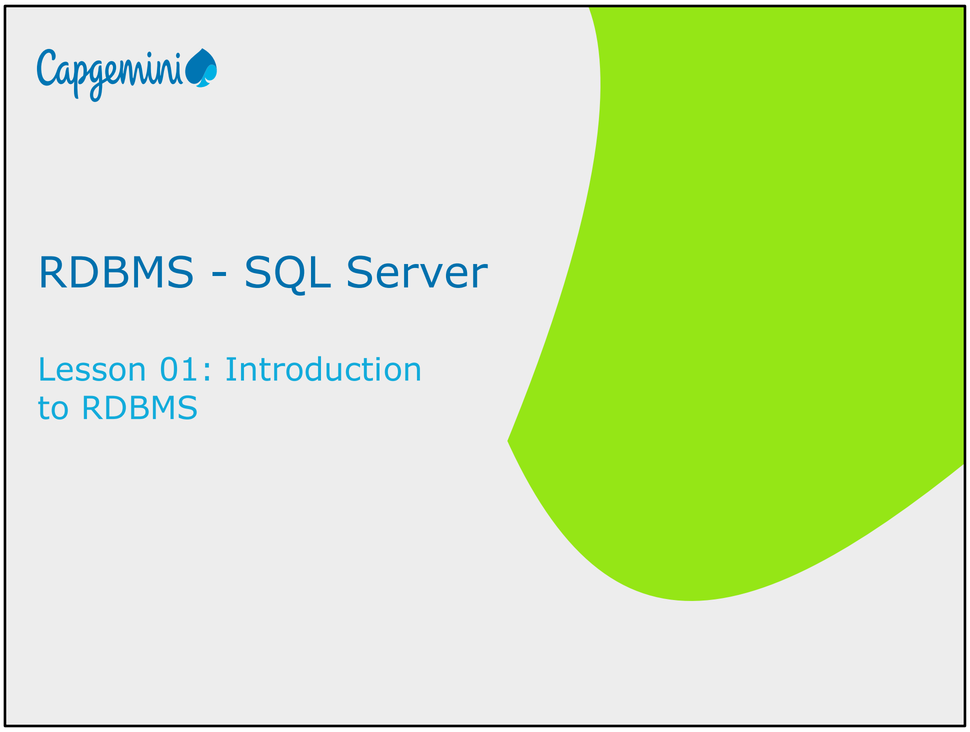
**Instructor Notes:**

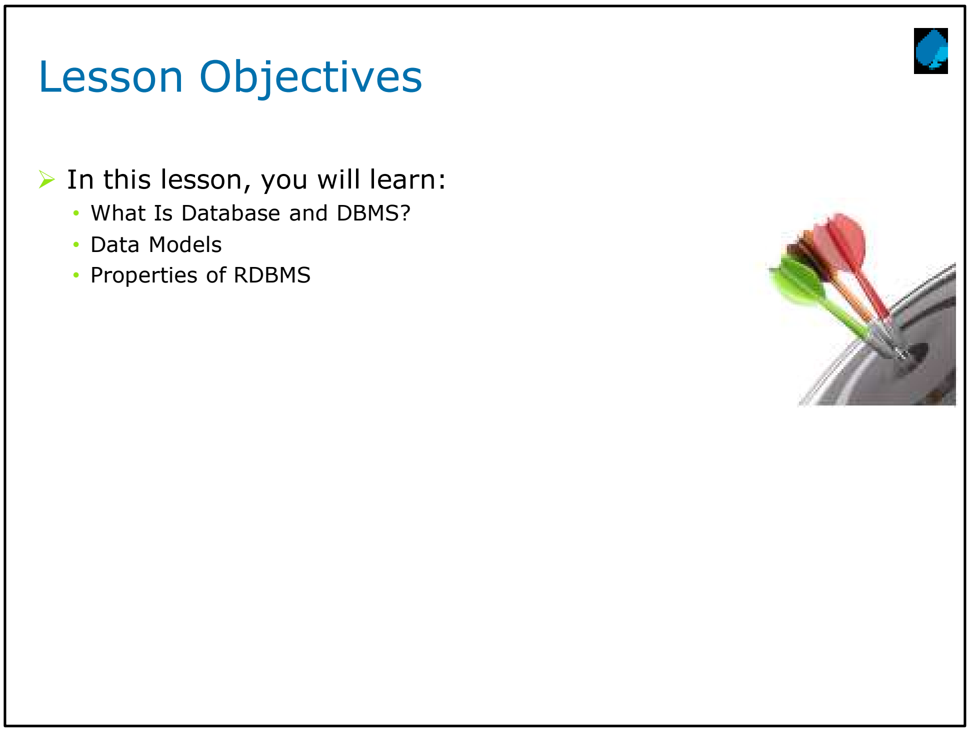


Capgemini

# RDBMS - SQL Server

Lesson 01: Introduction to RDBMS

**Instructor Notes:**



## Lesson Objectives

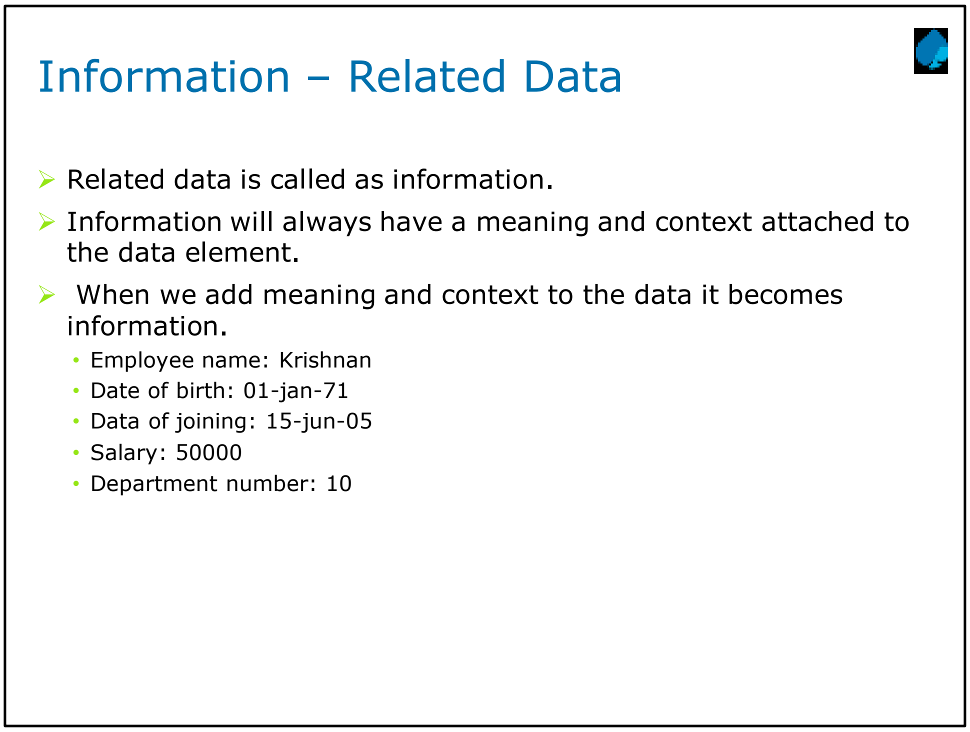
- In this lesson, you will learn:
  - What Is Database and DBMS?
  - Data Models
  - Properties of RDBMS

**Instructor Notes:**

## What is Data?

- Data (plural of the word datum) is a factual information used as a basis for reasoning, discussion, or calculation
- Data may be numerical data which may be integers or floating point numbers, and non-numerical data such as characters, date etc.
- Data by itself normally doesn’t have a meaning associated with it.
  - e.g.:- Krishnan ,01-jan-71,15-jun-05,50000

**Instructor Notes:**



## Information – Related Data

- Related data is called as information.
- Information will always have a meaning and context attached to the data element.
- When we add meaning and context to the data it becomes information.
  - Employee name: Krishnan
  - Date of birth: 01-jan-71
  - Data of joining: 15-jun-05
  - Salary: 50000
  - Department number: 10

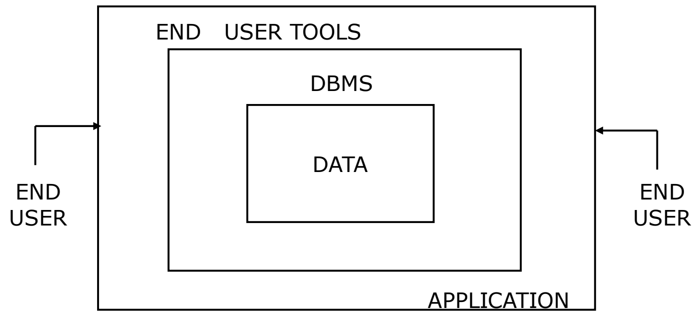
**Instructor Notes:**

Start with the need of Database. Talk about the 3 tier architecture and then move to the data storage.

Explain the importance of data management in brief and then introduce this slide. This will cover WHY part and then we come to WHAT part Explained on the slide.

## Introduction to Database

- DATABASE - A set of inter-related data
- DBMS - A software that manages the data
- SCHEMA - A set of structures and relationships, which meet a specific need



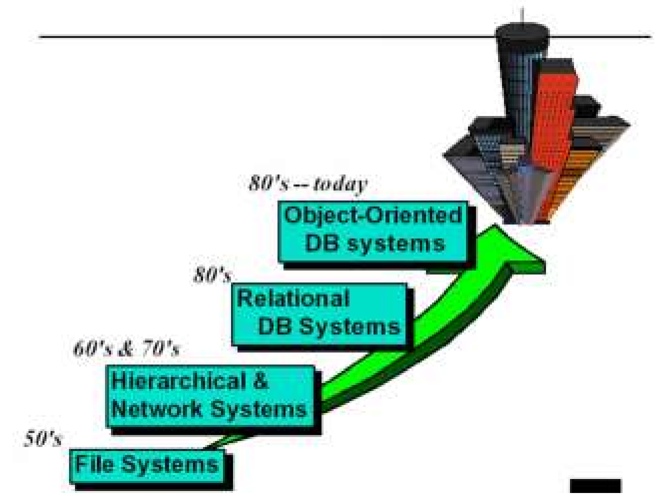
**Introduction**

A set of inter-related data is known as *database* and the software that manages it is known as *database management system* or DBMS. Hence DBMS can be described as "a computer-based record keeping system which consists of software for processing a collection of interrelated data". A set of structures and relationships that meet a specific need is called as a *schema*.

The database is centrally managed by a person known as the database administrator or the DBA. The DBA initially studies the System and accordingly decides the types of data to be used, then the structures to be used to hold the data and the interrelationships between the data structures. He then defines data to the DBMS. The DBA also ensures the security of the database. The DBA usually controls access to the data through the user codes and passwords and by restricting the views or operations that the users can perform on the database.

**Instructor Notes:**

## Evolution of Database



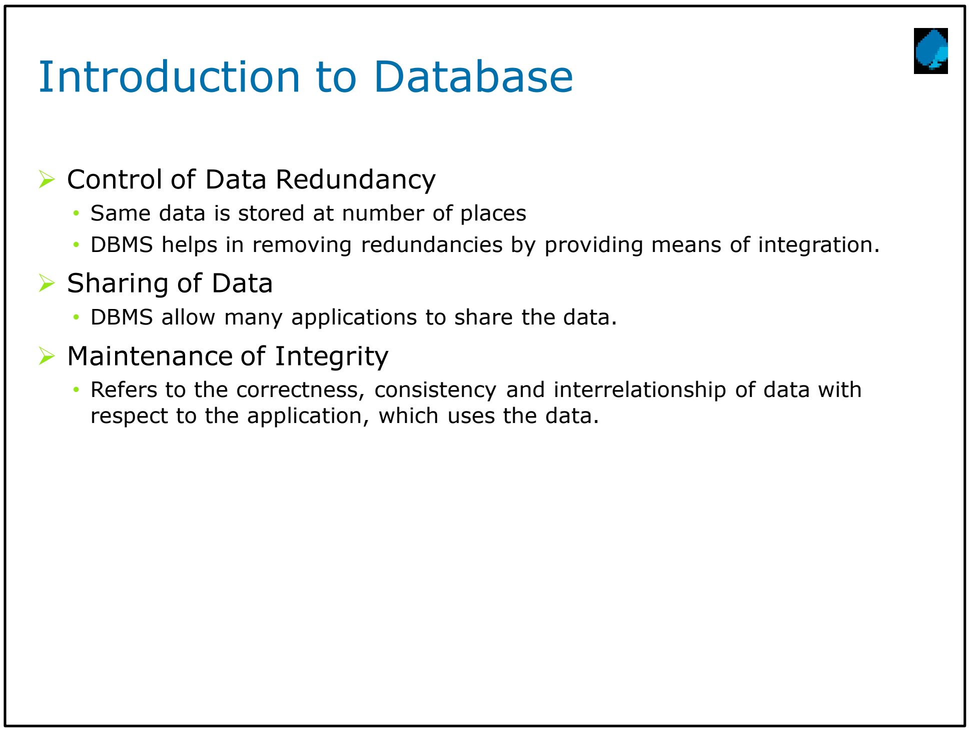
**Instructor Notes:**

## Database Management System

- A database is a collection of logically related information.
- Database Management is the task of maintaining databases so that information is readily and accurately available.
- The software required to perform the task of database management is called a Database Management System (DBMS).

