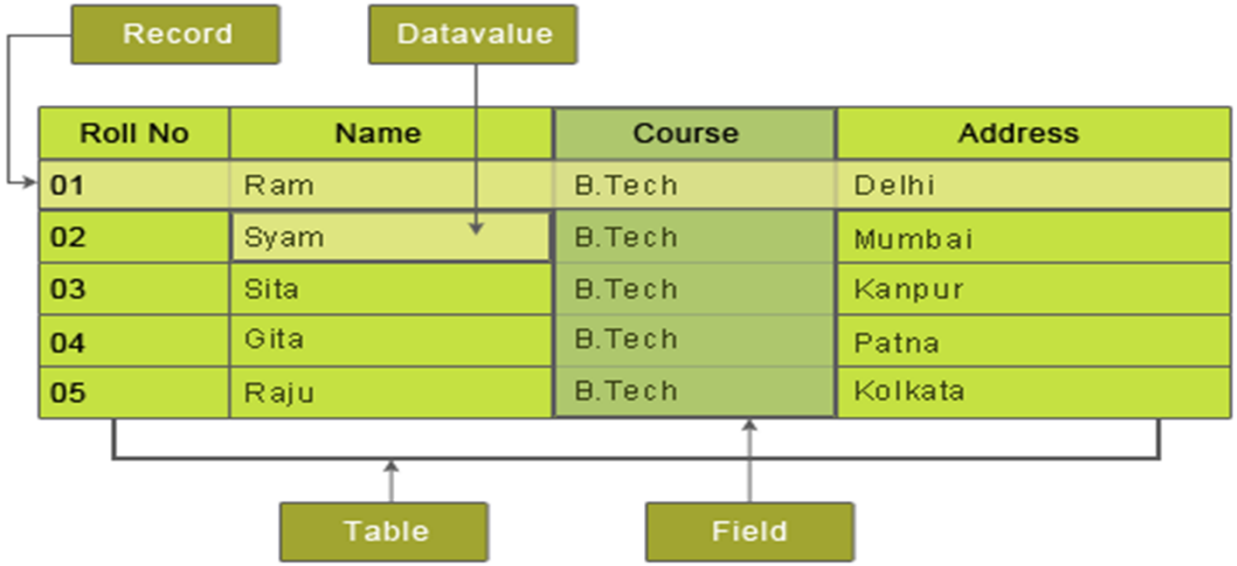
Concepts of Database Management System

Agenda

- Definition of DBMS
- Database System Applications
- View of Data
- Database Users and Administrators
- Different Types of Keys
- Introduction to Normalization
- Entity-Relationship Diagram (ERD)
- Other Terminologies of DBMS
- Introduction to NoSQL Databases
- CAP Theorem

Definition of DBMS

A **database-management system (DBMS)** is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.



Student_table(roll_no integer, name text, course text, address text)

Terminologies:

- DBMS
- Tuple / Record / Rows
- Attribute / Field
- Domain
- Instance
- Schema
- View

Database System Applications

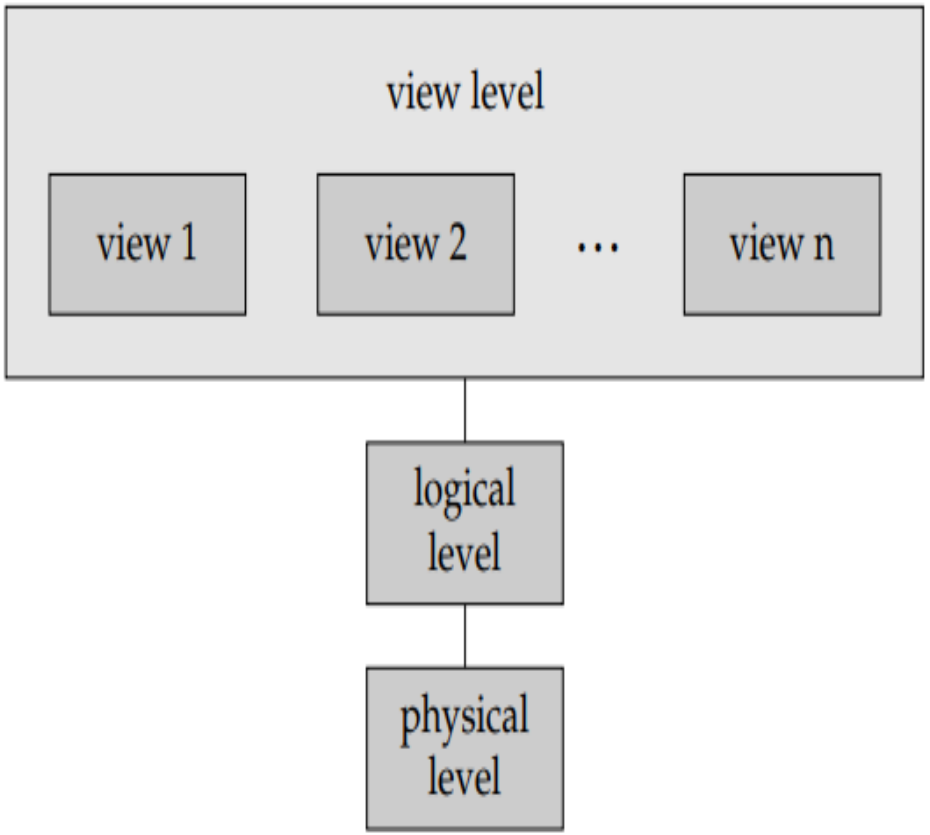
Databases are widely used. Here are some representative applications:

- **Banking:** For customer information, accounts, and loans, and banking transactions.
- **Airlines:** For reservations and schedule information.
- **Universities:** For student information, course registrations, and grades.
- **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
- **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
- **Sales:** For customer, product, and purchase information.
- **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.
- **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

Data Abstraction

Three levels of Data Abstraction:

- **Physical level:** The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.
- **Logical level:** It describes what data are stored in the database, and what relationships exist among those data.
- **View level:** Many users of the database system do not need all information stored in the database; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.



The three levels of data abstraction.

Database Users and Administrators

Database User Types

There are four different types of database-system users:

- **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a bank teller.
- **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.
- **Sophisticated users** interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor to get their query executed.
- **Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Database Administrator Roles

The functions of a Database Administrator (DBA) include:

- **Schema definition.** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure** and access-method definition.
- **Schema and physical-organization modification.** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
- **Granting of authorization for data access.** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance.** Examples of the database administrator's routine maintenance activities are: • Periodical backup, • Upgrading disk space as required, • Monitoring jobs running on the database and • Ensuring that performance is not degraded by very expensive tasks submitted by some users.

Different Types of Keys

Types of Keys

There are multiple different types of Keys:

- **Super Key**
- **Candidate Key**
- **Primary Key**
- **Alternate Key**
- **Natural Key**
- **Surrogate Key**
- **Composite Key**
- **Unique Key**
- **Foreign Key**
- **Secondary Key**
- **Partial Key**

Etc.

