approaches are possible.

- Refuse the writing of data to guarantee consistency of the replica: this is what relational databases do
- Allow the writing of data and accept inconsistent replicas: this is what NoSQL databases do-



Graph database
Relationships are mono-directional.
Every nodes knows the adjacent nodes, indexes are unnecessary.
They are based on graph theory.
It can be built also using RDBMS but performance are very bad.
Neo4j is a graph dbms.
You can save properties on nodes and on relationships.
It can be embedded with jar in the application locally or with server.
Cypher is a graph query language. Easy on
the eyes, while expressive and powerful.