



Lesson 304.1

INTRODUCTION TO RDBMS, DATA MODELING, and NORMALIZATION



Introduction to RDBMS, Data Modeling, and Normalization



Learning Objectives:

This lesson provides basic and advanced concepts of DBMS, RDBMS, Data modeling, ACID, and Normalization. This presentation is designed for both beginners and professionals.

By the end of this lesson, learners will be able to:

- Describe the DBMS and RDBMS concepts.
- Describe the different relationships in RDBMS.
- Define database standards.
- Describe the Data Modeling concept, including ERD.
- Describe the normalization and ACID database properties.
- Define SQL.

Table of Contents

- ❑ Database Management Systems.
- ❑ Relational Database Management Systems.
- ❑ American National Standards Institute - ANSI
- ❑ Structured Query Language SQL.
- ❑ ACID - Atomicity, Consistency, Isolation, and Durability Properties
- ❑ Data Modeling.
 - Relational Database Modeling.
 - Database Modeling
 - ▶ Conceptual, Logical, and Physical Data Model Diagrams
- ❑ Introduction to Relationships
- ❑ Normalization.
- ❑ Normal Forms.
 - First Normal Form.
 - Second Normal Form.
 - Third Normal Form.
- ❑ Advantages of Relational Database Management.

Database Management Systems

Overview:

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update, and manage data.

The DBMS manages three important things:

- ❑ The **data**.
- ❑ The **database engine** that allows data to be accessed, locked, and modified.
- ❑ The **database schema**, which defines the database's logical structure.

The above foundational elements help provide **concurrency, security, data integrity, and uniformity** to administration procedures.

The DBMS is most useful for providing a centralized view of data that can be accessed by multiple users from multiple locations in a controlled manner.

Relational Database Management Systems

A DBMS that is based on a *relational* model is called a Relational Database Management System (RDBMS).

- ❑ A relational model represents data in the form of a table. A table is a two-dimensional array containing rows and columns, and each row contains data related to an entity such as *a student*. Each column contains the data related to a single attribute of the entity such as *a student name*.
- ❑ One of the reasons behind the success of the relational model is its simplicity. It is easy to understand the data and easy to manipulate.
- ❑ RDBMS is the basis for SQL and for all modern database systems such as *MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access*.

RDBMS Terminologies

Table: A table is also considered a convenient representation of relations. But a table can have duplicate rows of data while a true relation cannot have duplicate data. A table is the simplest form of data storage.

- ❑ **Table Name:** Each table is given a name, which is used to refer to the table.
- ❑ **Row:** A single row in the table is called as **tuple**. Each row represents the data of a single entity.
- ❑ **Attribute/Column:** A column stores an attribute of the entity. For example, if the details of students are stored, the **student name** is an attribute, and the **course** is another attribute, and so on. Each column in the table is given a name. This name is used to refer to the value in the column.
- ❑ **Primary Key:** The primary key can be defined as a set of columns used to uniquely identify rows of a table.
- ❑ **Composite Key:** A primary key, having two or more attributes is called a composite key, which is a combination of two or more columns.
- ❑ **Relation Schema:** A relation schema describes the structure of the relation with the name of the relation (name of table) and its attributes names and types.

RDBMS Sample Table Illustration

SUPPLIER

Columns (Attributes, Fields)

Supplier_Number	Supplier_Name	Supplier_Street	Supplier_City	Supplier_State	Supplier_Zip
8259	CBM Inc.	74 5 th Avenue	Dayton	OH	45220
8261	B.R. Molds	1277 Gandolly Street	Cleveland	OH	49345
8263	Jackson Composites	8233 Micklin Street	Lexington	KY	56723
8444	Bryant Corporation	4315 Mill Drive	Rochester	NY	11344

Rows
(Records,
Tuples)

Key Field
(Primary Key)

PART

Part_Number	Part_Name	Unit_Price	Supplier_Number
137	Door latch	22.00	8259
145	Side mirror	12.00	8444
150	Door molding	6.00	8263
152	Door lock	31.00	8259
155	Compressor	54.00	8261
178	Door handle	10.00	8259

Primary Key

Foreign Key

Image source: w3resource.com

American National Standards Institute

- ❑ The American National Standards Institute (ANSI) is a private non-profit organization that is responsible for development of voluntary standards for products, services, systems, and personnel in the United States.
- ❑ RDBMS databases, such as Structured Query Language ([SQL](#)), were adopted as an ANSI standard in 1986 as SQL-86, and as an International Organization for Standardization (ISO) standard in 1987.

Why Standardize?

- ❑ Portability.
- ❑ Consistency.
- ❑ Monopoly Avoidance.

Note: Without a standard, each vendor would develop their own syntax.

Caveat: The standard better be good!

Structured Query Language - SQL

Structured Query Language (SQL) is a [domain-specific language](#) used in programming, and designed for managing data held in an [RDBMS](#). It is particularly useful in handling [structured data](#) (i.e., data incorporating relations among entities and variables).

- ❖ SQL is the set of statements with which all programs (software and applications) and users access data in a database.
- ❖ SQL is a language for organizing information, storing and updating information, accessing information, and protecting information.
- ❖ SQL provides statements for a variety of tasks, including:
 - Querying data.
 - Inserting, updating, and deleting rows in a table.
 - Creating, replacing and altering table properties using (DML)
 - Controlling access to the database and its objects.
 - Guaranteeing database consistency and integrity.

Structured Query Language - SQL (continued)

SQL can:

- Execute query data against a database.
- Retrieve data from a database.
- Insert records into a database.
- Update/replace/alter records in a database.
- Delete records from a database.
- Create new databases.
- Create new tables in a database.
- Create stored procedures and functions in a database.
- Create views in a database.
- Set permissions on tables, procedures, and views.
- Control access to the database and its objects.
- Provide guarantees of database consistency and integrity.