Different Types of Keys

Definition of Keys

Super Key: A super key is a set of one or more attributes (columns), which can uniquely identify a row in a table. As example in Student table, (Class, Sec, RollNo), (Class, Sec, RollNo, Gender), (Name, Address) are super keys. All attributes in a super key are sufficient but may not be essential.

Candidate Key: A candidate key is a set of one or more attributes (columns), which can uniquely identify a row in a table. As example in Student table, (Class, Sec, RollNo), (Name, Address) are candidate keys. All attributes in a candidate key are sufficient and also essential.

Primary Key: That very candidate key selected by DBA or Database Designer for the implementation purpose will be called as a primary key. As example if in Student table, (Class, Sec, RollNo) is used for implementation then it will be a primary key.
Attributes taking part in the construction of primary key are called prime attributes, others will be called non-prime attributes. Prime attributes will have NOT NULL constraints and not updatable.

Alternate Key: Unimplemented candidate key is called an alternate key. As example in Student table, unimplemented candidate key (Name, Address) can be considered as an alternate key.

Different Types of Keys

Definition of Keys

Natural Key: A natural key is an attribute that exists in the real world or is used by the business. It can be used to uniquely identify the row. As example Social Security Number (for US citizens), Tax File Number (TFN) (for Australian citizens), Aadhaar Card Number (for Indian citizen) etc.

Surrogate Key: The word surrogate means substitute. A surrogate key is an attribute that is invented or made up for the sole purpose of being used as the primary key. It has no value to the business or the real world.
An example of a surrogate key is an “address ID” for a table of addresses. Outside of the system, an address ID has no value to anyone. But for the database, it’s used to uniquely identify the record. These key values can get changed and NOT NULL. Mobile number in Student table is a surrogate key.

Composite Key: A composite key is a primary key, or unique identifier, that is made up of more than one attributes. As example in Student table (Class, Sec, RollNo) is a composite key.

Different Types of Keys

Definition of Keys

Unique Key: A unique key is an attribute in the table which is unique. It can be used to identify a row but may not be the primary key. PAN card number is an example. This is not updatable and can have NULL values.

Foreign Key: A foreign key is a column or set of columns in a table that refers to a primary key in another table. Foreign key will have domain constraints. As example -
Department (**Deptno**, Dept_name)
Teacher (**TeacherID**, Teacher_name, Teacher_qualification, **Deptno**)
Here Deptno in Teacher table is a foreign key. We can not allocate any teacher to any Deptno which is not existing in the Deptno column in Department table.

Secondary Key: Secondary key is a combination one or more attributes used for sorting or searching purpose on database tables.

Different Types of Keys

Definition of Keys

Partial Key: The set of attributes that are used to uniquely identify a weak entity set is called the Partial key. Only a bunch of the tuples can be identified using the partial keys. The partial Key of the weak entity set is also known as a discriminator.
In the following table EmpID is the partial key.

Empname	Dname	Relationship
Abir	Tripti	Mother
Abir	Sankha	Son
Abir	Briti	Daughter
Soumen	Jayita	Sister
Soumen	Aritra	Brother
Soumen	Agnibha	Son

Comparison between:

- Primary Key
- Unique Key
- Surrogate Key

Key	NOT NULL	Updatable	Example
Primary	TRUE	Not	Student_Reg_No
Unique	FALSE	Not	PAN_No
Surrogate	TRUE	Yes	Mobile_No

Definition of Normalization

Normalization: Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistency. Normalized databases will have no INSERT, DELETE and UPDATE anomalies. Whenever one database is unnormalized then we shall have three anomalies - Insert, Delete and Update.

Stu_Roll	Stu_Name	Stu_Addr	Course_No	Course_Name	Marks
1	Amit	Kolkata	C1	Mathematics	98
2	Dhiman	Patna	C1	Mathematics	97
3	Tripti	Noida	C2	Physics	95
2	Dhiman	Patna	C2	Physics	92
3	Tripti	Noida	C3	Chemistry	99
4	Chiranjib	Delhi	C2	Physics	91

Student_Course Table

Student_Course Table has following anomalies:

- Insert Anomaly
- Delete Anomaly
- Update Anomaly

Applying Normalization

After carrying out Normalization process, we have following three split tables:

Stu_Roll	Stu_Name	Stu_Addr
1	Amit	Kolkata
2	Dhiman	Patna
3	Tripti	Noida
4	Chiranjib	Delhi

Student Table

Stu_Roll	Course_No	Marks
1	C1	98
2	C1	97
3	C2	95
2	C2	92
3	C3	99
4	C2	91

Evaluation Table

Course_No	Course_Name
C1	Mathematics
C2	Physics
C3	Chemistry

Course Table

So all following three anomalies got resolved:

- Insert Anomaly
- Delete Anomaly
- Update Anomaly

Some of the Normalization rules are:

- INF, 2NF, 3NF, 4NF, 5NF
- BCNF (Boyce-Codd Normal Form)
- DKNF (Domain-Key Normal Form)
- PJNF (Project-Join Normal Form) etc.

Entity-Relationship Diagram

What is Entity-Relationship Diagram?

An Entity–Relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: **Entity set** and **Relationship set**.

Entity Relationship Diagrams are having following symbols:

- **Rectangle:** Represents Entity sets.
- **Ellipses:** Attributes
- **Diamonds:** Relationship Set
- **Lines:** They link attributes to Entity Sets and Entity sets to Relationship Set
- **Double Ellipses:** Multivalued Attributes
- **Dashed Ellipses:** Derived Attributes
- **Double Rectangles:** Weak Entity Sets
- **Double Lines:** Total participation of an entity in a relationship set
- **Double Diamond:** Identifying relationship

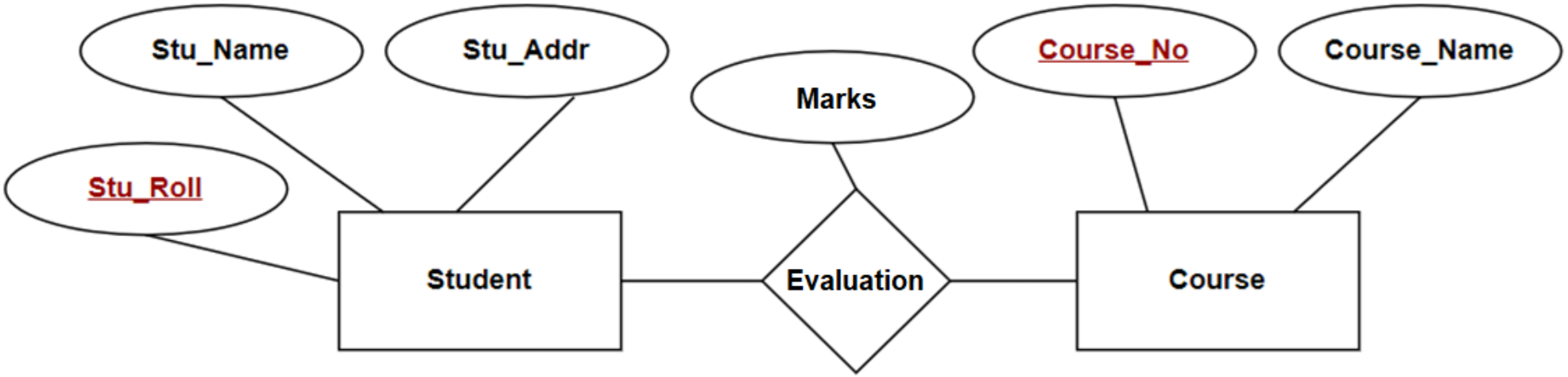
Entity-Relationship Diagram for the Student Database

Entity Set: Entity is something which has got some independent existence in an enterprise. With an Entity some set of attributes are connected. Depending upon the values of some attribute(s) an Entity can be identified uniquely. Collection of alike entities will form entity set.

