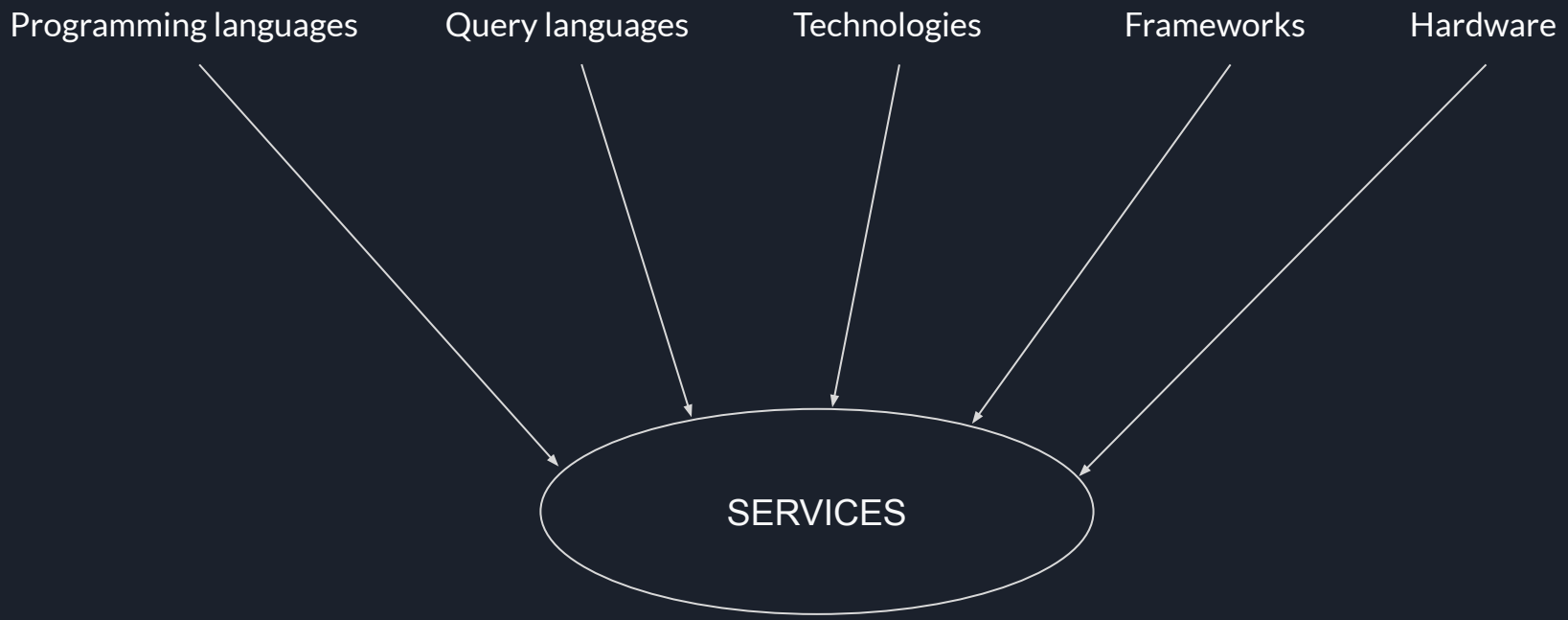