Atomicity, Consistency, Isolation, and Durability Properties

A good way to differentiate databases and test overall quality is to perform an atomicity, consistency, isolation, and durability (**ACID**) test. These four properties are scoped to a transaction, which is a unit of work that the programmer can define. A transaction can combine one or more database operations. For example:

1. **Atomicity is an all-or-none proposition.** Suppose you define a transaction that contains an UPDATE, an INSERT, and a DELETE statement. With atomicity, these statements are treated as a single unit, and thanks to consistency (the **C** in ACID), there are only two possible outcomes: they all change the database or none of them change the database. This is important in situations such as bank transactions, where transferring money between accounts could result in disaster if the server were to go down after a DELETE statement but before the corresponding INSERT statement.
2. **Consistency guarantees that a transaction never leaves your database in a half-finished state.** If one part of the transaction fails, all of the pending changes are rolled back, leaving the database as it was before you initiated the transaction. For instance, when you delete a customer record, you should also delete all of that customer's records from associated tables (such as invoices and line items). A properly configured database would not allow you delete the customer record if that meant leaving its invoices and other associated records stranded.

Atomicity, Consistency, Isolation, and Durability Properties

(continued)

3. Isolation keeps transactions separated from each other until they are finished. Transaction isolation is generally configurable in a variety of modes. For example, in one mode, a transaction blocks until the other transaction finishes. In a different mode, a transaction sees obsolete data (from the state the database was in before the previous transaction started).

Suppose a user deletes a customer, and before the customer's invoices are deleted, a second user updates one of those invoices. In a blocking transaction scenario, the second user would have to wait for the first user's deletions to complete before issuing the update. The second user would then find out that the customer had been deleted, which is much better than losing changes without knowing about it.

4. Durability guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination. Hence, even if the database server is unplugged in the middle of a transaction, it will return to a consistent state when it is restarted. The database handles this by storing uncommitted transactions in a transaction log.

Data Modeling

According to the **Data Management Book of Knowledge (DMBOK)**, Data Modeling is:

“The process of discovering, analyzing, representing, and communicating data requirements in a precise form called the data model.” And “data models depict and enable an organization to understand its data assets.”

Data Management Book of Knowledge (DMBOK)

- ❑ Data Modeling is an analysis and design method used to:
 - ❖ Define and analyze data requirements.
 - ❖ Define logical and physical structures that support the requirements.
 - ❖ Document software and business system designs.
- ❑ The “**modeling**” of various systems and processes often involves the use of diagrams, symbols, and textual references to represent the way the data flows through a software application or the Data Architecture within an enterprise.
- ❑ Mostly, we use three different data models. They allow developers to view and manipulate relationships between entities (tables). Each has its own way of storing the data. The following are the three different data models:
 1. Hierarchical Data Model.
 2. Network Data Model.
 3. Relational Data Model.

Relational Database Modeling Diagram

Relational Database Modeling Diagram also called *Entity Relationship Diagram (ERD)*.

- ❑ A Relational Database Model is a representation of a real-world situation about which data is to be **collected** and **stored** in a database and how data is **stored in Relational Databases**.
- ❑ An ERD is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system.
- ❑ Relational Database Modeling also includes practices, such as business process modeling, which deals with larger conceptual business processes and decision-making flows of entire organizations.

Database Modeling

Database modeling is a technique for documenting a database system using diagrams and symbols; it is used to represent communication of data.

- ❑ The highest level of abstraction for the data model is called the Entity Relationship Diagram (**ERD**), which is a graphical representation of data requirements for a database.
- ❑ For RDBMS, an **ERD** is most commonly used to design or debug databases.
- ❑ There are three components in ERD.
 1. **Entities**: Number of tables you need for your database.
 2. **Attributes**: Information such as property and facts that you need to describe each table.
 3. **Relationships**: How tables are linked together.

[Click here for more information about ERD.](#)

Introduction to Relationships

Relationships are the associations between the entities. Verbs often describe relationships between entities. Three types of relationships are discussed in this session. If you read about or hear of cardinality ratios, it also refers to types of relationships.

❖ One to One Relationship (1:1)

A single entity instance in one entity class is related to a single entity instance in another entity class. For example: Each professor has one office space.


❖ One to Many Relationship (1:M)


A single entity instance in one entity class (parent) is related to multiple entity instances in another entity class (child). For example: One instructor can teach many courses, but one course can only be taught by one instructor.


❖ Many to Many Relationship (M:M)

Each entity instance in one entity class is related to multiple entity instances in another entity class; and vice versa. For example: Each student can take many classes, and each class can be taken by many students.

We use Crow's Foot Symbols to represent the relationships.

■ one to one: 

■ one to many: 

■ many to many: 

Levels of Entity Relationship Diagram

Typically, three levels are used to produce an ERD.

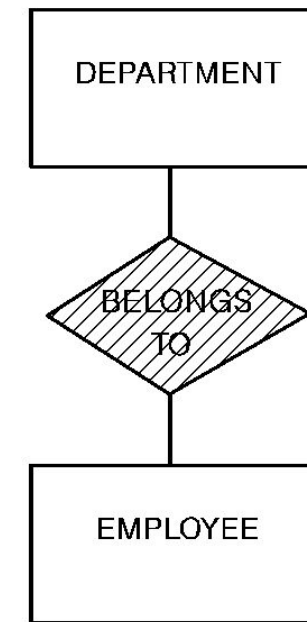
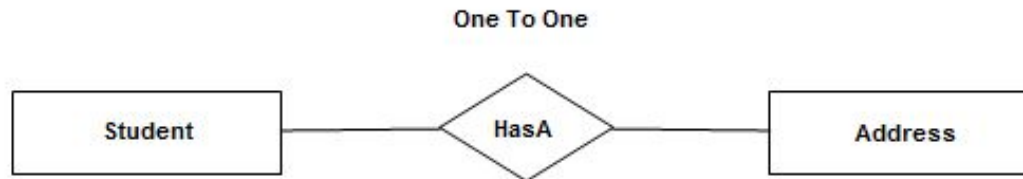
1. **Conceptual data model diagram.**
2. **Logical data model diagram.**
3. **Physical data model diagram**