Relationship Set: Relationship connects two or more than two entities. Collection of alike relationships will form relationship set.

$$primary\text{-}key(E_1) \cup primary\text{-}key(E_2) \cup \cdots \cup primary\text{-}key(E_n) \cup \{a_1, a_2, \ldots, a_m\}$$

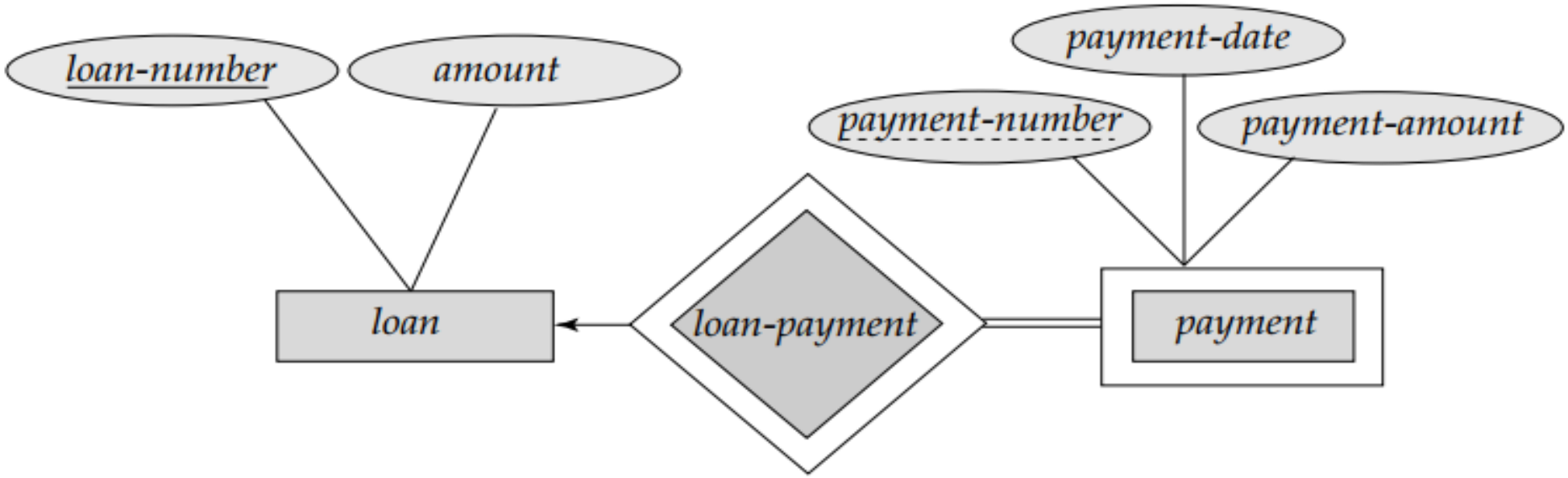
Entity–Relationship diagram against our Student database is shown below -



Types of Entity Sets

An Entity can be of following two types:

- **Strong Entity:** Strong Entity will have primary key.
- **Weak Entity:** Weak Entity will have partial key.



E-R diagram with a weak entity set.

Types of Attributes

An Attribute can be of following types:

- **Prime / Key Attribute:** Attribute which takes part in the construction of Primary Key.
- **Non-Prime / Non-Key Attribute:** Attribute which does not contribute to the construction of Primary Key.
- **Derived Attribute:** Attribute whose value will be calculated by some (business) logic and the value needs not to be supplied from the outside.
- **Single-valued Attribute:** Attributes which take only one value at a time.
- **Multi-valued Attribute:** Attributes which can hold multiple values.
- **Simple Attribute:** Simple attribute can not be divided into subparts.
- **Composite Attribute:** Composite attributes, can be divided into subparts.

Types of Attributes

An Attribute can be of following types:

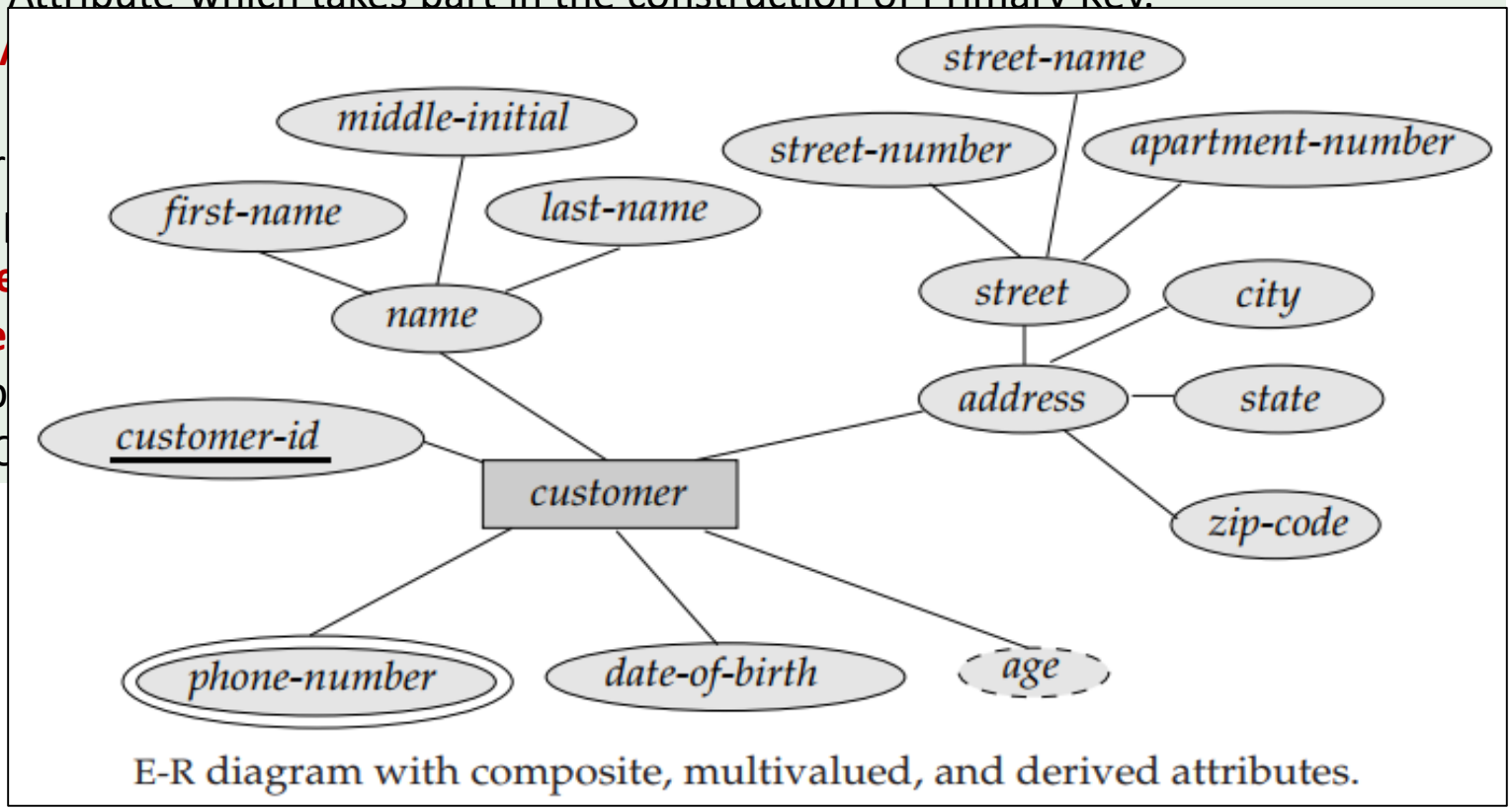
- **Prime / Key Attribute:** Attribute which takes part in the construction of Primary Key.
- **Non-Prime / Non-Key A**

