Chapter 2-Slide

Data Models

Compiled by Yaynshet. M

Data Model

- The Database model in the DBMS explains
  - The logic behind the structure of a Database system that should usually include
  - All the tables, which are represented as entities in ER model,
  - The relationships between the tables and objects, and
  - The requirement provided by the project team in order to settle on how data can be stored & accessed, granted
- Data Model: higher-level language with sets of concepts to describe the structure of a database, and certain constraints that the database should obey
- A data model is a description of the way that data is stored in a database

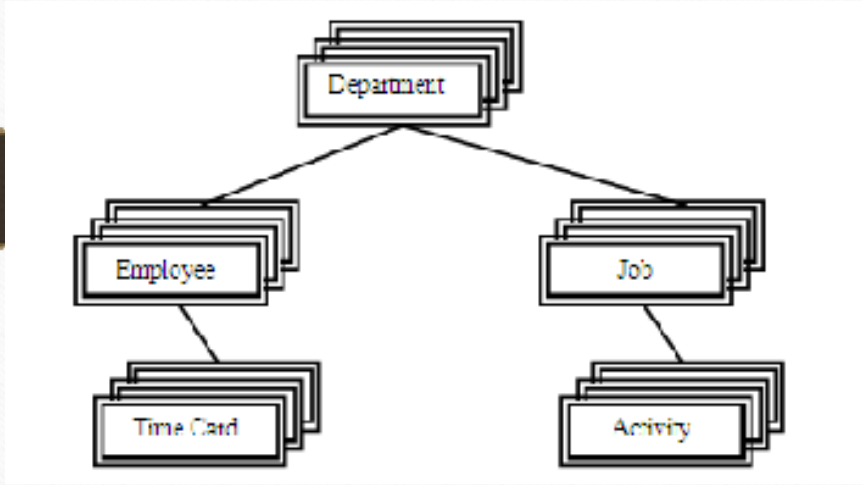
Data model

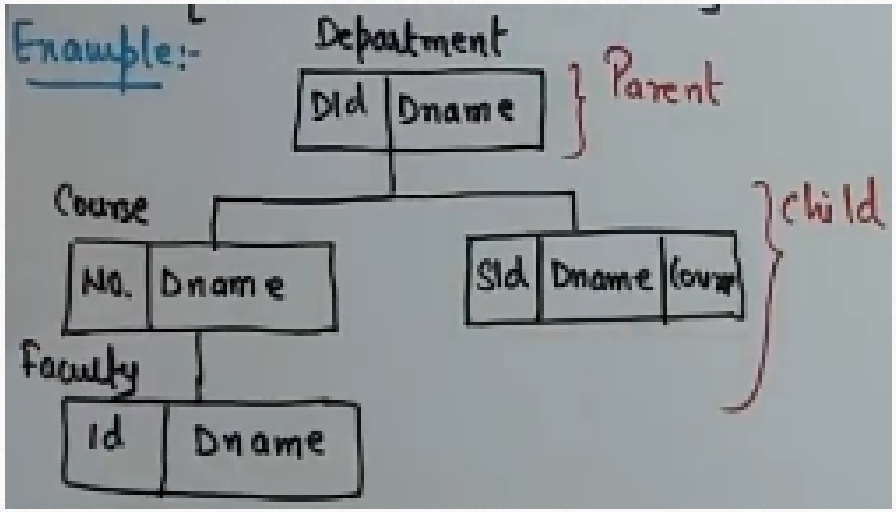
- *Generally it is* a collection of tools or concepts for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
  - Concepts to represent the data in an understandable way
- **Some of the different Data Models**
  - Hierarchical Data Model
  - Network Data Model
  - Relational Data Model

Hierarchical Model

- The hierarchical model is based on the parent-child hierarchical relationship.
- In this model, there is one parent entity with several children entity.
  - A parent node can have more than one child node
  - A child node can only have one parent node
  - The relationship between parent and child is one-to-many
- At the top, there should be only one entity which is called root.
- Nodes are arranged in a hierarchical structure as sort of upside-down tree
- To add new record type or relationship, the database must be redefined and then stored in a new form
- For example, an organization is the parent entity called root and it has several children entities like clerk, officer, and many more.

Hierarchical data model





ADVANTAGES
Hierarchical Model

- **Clear Chain of Command**
  - Members know to whom they report and who reports to them.
  - Communication gets channeled along defined and predictable paths,
    - which allows those higher in the organization to direct questions to the appropriate parties.
  - Individuals tend to know who does and does not possess the authority to assign or change tasks
- **Clear Paths of Advancement**
  - Hierarchical structures offer very clear, if not always easy, advancement paths.
- **Specialization**
  - The hierarchical structure divides larger tasks and areas of concern into various department configurations that specialize.
  - Specialization allows organizations to concentrate particular skill sets and resources to achieve maximum efficiency.

DISADVANTAGES Hierarchical Model

Poor Flexibility

- Hierarchical structures tend to adapt slowly to changing needs.

Communication Barriers

- As hierarchical organizational structures tend to channel communication vertically, interdepartmental or inter-agency communication suffers.
- Departmental specialization can lead to communication barriers when no shared jargon exists that allows members of different departments to communicate on the same level.