**Instructor Notes:**

Link this to the need of Database that you talked about before first slide. You can also compare redundancy and integrity problem in shared files.
Transaction control feature (on next slide) and SQL Language features can only be obtained in DBMS.



## Introduction to Database

- Control of Data Redundancy
  - Same data is stored at number of places
  - DBMS helps in removing redundancies by providing means of integration.
- Sharing of Data
  - DBMS allow many applications to share the data.
- Maintenance of Integrity
  - Refers to the correctness, consistency and interrelationship of data with respect to the application, which uses the data.

**Characteristics of DBMS**

Some of the characteristics of the DBMS have been discussed below:

**Control of Data Redundancy**

When the same data is stored in a number of files it brings in data redundancy. In such cases, if the data is changed at one place, the change has to be duplicated in each of the files.
The main disadvantages of data redundancy are:
Storage space gets wasted.
Processing time may be wasted as more data is to be handled.
Inconsistencies may creep in.
DBMS help in removing redundancies by providing means of integration.

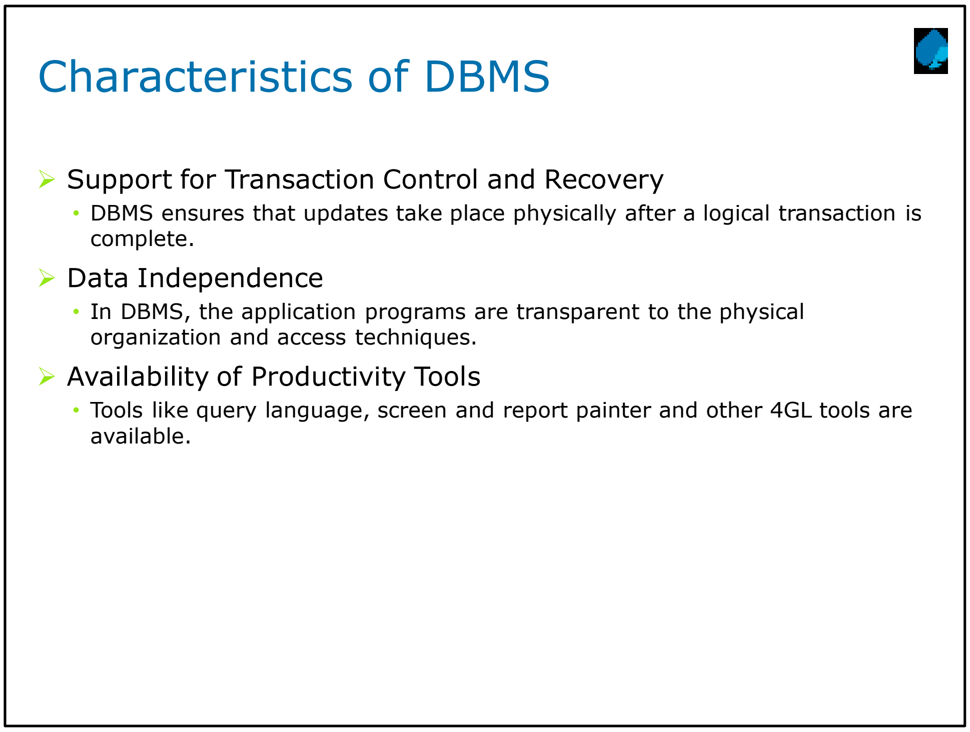
**Sharing of Data**

DBMS allow many applications to share the data.

**Maintenance of Integrity**

Integrity of data refers to the correctness, consistency and interrelationship of data with respect to the application that uses the data. Some of the aspects of data integrity are:
Many data items can only take a restricted set of values.
Certain field values are not to be duplicated across records. Such restrictions, called *primary key* constraints can be defined to the DBMS.
Data integrity, which defines the relationships between different files, is called *referential integrity* rule, which can also be specified to the DBMS.

**Instructor Notes:**



## Characteristics of DBMS

- Support for Transaction Control and Recovery
  - DBMS ensures that updates take place physically after a logical transaction is complete.
- Data Independence
  - In DBMS, the application programs are transparent to the physical organization and access techniques.
- Availability of Productivity Tools
  - Tools like query language, screen and report painter and other 4GL tools are available.

**Support for Transaction Control and Recovery**

Multiple changes to the database can be clubbed together as a single 'logical transaction'. The DBMS will ensure that the updates take place physically only when the logical transaction is complete.

**Data Independence**

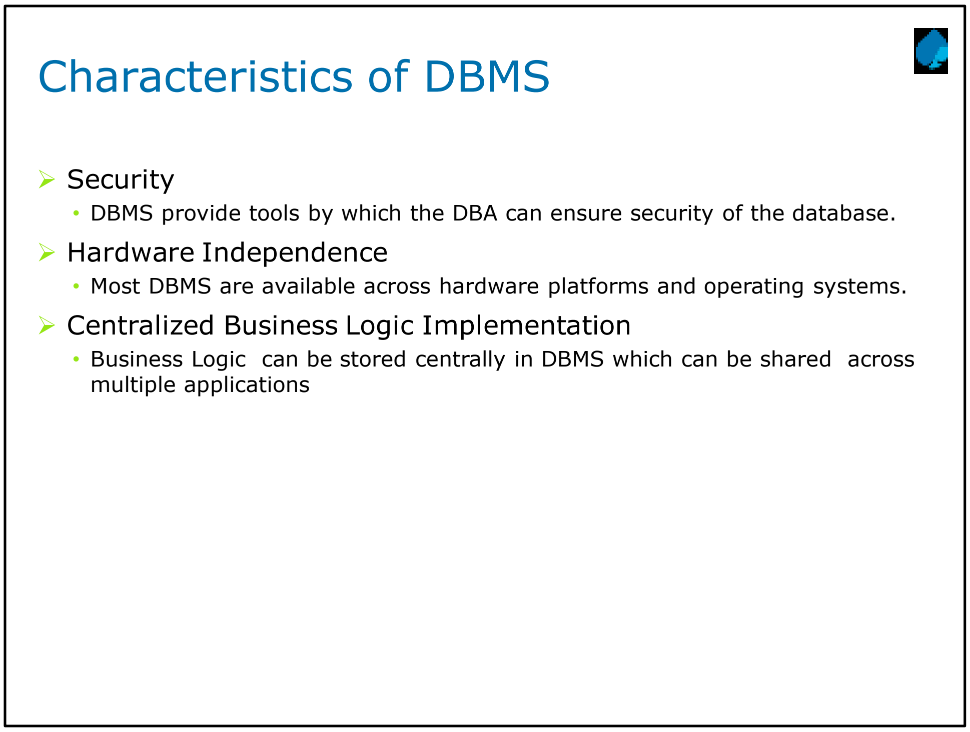
In conventional file based applications, programs need to know the data organization and access technique to be able to access the data. This means that if you make any change in the way the data is organized you will also have to take care to make changes to the application programs that apply to the data. In DBMS, the application programs are transparent to the physical organization and access techniques.

**Availability of Productivity Tools**

Tools like query language, screen and report painter and other 4GL tools are available. These tools can be utilized by the end-users to query, print reports etc. SQL is one such language, which has emerged as standard.

**Instructor Notes:**

Explain why security is essential. Give example like different roles in an Organization would like to access different data from the database.
E.g.. Accounts people should have access
To payroll data but other employees should not be given access to it.



## Characteristics of DBMS

- Security
  - DBMS provide tools by which the DBA can ensure security of the database.
- Hardware Independence
  - Most DBMS are available across hardware platforms and operating systems.
- Centralized Business Logic Implementation
  - Business Logic can be stored centrally in DBMS which can be shared across multiple applications

**Security**
DBMSs provide tools by which the DBA can ensure security of the database.
**Hardware Independence**
Most DBMSs are available across hardware platforms and operating systems. Thus the application programs need not be changed or rewritten when the hardware platform or operating system is changed or upgraded.

**Instructor Notes:**

Discuss the various definitions of a Model.

Explain how it can help them to explore the requirements further.

## Definition of a Model

- An integrated collection of concepts for describing data, relationships between data, and constraints on the data used by an organization.
- A representation of 'real world' objects and events, and their associations.
- It attempts to represent the data requirements of the organization that you wish to model
- Modeling is an integral part of the design and development of any system.
- A correct model is essential.

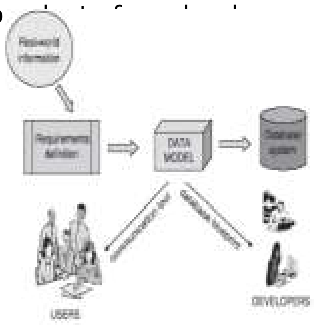
What is a model?
A model serves two primary purposes:
As a true representation of some aspects of the real world, a model enables clearer communication about those aspects.
A model serves as a blueprint to shape and construct the proposed structures in the real world.
So, what is a data model? A data model is an instrument that is useful in the following ways:
1) A model helps the users or stakeholders clearly understand the database system that is being implemented. It helps them understand the system with reference to the information requirements of an organization.
2) It enables the database practitioners to implement the database system exactly conforming to the information requirements.

A data model, therefore, serves as a critical tool for communication with the users and it also serves as a blueprint of the database system for the developers.
Without a proper data model, an adequate database system cannot be correctly designed and implemented. A good data model forms an essential prerequisite for any successful database system. Unless the data modelers represent the information requirements of the organization in a proper data model, the database design will be totally ineffective.

**Instructor Notes:**

## What is Data Modeling?

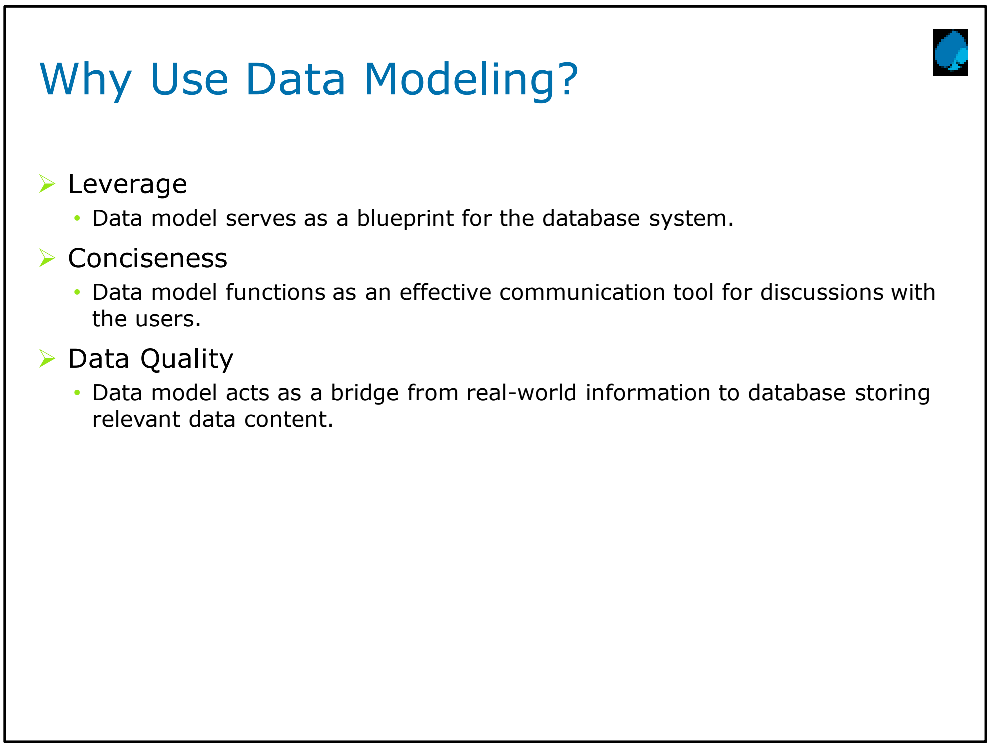
- Data modeling is a technique for exploring the data structures needed to support an organization's information need.
- It would be a conceptual representation or a replica of the data structure required in the database system.
- A data model focuses on which data is required and how the data should be organized.
- At the conceptual level, the data model is indep or software constraints.