Primary Key.

- **Derived Attribute:** Attr

value needs not to be su

- **Single-valued Attribute**
- **Multi-valued Attribute**
- **Simple Attribute:** Simp
- **Composite Attribute:** C

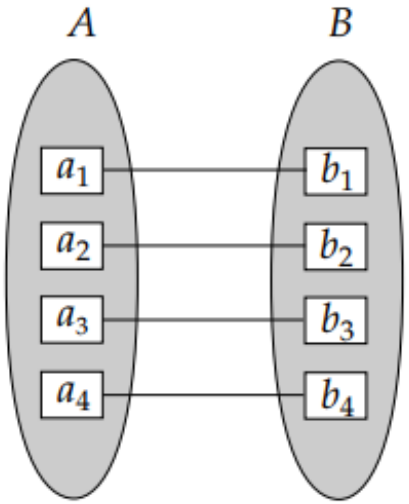


E-R diagram with composite, multivalued, and derived attributes.

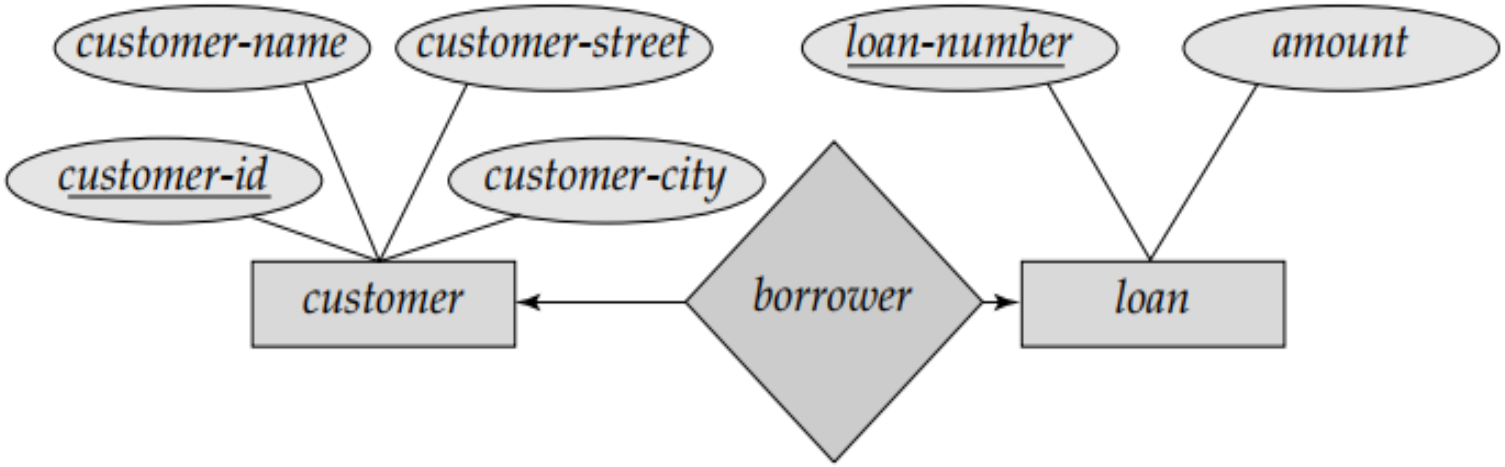
Mapping Cardinalities

Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

• **One to one:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



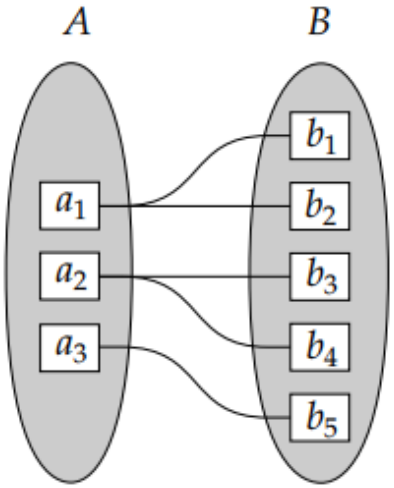
One to one.



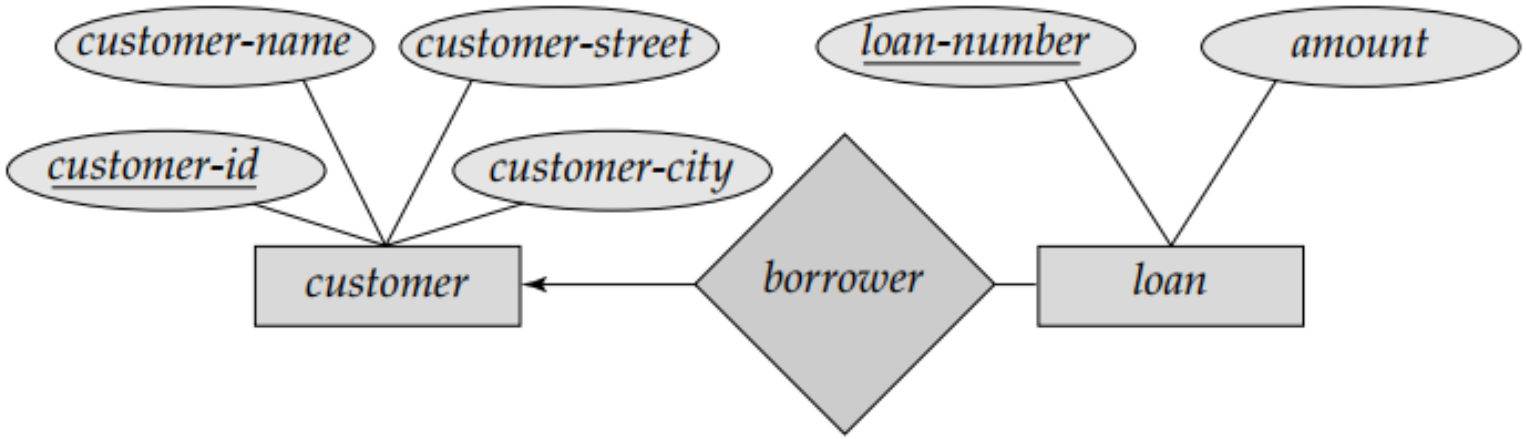
This bank issues non-parallel personal loans only.