We compare these three types of data model levels. The table compares the different features:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	✓
Foreign Keys		✓	✓
Table Names			✓
Column Names			✓
Column Data Types			✓

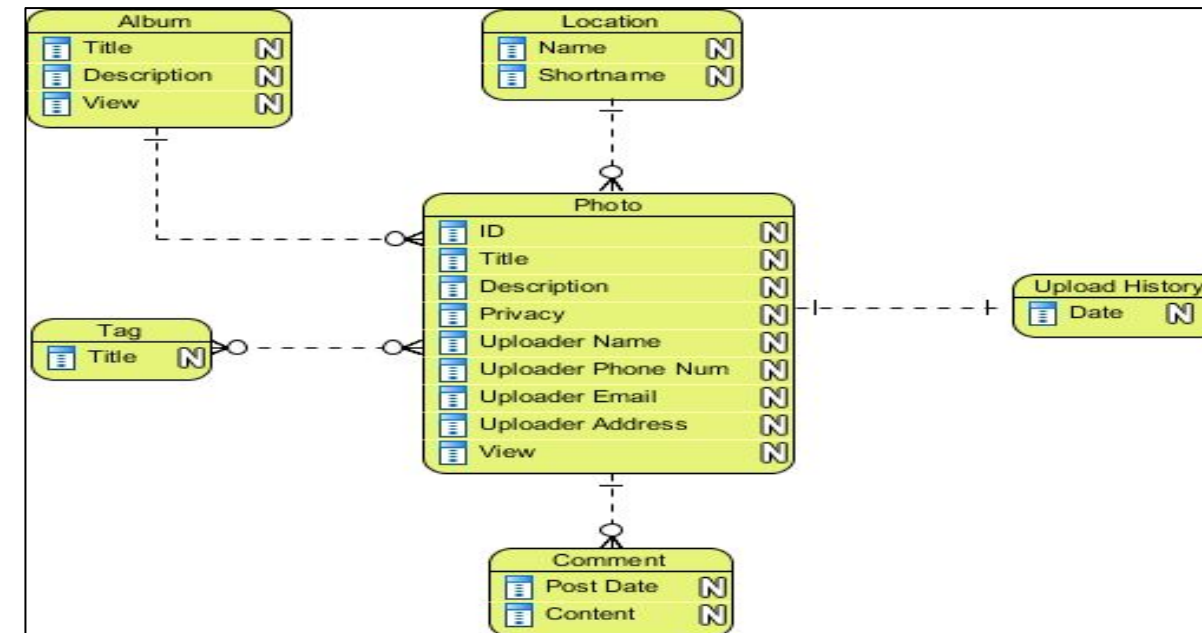
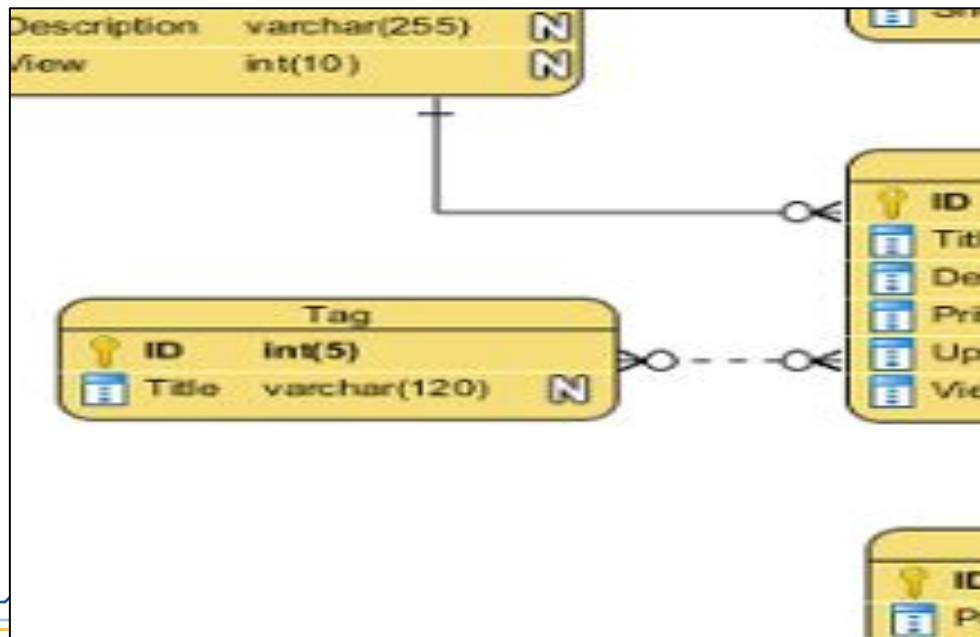
1 - Conceptual Data Model Diagram

- ❑ A **Conceptual Data Model** is a representation of a system, made up of the composition of **concepts**, which are used to help people know, understand, or simulate a subject that the model represents. It is also a set of concepts.
- ❑ **BrainStorming** involves database developers and programmers who gather information from business requirements, SRS, or a product backlog. The need of satisfying the database design is not considered yet.
- ❑ A **conceptual data model** identifies the highest-level relationships between the different entities.