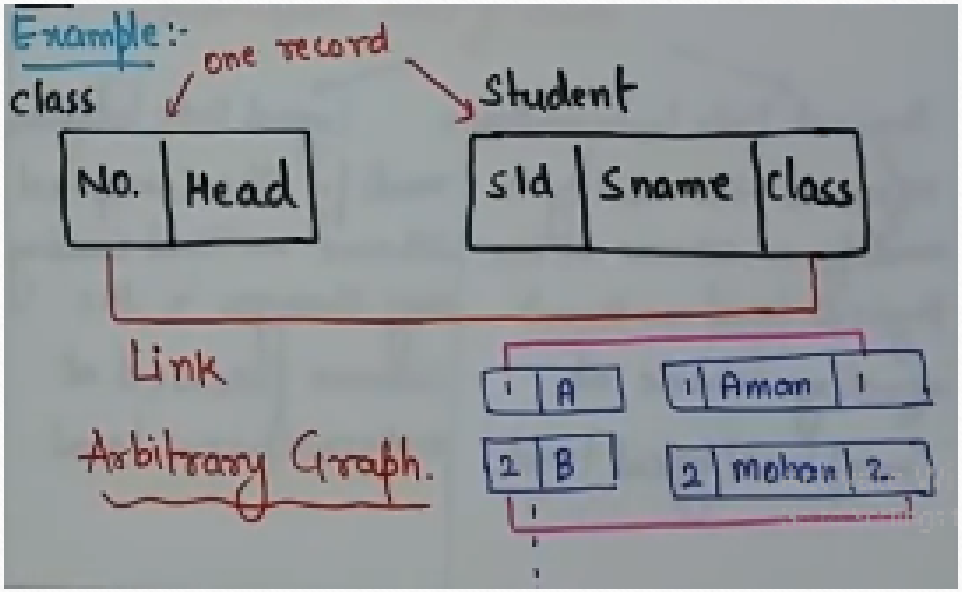
Organizational Disunity

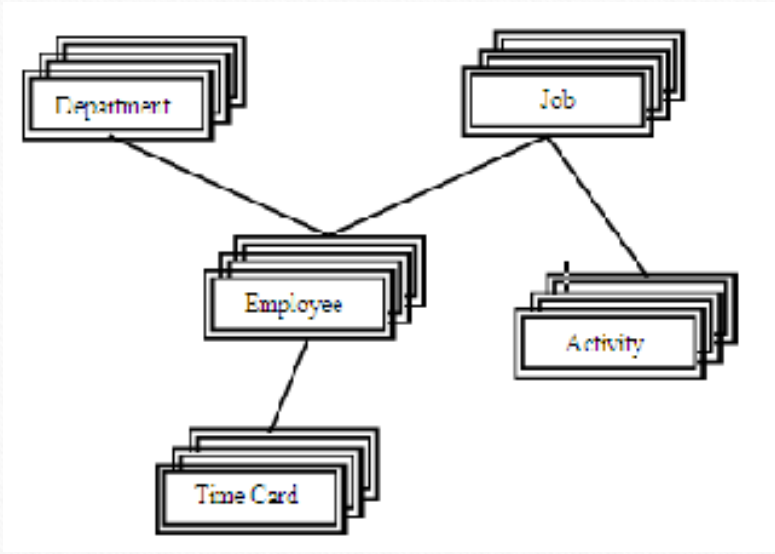
- The departmentalizing of specializations leads, in some cases, to decisions made to benefit a department rather than the organization goals.

Network Model

- In the network data model, all the entities are organized in graphical representations.
- There may be several parts in the graph in which the entities can be accessed.
- Data is represented by collection of records
  - Relationships among data is represented by links
- Record is a collection of attributes, each which contain only one data value.
- The link is an association between two records
- Allows record types to have more that one parent unlike hierarchical model
- A network data models sees records as set members
- Each set has an owner and one or more members
- Like hierarchical model network model is a collection of physically linked records.
- Allow member records to have more than one owner

Network Model





ADVANTAGES of Network Data Model:

- Some advantages include
  - conceptual simplicity
  - data access flexibility
  - conformance to standards
  - handle more relationship types
  - promote database integrity
  - allows for data independence.

DISADVANTAGES of Network Data Model:

- The structure is difficult to change
- This type of system is very complex
- There is a lack of structural independence.
- In summary the network database model is similar but different than the hierarchal database model.
- But the network database model should be used when it is necessary to have a flexible way of representing objects and their relationships.

Relational Data Model

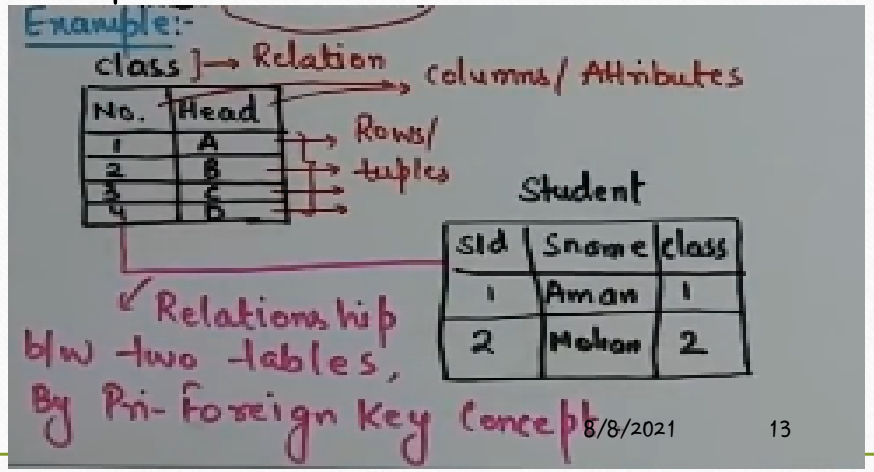
- The most popular and extensively used data model is the relational data model.
- The data model allows the data to be stored in tables called a relation.
- The relations are normalized and the normalized relation values are known as atomic values.
- Each of the rows in a relation is called tuples which contains the unique value.
  - A **row** of the table is called tuple => equivalent to **record**
- The attributes are the values in each of the columns which are of the same domain.
  - A **column** of a table is called attribute => equivalent to **fields**
- Can define more flexible and complex relationship