What is Data Modeling?
At this level, the data model is generic; it does not vary whether you want to implement an object-relational database, a relational database, a hierarchical database, or a network database.
At the next level down, a data model is a logical model relating to the particular type of database relational, hierarchical, network, and so on. This is because in each of these types, data structures are perceived differently.
If you proceed further down, a data model is a physical model relating to the particular database management system (DBMS) you may use to implement the database.

**Instructor Notes:**

Explain how a model will help in understanding and communicating requirements better.



## Why Use Data Modeling?

- Leverage
  - Data model serves as a blueprint for the database system.
- Conciseness
  - Data model functions as an effective communication tool for discussions with the users.
- Data Quality
  - Data model acts as a bridge from real-world information to database storing relevant data content.

Why Use Data Modeling?

**Leverage:** The key reason for giving special attention to data organization is the leverage. A small change to a data model may have a major impact on the whole system. Therefore, you can opt for modifying the data model instead of the system. For the most commercial information systems, the programs are far more complex. Also, considerable time is consumed in specifying and constructing them, as compared to the database. However, their contents and structures are heavily influenced by the database design.

**Conciseness:** A data model is a very powerful tool for establishing requirements and capabilities of information systems. Its valuable because of its conciseness. It implicitly defines a whole set of screens, reports, and processes needed to capture, update, retrieve, and delete the specified data. The data modeling process can tremendously facilitate our understanding of the essence of business requirements.

**Data Quality:** The data held in a database is usually a valuable business asset built up over a long period. Inaccurate data (poor data quality) reduces the value of the asset and can be expensive or impossible to correct. Frequently, problems with data quality can be traced back to a lack of consistency in (a) defining and interpreting data, and (b) implementing mechanisms to enforce the definitions.

**Instructor Notes:**

Data modeling is a technique of creating a conceptual schema based on the entities identified in the problem domain and finally arriving at the physical database schema

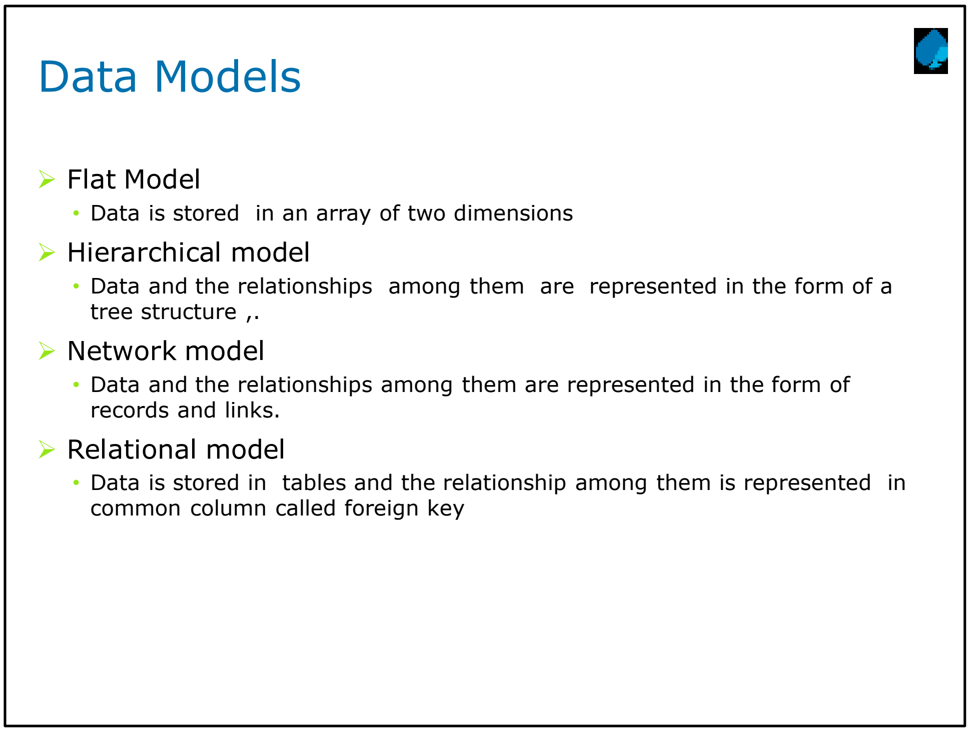
Database modeling is how the data and its relationship is organized in the database .

## Data Models

- It is a specification of how the data in a database is structured and accessed .
- Common models are
  - Flat Model
  - Hierarchical
  - Network
  - Relational

**Instructor Notes:**

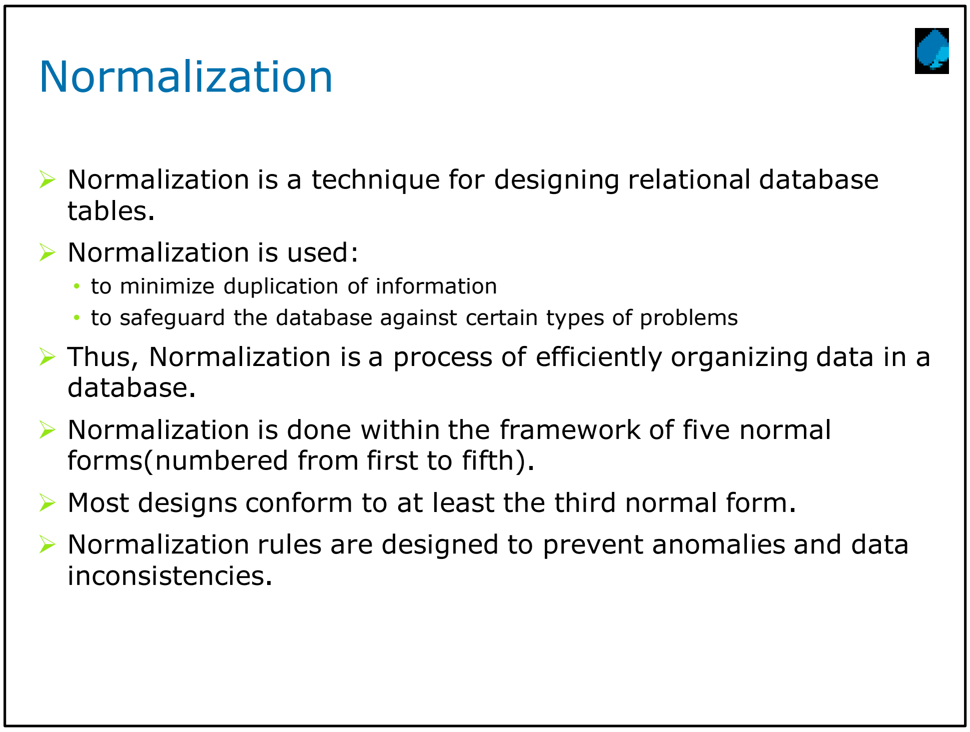
The instructor can talk about Object relation model also , if he /she is comfortable



## Data Models

- Flat Model
  - Data is stored in an array of two dimensions
- Hierarchical model
  - Data and the relationships among them are represented in the form of a tree structure ,.
- Network model
  - Data and the relationships among them are represented in the form of records and links.
- Relational model
  - Data is stored in tables and the relationship among them is represented in common column called foreign key

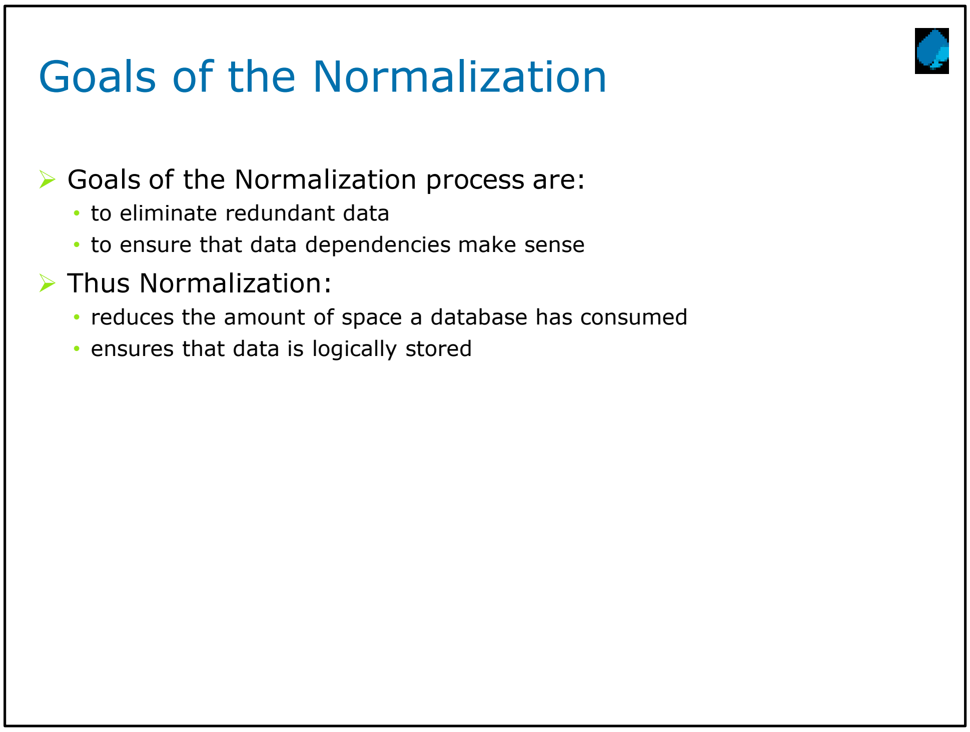
**Instructor Notes:**



## Normalization

- Normalization is a technique for designing relational database tables.
- Normalization is used:
  - to minimize duplication of information
  - to safeguard the database against certain types of problems
- Thus, Normalization is a process of efficiently organizing data in a database.
- Normalization is done within the framework of five normal forms(numbered from first to fifth).
- Most designs conform to at least the third normal form.
- Normalization rules are designed to prevent anomalies and data inconsistencies.

**Instructor Notes:**



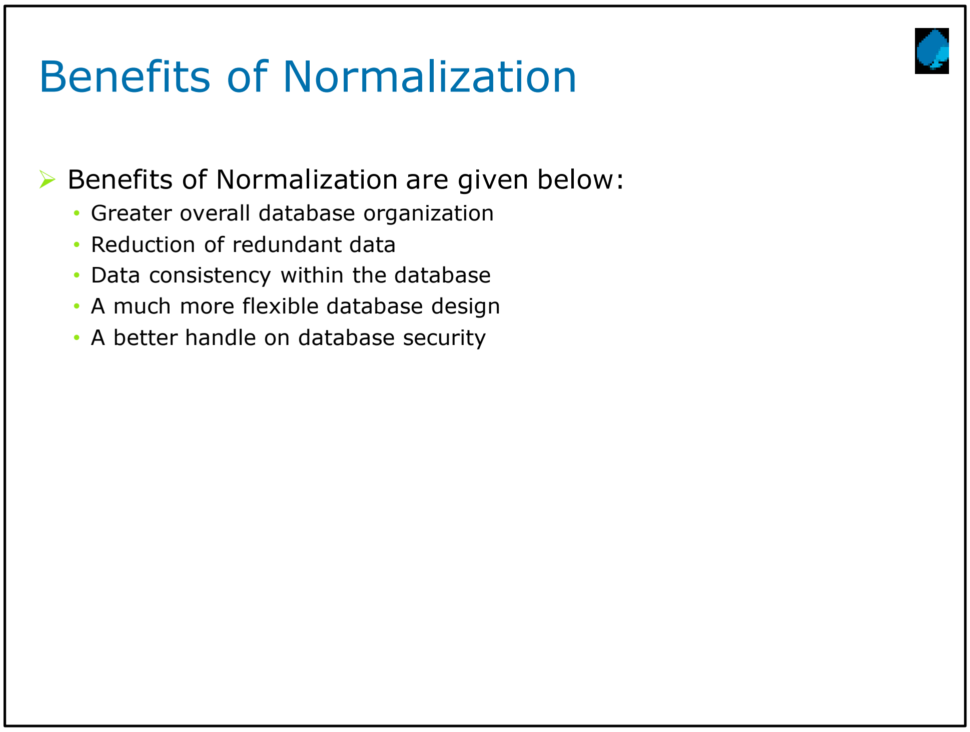
## Goals of the Normalization

- Goals of the Normalization process are:
  - to eliminate redundant data
  - to ensure that data dependencies make sense
- Thus Normalization:
  - reduces the amount of space a database has consumed
  - ensures that data is logically stored

**<u>Normalization (contd.)</u>:**

- Higher degrees of normalization typically involve more tables and create the need for a larger number of joins, which can reduce performance.
- Accordingly:
- More highly normalized tables are typically used in database applications involving many isolated transactions
  **For example:** an Automated teller machine
- While less normalized tables tend to be used in database applications that need to map complex relationships between data entities and data attributes
  **For example:** a reporting application, or a full-text search application
- Database theory describes a table's degree of normalization in terms of Normal Forms of successively higher degrees of strictness.
  **For example:** A table in Third Normal form (3NF), is consequently in Second Normal form (2NF) as well but the reverse is not necessarily true.
- Although the Normal Forms are often defined informally in terms of the characteristics of tables, rigorous definitions of the Normal Forms are concerned with the characteristics of mathematical constructs known as "Relations". Whenever information is represented relationally, it is meaningful to consider the extent to which the representation is normalized.