Mapping Cardinalities

• **One to many:** An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.



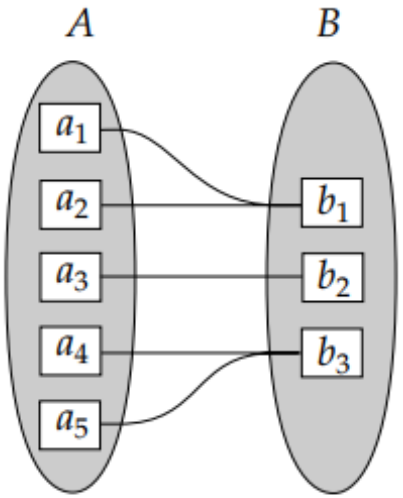
One to many.



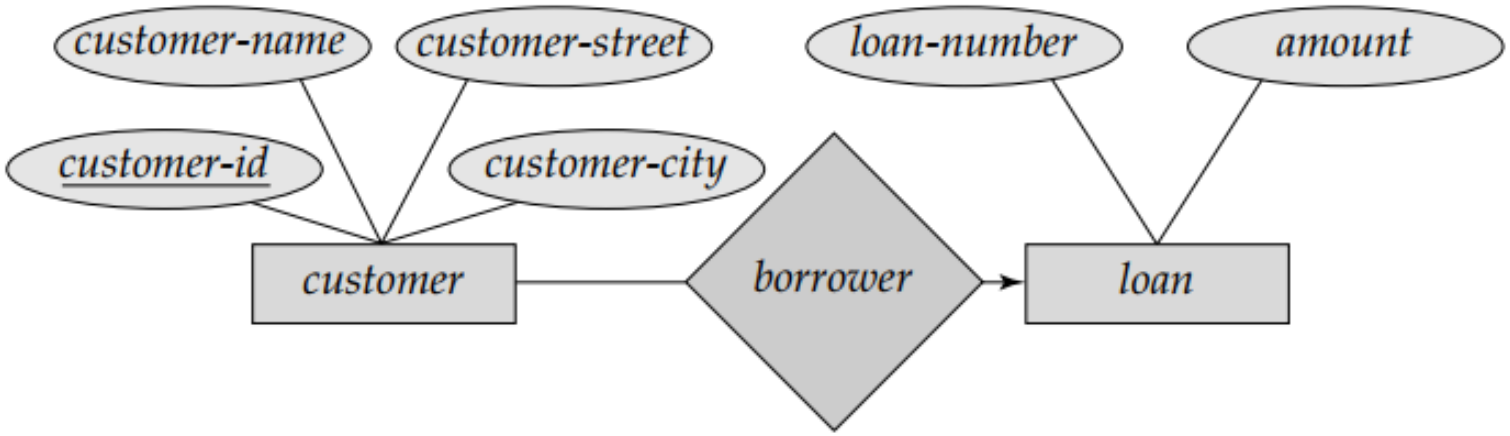
This bank issues parallel personal loans only. One loan can have only one customer as a loan holder and against one customer he/she can have multiple personal loan numbers.

Mapping Cardinalities

• **Many to one:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.



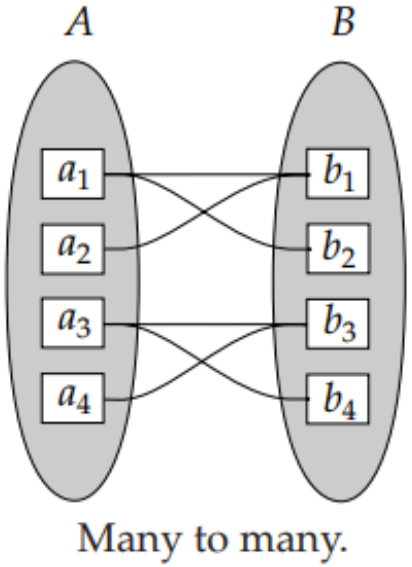
Many to one.



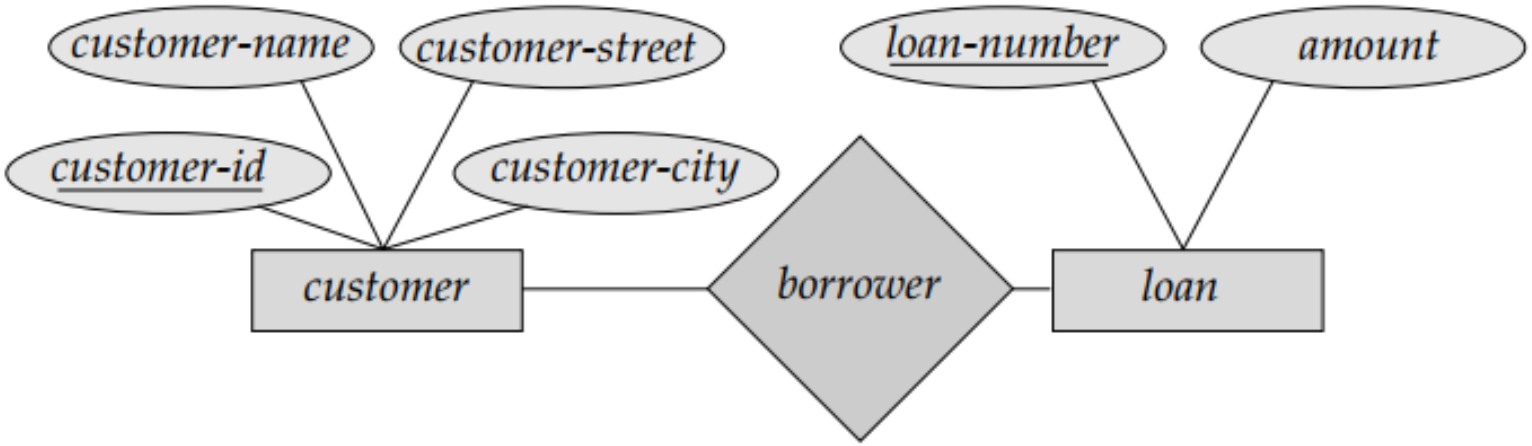
This bank issues non-parallel personal and group loans only. One loan can have multiple customers as loan holders and against one customer he/she can have only one loan number.