Relational model

- The tables seem to be independent but are related some how
- No physical consideration of the storage is required by the user
- Many tables are merged together to come up with a new virtual view of the relationship

Alternative terminologies

- Relation=Table = File
- Tuple = Row = Record
- Attribute = Column = Field



Relational model

- The rows represent records (collections of information about separate items)
- The columns represent fields (particular attributes of a record)
- Conducts searches by using data in specified columns of one table to find additional data in another table
- In conducting searches, a relational database matches information from a field in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables

Properties of Relational model

- Each row of a table is uniquely identified by a **PRIMARY KEY** composed of one or more columns
- Each tuple in a relation must be unique
- Group of columns, that uniquely identifies a row in a table is called a **CANDIDATE KEY**
- **ENTITY INTEGRITY RULE** of the model states that no component of the primary key may contain a **NULL** value.
- A column or combination of columns that matches the primary key of another table is called a **FOREIGN KEY**. Used to cross-reference tables.

Properties of Relational model

- The **REFERENTIAL INTEGRITY RULE** of the model states that, for every foreign key value in a table there must be a corresponding primary key value in another table in the database or it should be **NULL.**
- All tables are **LOGICAL ENTITIES**
- A table is either a BASE TABLES (Named Relations) or VIEWS (Unnamed Relations)
- Only Base Tables are physically stored
- VIEWS are derived from BASE TABLES with SQL instructions like: [SELECT .. FROM .. WHERE .. ORDER BY]

Properties of Relational model

- Order of rows and columns is immaterial
- Entries with repeating groups are said to be un-normalized
- Entries are single-valued
- Each column (field or attribute) has a distinct name

NB: All values in a column represent the same attribute and have the same data format

Properties of Relational model

Building Blocks of the Relational Data Model

The building blocks of the relational data model are:

- **Entities**: real world physical or logical object
- **Attributes**: properties used to describe each Entity or real world object.
- **Relationship**: the association between Entities
- **Constraints**: rules that should be obeyed while manipulating the data

Building blocks of Relational Model

1. **ENTITIES**

- persons, places, things etc. which the organization has to deal with.
- The name given to an entity should always be a singular noun descriptive of each item to be stored in it.

  E.g.: student NOT students

Building blocks of Relational Model

- Every relation has a schema, which describes the columns, or fields and the relation itself
- **Existence Dependency**: the dependence of an entity on the existence of one or more entities.
- **Weak entity**: an entity that cannot exist without the entity with which it has a relationship

Building blocks of Relational Model

2. **ATTRIBUTES –**

- the items of information which characterize and describe these entities.
- Attributes are pieces of information ABOUT entities

Building blocks of Relational Model

- At this level we need to know such things as:

- Attribute name (be explanatory words or phrases)
- **The domain** from which attribute values are taken
  - (A DOMAIN is a set of values from which attribute values may be taken.)
    - Each attribute has values taken from a *domain*
      - For example, the domain of **Name** is **string** and that for **salary** is **real**
- Whether the attribute is part of the **entity identifier** (attributes which just describe an entity and those which help to identify it uniquely)
- Whether it is **permanent or time-varying** (which attributes may change their values over time)
- Whether it is **required or optional** for the entity (whose values will sometimes be unknown or irrelevant)

Types of Attributes

(1)Simple (atomic) Vs Composite attributes

- **Simple:** contains a single value (not divided into sub parts)

  E.g. Age, gender

- **Composite**: Divided into sub parts (composed of other attributes)

  E.g. Name, address

(2)Single-valued Vs multi-valued attributes

Single-valued : have only single value(the value may change but has only one value at one time)

E.g. Name, Sex, Id. No., color_of_eyes

Multi-Valued: have more than one value

E.g. Address, dependent-name Person may have several college degrees, several languages

Types of Attributes

(3)Stored vs. Derived Attribute

- Stored : not possible to derive or compute

  E.g. Name, Address

- Derived: The value may be derived (computed) from the values of other attributes.

E.g. :

- Age (current year – year of birth
- Length of employment (current date- start date)
- Profit (earning-cost)
- G.P.A (grade point/credit hours)

(4) Null Values

- NULL applies to attributes which are not applicable or which do not have values.
  - You may enter the value NA (meaning not applicable)

NB: Value of a key attribute cannot be null.