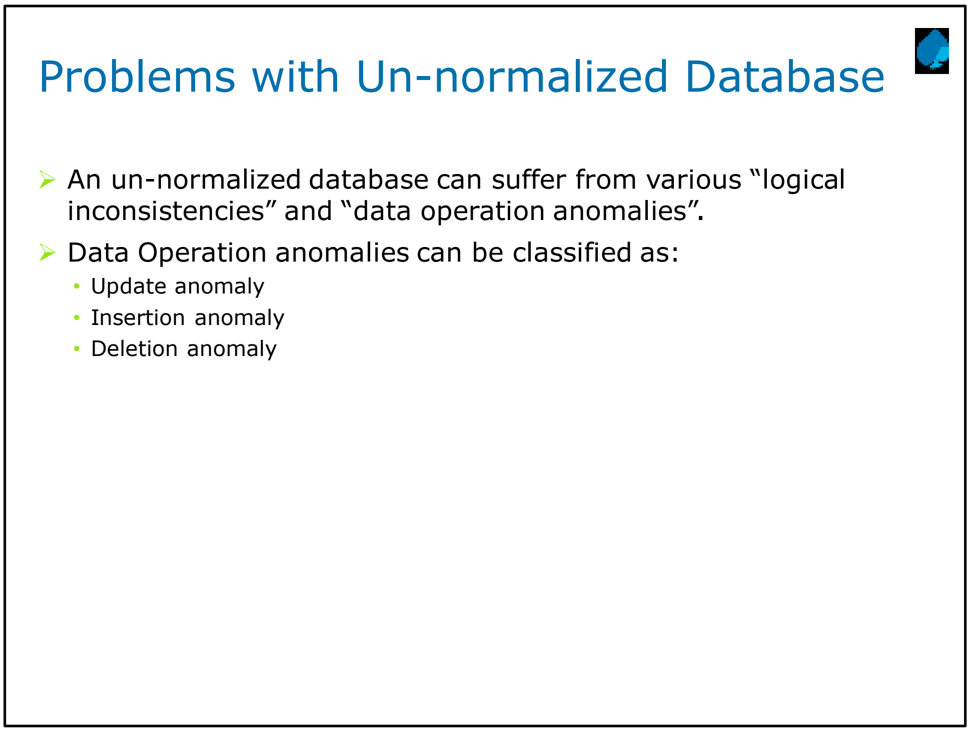
**Instructor Notes:**



## Benefits of Normalization

- Benefits of Normalization are given below:
  - Greater overall database organization
  - Reduction of redundant data
  - Data consistency within the database
  - A much more flexible database design
  - A better handle on database security

**Instructor Notes:**

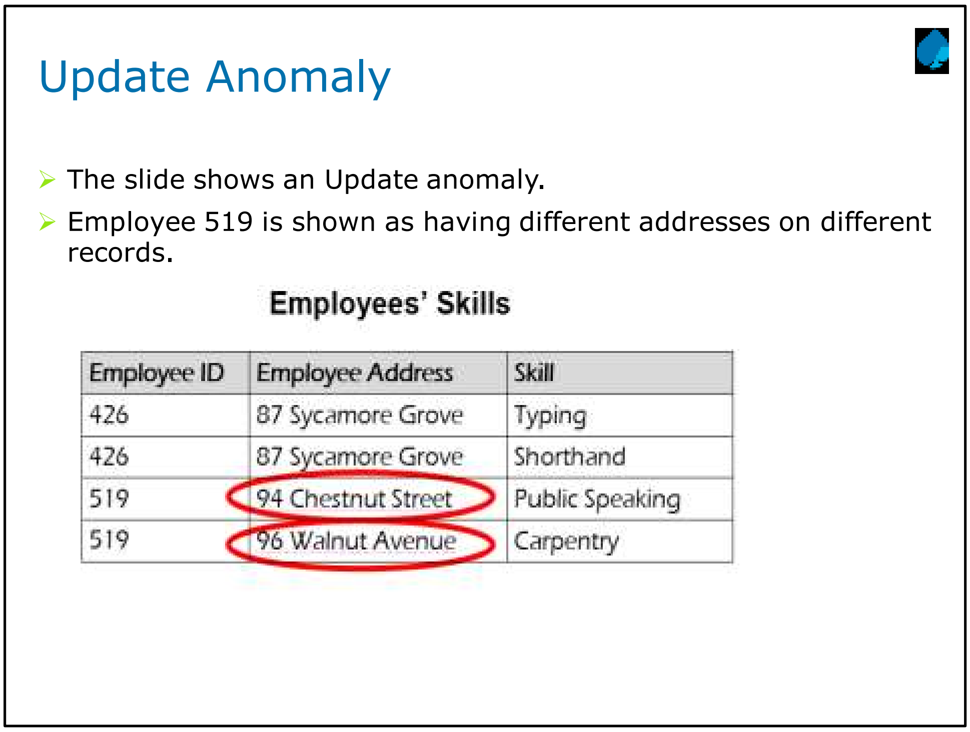


## Problems with Un-normalized Database

- An un-normalized database can suffer from various “logical inconsistencies” and “data operation anomalies”.
- Data Operation anomalies can be classified as:
  - Update anomaly
  - Insertion anomaly
  - Deletion anomaly

The purpose of Normalization is to ensure:
there is no duplication of data
there is no unnecessary data stored within a Database
all the attributes (columns) of a table are completely dependent on the primary key of a table only, and not on any other attribute.

**Instructor Notes:**



## Update Anomaly

- The slide shows an Update anomaly.
- Employee 519 is shown as having different addresses on different records.

**Employees' Skills**

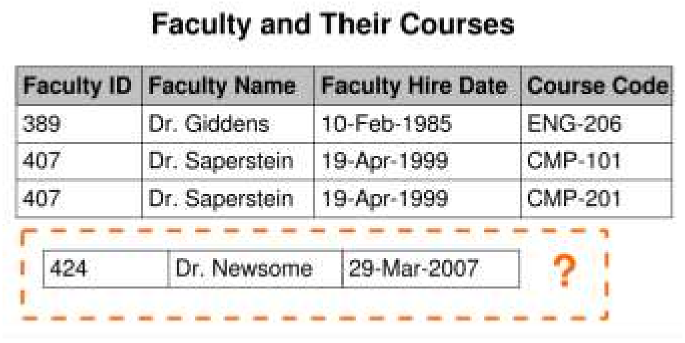
| Employee ID | Employee Address | Skill |
|---|---|---|
| 426 | 87 Sycamore Grove | Typing |
| 426 | 87 Sycamore Grove | Shorthand |
| 519 | 94 Chestnut Street | Public Speaking |
| 519 | 96 Walnut Avenue | Carpentry |

Update anomaly:
The same information can be expressed on multiple records. Therefore updates to the table may result in logical inconsistencies.
For example: Each record in an "Employees' Skills" table might contain an Employee ID, Employee Address, and Skill. Thus a change of address for a particular employee will potentially need to be applied to multiple records (one for each of his skills). If the update is not carried through successfully — that is, if the employee's address is updated on some records but not on others — then the table is left in an inconsistent state. Specifically, the table provides conflicting answers to the question of what is the address of a particular employee. This phenomenon is known as an "Update anomaly".
The slide shows an Update anomaly. Employee 519 is shown as having different addresses on different records.

**Instructor Notes:**

## Insertion Anomaly

- The slide shows an example of an Insertion anomaly.
- Until the new faculty member is assigned to teach at least one course, the details cannot be recorded.



**Faculty and Their Courses**

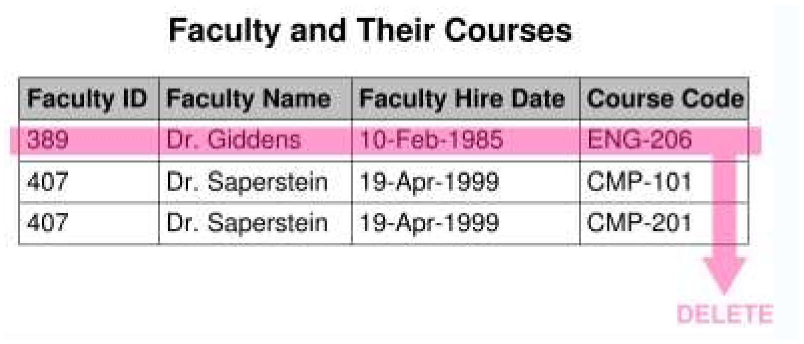
| Faculty ID | Faculty Name | Faculty Hire Date | Course Code |
|---|---|---|---|
| 389 | Dr. Giddens | 10-Feb-1985 | ENG-206 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-101 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-201 |
| 424 | Dr. Newsome | 29-Mar-2007 | ? |

Insertion anomaly:
There are circumstances in which certain facts cannot be recorded at all. For example: Each record in "Faculty and Their Courses" table might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code — thus we can record the details of any faculty member who teaches at least one course. However, we cannot record the details of a newly-hired faculty member who has not yet been assigned to teach any courses. This phenomenon is known as an "Insertion anomaly".
The slide shows an Insertion anomaly. Until the new faculty member is assigned to teach at least one course, his details cannot be recorded.

**Instructor Notes:**

## Deletion Anomaly

- The slide shows an example of a Deletion anomaly.
- All information about Dr. Giddens is lost when he temporarily ceases to be assigned to any course.



**Faculty and Their Courses**

| Faculty ID | Faculty Name | Faculty Hire Date | Course Code |
|---|---|---|---|
| 389 | Dr. Giddens | 10-Feb-1985 | ENG-206 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-101 |
| 407 | Dr. Saperstein | 19-Apr-1999 | CMP-201 |

**<u>Deletion Anomaly</u>**:

- There are circumstances in which the deletion of data representing certain facts necessitates the deletion of data representing completely different facts. The “Faculty and Their Courses” table described in the previous example suffers from this type of anomaly. If a faculty member temporarily ceases to be assigned to any course, we must delete the last of the records on which that faculty member is displayed. This phenomenon is known as a “Deletion anomaly”.
- The slide shows a Deletion anomaly. All information about Dr. Giddens is lost when he temporarily ceases to be assigned to any courses.

**Instructor Notes:**

## Background to Normalization: Functional dependency

- Let us suppose we have two columns A and B, such that, for given value of column A there is a single value of column B associated with it. Then column B is said to be functionally dependent on column A.
- Column B is functionally dependent on column A is equivalent to saying that column A determines(identifies) column B.
  - The same can be notified as A → B
- Eg: Employee Address has a functional dependency on Employee ID because a particular Employee ID value corresponds to one and only one Employee Address value.
  - EmpID → EmpAddress

Note that the reverse need not be true. Several employees can live at the same address and therefore one Employee Address value can correspond to more than one Employee ID. Employee ID is therefore not functionally dependent on Employee Address.

An attribute may be functionally dependent either on a single attribute or on a combination of attributes. It is not possible to determine the extent to which a design is normalized without understanding what functional dependencies apply to the attributes within its tables. Understanding this concept, in turn, requires knowledge of the problem domain.

**Instructor Notes:**

## Background to Normalization: Functional dependency

<u>**Background to Normalization (contd.)**</u>**:**

**Three Types of Functional Dependencies**

- **Full Dependency:** In a relation, the attribute(s) B is fully functional dependent on A, if B is functionally dependent on A but not on any proper subset of A.
- **Partial Dependency:** It is a relation, where there is some attribute that can be removed from A and the dependency still holds.
  **For example:** Staff_No, Sname → Branch_No
- **Transitive Dependency:** In a relation, if attribute(s) A→B and B→C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C)
  **For example:** Staff_No → Branch_No, and Branch_No → BAddress
  **For example:** {Employee Address} has a functional dependency on {Employee ID, Skill}, but not a full functional dependency, because it is also dependent on {Employee ID}.

➢ **Trivial functional dependency:** A trivial functional dependency is a functional dependency of an attribute on a superset of itself. {Employee ID, Employee Address} → {Employee Address} is trivial, as is {Employee Address} → {Employee Address}.

➢ **Transitive dependency:** A transitive dependency is an indirect functional dependency, one in which X → Z only by virtue of X→Y and Y→Z.