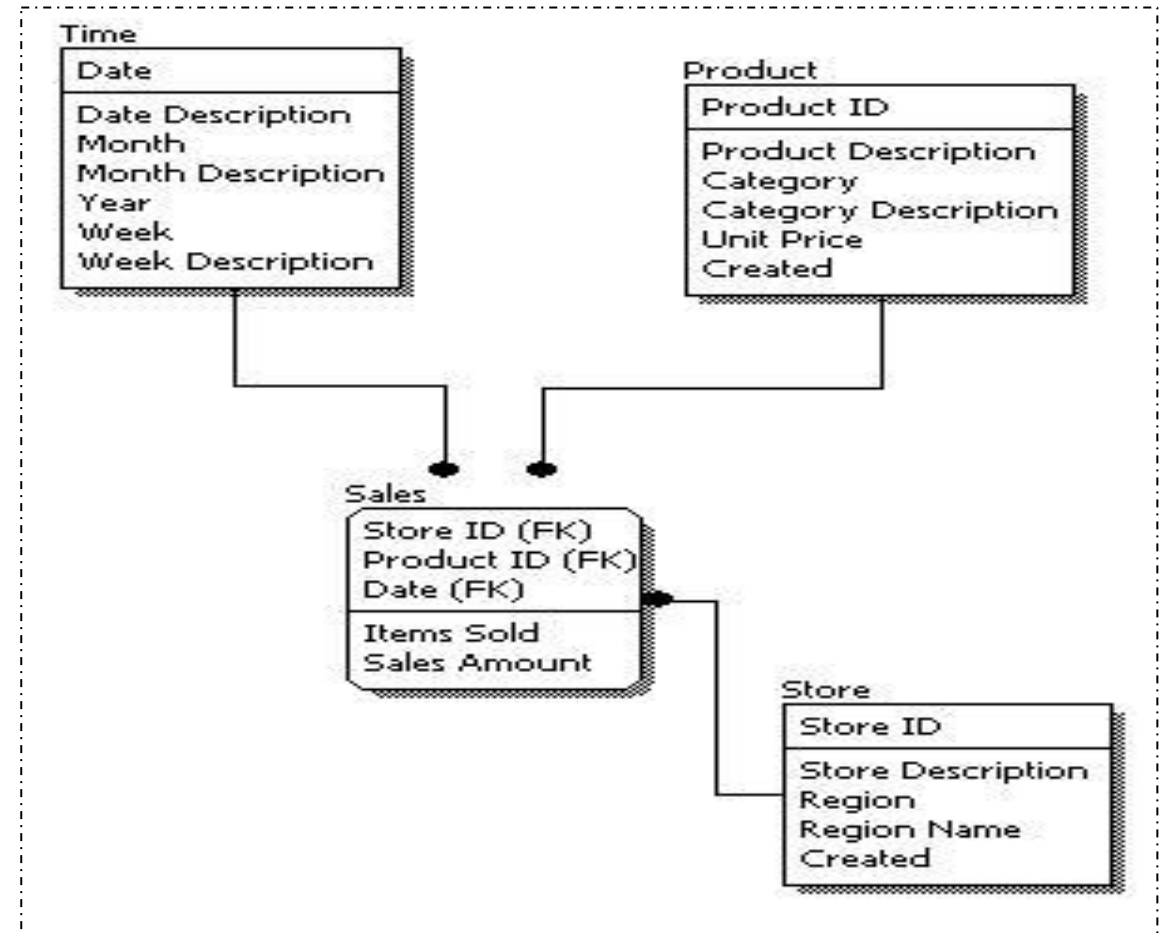
2 - Logical Data Model Diagram

- ❏ In the Logical Data Model Diagram, the data points should be as detailed as possible, without regard to how they will be physically implemented in the database. Features of a Logical Data Model Diagram include:
 - Includes all entities and relationships among them.
 - Specifies all attributes for each entity.
 - Specifies the primary key and composite key for each entity.
 - Specifies foreign keys (keys identifying the relationship between different entities).
 - Includes normalization (resolves many-to-many relationships).
- ❏ Only Database developers create Logical Data Model Diagrams, which is helpful for normalization.



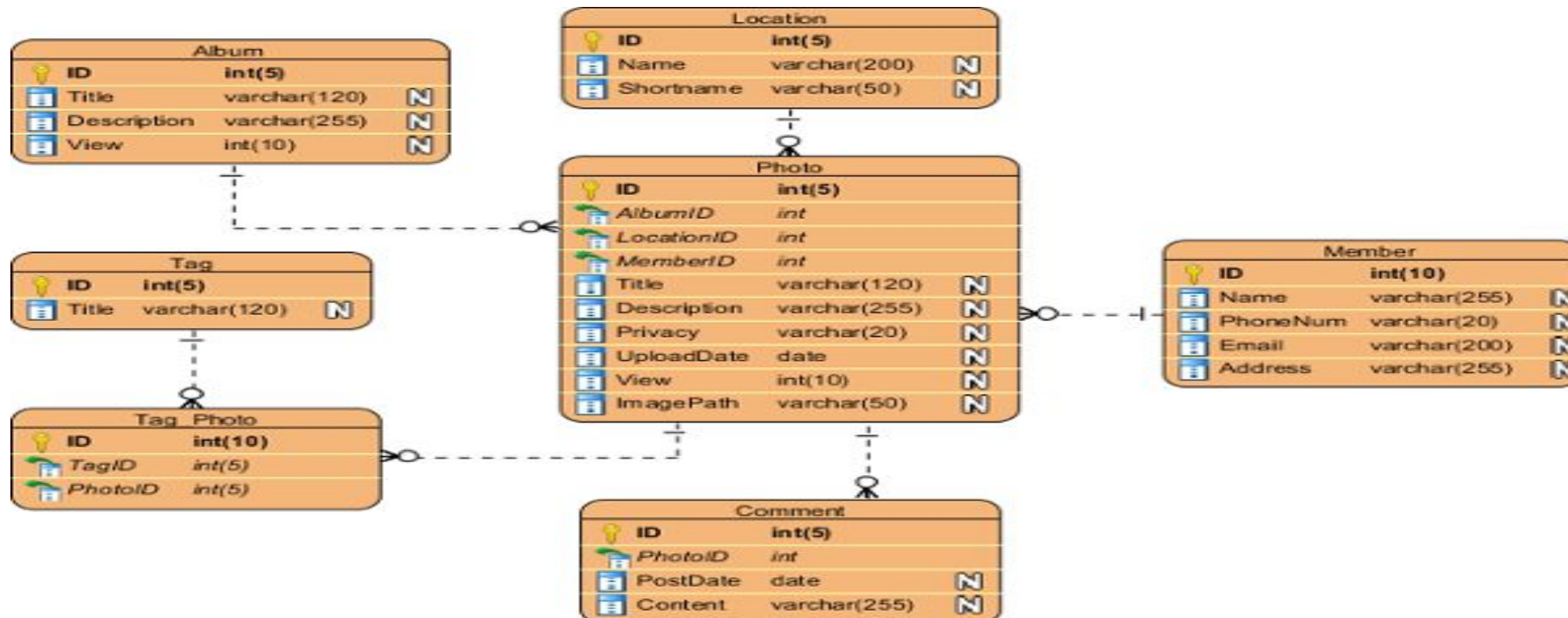
2 - Logical Data Model Diagram (continued)