Default value – is the value assumed value if no explicit value

Entity vs. Attributes

- When designing the conceptual specification of the database,
  - one should pay attention to the distinction between an Entity and an Attribute.
- Consider designing a database of employees for an organization

  Argument: Should *address* be an attribute of Employees or an entity ?
  - If we have several addresses per employee,
    - *address* must be an entity (*attributes cannot be set-valued/multi valued*)
  - If the structure (city, Woreda, Kebele, etc.) is important,

    e.g.: want to retrieve employees in a given city,
    - **address** must be modeled as an **entity** (*attribute values are atomic*

Building blocks of Relational Model

3. RELATIONSHIP

- In any business processing one object may be associated with another object due to some event.
  - Such kind of association is what we call a RELATIONSHIP between entity objects.
- Related entities require setting of LINKS from one part of the database to another.
- A relationship should be named by a word or phrase which explains its function

Degree of relationship

- An important point about a relationship is how many entities participate in it.
- The number of entities participating in a relationship is called the **DEGREE** of the relationship.
  - **UNARY/RECURSIVE RELATIONSHIP:** *Tuples/records of a Single entity are related with each other.*
  - **BINARY RELATIONSHIPS:** *Tuples/records of two entities are associated in a relationship*
  - **TERNARY RELATIONSHIP:** *Tuples/records of three different entities are associated*
  - And a generalized one:
    - **N-NARY RELATIONSHIP:** *Tuples from arbitrary number of entity sets are participating in a relationship*

Cardinality of a Relationship

- The number of instances participating or associated with a single instance from an entity in a relationship is called the **CARDINALITY** of the relationship.
- **ONE-TO-ONE:** one tuple is associated with only one other tuple.
  - E.g. Building – Location: – as a single building will be **located** in a single location and as a single location will only **accommodate** a single Building.

Cardinality of a Relationship

- ONE-TO-MANY: one tuple can be associated with many other tuples, but not the reverse.
  - E.g. Department-Student: - as one department can have multiple students.
- MANY-TO-ONE: many tuples are associated with one tuple but not the reverse.
  - E.g. Employee – Department: as many employees belong to a single department
- MANY-TO-MANY: one tuple is associated with many other tuples and from the other side, with a different role name one tuple will be associated with many tuples
  - E.g. Student – Course: - as a student can take many courses and a single course can be attended by many students.

SQL Components

- The Structured Query Language (SQL) is the set of instructions used to interact with a relational database.
- In fact, SQL is the only language that most Relational databases understand.
- Whenever you interact with such a database, the software translates your commands (whether they are mouse clicks or form entries) into a SQL statement that the database knows how to interpret.

SQL Components

- As a user of any database-driven software program, you're probably using SQL, even if you don't know it.
- For example, a database-driven dynamic web page (like most websites) takes user input from forms and clicks and uses it to compose a SQL query that retrieves information from the database required to generate the next web page.

SQL Components

- Consider the example of a simple online catalog with a search function.
- The search page might consist of a form containing just a text box in which you enter a search term and then click a search button.
- When you click the button, the web server retrieves any records from the product database containing the search term and uses the results to create a web page specific to your request.

SQL Components

- For example, if you searched for products containing the term "Irish,"
  - the server might use the following SQL statement to retrieve related products:

```
SELECT *
FROM products
WHERE name LIKE '%irish%';
```

- Translated, this command retrieves any records from the database table named "products" that contain the characters "irish" anywhere within the product name.

SQL components

- SQL has three main components:
  - the Data Manipulation Language (DML),
  - the Data Definition Language (DDL), and
  - the Data Control Language (DCL).

The Data Manipulation Language

- The Data Manipulation Language (DML) contains the subset of SQL commands used most frequently
- They simply manipulate the contents of a database in some form.
- The four most common DML commands are
  - The SELECT command which is used to retrieve information from a database
  - The INSERT command used to add new information to a database
  - The UPDATE command which is used to modify information currently stored in a database
  - The DELETE command which is used to remove information from a database