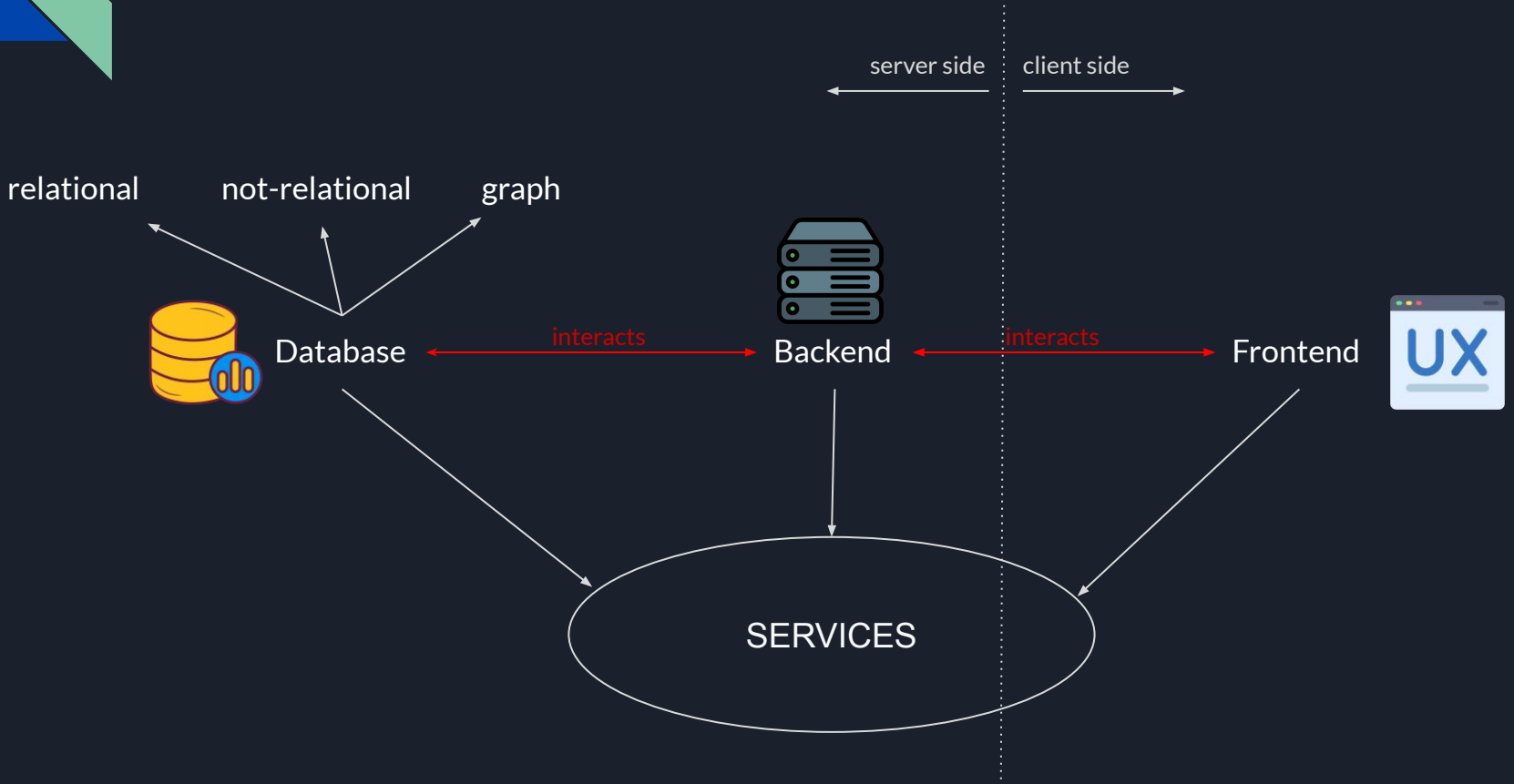
Database noSQL