➢ **Multivalued dependency:** A multi-valued dependency is a constraint according to which the presence of certain rows in a table implies the presence of certain other rows. Refer the Multivalued Dependency article for a rigorous definition.

➢ **Join dependency:** A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T.

➢ **Superkey:** A super key is an attribute or set of attributes that uniquely identifies rows within a table. In other words, two distinct rows are always guaranteed to have distinct superkeys. {Employee ID, Employee Address, Skill} will be a superkey for the “Employees’ Skills” table; {Employee ID, Skill} will also be a superkey.

➢ **Candidate key:** A candidate key is a minimal superkey, that is a superkey for which we can say that no proper subset of it is also a superkey. {Employee Id, Skill} will be a candidate key for the “Employees’ Skills” table.

➢ **Non-prime attribute:** A non-prime attribute is an attribute that does not occur in any candidate key. Employee Address will be a non-prime attribute in the “Employees’ Skills” table.

➢ **Primary key:** Most DBMSs require a table to be defined as having a single unique key, rather than a number of possible unique keys. A primary key is a key which the database designer has designated for this purpose.

**Instructor Notes:**

## Normal Forms

- Normal Form is abbreviated as NF.
- Normal Form provides criteria for determining a table’s degree of vulnerability to logical inconsistencies and anomalies.
- The higher the NF applicable to a table, the less vulnerable it is to inconsistencies and anomalies.

Normal Forms:
The Normal Forms (abbrev. NF) of relational database theory provide criteria for determining a table’s degree of vulnerability to logical inconsistencies and anomalies. The higher the Normal Form applicable to a table, the less vulnerable it is to inconsistencies and anomalies.
Each table has a “highest normal form” (HNF). By definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF. Again by definition, a table fails to meet the requirements of any Normal Form higher than its HNF.
The Normal Forms are applicable to individual tables. To say that an entire database is in normal form n implies that all its tables are in normal form n.
Newcomers to database design sometimes suppose that normalization proceeds in an iterative fashion, i.e. a 1NF design are first normalized to 2NF, then to 3NF, and so on. This is not an accurate description of how normalization typically works. A sensibly designed table is likely to be in 3NF on the first attempt. Furthermore, if it is 3NF, it is overwhelmingly likely to have an HNF of 5NF. Achieving the “higher” normal forms (above 3NF) does not usually require an extra expenditure of effort on the part of the designer, because 3NF tables usually need no modification to meet the requirements of these higher normal forms.
Edgar F. Codd originally defined the first three normal forms (1NF, 2NF, and 3NF). These normal forms have been summarized as requiring that all non-key attributes be dependent on “the key, the whole key and nothing but the key”. The fourth and fifth normal forms (4NF and 5NF) deal specifically with the representation of many-to-many and one-to-many relationships among attributes. Sixth normal form (6NF) incorporates considerations relevant to temporal databases.

**Instructor Notes:**

## Case Study : Normalization

- Given an EmpProject Table structure
- Note: EmpProject is an un-normalized relation

| Proj Code | Proj Type | Proj Desc | Empno | Ename | Grade | Sal scale | Proj Join Date | Alloc Time |
|---|---|---|---|---|---|---|---|---|
| 001 | APP | LNG | 46 | JONES | A1 | 5 | 12/1/1998 | 24 |
| 001 | APP | LNG | 92 | SMITH | A2 | 4 | 2/1/1999 | 24 |
| 001 | APP | LNG | 96 | BLACK | B1 | 9 | 2/1/1999 | 18 |
| 004 | MAI | SHO | 72 | JACK | A2 | 4 | 2/4/1999 | 6 |
| 004 | MAI | SHO | 92 | SMITH | A2 | 4 | 5/5/1999 | 6 |

**Instructor Notes:**

## Normalization – First Normal Form (1NF)

- A relation is said to be in "First Normal Form" (1NF) if and only if all its attributes assume only atomic values and there are no repeating groups.
- 1NF requires that the values in each column of a table are "atomic".
  - "Atomic" implies that there are no sets of values within a column.
- To Transform un-normalized relation in 1 NF:
  - Eliminate repeating groups.
  - Make a separate table for each set of related attributes, and give each table a primary key.

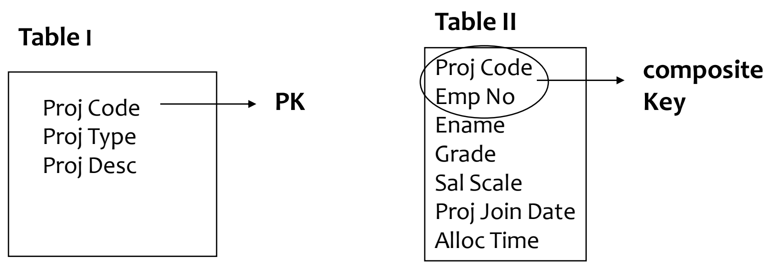
First Normal Form (1NF):
A table is in first normal form (1NF) if and only if it faithfully represents a relation. Given that database tables embody a relation-like form, the defining characteristic of one in First Normal form is that it does not allow duplicate rows or nulls. To put it simply, a table with a unique key (which, by definition, prevents duplicate rows) and without any nullable columns is in 1NF.
Note:

The restriction on nullable columns as a 1NF requirement, as espoused by Chris Date, et. al., is controversial. This particular requirement for 1NF is a direct contradiction to Dr. Codd's vision of the relational database, in which he stated that "null values" must be supported in a fully relational DBMS in order to represent "missing information and inapplicable information in a systematic way, independent of data type".
By redefining 1NF to exclude nullable columns in 1NF, no level of normalization can ever be achieved unless all nullable columns are completely eliminated from the entire database. This is in line with Date's and Darwen's vision of the perfect relational database, however it can introduce additional complexities in SQL databases to the point of impracticality.

**Instructor Notes:**

## Normalization – First Normal Form (1NF)

- Rule:
  - A relation is in the first normal form if the intersection of any column and row contains only one value
  - Identify any suitable primary key
  - Remove repeating groups to simplify relationship that may lead to multiple table design



**Instructor Notes:**

## Normalization – Second Normal Form (2NF)

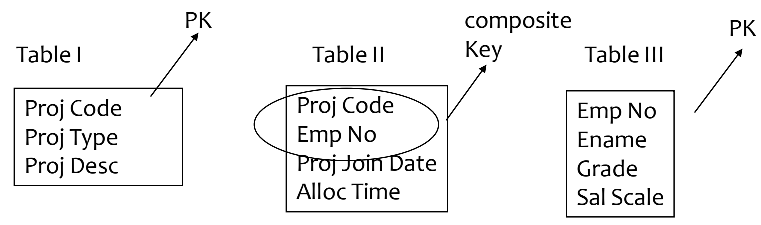
- A relation is in Second Normal Form (2NF) if and only if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of the relation.
- 2NF is based on the concept of Full Functional Dependency.

Second Normal Form (2NF):
Where the First Normal Form deals with atomicity of data, the Second Normal Form (or 2NF) deals with relationships between composite key columns and non-key columns. As stated earlier, the normal forms are progressive, so to achieve Second Normal form, your tables must already be in First Normal Form.
In case of the Second Normal form (or 2NF), any non-key columns must depend on the entire primary key. In the case of a composite primary key, this means that a non-key column cannot depend on only part of the composite key.

**Instructor Notes:**

## Normalization – Second Normal Form (2NF)

- Transform 1 NF to 2 NF :
- If a non-key attribute depends on only part of a composite key,
- remove it to a separate table.
  - For every relation with a single data item making up the primary key, this rule should "always be true".
  - For those with the composite key, examine every column and ask whether its value depends on the whole of the composite key or just some part of it.
  - Remove those that depend only on part of the key to a new relation with that part as the primary key.



Rules for 2NF:
The Second Normal Form involves the idea of Functional Dependency.
Functional Dependency: Attribute B has a functional dependency on attribute A, if for each value of attribute A, there is exactly one value of attribute B.

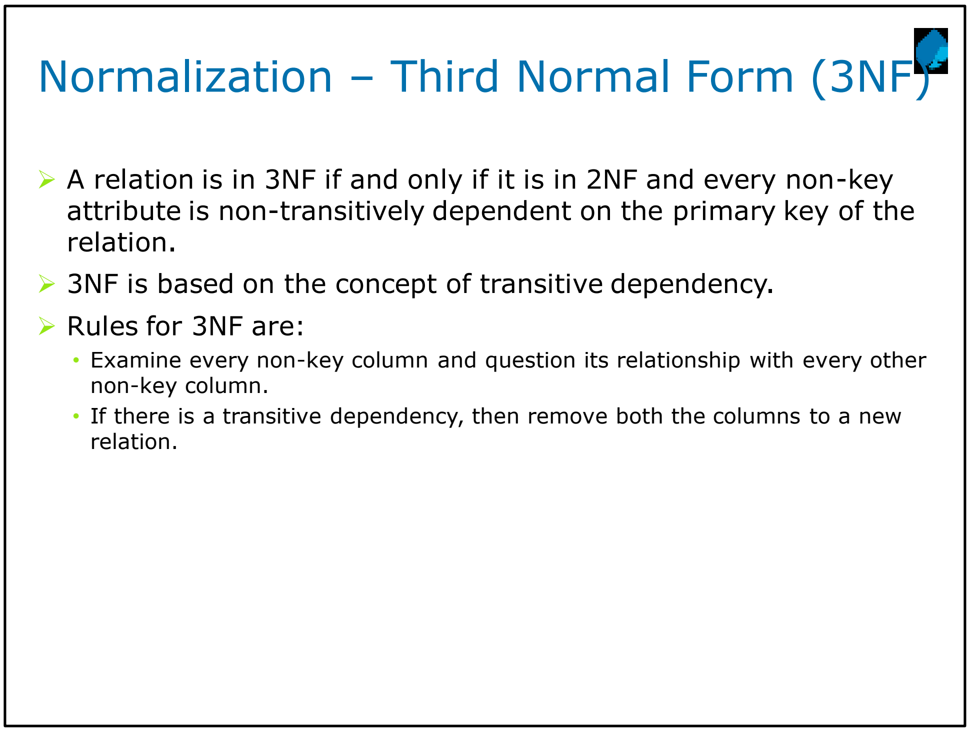
For example: In an Emp table, Employee Address has a functional dependency on Employee ID because a particular Employee ID value corresponds to one and only one Employee Address value.
Note that the reverse need not be true.

For example: Several employees can live at the same address, and therefore one Employee Address value can correspond to more than one Employee ID. Employee ID is therefore not functionally dependent on Employee Address.

An attribute may be functionally dependent either on a single attribute or on a combination of attributes. It is not possible to determine "the extent to which a design is normalized" without understanding the "functional dependencies" that apply to the attributes within its tables. Understanding this concept, in turn, requires knowledge of the problem domain.

For example: An Employer may require certain employees to split their time between two locations, such as New York City and London, and therefore want to allow Employees to have more than one Employee Address. In this case, Employee Address will no longer be functionally dependent on Employee ID.

**Instructor Notes:**



## Normalization – Third Normal Form (3NF)

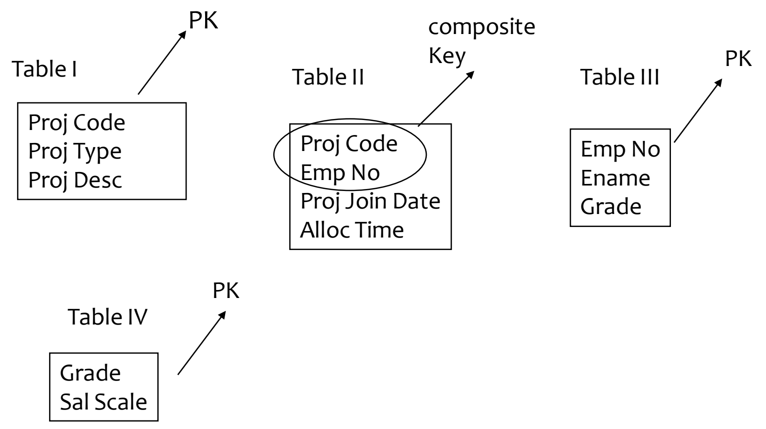
- A relation is in 3NF if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key of the relation.
- 3NF is based on the concept of transitive dependency.
- Rules for 3NF are:
  - Examine every non-key column and question its relationship with every other non-key column.
  - If there is a transitive dependency, then remove both the columns to a new relation.