- Any database should be designed with the end user in mind. The logical data model diagram, also referred to as the logical model, is the process of arranging data into logical, organized groups of objects that can easily be maintained.
- The logical design should reduce data repetition or go so far as to completely eliminate it. After all, why store the same data twice? Naming conventions used in a database should also be standard and logical.
- The Logical Data Model Diagram is very important and helpful for the database normalization process.



3 - Physical Data Model Diagram

The Physical Data Model Diagram represents the **actual design blueprint** of a **relational database**. It represents how data should be structured and related in a specific DBMS; therefore, it is important to consider the convention and restriction of the DBMS you use when you design a physical ERD. This means that an accurate use of data type is needed for entity columns, and the use of reserved words has to be avoided in naming entities and columns. Database designers may also add primary keys, foreign keys, and constraints to the design.



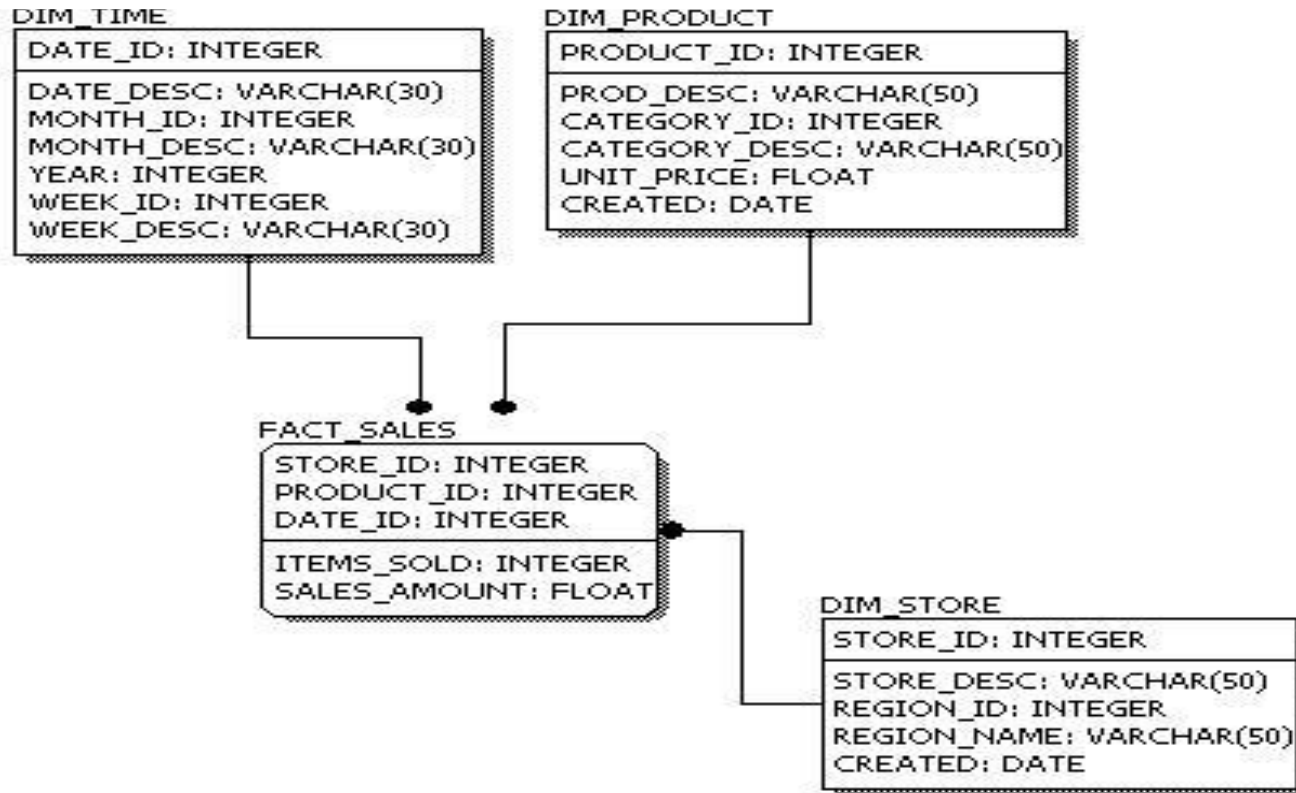


3 - Physical Data Model Diagram (continued)

- ❑ The Physical Data Model Diagram represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.
- ❑ Features of a Physical Data Model Diagram include:
 1. Conversion of entities into tables.
 2. Conversion of relationships into foreign keys.
 3. Denormalization based on user requirements.
 4. Physical considerations may cause the physical data model to be quite different from the logical data model.
 5. Definition of constraints for relationships.
- ❑ Physical data model diagram is also called **Schema of Database**.



3 - Physical Data Model Diagram (continued)



When comparing the physical model with the logical model diagram, we can see that the main differences include:

- Entity names are now table names.
- Attributes are now column names.
- Data type for each column is specified.
- Data types can be different depending on the actual database being used.