Ing. Andrea Ferrari

Introduction



Introduction



SQL in short

Structured Query Language, the name is quite clear :)

Simplifying, it is a language used to ask for, and manipulate data.

It is used to querying a SQL database, a **relational database** (Db2, mySQL, PostgreSQL).

A relational database is particularly useful in handling structured data, because allow you to define and implement relations between groups of structured data (tables, rows).

SQL in short

Tables

order

id	user_id	item_id	quantity
1	4	1	10
2	4	2	5
5	6	1	1

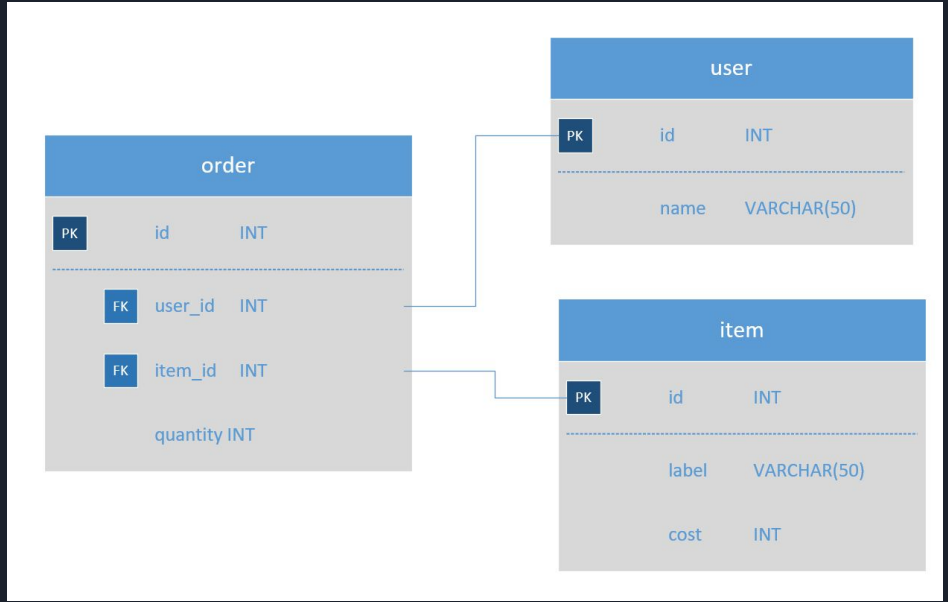
user

id	name
...	...
4	Paperino
6	Pluto

item

id	label	cost
1	pencil	1
2	bundle of sheets	5

ER



when SQL is hard to use?

Object–relational impedance mismatch: it is a set of conceptual and technical difficulties that are often encountered when a relational database is being served by an application program (or multiple application programs) written in an object-oriented programming language or style, particularly because **objects or class definitions must be mapped to database tables** defined by a relational schema.

Solution using SQL → DAO(Data Access Object) PATTERN

when SQL is hard to use?

Based on OOP (object-oriented programming) example 1

```
public class Item {
    int id;
    String label;
    int cost;
}
```


```
public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```

??

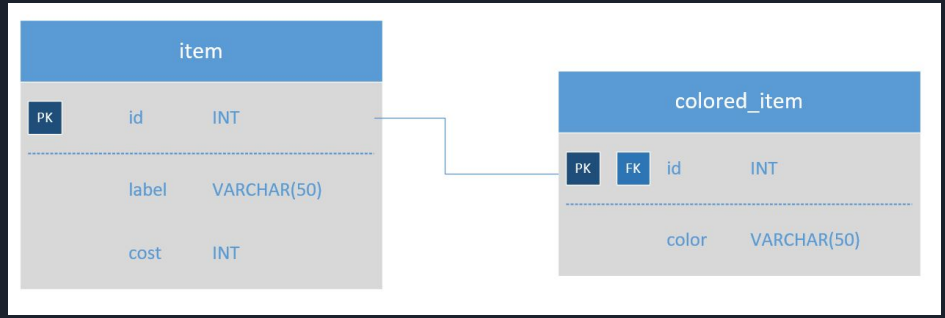
when SQL is hard to use?

Based on OOP (object-oriented programming) example 1

Solution A

```
public class Item {
    int id;
    String label;
    int cost;
}

public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```



If we want to fill a ColoredItem object, we need a join operation

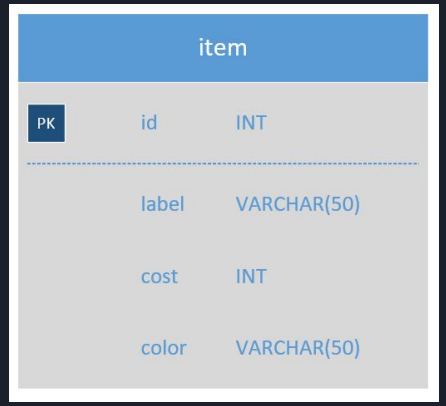
when SQL is hard to use?

Based on OOP (object-oriented programming) example 1

Solution B

```
public class Item {
    int id;
    String label;
    int cost;
}

public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```



If we want to fill an Item object, we need to filter out the “color” column.
if the extension class is new and we already have data inside our Item table?

when SQL is hard to use?

Based on OOP (object-oriented programming) example 1

Solution B

```
public class Item {
    int id;
    String label;
    int cost;
}

public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```

id	label	cost	color
1	pencil	1	??
2	boundle of sheets	5	??
3	new colored item	10	red

when SQL is hard to use?