Third Normal Form (3NF):
Third Normal Form (3NF) requires that all columns depend directly on the primary key. Tables violate the Third Normal Form when one column depends on another column, which in turn depends on the primary key (a transitive dependency).
One way to identify transitive dependencies is to look at your table, and see if any columns require updating if another column in the table is updated. If such a column exists, it probably violates 3NF.
Third Normal Form: Eliminate Data Not Dependent On Key
Criteria for 3NF:
The table must be in 2NF.
Every non-prime attribute of the table must be non-transitively dependent on each candidate key. A violation of 3NF will mean that at least one non-prime attribute is only indirectly dependent (transitively dependent) on a candidate key.
For example:

> Consider a “Departments” table whose attributes are Department ID, Department Name, Manager ID, and Manager Hire Date. And suppose that each manager can manage one or more departments. {Department ID} is a candidate key.
> Although Manager Hire Date is functionally dependent on the candidate key {Department ID}, this is only because Manager Hire Date depends on Manager ID, which in turn depends on Department ID.
> This transitive dependency means the table is not in 3NF unless the manager can be hired for more than one department and the date represents the hiring date of that manager only for this department.

**Instructor Notes:**

## Normalization – Third Normal Form (3NF)

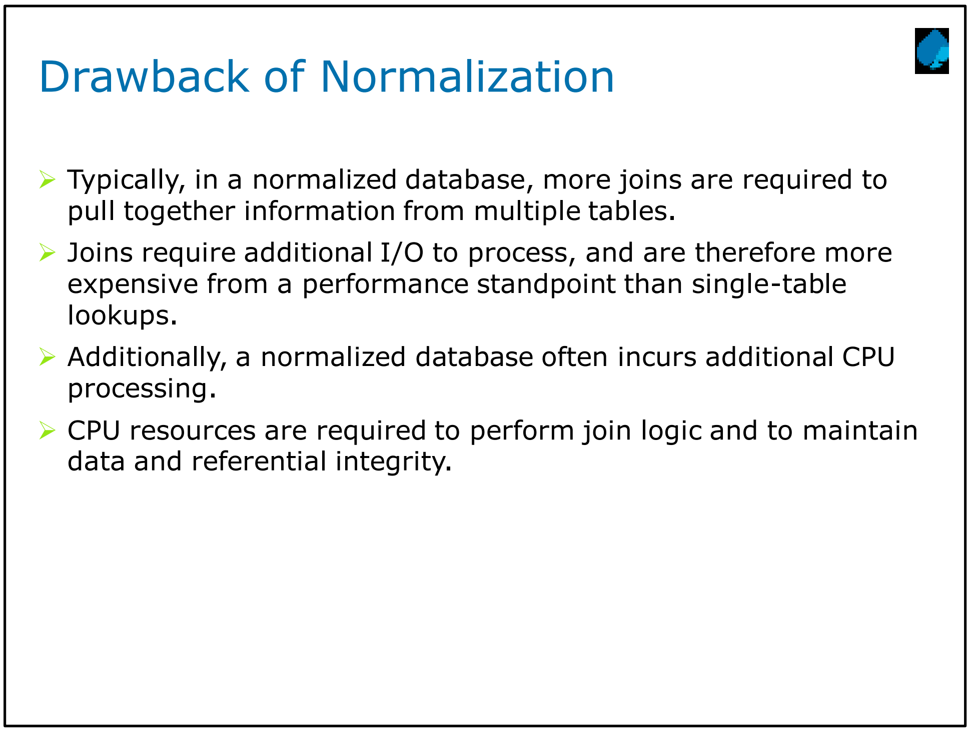


**Instructor Notes:**

## Normalization Summarization

- 1 NF- Ensure all values are atomic and Eliminate Repeating Groups
- 2 NF- Eliminate Partial Dependencies
- 3 NF- Eliminate Transitive Dependencies

**Instructor Notes:**



## Drawback of Normalization

- Typically, in a normalized database, more joins are required to pull together information from multiple tables.
- Joins require additional I/O to process, and are therefore more expensive from a performance standpoint than single-table lookups.
- Additionally, a normalized database often incurs additional CPU processing.
- CPU resources are required to perform join logic and to maintain data and referential integrity.

**Instructor Notes:**

## CODD's Rules

- 0. Relational Database Management.
- 1. Information Representation
- 2. Logical Accessibility
- 3. Representation of Null Values
- 4. Catalog Facilities
- 5. Data Languages
- 6. View Updatability
- 7. Update and Delete
- 8. Physical Data Independence
- 9. Logical Data Independence
- 10. Integrity Constraints
- 11. Database Distribution
- 12. Non-subversion

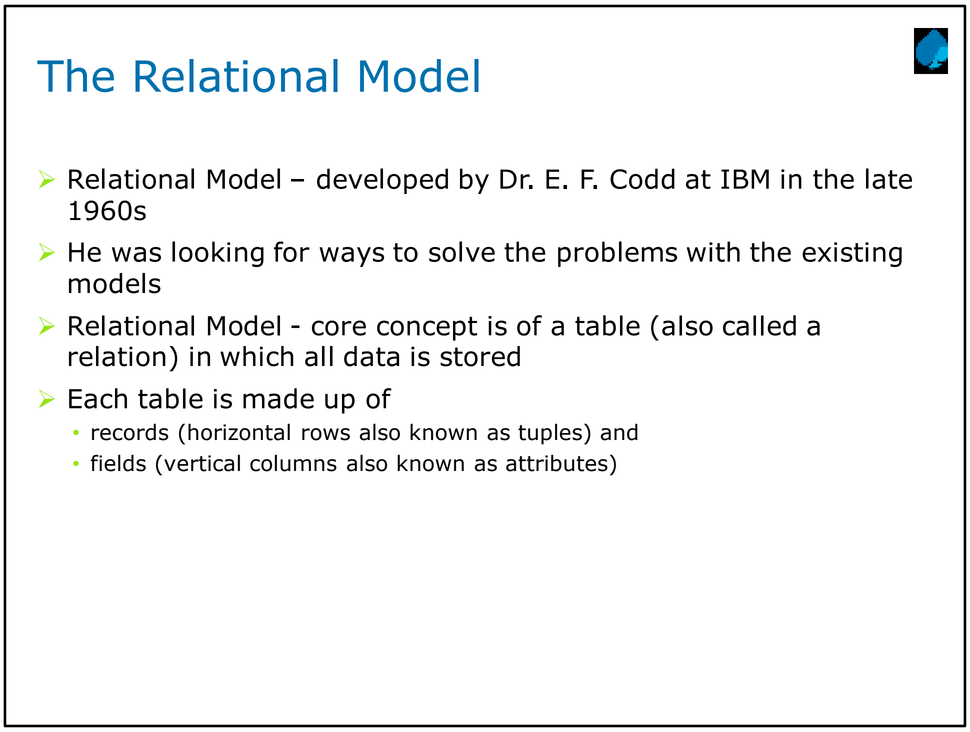
CODD'S Rules For "FULLY" Functional System
Rule 0 : Any truly relational database must be manageable entirely through its relational capability.
Rule 1 : Information rule : All information is explicitly and logically represented in exactly one way by data values in tables. It means, that if an item of data doesn't reside somewhere in a table in the database, then it doesn't exists. This necessitates the provision of an active data dictionary that is itself relational.
Rule 2 : The rule of guaranteed access : Every item of data must be logically addressable by restoring to a combination of the table name and the column name.This rule says that at the intersection of a column and a row, you 'll necessarily find one value of data item (or NULL).
Rule 3 : The systematic treatment of null values : NULL values are supported in the representation of missing and inapplicable information. This support for NULL values must be consistent throughout the RDBMS and independent of data type.
Rule 4 : The database description rule : A description of a database is held and maintained using the same logical structure used to define the data, thus allowing users with appropriate authority to query such information in the same ways and using the same language as they would any other data in the database (It refers to DD).There must be a DD within the RDBMS that is constructed of tables/views that can be examined using SQL. This rule states that a DD is mandatory and must be a table. Every data must be in table and every DD is a table . So SQL statements can work on a table as well as DD.

**Instructor Notes:**

## CODD’s Rules

**Rule 5 : The comprehensive sub language rule** : There must be at least one language whose statements can be expressed as character strings confirming to some well defined syntax. i.e Comprehensive in supporting the following :
a. Data Definition (DDL)
b. View Definition (DDL)
c. Data Manipulation (DML)
d. Integrity constraints (DDL)
e. Authorization (DCL)
f. Transactional boundaries (DML)
This means that the RDBMS must be completely manageable through its own direct of SQL.
**Rule 6 : The view updating rule** : All views that can be updated in theory can also be updated by the system.
**Rule 7 : The insert and the update rule :** The capability of handling a base relation or in-fact a derived relation as a single operand must hold good for all retrieve, update, delete and insert activity. It means that major DML commands namely select, Update, Delete & Insert must be available & operational on set of rows in a relational, i.e you should use the same commands for modifications in all tables & views.
**Rule 8 : The physical independence rule :** User access to the database or a terminal monitors or application programs must remain logically consistent whenever changes to the storage representation or access methods to the data are changed.
If an index is built or destroyed by the DBA on a table, any user should retrieve the same data from the table. Applications must be limited to interfacing with the logical layer to enable the enforcement of this rule.
**Rule 9 : Logical data Independence :** Application programs & terminal activities must remain logically unimpaired whenever that are theoretically permitted are made the base table. This rule allows many types of database design changes to be made dynamically w/o users being aware of it.
**Rule 10 : Integrity Independence rule :** All Integrity constraints defined for a database must be definable in the language referred to in rule 5. The following integrity rules should apply to every relational database:
a. No component of a primary key can have missing values. This is the basic rule for entity integrity.
b. For each distinct foreign key in value there must be a matching primary key in the same domain.
**Rule 11 : Distribution rule :** A RDBMS must have distribution independence.
Applications running on a non-distributed database must remain logically unimpaired if that data should then become distributed. In the context of a distributed relational database.
**Rule 12 : No sub - version rule :** If an RDBMS supports a low level language that permits row at a time processing then this language must not be able to bypass any integrity rules or constraints defined in the high level relational language.

**Instructor Notes:**



## The Relational Model

- Relational Model – developed by Dr. E. F. Codd at IBM in the late 1960s
- He was looking for ways to solve the problems with the existing models
- Relational Model - core concept is of a table (also called a relation) in which all data is stored
- Each table is made up of
  - records (horizontal rows also known as tuples) and
  - fields (vertical columns also known as attributes)

**Instructor Notes:**

Make it clear that TABLE itself means RELATION

## The Relational Model

- Because of lack of linkages relational model is easier to understand and implement.

**Student Table**

| Scode | Sname |
|---|---|
| S1 | A |
| S2 | B |

**Course Table**

| Ccode | Cname |
|---|---|
| C1 | Physics |
| C2 | Chemistry |
| C3 | Maths |
| C4 | Biology |

**Marks Table**

| Ccode | Scode | Marks |
|---|---|---|
| C1 | S1 | 65 |
| C2 | S1 | 78 |
| C3 | S1 | 83 |
| C4 | S1 | 85 |
| C3 | S2 | 83 |
| C4 | S2 | 85 |

**Instructor Notes:**

## Possibilities in Relational Model

- INSERT
  - Inserting a course record or student record poses no problems because tables are separate
- DELETE
  - Deleting any record affects only a particular table
- UPDATE
  - Update can be done only to a particular table

**Instructor Notes:**

## The Relational DBMS

- **Examples of Relational Tables**

**DEPT table**

| **Deptno** | **Dname** | **Loc** |
|---|---|---|
| 10 | Accounting | New York |
| 20 | Research | Dallas |
| 30 | Sales | Chicago |
| 40 | Operations | Boston |

**ß 'row' or 'tuple'**

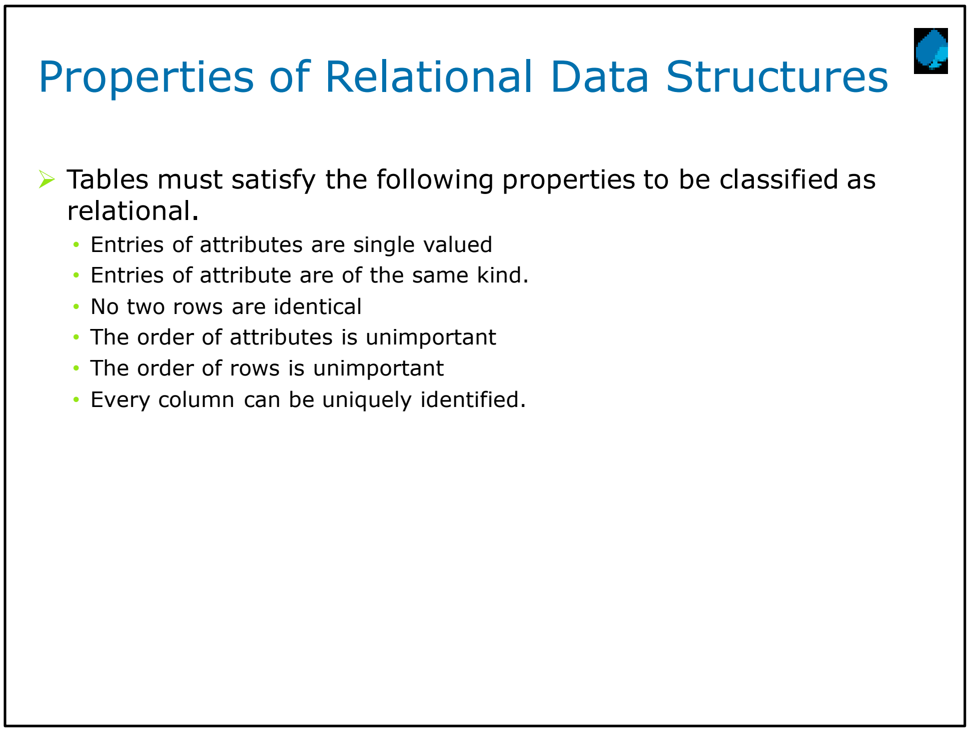
á

**'column' or 'attribute'**

**EMPLOYEE table**

| **Empno** | **Empname** | **Job** | **Mgr** | **Deptno** |
|---|---|---|---|---|
| 7369 | Smith | Clerk | 7902 | 20 |
| 7499 | Allen | Salesman | 7839 | 30 |
| 7566 | Jones | Manager | 7839 | 20 |
| 7839 | King | President | | 10 |
| 7902 | Ford | Analyst | 7566 | 20 |

**Instructor Notes:**



## Properties of Relational Data Structures

- Tables must satisfy the following properties to be classified as relational.
  - Entries of attributes are single valued
  - Entries of attribute are of the same kind.
  - No two rows are identical
  - The order of attributes is unimportant
  - The order of rows is unimportant
  - Every column can be uniquely identified.