Mapping Cardinalities

• **Many to many:** An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



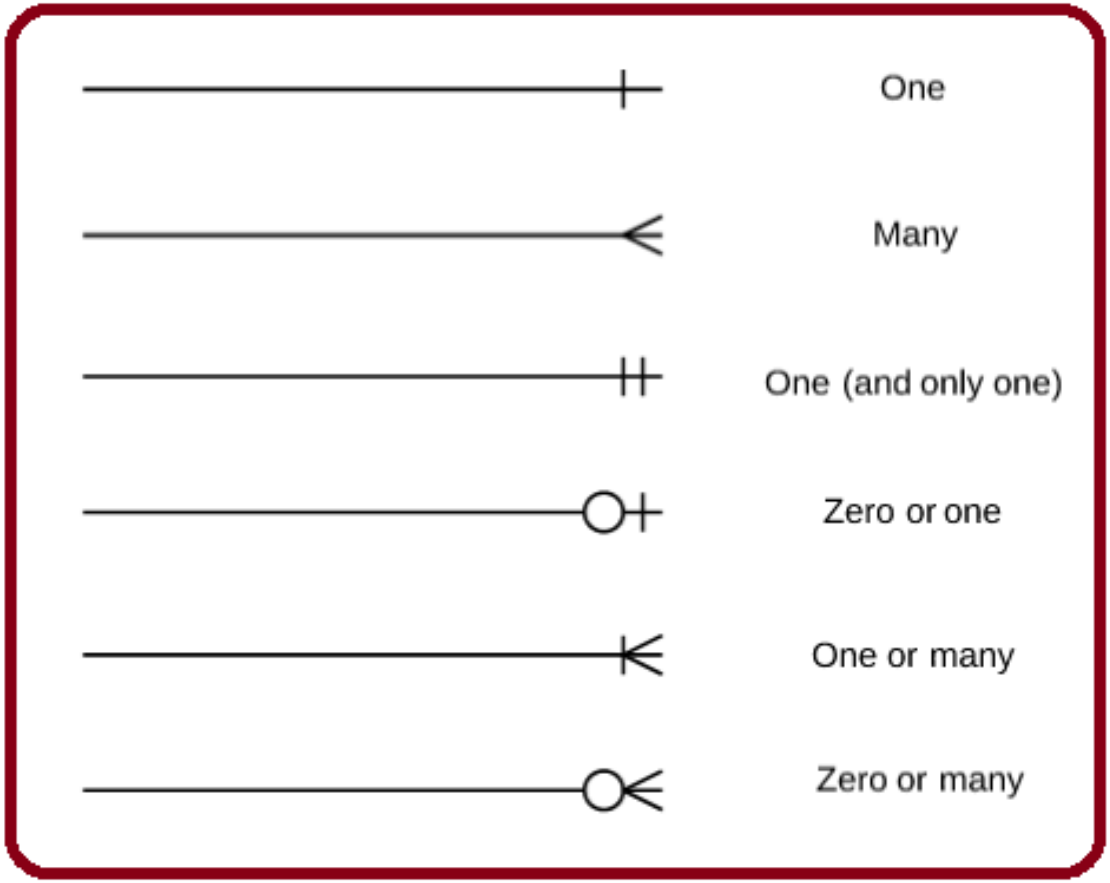
Many to many.

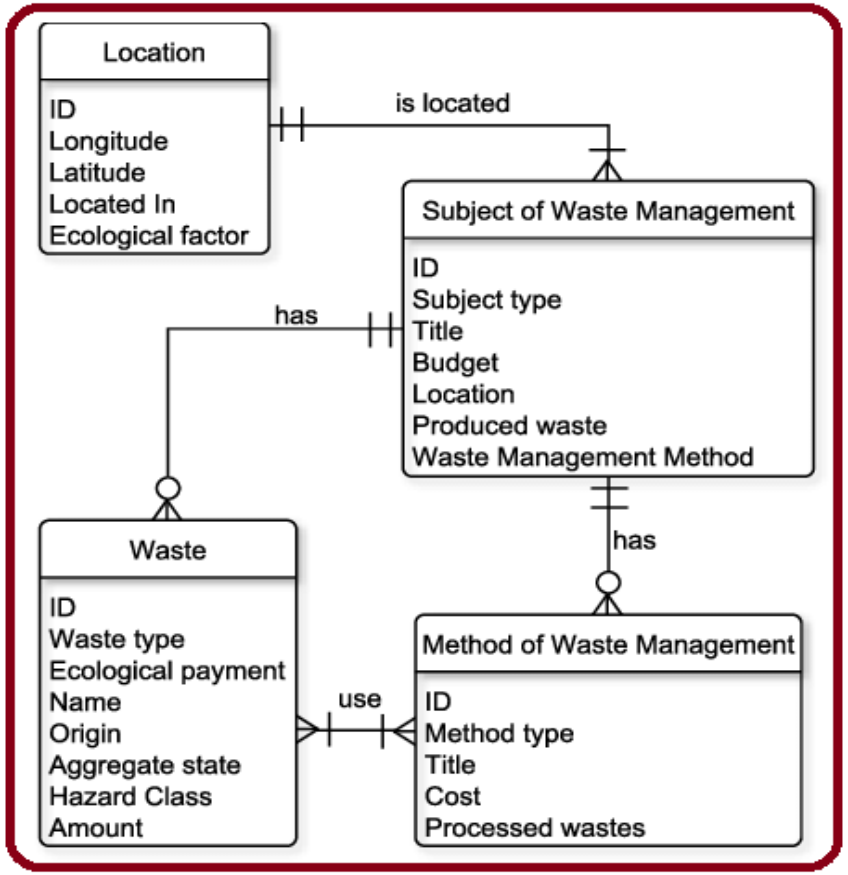


This bank issues parallel personal and group loans only. One loan can have multiple customers as loan holders and against one customer he/she can have multiple loan numbers.

Entity-Relationship Diagram Using Crow Foot Notation

One sample Entity-Relationship Diagram using Crow Foot Notation





Other Terminologies of DBMS

- **ACID:** The acronym standing for the properties maintained by standard database management systems, standing for **Atomicity**, **Consistency**, **Isolation**, and **Durability**.
- **Atomicity:** The property of a transaction that guarantees that either all or none of the changes made by the transaction are written to the database.
- **BLOB:** An abbreviation for **Binary Large OBject**. In SQL, BLOB can be a general term for any data of type long varbinary, long varchar, or long wvarchar. It is also a specific term (and synonym) for data of type long varbinary.
- **Commit:** The action that causes all the changes made by a particular transaction to be reliably written to the database files and made visible to other users.
- **Concurrency:** The property in which two or more computing processes are executing at the same time.
- **Cursor:** A collection of rows grouped by common criteria (key sequence, set membership, SELECT result set) that can be navigated and updated.