Tools for Entity Relationship Diagram

- ❑ [MySQL-Workbench Reverse engineer](#)
- ❑ [dbdesigner](#)
- ❑ [drawio-app.com](#)
- ❑ [Lucidchart](#)
- ❑ [Creately](#)
- ❑ [smartdraw](#)

Normalization

- ❏ Normalization is a process of reducing redundancies of data in a database.
- ❏ Normalization is a technique that is used when designing and redesigning a database.
- ❏ Normalization is a **process** or **set of guidelines** used to optimally design a database to ***reduce redundant data***. The actual guidelines of normalization are called **normal forms**.

Normalization (continued)

The image on the right illustrates the database before it was normalized.

- ❑ A database that is not normalized may include data that is contained in one or more different tables for no apparent reason. This could be bad for **security reasons, disk space usage, speed of queries, efficiency of database updates, and most importantly, data integrity.**
- ❑ A database before normalization is one that has not been broken down logically into smaller, more manageable tables.

COMPANY_DATABASE	
emp_id	cust_id
last_name	cust_name
first_name	cust_address
middle_name	cust_city
address	cust_state
city	cust_zip
state	cust_phone
zip	cust_fax
phone	ord_num
pager	qty
position	ord_date
date_hire	prod_id
pay_rate	prod_desc
bonus	cost
date_last_raise	

Normalization (continued)

Data Redundancy

Data **should not** be redundant, which means that the duplication of data should be kept to a minimum for several reasons. For example, it is unnecessary to store an employee's home address in more than one table. With duplicate data, unnecessary space is used, and confusion is always a threat. For instance, an employee's address in one table does not match the address of the same employee in another table.

- Which table is correct?
- Do you have documentation to verify the employee's current address?

As if data management were not difficult enough, redundancy of data could prove to be a disaster.

Normalization (continued)

End User's Needs

- The needs of the end user should be one of the top considerations when designing a database. Remember that the end user is the person who ultimately uses the database. There should be ease of use through the user's front-end tool (a client program that allows a user access to a database), but this, along with optimal performance, cannot be achieved if the user's needs are not taken into consideration
- Some user-related design considerations include:
 - What data should be stored in the database?
 - How will the user access the database?
 - What privileges does the user require?
 - How should the data be grouped in the database?
 - What data is the most commonly accessed?
 - How is all data related in the database?
 - What measures should be taken to ensure accurate data?

Normal Forms

- ❑ Normal form is a way of measuring the levels, or depth to which a database has been normalized. A database's level of normalization is determined by the normal form.
- ❑ The following are the three most common normal forms in the normalization process:
 - ▶ The **first** normal form.
 - ▶ The **second** normal form.
 - ▶ The **third** normal form.
- ❑ Of the three normal forms, each subsequent normal form depends on normalization steps taken in the previous normal form. For example, to normalize a database using the second normal form, the database must first be in the first normal form.

The First Normal Form

The objective of the first normal form is to divide the base data into logical units called **tables**. When each table has been designed, a primary key is assigned to most or all tables.

First Normal Form (1NF) rules:

1. It should only have single (atomic) valued attributes/columns.
2. Each record needs to be unique.
3. Values stored in a column should be of the same domain.
4. All of the columns in a table should have unique names.
5. The order in which data is stored does not matter.

Example - The First Normal Form

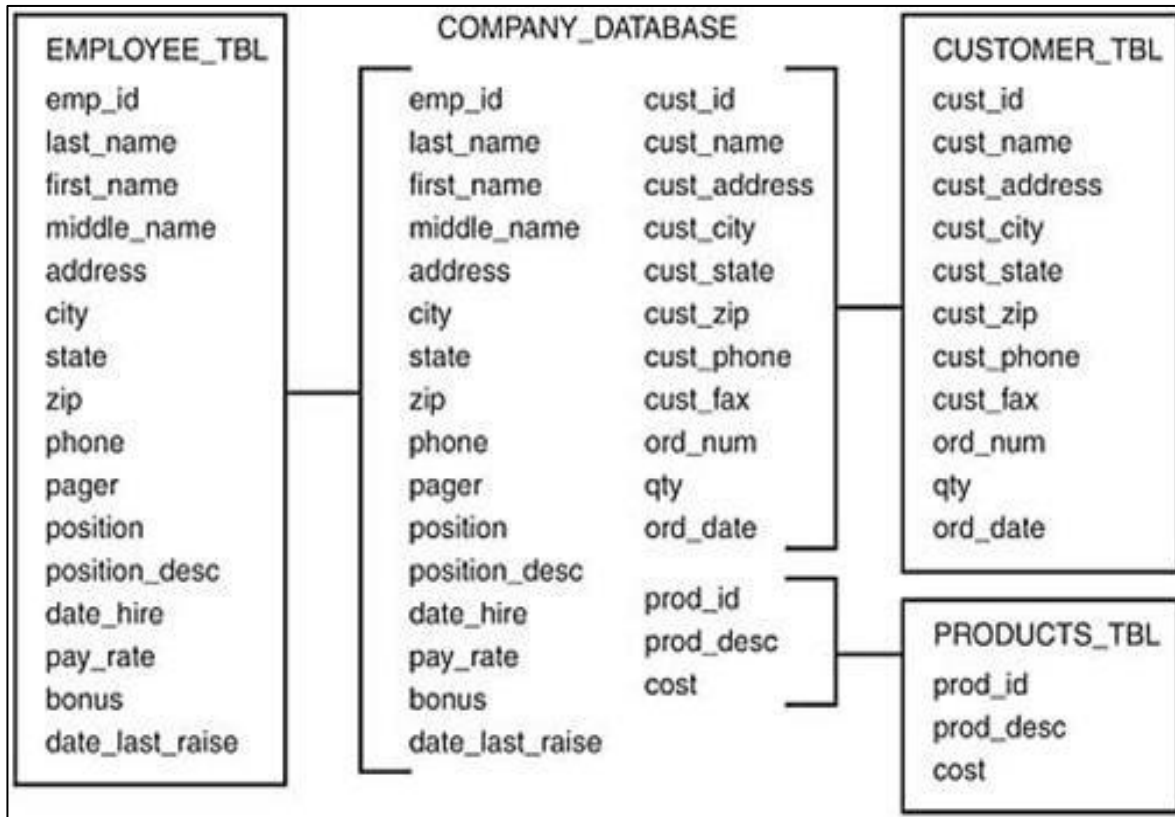
Examine Figure: Assume that this is a raw database with ONE table...

