Class6-DB-Dec15

Assign	
Status	Completed
Priority	
Date Created	@December 15, 2021 5:26 PM
Due Date	
Property	

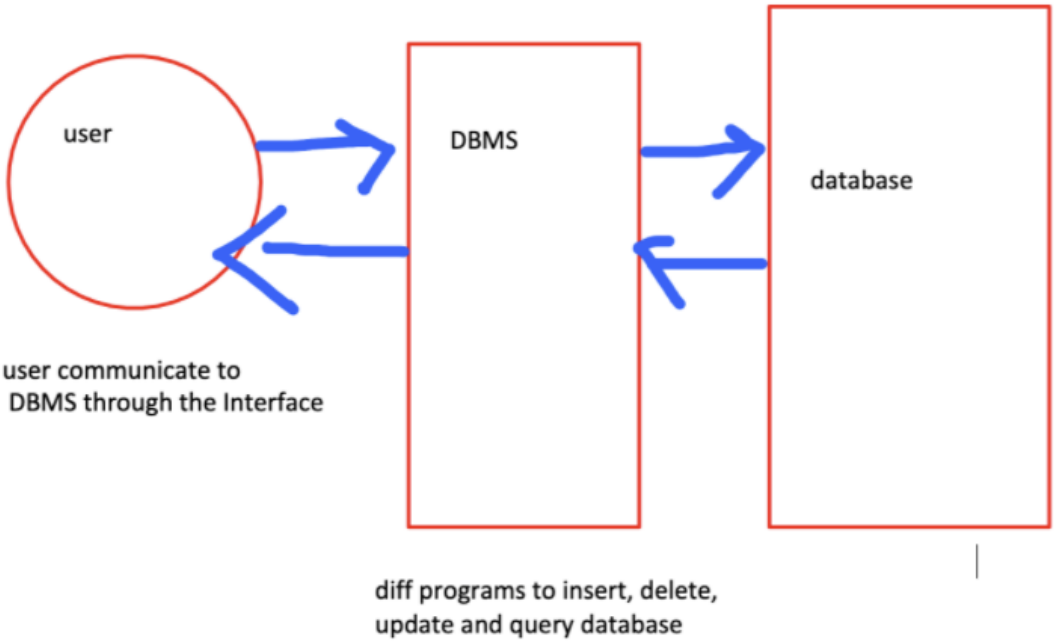
1. Relational Database

Database: is a collection of interrelated data which helps in efficient retrieval, insertion and deletion of data from it

DBMS: Database Management System

The software is designed to store, retrieve, define and manage data in database

DBMS help to manipulate database



SQL: structure query language

syntax is different among different DBMS

limit (offset), (num)

to get 11 - 20
MySQL:

```
select *
from tableName
order by name
limit 10, 10
```

Oracle:

```
select *
from tableName
order by name
offset 10 rows fetch next 10 rows only;
```

database normalization

is to eliminate redundant and ensure data is stored logically

There are 6 normal forms, achieves best in 3rd normal form

1NF(first normal form)

- each table cell should contain a single value
- each record needs to be unique

The second row should be eliminated since it's not unique.

id	name	address
1	James Xu	123 street, NY, US
~~1~~	~~James Xu~~	~~123 street, NY, US~~

2NF

- be in 1NF (based on)
- single Column primary key

id	firstName	lastName	age	city	street	zipcode
1	James	Xu	12	NY	234 street	123456
2	James	Liu	12	NY	234 street	123456

add id column can uniquely identify each row

~~firsName + lastName: composite primary key~~

the two columns form the primary key

so if the table should follow the 2NF, **it should only have one column as the PK**

we can add an id column as PK to follow 2NF

3NF

- be in 2NF(based on)
- has no transitive functional dependencies

transitive functional dependencies: is when changing a non-key column, might cause any of the other non-key columns to change

id	name	birth_year	age
1	A	~~1996~~2000	25 21
2	B	1990	31

When a table is updated birth_year from 1996 to 2000, the age column will be changed from 25 to 21. That are transitive functional dependencies.