Terminologies of DBMS

Other Terminologies of DBMS

- **DDL: Database Definition Language**. The SQL DDL provides commands for defining relation schemas, deleting relations, and modifying relation schemas
- **Deadlock:** A situation in which resources (i.e., locks) are held by two or more connections that are each needed by the other connections so that they are stuck in an infinite wait loop. For example, connection 1 has a lock on table1 and is requesting a lock on table2 that is currently held by connection 2, which is also requesting a lock on table1.
- **DML: Database Manipulation Language**. In SQL, such statements as UPDATE, INSERT and DELETE are considered DML.
- **NoSQL:** A classification of data storage systems that are not primarily designed to be relationally accessed through the common SQL language. NoSQL systems are characterized by dynamic creation and deletion of key/value pairs and are structured to be highly scalable to multiple computers.
- **SQL:** The standardized and commonly accepted language used for defining, querying and manipulating a relational database.

Other Terminologies of DBMS

- **Thread:** A single, sequential execution of a computer program or program segment. A program can consist of one or more concurrently executing threads. Where multiple threads access the same data, synchronization method needs to be employed to ensure that the data is accessed only by one thread at a time.
- **Transaction:** A set of logically related database modifications that is written to the database as a unit. The database changes associated with a given transaction are guaranteed by the DBMS to be written completely to the database; in the event of a system failure, none are written. The state of the database both before and after a transaction will be consistent with its design.

What is a NoSQL Database?

In general terms, Databases, which store data in a format different from relational databases, are known as NoSQL databases. NoSQL stands for **"Not Only SQL"**, which pertains to the fact that either the database can store and manage data using "no SQL" at all or it can work in a combination that combines the flexibility of the newer approach (NoSQL) with the power of the traditional relational system (SQL).

Unlike relational databases, related data doesn't have to be split up between multiple tables; instead, it can be nested within the same data structure.

Features of NoSQL

- **Multi-Model:** Unlike relational databases, where data is stored in relations, different data models in NoSQL databases make them flexible to manage data. Each data model is designed for specific requirements.
- **Distributed:** NoSQL databases use the shared-nothing architecture, implying that the database has no single control unit or storage. The advantage of using a distributed database is that data is continuously available because data remains distributed between multiple copies.
- **Flexible Schema:** Unlike RDBMS, NoSQL databases are quite flexible while managing data. NoSQL databases can process structured, semi-structured or unstructured data with the same ease, thereby increasing performance.
- **Eliminated Downtime:** One of the essential features is the eliminated downtime. Since the data is maintained at various nodes owing to its architecture, so there is failure transparency.
- **High Scalability:** One of the reasons for preferring NoSQL databases over relational databases is their high scalability. Since the data is clustered onto a single node in a relational database, scaling up poses a considerable problem. On the other hand, NoSQL databases use horizontal scaling, and thus the data remains accessible even when one or more nodes go down.

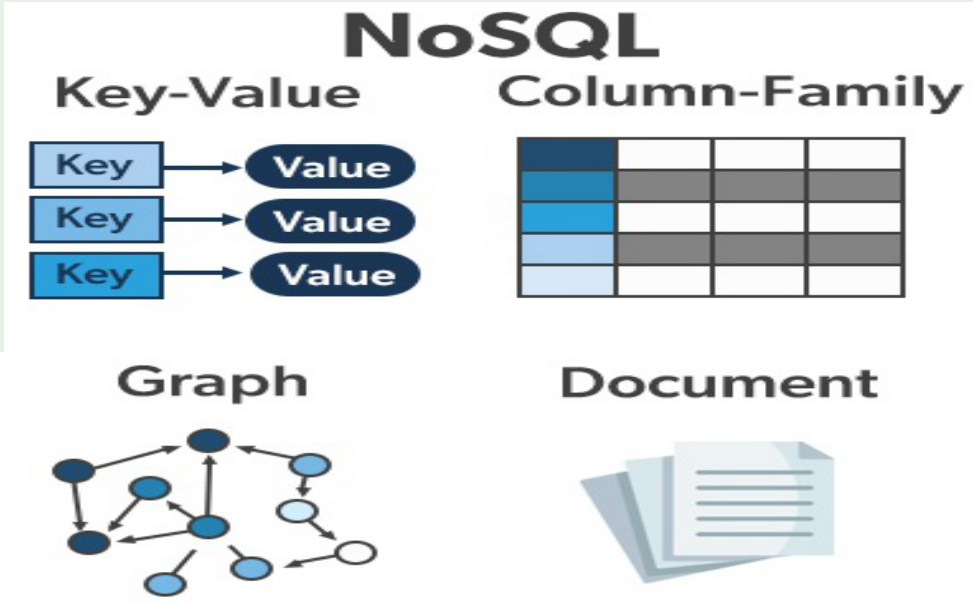
Types of NoSQL Databases

A database is a collection of structured data or information which is stored electronically in a computer system and can be accessed easily. A database is usually managed by a Database Management System (DBMS).

NoSQL is a non-relational database that is used to store the data in the nontabular form. NoSQL stands for Not only SQL. The main types are documents, key-value, wide-column, and graphs.

Types of NoSQL Database:

- **Document-based databases**
- **Key-value stores**
- **Column-oriented databases**
- **Graph-based databases**



Document-Based Database

The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents.

In the Document database, the particular elements can be accessed by using the index value that is assigned for faster querying. Collections are the group of documents that store documents that have similar contents. Not all the documents are in any collection as they require a similar schema because document databases have a flexible schema.

Key features of documents database:

- **Flexible schema:** Documents in the database has a flexible schema. It means the documents in the database need not be the same schema.
- **Faster creation and maintenance:** the creation of documents is easy and minimal maintenance is required once we create the document.
- **No foreign keys:** There is no dynamic relationship between two documents.
- **Open formats:** To build a document we use XML, JSON, and others.

Some popular Document-Based databases are **MongoDB**, **CouchDB, Lotus Notes**.

Sample JSON Document

```
{ "products" : [
  { "Name": "Cheese", "Price" : 2.50, "Location": "Refrigerated foods"},
  { "Name": "Crisps", "Price" : 3, "Location": "the Snack isle"},
  { "Name": "Pizza", "Price" : 4, "Location": "Refrigerated foods"},
  { "Name": "Chocolate", "Price" : 1.50, "Location": "the Snack isle"},
  { "Name": "Self-raising flour", "Price" : 1.50, "Location": "Home baking"},
  { "Name": "Ground almonds", "Price" : 3, "Location": "Home baking"}
]}
```