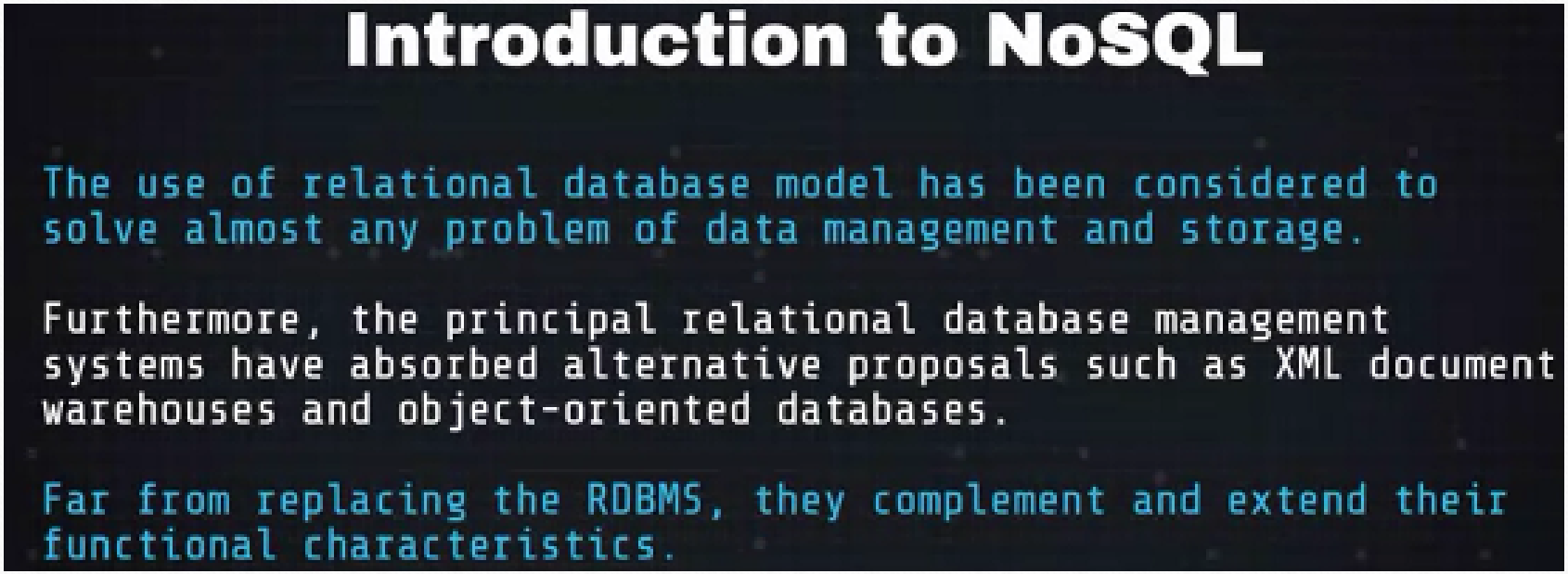
Data Definition Language

- The Data Definition Language (DDL) contains commands that are less frequently used.
- DDL commands modify the actual structure of a database, rather than the database's contents.
- Examples of commonly used DDL commands include those used to
  - generate a new database table (CREATE TABLE),
  - modify the structure of a database table (ALTER TABLE), and
  - delete a database table (DROP TABLE).

Data Control Language

- The Data Control Language (DCL) is used to manage user access to databases.
- It consists of two commands:
  - the GRANT command, used to give database permissions for a user, and
  - the REVOKE command, used to remove existing permissions.
- These two commands form the core of the relational database security model.

NOSQL Systems



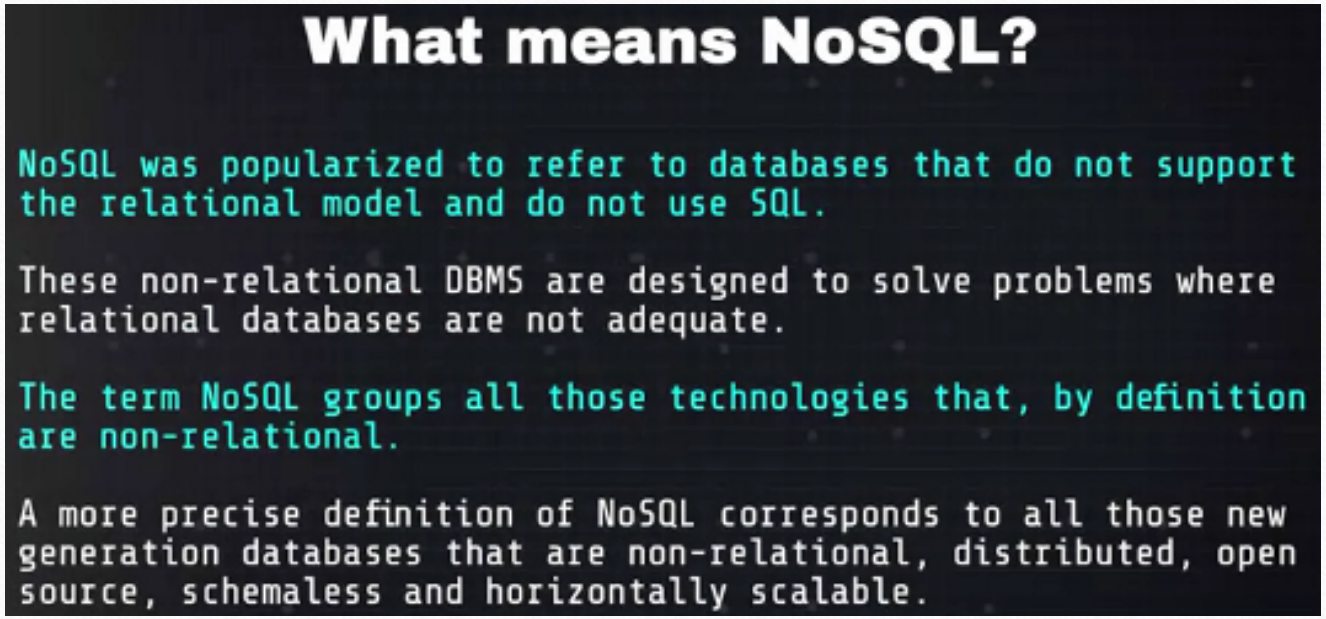
How NoSQL Differ from SQL

Data models: A NoSQL database lets you build an application without having to define the schema first unlike relational databases which make you define your schema before you can add any data to the system.

Regarding data structure: NoSQL databases are designed to handle unstructured data (e.g., texts, social media posts, video, email) which makes up much of the data that exists today.

Development model: NoSQL databases are open source whereas relational databases typically are closed source with licensing fees baked into the use of their software. With NoSQL, you can get started on a project without any heavy investments in software fees upfront.

NoSQL?


What means NoSQL?
NoSQL was popularized to refer to databases that do not support the relational model and do not use SQL.
These non-relational DBMS are designed to solve problems where relational databases are not adequate.
The term NoSQL groups all those technologies that, by definition are non-relational.
A more precise definition of NoSQL corresponds to all those new generation databases that are non-relational, distributed, open source, schemaless and horizontally scalable.

NoSQL

Cassandra, Ubuntu One, CouchDB

Emerging web companies were in·uenced by the BigTable columnar DBMS and by Dynamo's key-value DMBS; who developed their own non-relational database systems.

The NoSQL movement also caught the attention of companies like Facebook, who developed the Cassandra system, later used by Twitter. In addition, LinkedIn developed Project Voldemort and Ubuntu One, a synchronized cloud storage system based on CouchDB, emerged.

NoSQL

Cassandra, Ubuntu One, CouchDB

All these companies have in common that they offer web services for a large number of users and daily execute processes with many read/ write operations.