Avoid that when 3NF is followed.

normalization level higher != better

separate data(a big table) to different small tables, and use join to merge all the data

join is not efficient, which will cost lots of time → no efficient

try to avoid join operation

→ de-normalization (just can follow 1 or 2 normal form to increase performance, depend on business logic)

merge some tables into a big table

the big table could not follow the normal forms, but it can make us query data efficiently

Database VS. file system

difference between file system and database

file system	dnms
manage and organize the files in a storage medium	manage the database
redundant data	no redundant data
don't provide backup and recovery of data, if it is lost	provide backup and recovery of data if lost
no efficient query processing	efficient query processing
less data consistency	more data consistency
less security	more security

less expensive cost high

Non-relational database(no-sql database)

store data use different formats

all data are un-organized

2. Non-relational Database

no-sql database

- document data stores
- column familiar data stores
- key/value data store
- graph data store

1. document data store (MongoDB)
    a. collection - table
    b. document - row
        i. BSON: binary json data

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

2. Column family data store

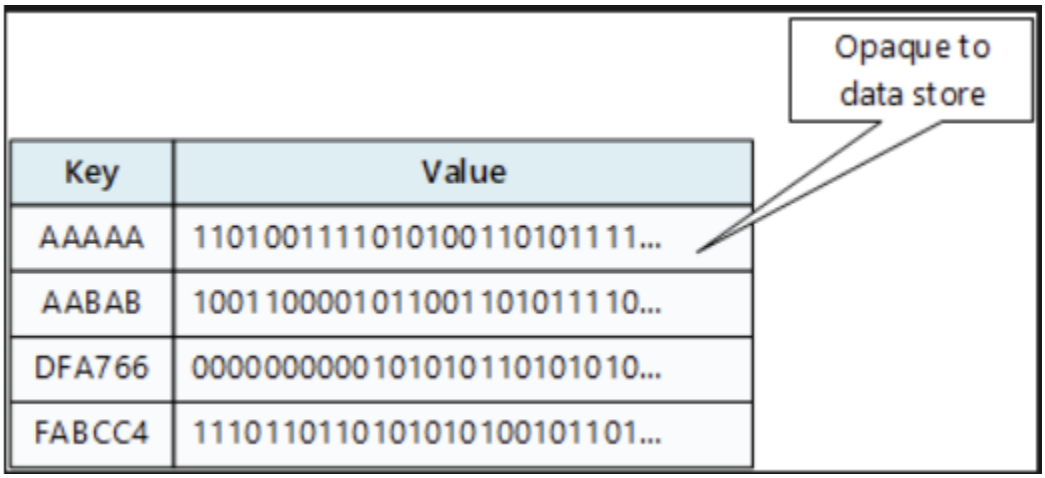
CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