Based on OOP (object-oriented programming) example 1

Solution B

```
public class Item {
    int id;
    String label;
    int cost;
}

public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```

id	label	cost	color
1	pencil	1	NULL
2	boundle of sheets	5	NULL
3	new colored item	10	red

when SQL is hard to use?

Based on OOP (object-oriented programming) example 2

```
public class Item {
    int id;
    String label;
    int cost;
}
```

```
public class ColoredItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String color;
}
```

```
public class ComplexItem
extends Item {
    //int id;
    //String label;
    //int cost;
    String[] parts;
}
```

We have an array here

We have 2 different extensions

Open questions, in that use-case?

Object–relational impedance mismatch

Objects (instances) reference one another and therefore form a graph in the mathematical sense.
Relational schemas are, in contrast, tabular and based on relational algebra, which defines linked heterogeneous tuples.

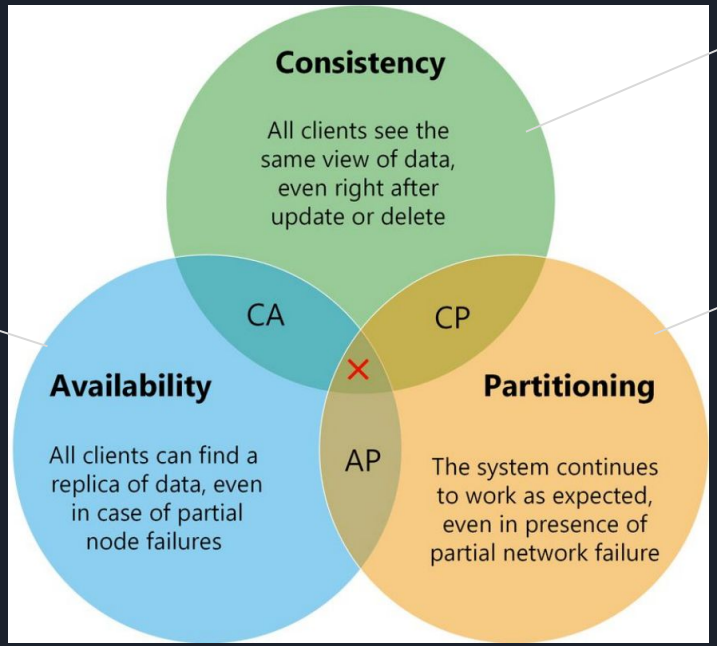