In addition to web applications, NoSQL databases also support diverse activities, including predictive analysis and non-critical transactional systems.

Advantages

High performance. The read and write speeds in a NoSQL database are usually much higher than in a traditional RDBMS.

For example, Hypertable's technology is capable of storing one trillion data per day. Another example is the Google BigTable software, which can process 20 petabytes of data in a day.

Horizontal scaling: If the database experiences some growth, it is possible to add nodes to a distributed system in such a way that it provides processing and storage economically.

NoSQL databases have the necessary mechanisms for horizontal scalability.

Advantages

A simple storage mechanism: The storage mode is much simpler than the table schema of the relational model. This positively affects performance as there is no defined scheme for saving data.

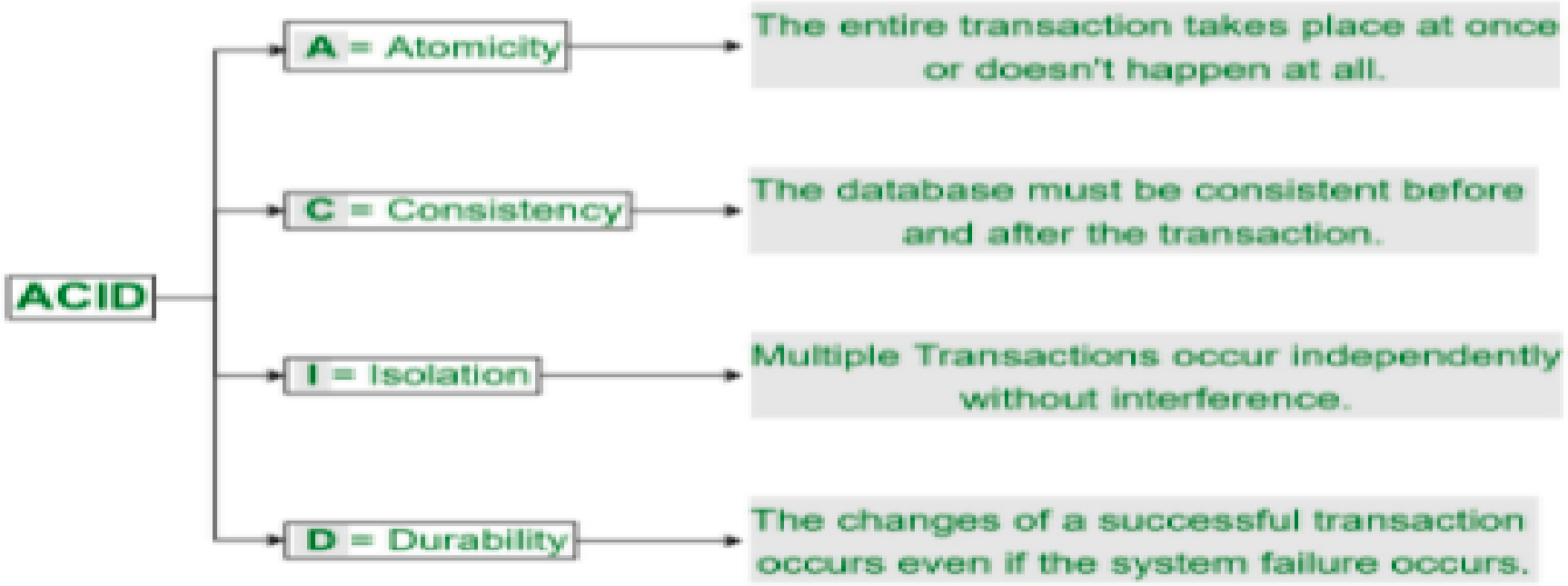
Flexible storage The storage schemas of the non-relational databases are much more flexible than the rigid schema of tables. In reality they lack a fixed scheme. When the data to be stored can not be translated into tables of the relational model, it is convenient to look for a solution by NoSQL.

No ACID. Without a strict management of transactions, the control of concurrency is not done by blocking but there are other mechanisms.

Since durability is not a critical element in some web applications, it is not necessary to write logs for each transaction.
Examples of data not sensitive to durability include copies of user session data and text messages in forums or social networks.

- A **transaction** is a single logical unit of work which accesses and possibly modifies the contents of a database.
- Transactions access data using read and write operations.
- In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties. ACID is very Fundamental

ACID Properties in DBMS



Disadvantages of NoSQL

Non-standardized query language.
In the absence of a standard query language for NoSQL databases, it is necessary to learn each query language for a given DBMS. This is true even within each NoSQL storage category.

Problems in transactions.
As a result of not following ACID, there is weaker control over the consistency, durability and isolation of transactions.

Integrity problems:
Ensuring the integrity of the data requires extra programming manually and is not a function of the DBMS. Integrity is understood from the point of view of the restrictions: of domain, referential of null value, etc.

RDBMS or NoSQL?

The decision on what type of DBMS should be used depends on a set of factors including but not limited to:

- The volume of data to store.
- The estimated concurrence.
- The number of operations that are done on the database per unit of time.
- The desired scalability of the database.
- The degree of integrity and consistency that is desired.
- The nature of the data to be stored.
- The most frequent types of operations that you want to do with the data.

Characteristics of NoSQL Databases

C- Consistency: It refers to whether a system is in a consistent state after the execution of an operation or not. A distributed system is considered consistent if after an update operation by a node, the rest of the nodes see the update in a shared data resource.

A - Availability: it means that a system is designed and implemented in such a way that it can continue its operation if there are software or hardware problems or a node fails.