Relational tables have six properties, which must be satisfied for any table to be classified as relational. These are :

**1. *Entries of attributes are single valued***
Entry in every row and column position in a table must be single valued. This means columns do not contain repeating groups

**2. *Entries of attribute are of the same kind***
Entries in a column must be of same kind. A column supposed to store sal of a employee should not store comm.

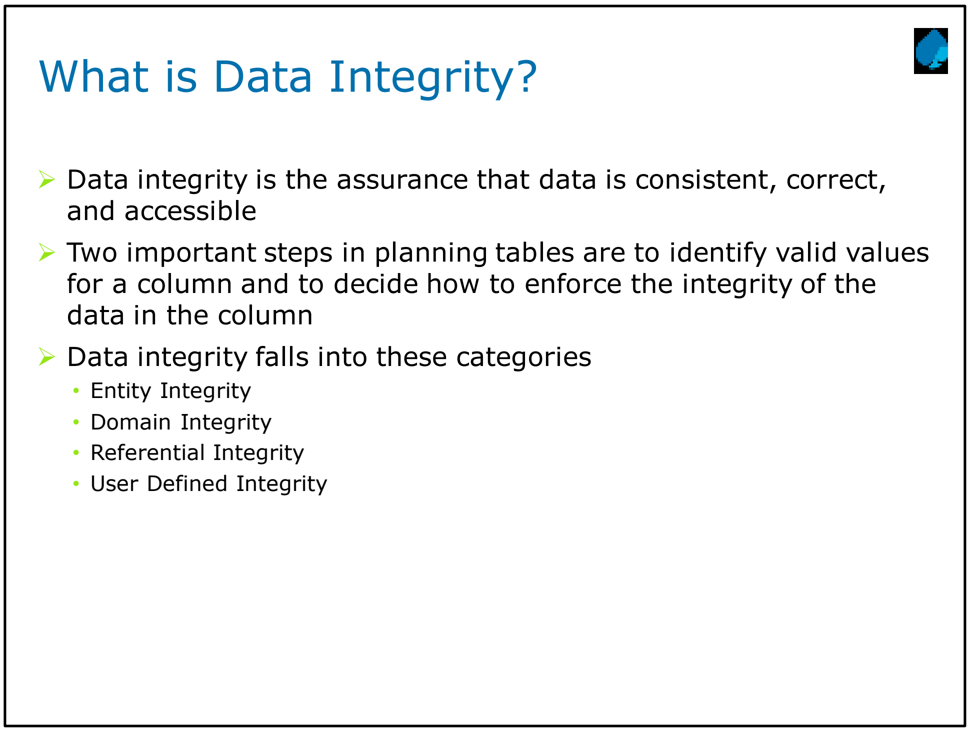
**3. *No two rows are identical***
Each row should be unique, this uniqueness is ensured by the values in a specific set of columns called the primary key.

**4. *The order of attributes is unimportant***
There is no significance attached to order in which columns are stored in the table. A user can retrieve columns in any order.

**5. *The order of rows is unimportant***
There is no significance attached to the order in which rows are stored in the table. A user can retrieve rows in any order.

**6. *Every column can be uniquely identified.***
Each column is identified by its name and not its position. A column name should be unique in the table.

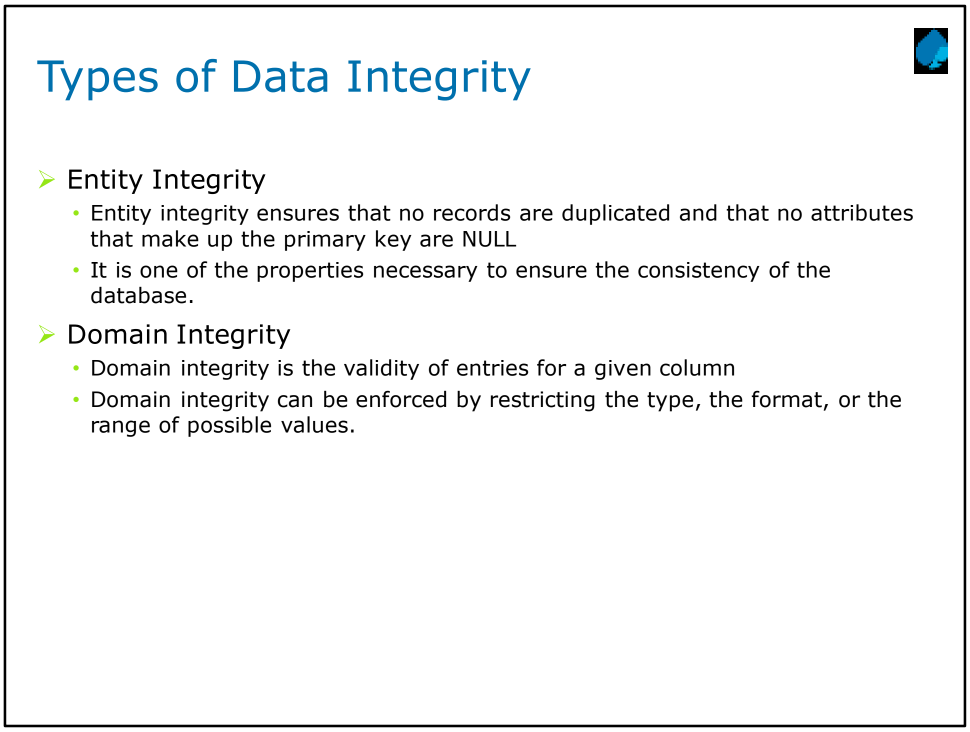
**Instructor Notes:**



## What is Data Integrity?

- Data integrity is the assurance that data is consistent, correct, and accessible
- Two important steps in planning tables are to identify valid values for a column and to decide how to enforce the integrity of the data in the column
- Data integrity falls into these categories
  - Entity Integrity
  - Domain Integrity
  - Referential Integrity
  - User Defined Integrity

**Instructor Notes:**



## Types of Data Integrity

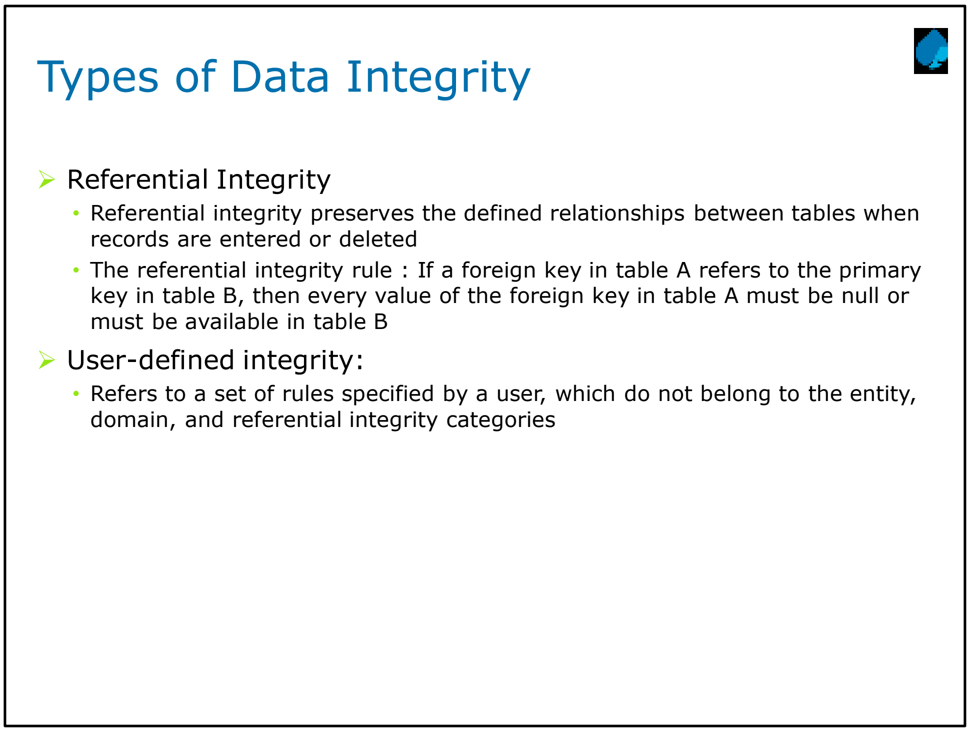
- Entity Integrity
  - Entity integrity ensures that no records are duplicated and that no attributes that make up the primary key are NULL
  - It is one of the properties necessary to ensure the consistency of the database.
- Domain Integrity
  - Domain integrity is the validity of entries for a given column
  - Domain integrity can be enforced by restricting the type, the format, or the range of possible values.

**Entity integrity** defines a row as a unique entity for a particular table. Entity integrity enforces the integrity of the identifier column(s) or the primary key of a table (through indexes, UNIQUE constraints, PRIMARY KEY constraints, or IDENTITY properties).

**Domain Integrity**

You can enforce domain integrity by restricting the type (through data types), the format (through CHECK constraints and rules), or the range of possible values (through FOREIGN KEY constraints, CHECK constraints, DEFAULT definitions, NOT NULL definitions, and rules).

**Instructor Notes:**

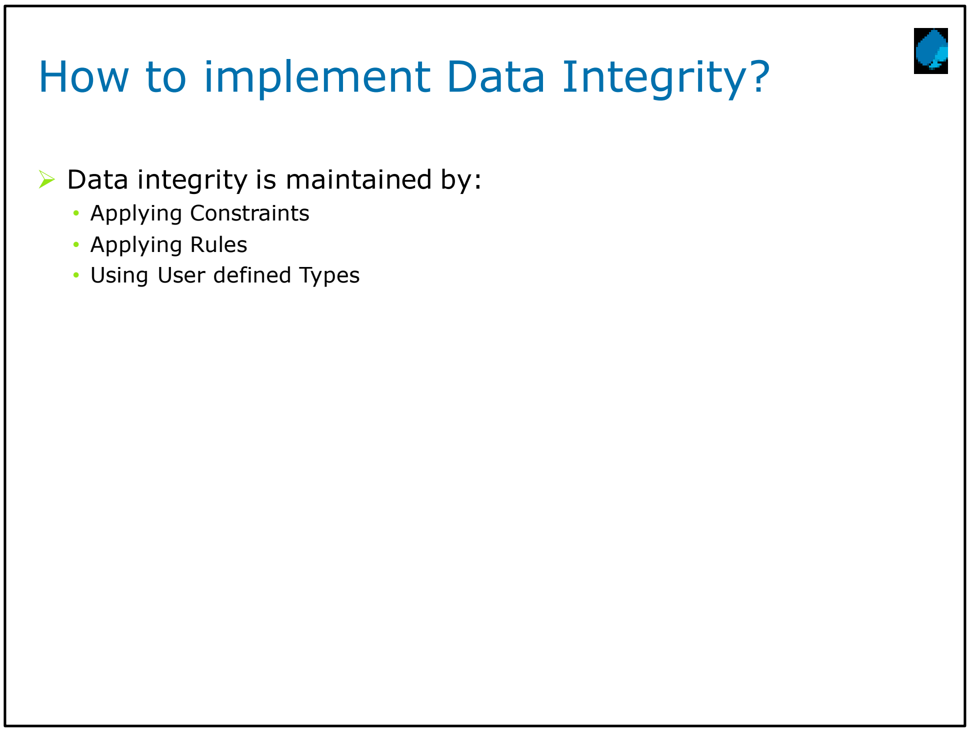


## Types of Data Integrity

- Referential Integrity
  - Referential integrity preserves the defined relationships between tables when records are entered or deleted
  - The referential integrity rule : If a foreign key in table A refers to the primary key in table B, then every value of the foreign key in table A must be null or must be available in table B
- User-defined integrity:
  - Refers to a set of rules specified by a user, which do not belong to the entity, domain, and referential integrity categories

In Microsoft SQL Server, referential integrity is based on relationships between foreign keys and primary keys or between foreign keys and unique keys (through FOREIGN KEY and CHECK constraints). Referential integrity ensures that key values are consistent across tables. Such consistency requires that there be no references to nonexistent values and that if a key value changes, all references to it change consistently throughout the database.