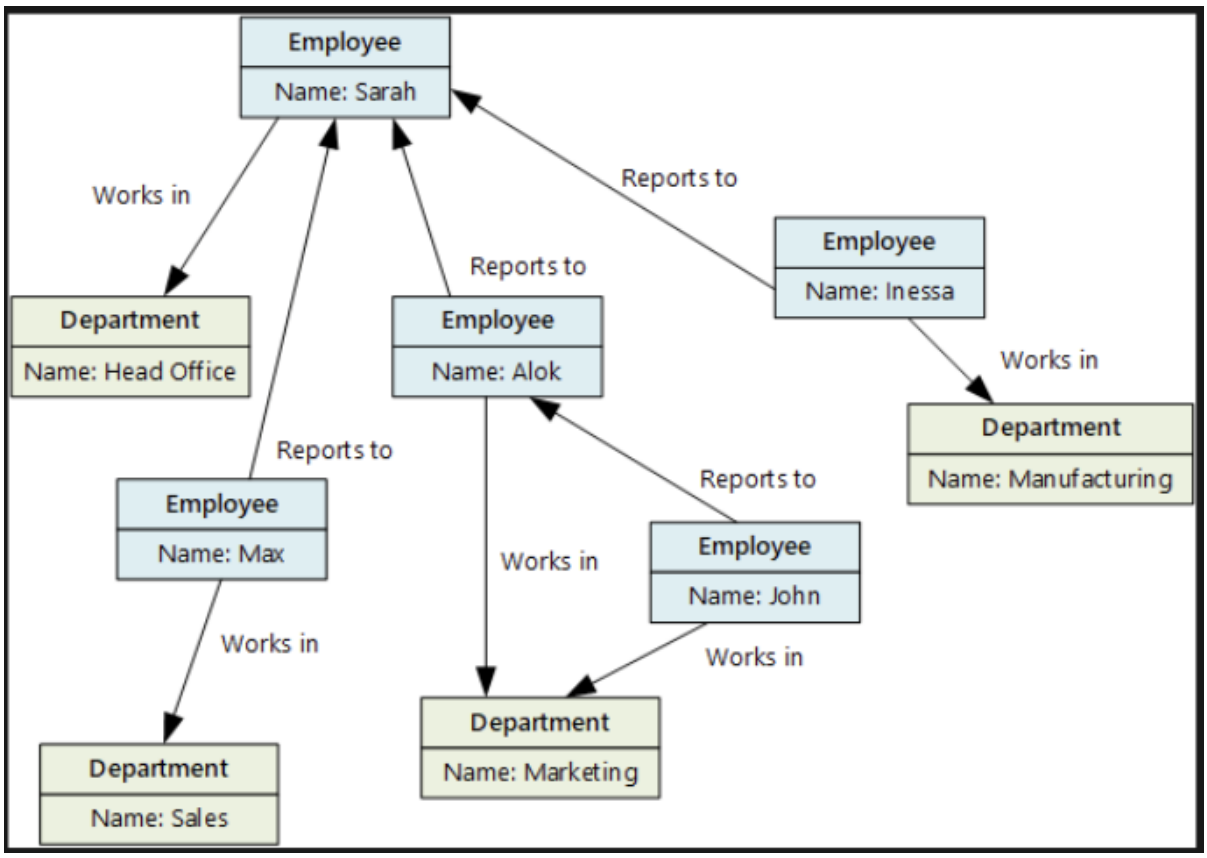
each data are logically related (related to content of Column Family)