Database Name: COMPANY_DATABASE	
Table Name: company_info_tbl	
emp_id	cust_id
last_name	cust_name
first_name	cust_address
middle_name	cust_city
address	cust_state
city	cust_zip
state	cust_phone
zip	cust_fax
phone	ord_num
pager	qty
position	ord_date
date_hire	prod_id
pay_rate	prod_desc
bonus	cost
date_last_raise	

Take a moment to consider how you will divide the ONE table into smaller tables.

Example - First Normal Form (continued)

The figure below illustrates how the raw database shown in the previous figure has been redeveloped using the 1NF.



You can see that to achieve 1NF, the data had to be broken into logical units of related information, each having a primary key and ensuring that there are no repeated groups in any of the tables. Instead of one large table, there are now smaller, more manageable tables:

- **EMPLOYEE_TBL**
- **CUSTOMER_TBL**
- **PRODUCTS_TBL.**

The primary keys are normally the first columns listed in a table. In this case, the primary keys are: EMP_ID, CUST_ID, and PROD_ID .

Second Normal Form

The objective of the Second Normal Form (2NF) is to take data that is only partly dependent upon the primary key, and enter that data into another table.

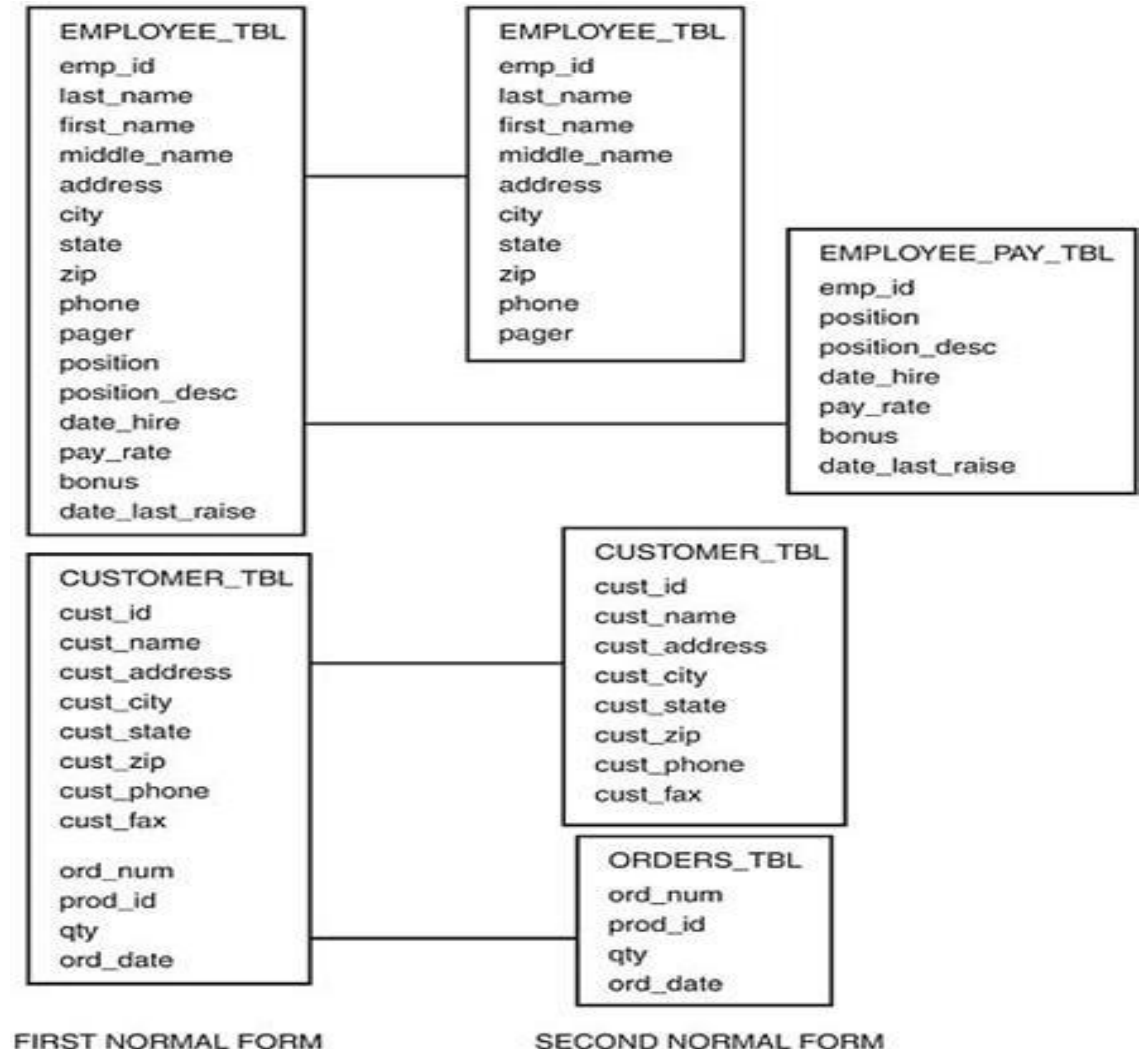
2NF Rules:

1. It should be in the First Normal Form.
2. It should not have partial dependency.

Example - The Second Normal Form

According to the figure, 2NF derived from 1NF by further breaking two tables down into more specific units:

- **EMPLOYEE_TBL** split into two tables called **EMPLOYEE_TBL** and **EMPLOYEE_PAY_TBL**.
- Personal employee information is dependent on the primary key (EMP_ID), so that information remained in the **EMPLOYEE_TBL** (**EMP_ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, ADDRESS, CITY, STATE, ZIP, PHONE, and PAGER**).
- On the other hand, the information that is only partly dependent on the EMP_ID (each individual employee) is used to populate **EMPLOYEE_PAY_TBL** (**EMP_ID, POSITION, POSITION_DESC, DATE_HIRE, PAY_RATE, and DATE_LAST_RAISE**). Notice that both tables contain the column EMP_ID. This is the primary key of each table and is used to match corresponding data between the two tables.
- **CUSTOMER_TBL** split into two tables called **CUSTOMER_TBL** and **ORDERS_TBL**. What took place is similar to what occurred in the EMPLOYEE_TBL. Columns that were partly dependent on the primary key were directed to another table. The order information for a customer is dependent on each CUST_ID, but does not directly depend on the general customer information in the original table.