Note: you can use SQL or noSQL in any scenario, in computer science there is always more than one solution, but usually one will fill better in your needs.

Database CAP

Transaction
Usually, enabled as default

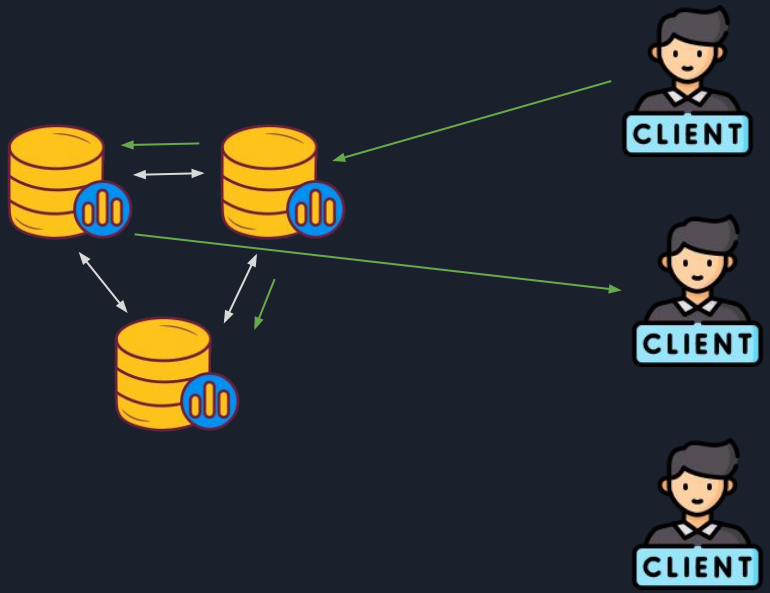
Distributed

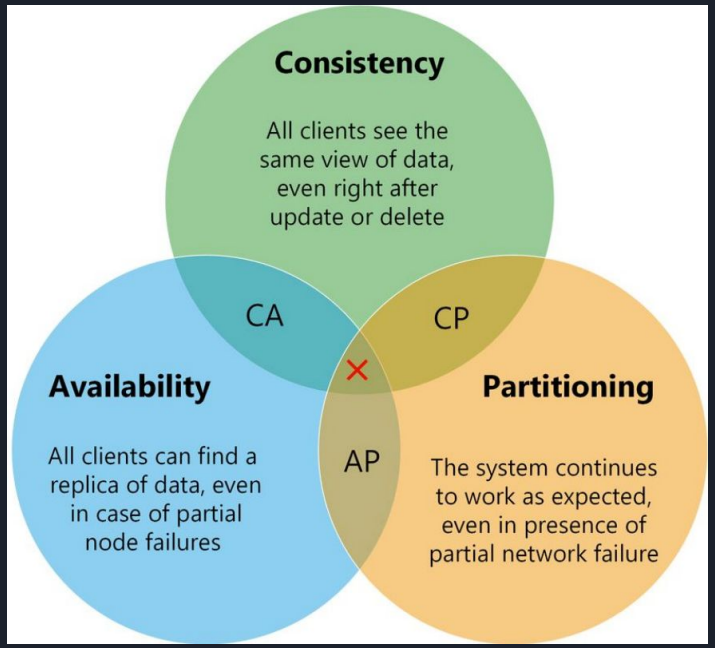
Replicated



Database CAP

Availability

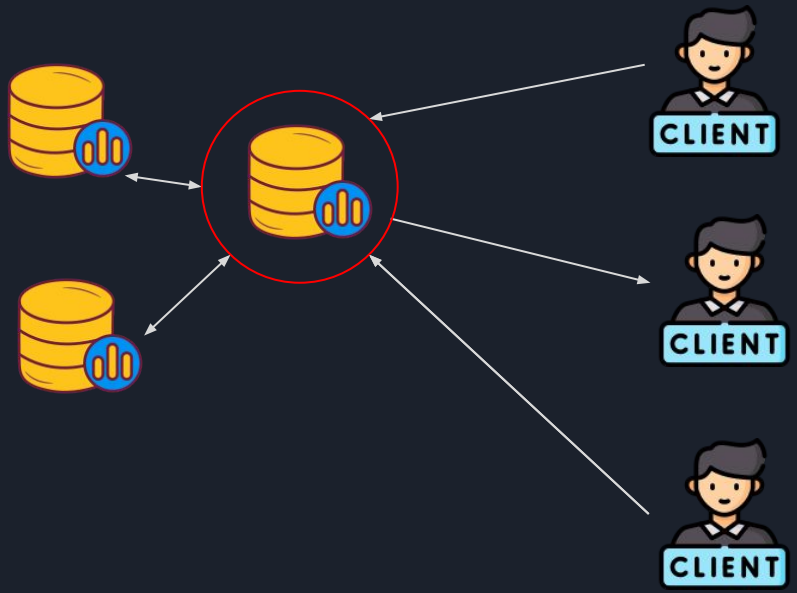


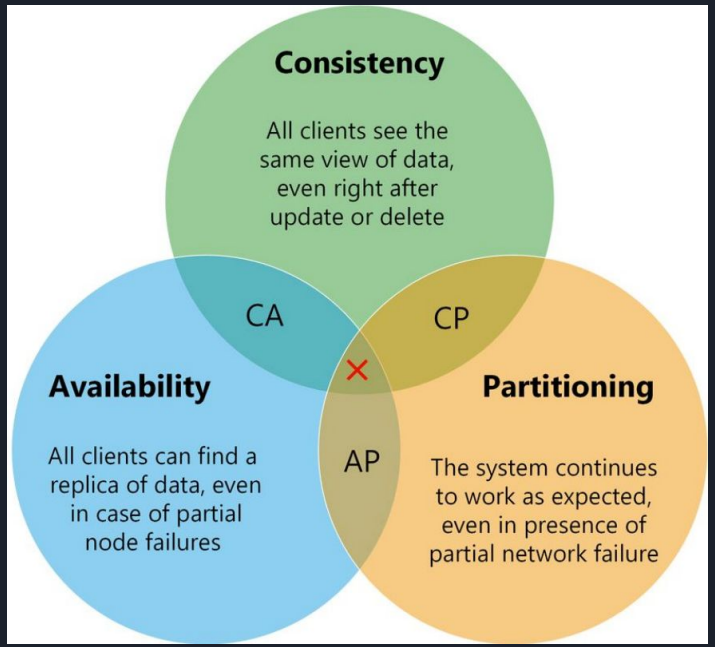


We have **more than one** node over the network in replication

Database CAP

Partitioning

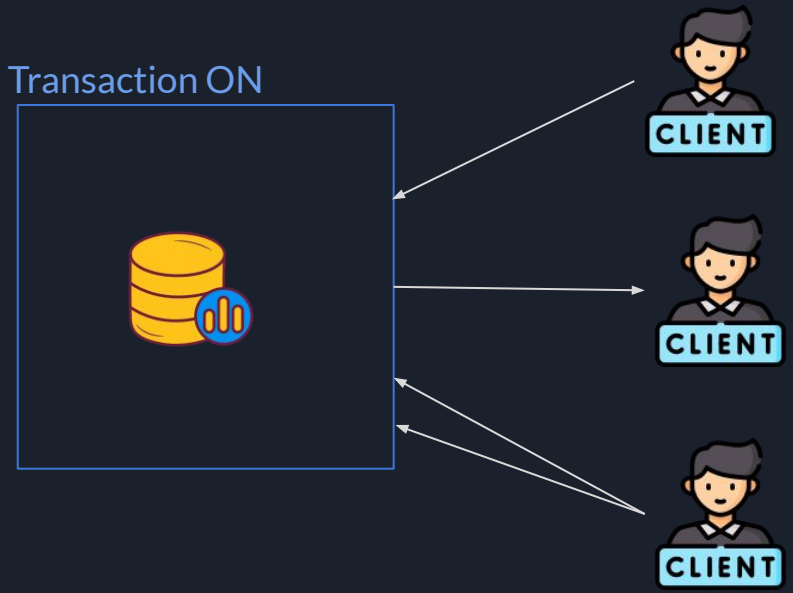


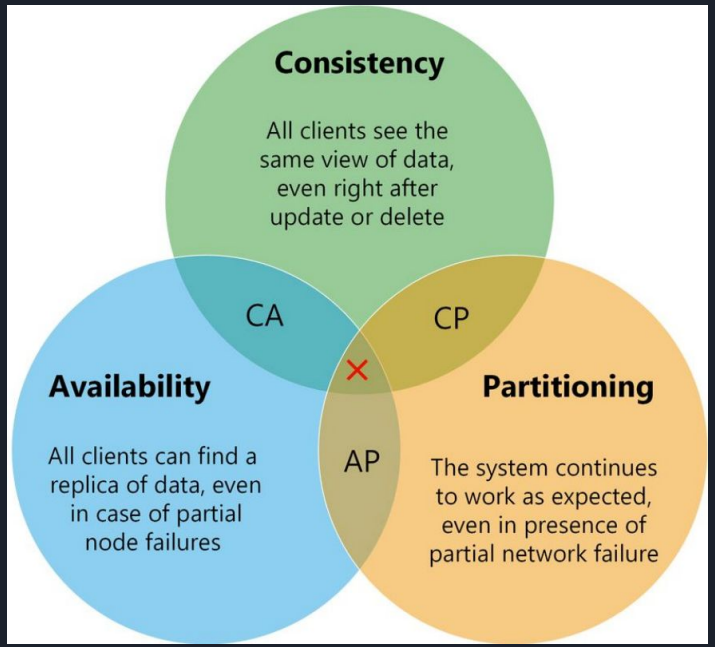


A single node is the entry point, but the data are **distributed in the same node**, we drastically reduce the network issues sensibility

Database CAP

Consistency

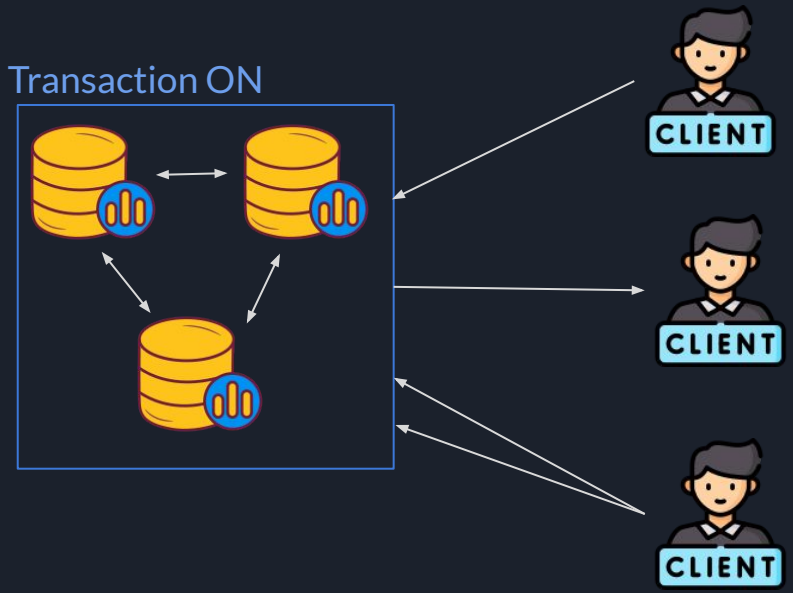


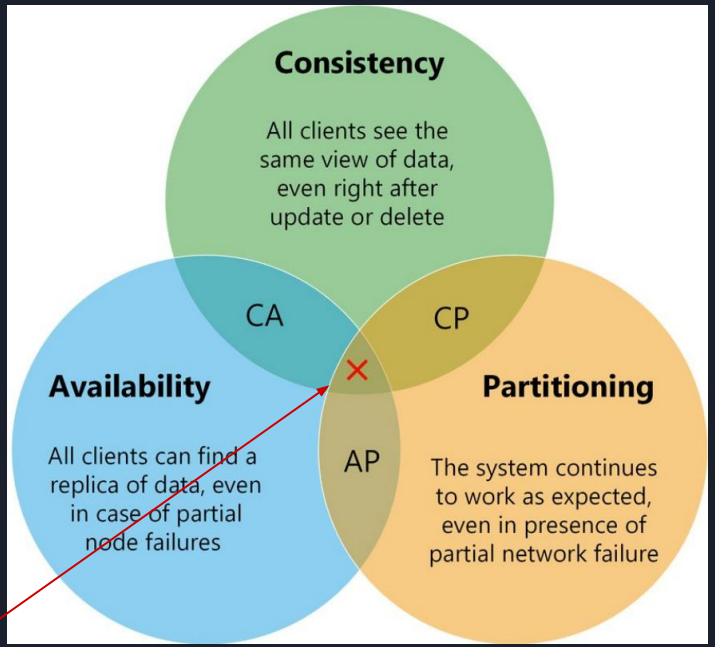


ACID Transaction
Atomicity, Consistency, Isolation, e Durability

Database CAP

CAP





if you see there is an X are we sure that is possible?

Database CAP

Usually, you can achieve these CA CP AP transparently