each row has additional elements flexibly

3. Key-Value data store: look like a very big HashMap (redis)



4. Graph Data store
    - Node
    - Edge



Examples for each kind of no-sql

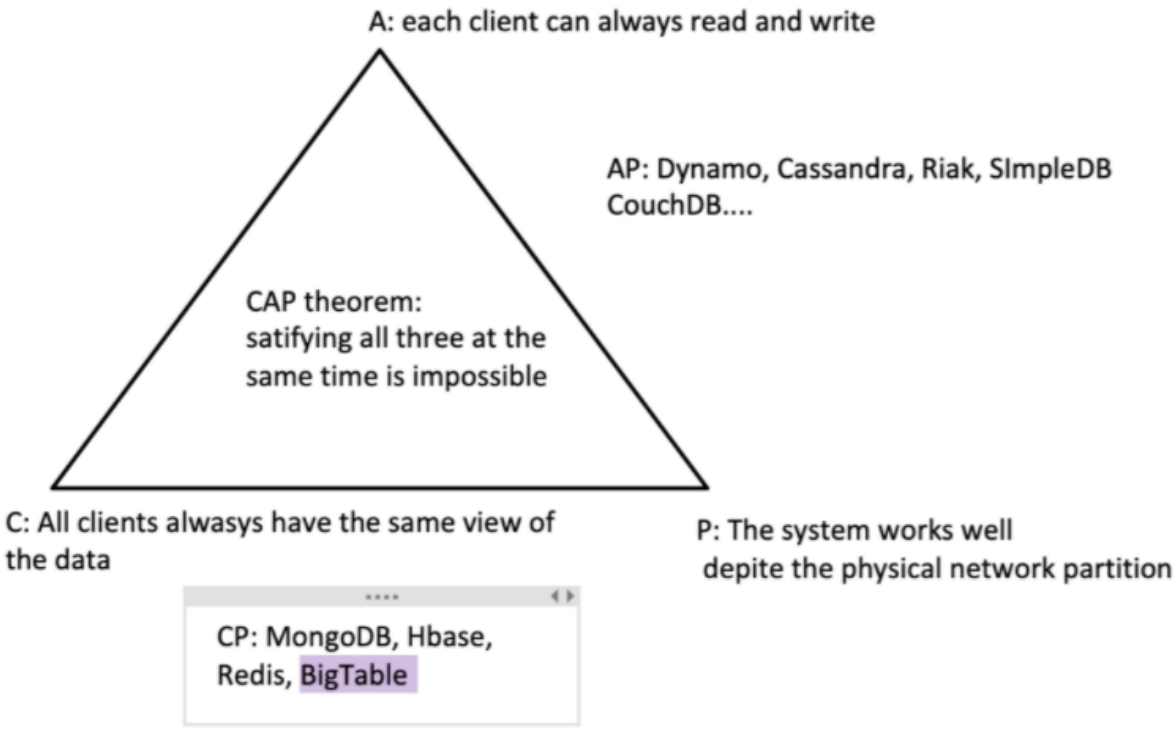
- document: **MongoDB**, CouchDB
- Column family: **Cassandra**, HBase
- Key-value: **Redis**, Riak
- Graph Datastore: Neo$j, GraphDB

need to be familiar with these three, MongoDB, Cassandra, and Redis

CAP principle in no-sql database

- Consistency
- Availability
- Partition tolerance

Satisfy these three are impossible.