The Third Normal Form

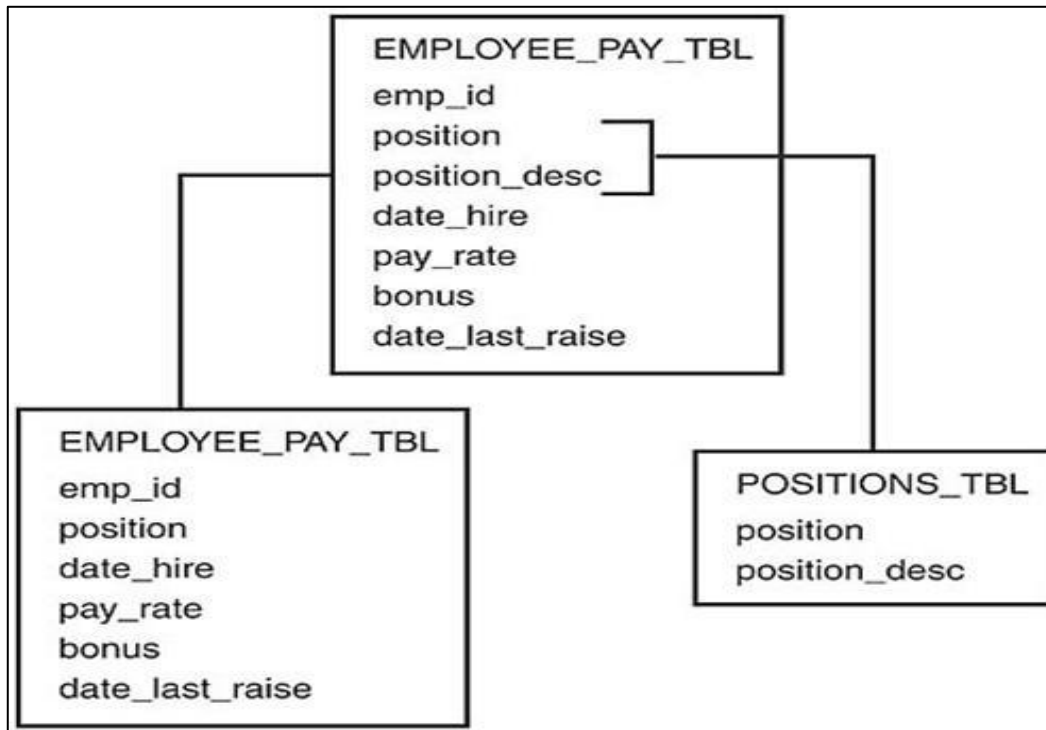
The objective of the Third Normal Form (3NF) is to remove data in a table that is not dependent upon the primary key.

3NF Rules:

1. It should be in the Second Normal Form.
2. It should not have Transitive Dependency.
 - ❖ The advantage of removing transitive dependency is that the amount of data duplication is reduced, and data integrity is achieved.



Example - Third Normal Form



Another table was created to display the use of the 3NF. **EMPLOYEE_PAY_TBL** table can split into two tables: one table containing the actual employee pay information, and the other table containing the position and position_descriptions, which really do not need to reside in **EMPLOYEE_PAY_TBL**. The **POSITION_DESC** column is totally independent of the primary key, EMP_ID.

Advantages of Relational Database

- **Simplicity:** A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** The Relational model in DBMS is easy, as tables consisting of rows and columns are quite natural and simple to understand.
- **Query capability:** It makes possible for a high-level query language like **SQL** to avoid complex database navigation.
- **Data independence:** The Structure of Relational database can be changed without having to change any application.

Knowledge Check

1. What is DBMS?
2. What is RDBMS?
3. What is Normalization?
4. What is data redundancy?
5. What is ERD, and why do we need it?

Summary

- The Relational Database Management System represents the database as a collection of relations (tables). Attribute, Tables, Tuple, Relation Schema, Degree, Cardinality, Column, and Relation instance, are some important components of Relational Database Modeling.
- Data Integrity constraints are referred to as *conditions*, which must be present for a valid Relation approach in DBMS.
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain, or if it is not of the appropriate data type.
- Insert, Select, Modify, and Delete are the operations performed in Relational Database Model Diagram constraints.
- The relational database is only concerned with data and not with a structure, which can improve the performance of the model.
- Advantages of the Relational Data Model Diagram in DBMS are simplicity, structural independence, ease of use, query capability, data independence, scalability, etc.
- Few relational databases have limits on field lengths, which cannot be exceeded.
- SQL is a language that is used to manage data that is held in a relational database management system. It uses tables to manipulate and retrieve information from databases for analysis.



Reference

- ❑ [https://docs.oracle.com/cd/E18283_01/server.112/e17118/intro.htm#:~:text=Structured%20Query%20Language%20\(SQL\)%20is,when%20executing%20the%20user's%20request.](https://docs.oracle.com/cd/E18283_01/server.112/e17118/intro.htm#:~:text=Structured%20Query%20Language%20(SQL)%20is,when%20executing%20the%20user's%20request.)
- ❑ <https://www.youtube.com/watch?v=RJ9TpkWKyU0>
- ❑ https://www.youtube.com/watch?v=tR_rOJPiEXc



Questions?