- Availability and Partitioning are managed by the database itself or through other technology, on top of the database structure
- Consistency is usually set on the database configuration or managed by the upper-level service (backend).

NoSQL

A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century. Triggered by the needs of Web 2.0 companies. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called Not only SQL to emphasize that they may support SQL-like query languages or sit alongside SQL databases.

NoSQL

Type	Notable examples of this type
Key–value cache	Apache Ignite, Couchbase, Coherence, eXtreme Scale, Hazelcast, Infinispan, Memcached, Redis, Velocity
Key–value store	Azure Cosmos DB, ArangoDB, Amazon DynamoDB, Aerospike, Couchbase, ScyllaDB
Key–value store (eventually consistent)	Azure Cosmos DB, Oracle NoSQL Database, Riak, Voldemort
Key–value store (ordered)	FoundationDB, InfinityDB, LMDB, MemcacheDB
Tuple store	Apache River, GigaSpaces, Tarantool, TIBCO ActiveSpaces, OpenLink Virtuoso
Triplestore	AllegroGraph, MarkLogic, Ontotext-OWLIM, Oracle NoSQL database, Profium Sense, Virtuoso Universal Server
Object database	Objectivity/DB, Perst, ZopeDB, db4o, GemStone/S, InterSystems Caché, JADE, ObjectDatabase++, ObjectDB, ObjectStore, ODABA, Realm, OpenLink Virtuoso, Versant Object Database, ZODB
Document store	Azure Cosmos DB, ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB, eXist-db, IBM Domino, MarkLogic, MongoDB, RavenDB, Qizx, RethinkDB, Elasticsearch, OrientDB
Wide Column Store	Azure Cosmos DB, Amazon DynamoDB, Bigtable, Cassandra, Google Cloud Datastore, HBase, Hypertable, ScyllaDB
Native multi-model database	ArangoDB, Azure Cosmos DB, OrientDB, MarkLogic, Apache Ignite,[22][23] Couchbase, FoundationDB, Oracle Database
Graph database	Azure Cosmos DB, AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Blazegraph
Multivalue database	D3 Pick database, Extensible Storage Engine (ESE/NT), InfinityDB, InterSystems Caché, jBASE Pick database, mvBase Rocket Software, mvEnterprise Rocket Software, Northgate Information Solutions Reality (the original Pick/MV Database), OpenQM, Revelation Software's OpenInsight (Windows) and Advanced Revelation (DOS), UniData Rocket U2, UniVerse Rocket U2