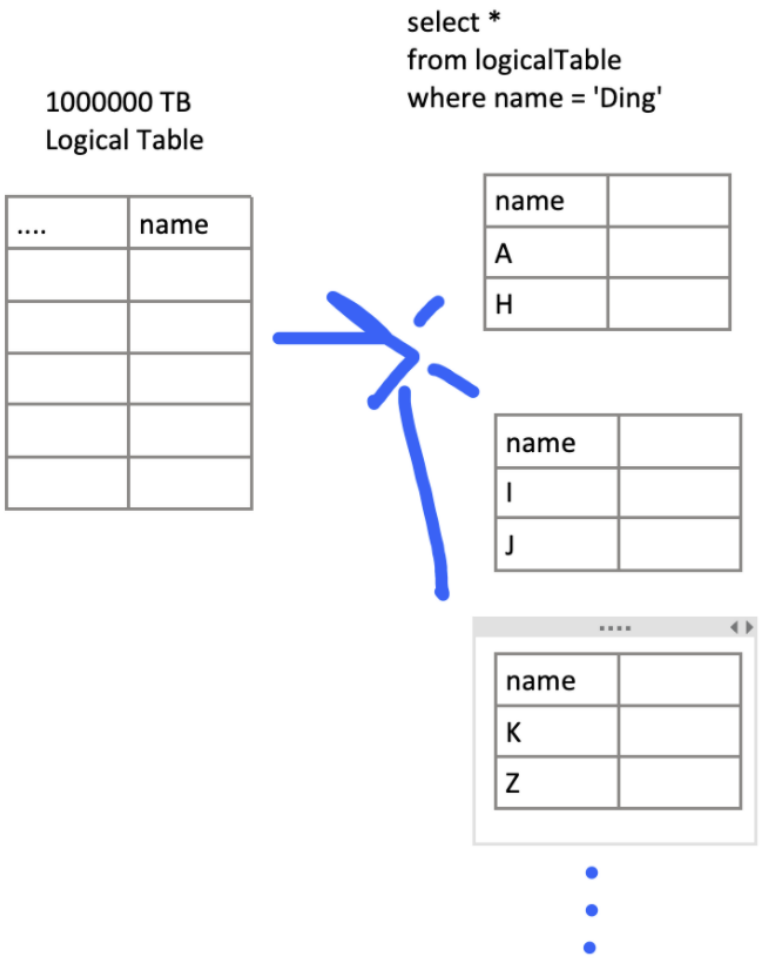
AP: Cassandra, Dynamo

Dynamo - AWS

BigTable - Google Cloud

3. Sharding and Replica

Sharding - distribute a single logical database across a cluster of machine

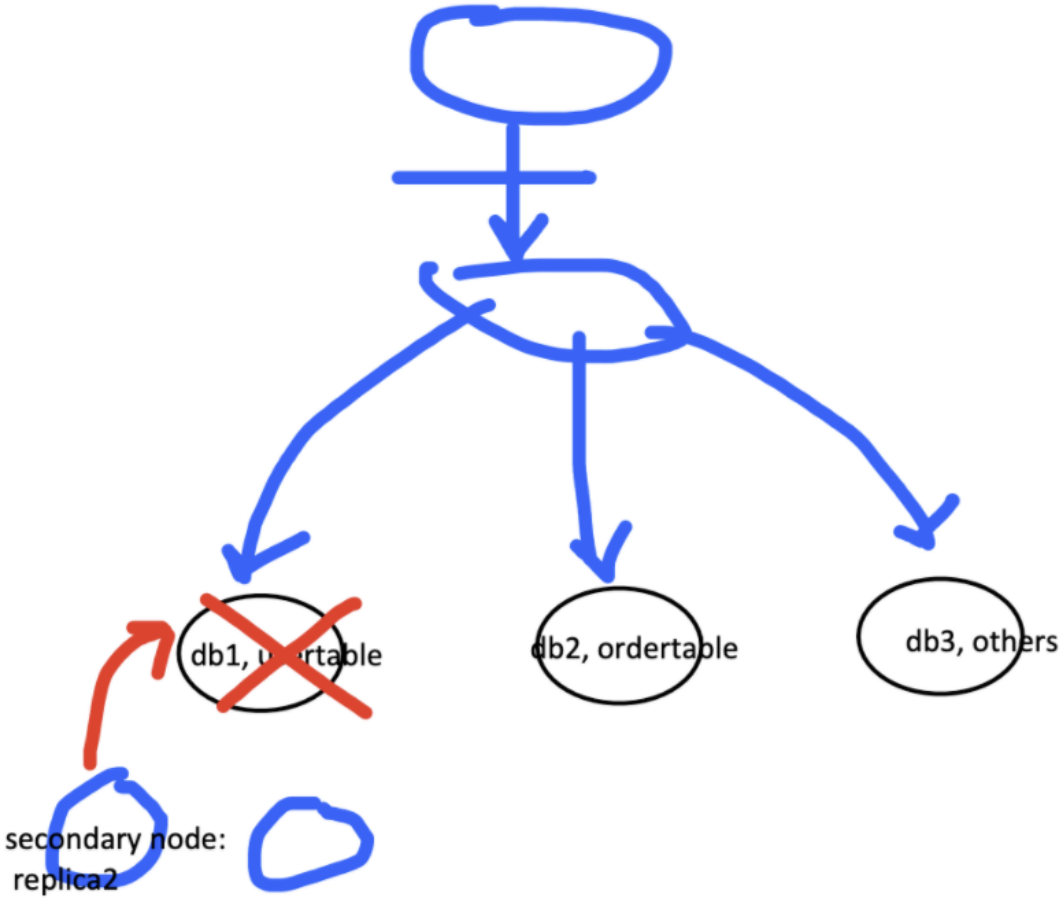


there is a large table with 1000000TB data, which's no single machine can handle these huge data.

divide data to many small tables, and partition table based on column, for example name

replica:

- redundancy
- failover


db1, ordertable
db2, ordertable
db3, others
secondary node:
replica2