P - Partition Tolerance: It is the ability of a system to continue its operation in the presence of network partitions. This happens if a set of nodes in a network loses connectivity with other nodes.

Properties of NoSQL Databases

A NoSQL database follows the paradigm of BASE Basically Available systems, instead of the ACID approach.

- Basically available.
- Soft state.
  (the information will expire unless it is refreshed).
- Eventual Consistency.

The BASE properties can be summarized as follows:
an application works basically all the time, it does not have to be consistent all the time but it will eventually reach a known state.

Types of Nonrelational Databases

Types of non-relational DBMS. There are four main non-relational DBMS categories:

- Key-value storage
- Column-oriented databases
- Document-oriented databases
- Graphs-oriented data bases

Key-Value database

It is a system that stores values indexed by keys, can store structured and unstructured data. (The possibility of storing any type of value is called schema-less).

- Values are stored as arrays of bytes.
- The content is not important for the database.
- High performance, very scalable, very flexible and low complexity.

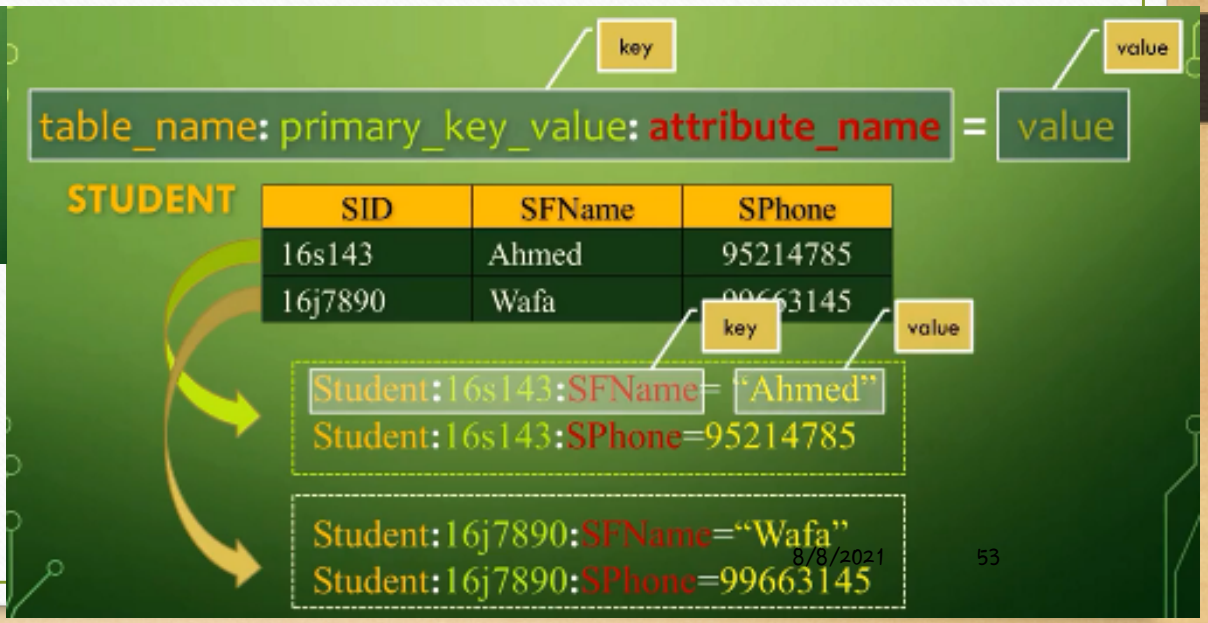
For instance: Amazon DynamoDB, Cassandra, Voldemort, RAMCloud and Flare.

- Redis, Riak and Oracle NoSQL database are another examples of key-value databases.

Key-value

The 3 operations performed on a key-value database are:

1. put(key, value)
2. get(key)
3. delete(key)



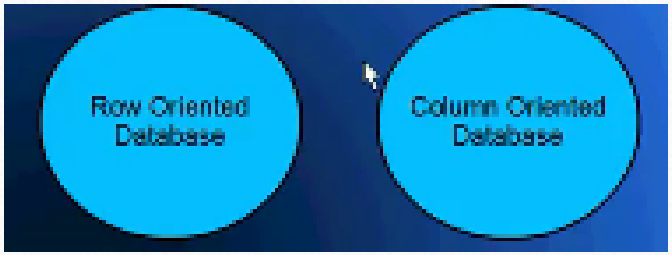
Column-oriented databases

- Best Example of Row-oriented data stores is Relational Database, which is a structured data storage and also a sophisticated query engine.
- It incurs a big penalty to improve performance as the data size increases.