**Instructor Notes:**



## How to implement Data Integrity?

- Data integrity is maintained by:
  - Applying Constraints
  - Applying Rules
  - Using User defined Types

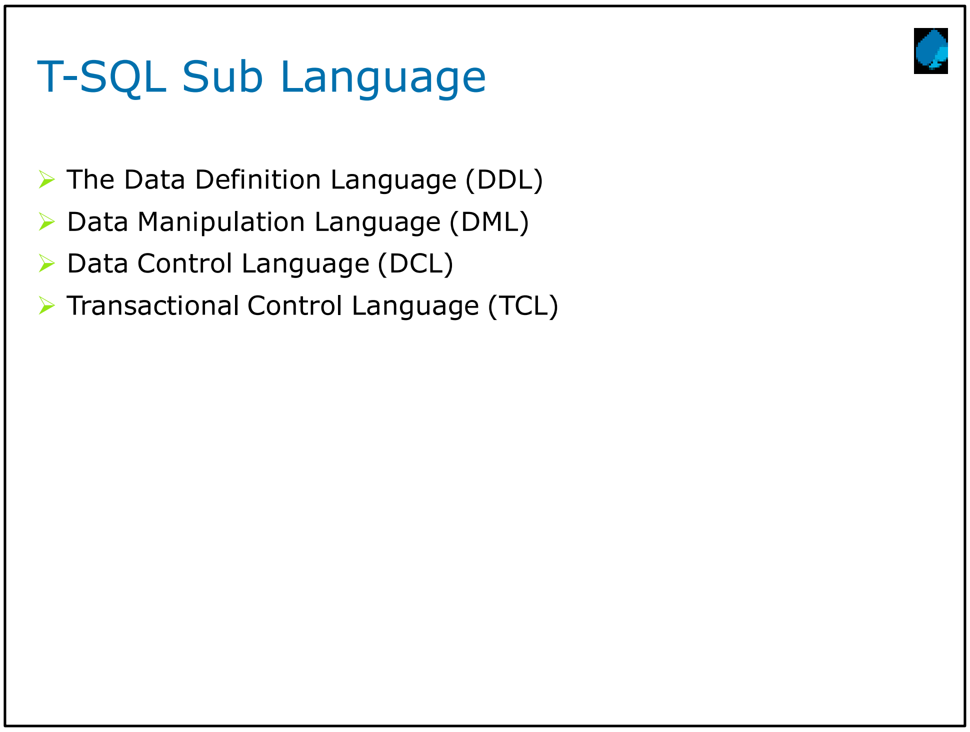
**Instructor Notes:**

Different database vendors will have different flavors of SQL . T-SQL from Microsoft and Sybase Inc , SQLPLUS from Oracle etc .
Also talk about ANSI-SQL

## T-SQL Languages

- SQL is a special-purpose language used to define, access, and manipulate data
- SQL is nonprocedural language, it only describes the necessary components like tables and desired results without specifying exactly how those results should be computed
- SQL comes in many flavors
- Microsoft SQL Server makes use of Transact-SQL

**Instructor Notes:**



## T-SQL Sub Language

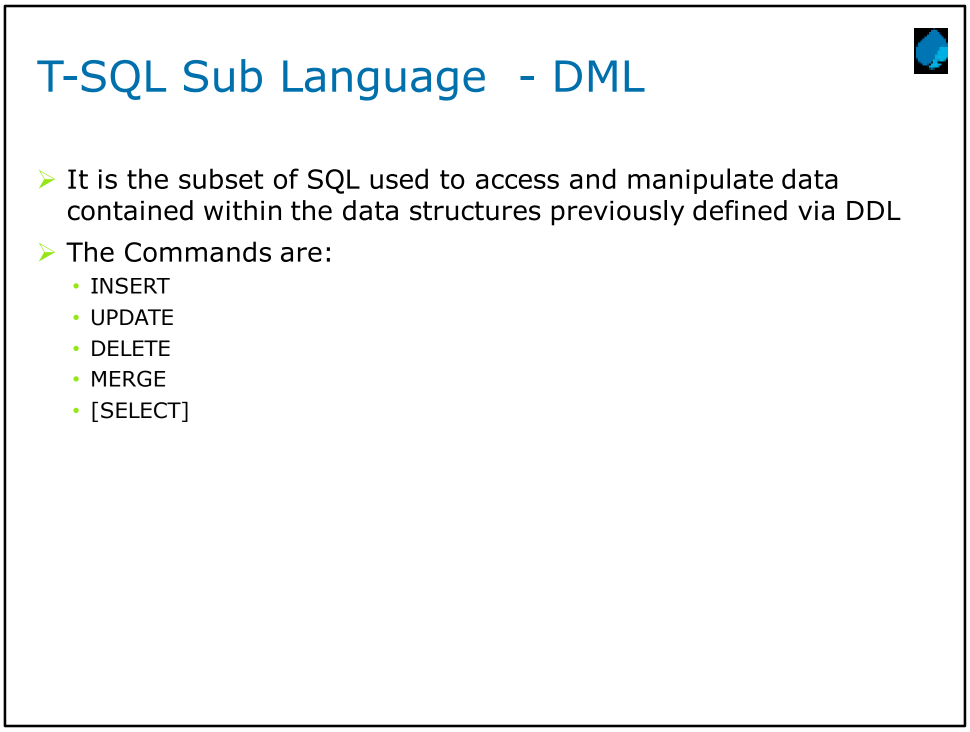
- The Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transactional Control Language (TCL)

**Instructor Notes:**

## T-SQL Sub Language - DDL

- It is the subset of SQL which contains the commands used to create and destroy databases and database objects
- DDL includes the commands for handling tasks such as creating tables, indexes, views, and constraints
- The commands are
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

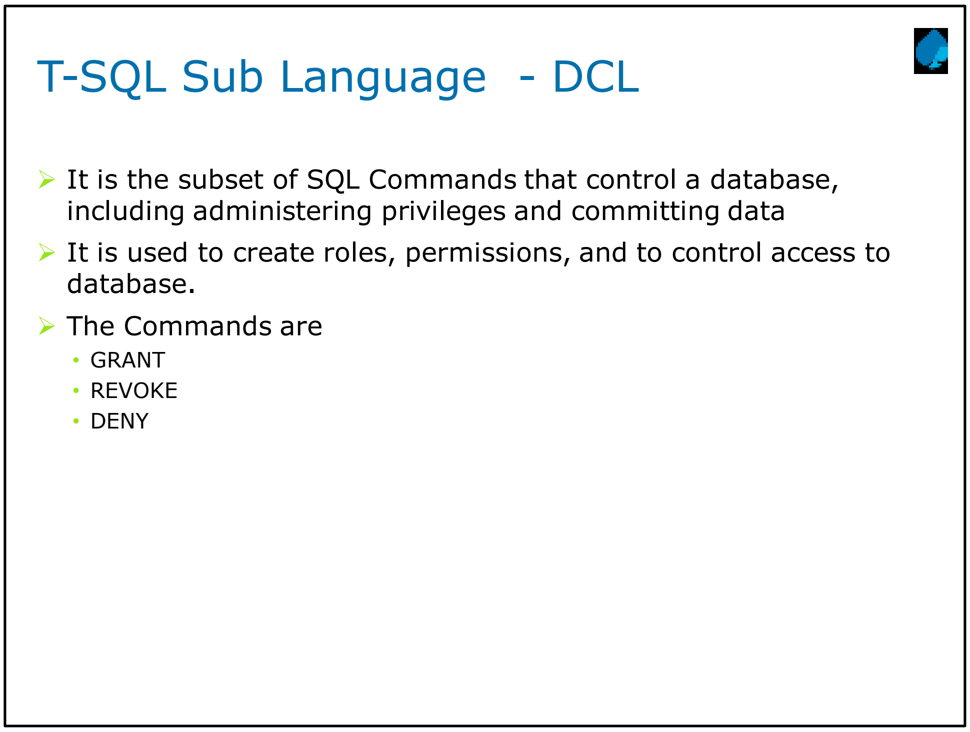
**Instructor Notes:**



## T-SQL Sub Language - DML

- It is the subset of SQL used to access and manipulate data contained within the data structures previously defined via DDL
- The Commands are:
  - INSERT
  - UPDATE
  - DELETE
  - MERGE
  - [SELECT]

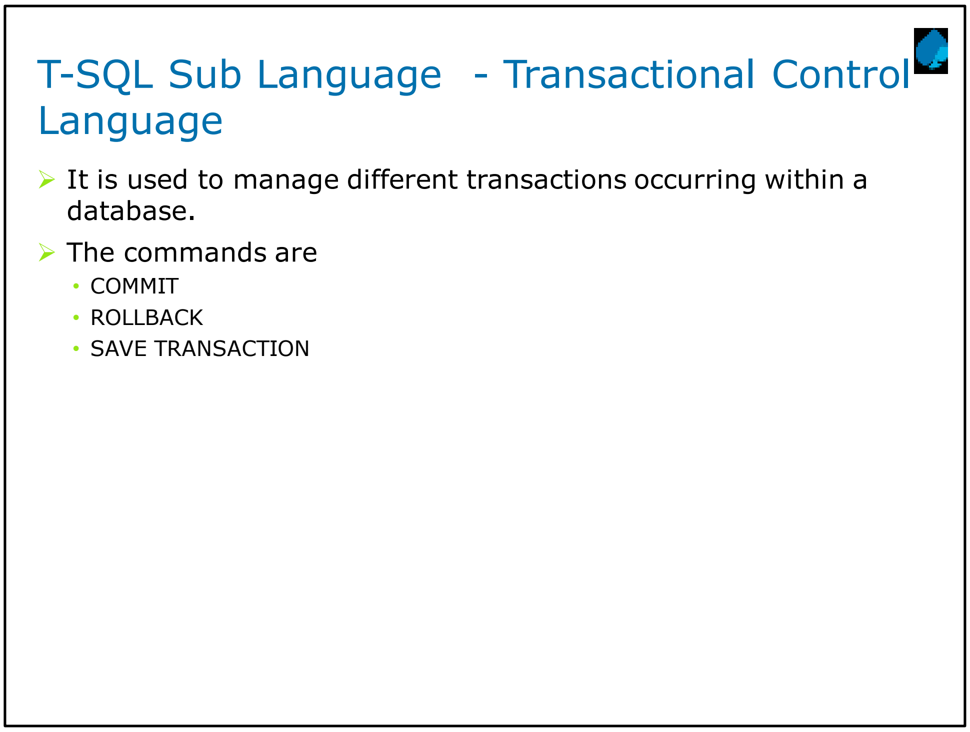
**Instructor Notes:**



## T-SQL Sub Language - DCL

- It is the subset of SQL Commands that control a database, including administering privileges and committing data
- It is used to create roles, permissions, and to control access to database.
- The Commands are
  - GRANT
  - REVOKE
  - DENY

**Instructor Notes:**



## T-SQL Sub Language - Transactional Control Language

- It is used to manage different transactions occurring within a database.
- The commands are
  - COMMIT
  - ROLLBACK
  - SAVE TRANSACTION

**Instructor Notes:**

None

## Summary

- In this lesson, you have learnt:
  - A set of inter-related data is known as database and the software that manages it is known as database management system or DBMS
  - Different data models are Hierarchical Model, Network Model and Relational Model
  - Data integrity is the assurance that data is consistent, correct, and accessible

**Instructor Notes:**

## Review Question

- Question 1: A set of structures and relationships that meet a specific need is called as a ________.
- Question 2: What are the characteristics of DBMS?
- Question 3: What are the various types on Data Integrity?
- Question 4: What are the SQL Sublanguages?