Sample XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
  </food>
</breakfast_menu>
```

Key-Value Pairs Database

A key-value store is a nonrelational database. The simplest form of a NoSQL database is a key-value store. Every data element in the database is stored in key-value pairs. The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings and numbers or complex objects.

A key-value store is like a relational database with only two columns which is the key and the value.

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Key features of the key-value store:

- **Simplicity.**
- **Scalability.**
- **Speed.**

Some popular Key-Value Pair databases are **Dynamo DB**, **Redis, BerkleyDB**.

Column Oriented Databases

A column-oriented database is a non-relational database that stores the data in columns instead of rows. That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.

Columnar databases are designed to read data more efficiently and retrieve the data with greater speed. A columnar database is used to store a large amount of data.

Column Oriented Database

date	price	size
2011-01-20	10.1	10
2011-01-21	10.3	20
2011-01-22	10.5	40
2011-01-23	10.4	5
2011-01-24	11.2	55
2011-01-25	11.4	66
...	...	...
2013-03-31	17.3	100

Key features of column-oriented database:

- **Scalability**
- **Compression**
- **Very responsive**

Some popular Column Oriented databases are **Google's Bigtable**, **Cassandra**, **HBase**.

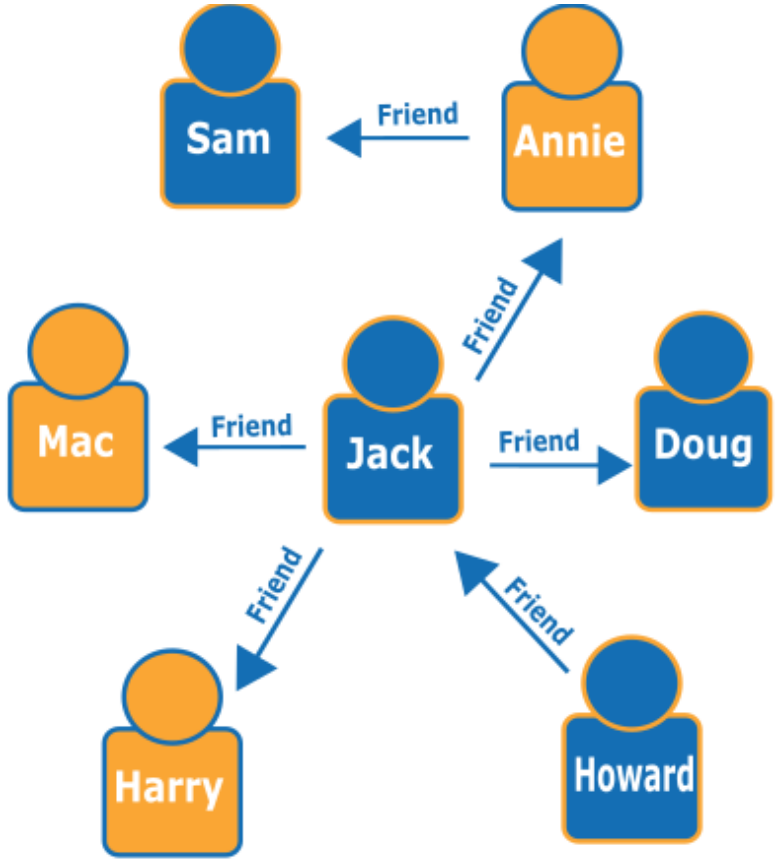
Graph-Based Databases

Graph-Based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.

Key features of graph database:

- **In a graph-based database, it is easy to identify the relationship between the data by using the links.**
- **The Query's output is real-time results.**
- **The speed depends upon the number of relationships among the database elements.**

Some popular Graph-Based databases are **Neo4j**, **OrientDB**, **Allegrograph**.



CAP Theorem

Definition

Any **shared-data** distributed system, can fulfil **at most two** of the following properties:

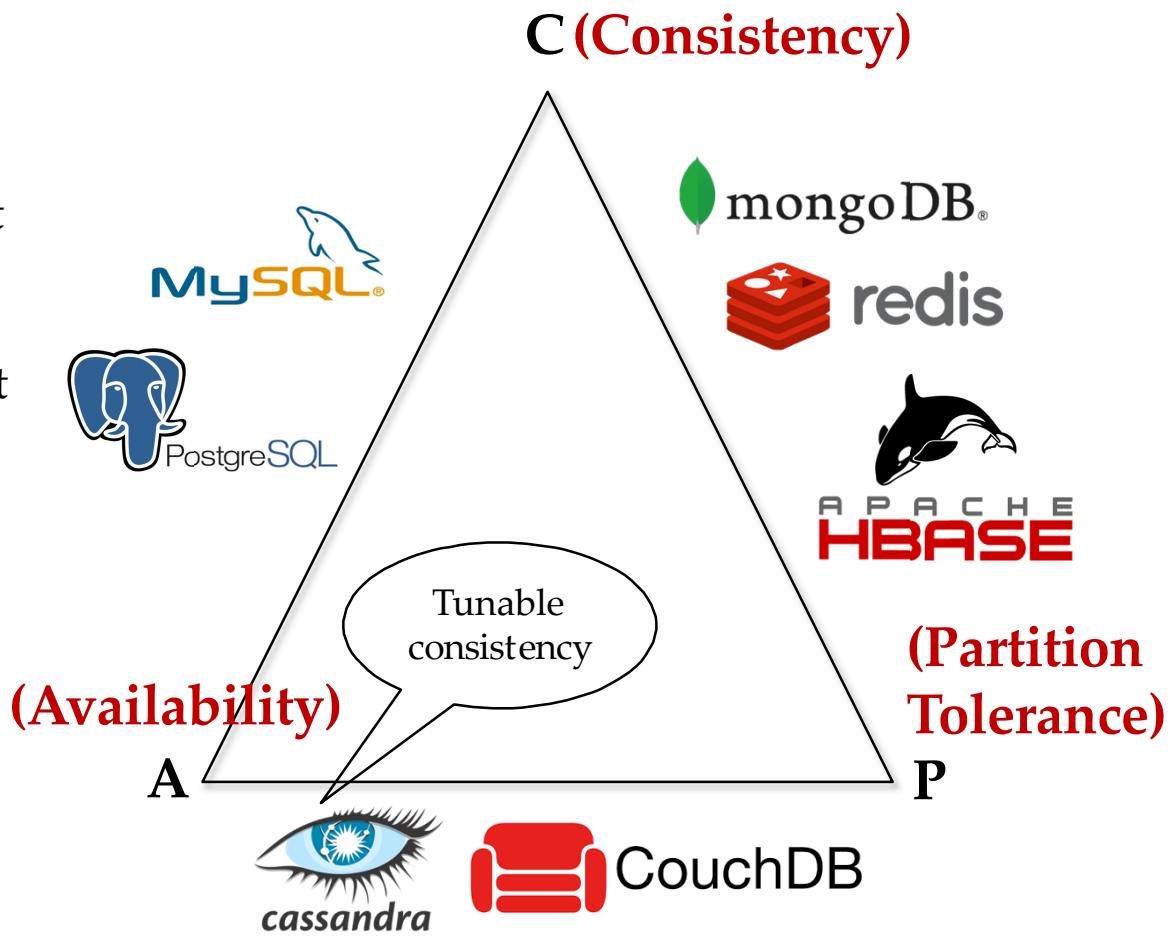
- **Consistency** – every read operation receives result of the most recent write
- **Availability** – every read operation receives a non-error response, but returned data may be stale
- **Partition Tolerance** – system continues to operate without errors despite network disruption between nodes (e.g. "split-brain")

Simplified:

In presence of network partitioning, system's designer must choose between data consistency and availability of his solution.

CAP Theorem

- **CA systems** are consistent and available, but do not support network partitioning
- **AP systems** always support read and write operations, but stored data may not be consistent at all time
- **CP systems** are always consistent, but not available during network partitioning
- Over-simplified database classification, correct only when using their default configuration settings
- Distributed systems battle-testing blog: https://jepsen.io/analyses



Thanks for listening !