Database Storage orientation methods

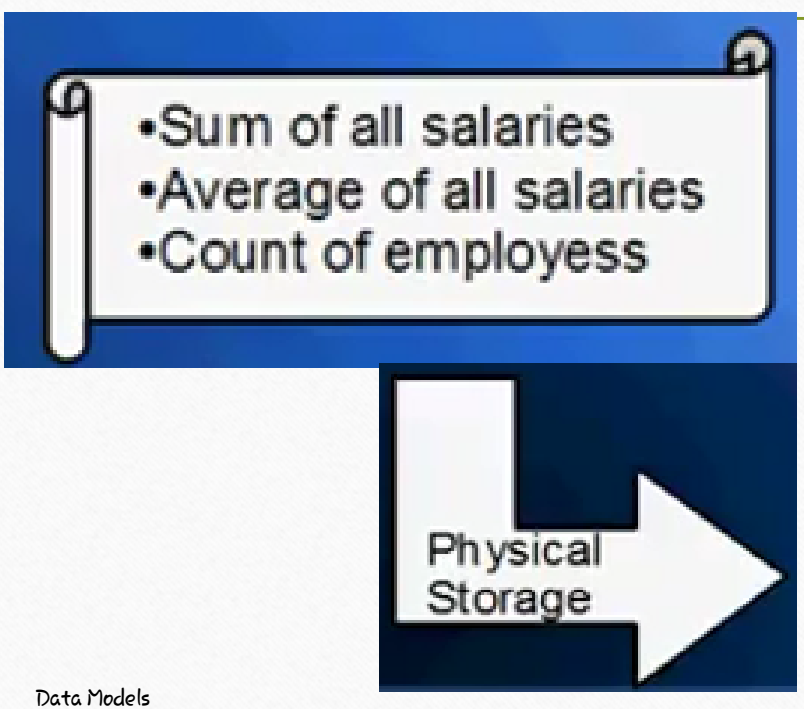
Table: Employee

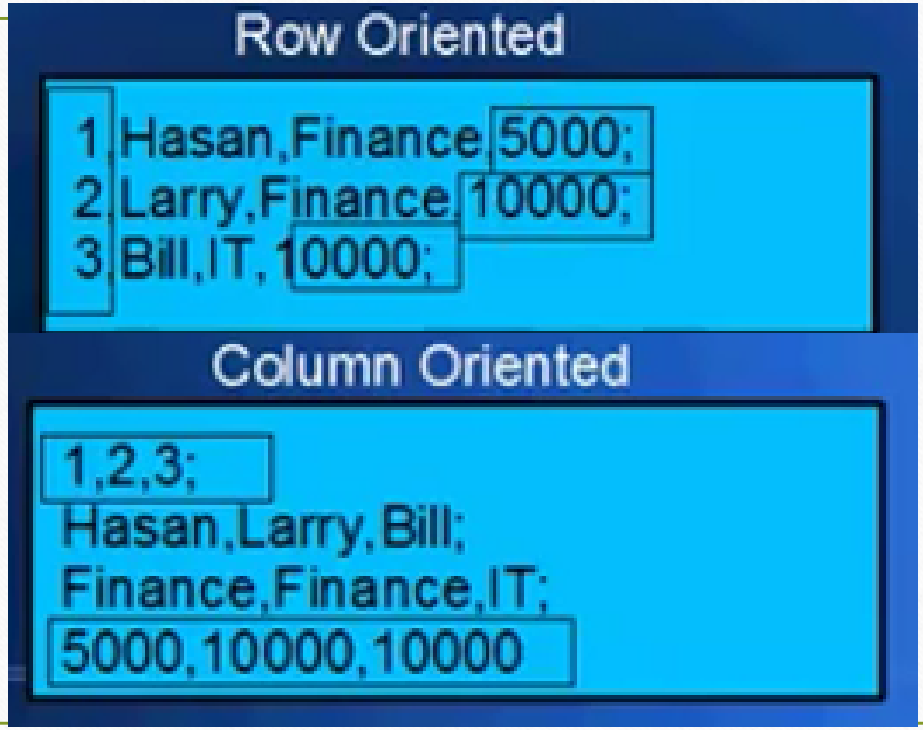
ID	NAME	DEPT	SALARY
1	Hasan	Finance	5,000
2	Larry	Finance	10,000
3	Bill	IT	10,000



- This table is logical view (how database is storing this information is not visible to us: how the table is stored in the backend could be very different than what you see on the screen)
- It stores 4 attributes of employee entity namely ID, Name, Dept. and Salary

Row vs column oriented





Column-oriented database

Column-oriented storage is especially efficient when data readings are massive, and writes to a few columns. This is because in a query only the data of the columns that interest are obtained, not all the columns of a record, which increases the efficiency.

The problem of the writings would be when you want to write in all the columns of the record, since the columns work as individual units and are not necessarily contiguous, as in the storage by rows.

Column-oriented databases are widely used in business intelligence.

- Examples of column oriented databases are :MariaDB, CrateDB, ClickHouse, Greenplum Database, Apache Hbase, Apache Kudu, Apache Parquet, Hypertable, MonetDB .

Document-oriented databases

This format is often a JSON document (JavaScript Object Notation), but it can be XML or any other.

Allows very advanced queries on the data.
Allows relations between data.
Do not allow join operations due to performance issues.

Examples of document-oriented DBMS include CouchDB, MongoDB, Cloudkit, and XML databases such as DB2 pureXML.

Graph-oriented databases

A graph is represented as a set of nodes (entities) interconnected by edges (relationships).

Graphs give importance not only to the data, but to the relationships between them.

Relationships can also have attributes and direct queries can be made to relationships, rather than to the nodes.

Being stored in this way, it is much more efficient to navigate between relationships than in a relational model.

- Examples of graph-oriented databases are Neo4j, infoGrid, Hyperbase-DB

Which on to choose ?

- Similarities and difference
- All in One in the table below

	Performance	Scalability	Flexibility	Complexity	Functionality
Key-Value Stores	high	high	high	none	variable (none)
Column stores	high	high	moderate	low	minimal
Document stores	high	variable (high)	high	low	variable (low)
Graph databases	variable	variable	high	high	graph theory
Relational databases	variable	variable	low	moderate	relational algebra

- MongoDB is the most popular NoSQL database.
- A free and open source, cross-platform, document-oriented database
- MongoDB uses JSON-like documents with schemas.

- END of Slide