Databases

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

LEARNING TOPICS

- Database Overview
  - What is a Database?
  - Database Types
  - Database Objects
  - Tables
  - Data Types
  - Primary Key
  - Indexes
- SQL Statements
  - DDL, DML, DQL, DCL
- Database Design Basics
  - What is Normalization?
  - Normalization Benefits
  - Understanding Relationships
  - Relationship Types
  - Understanding Integrity

Database - Overview

- Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information

- Mostly data represents recordable facts. Data aids in producing information, which is based on facts

- For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks

- A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information

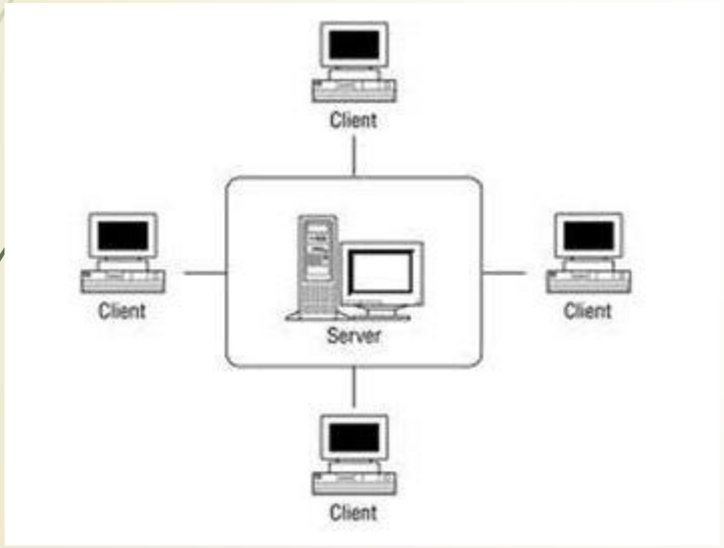
Types of Database

There are several types of database management systems. Here is a list of six common database management systems:

1. Hierarchical databases:- **IBM Information Management System (IMS) and Windows Registry**
2. Network databases:- **Raima Database Manager, TurboIMAGE**
3. Relational databases:- **MySQL, SQLite, and IBM DB2.**
4. Object-oriented databases:- **Versant Object Database, ODABA**
5. Graph databases:- **Cosmos DB, SAP HANA**
6. Document / No SQL databases:- **MongoDB, and Azure**

Relational Databases

In relational database management systems (RDBMS), the relationship between data is relational and data is stored in tabular form of columns and rows. Each column if a table represents an attribute and each row in a table represents a record. Each field in a table represents a data value.
Structured Query Language (SQL) is a the language used to query a RDBMS including inserting, updating, deleting, and searching records.
Relational databases work on each table has a key field that uniquely indicates each row, and that these key fields can be used to connect one table of data to another.



Relational databases are the most popular and widely used databases. Some of the popular DDBMS are Oracle, SQL Server, MySQL, SQLite, and IBM DB2.

The relational database has two major reasons

1.Relational databases can be used with little or no training.
2.Database entries can be modified without specify the entire body.

Properties of Relational Tables

In the relational database we have to follow some properties which are given below.

•It's Values are Atomic
•In Each Row is alone.
•Column Values are of the Same thing.
•Columns is undistinguished.
•Sequence of Rows is Insignificant.
•Each Column has a common Name.

RDBMs are the most popular databases.

Database Objects - Tables

A *database object* is any defined object in a database that is used to store or reference data. Some examples of database objects include tables, views, clusters, sequences, indexes, and synonyms. The table is this hour's focus because it is the primary and simplest form of data storage in a relational database.

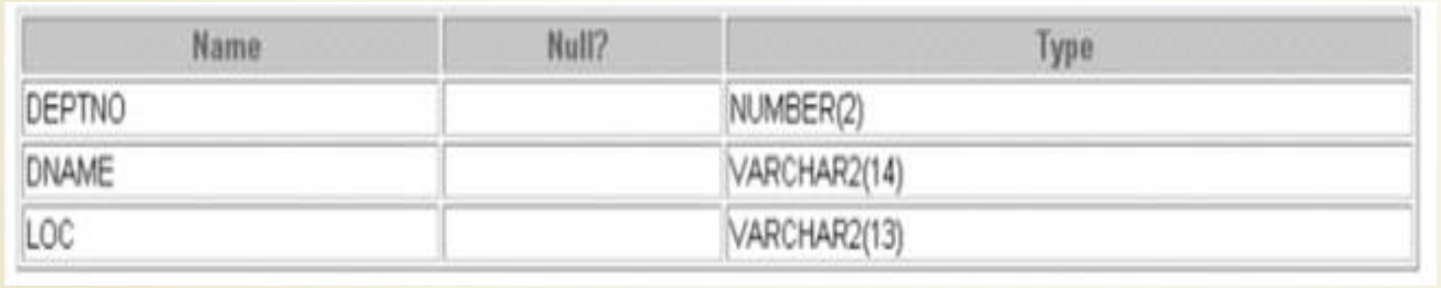
Table:-

This database object is used to create a table in database.

Syntax :
CREATE TABLE [schema.]table (column datatype [DEFAULT expr][, ...]);

Example :
CREATE TABLE dept (deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(13));

Output :
DESCRIBE dept;



Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

Database Objects - Views

View:-

This database object is used to create a view in database. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary.

Syntax :

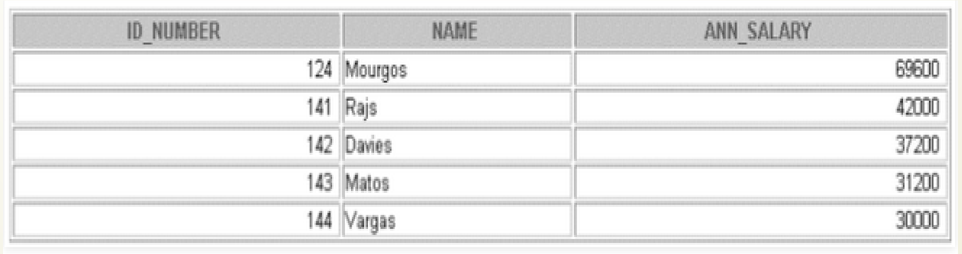
```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
[(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

Example :

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
salary*12 ANN_SALARY
FROM employees
WHERE department_id = 50;
```

Output :

```
SELECT * FROM salvu50;
```



ID_NUMBER	NAME	ANN_SALARY
124	Mourgos	69600
141	Rajs	42000
142	Davies	37200
143	Matos	31200
144	Vargas	30000

Database Objects – Indexes

Index:-

This database object is used to create a indexes in database. An Oracle server index is a schema object that can speed up the retrieval of rows by using a pointer. Indexes can be created explicitly or automatically. If you do not have an index on the column, then a full table scan occurs.
An index provides direct and fast access to rows in a table. Its purpose is to reduce the necessity of disk I/O by using an indexed path to locate data quickly. The index is used and maintained automatically by the Oracle server. Once an index is created, no direct activity is required by the user. Indexes are logically and physically independent of the table they index. This means that they can be created or dropped at any time and have no effect on the base tables or other indexes.

Syntax :

```
CREATE INDEX index
        ON table (column[, column]...);
```

Example :

```
CREATE INDEX emp_last_name_idx
        ON employees(last_name);
```

Database Objects - Synonyms

Synonym

This database object is used to create a indexes in database. It simplify access to objects by creating a synonym(another name for an object). With synonyms, you can Ease referring to a table owned by another user and shorten lengthy object names. To refer to a table owned by another user, you need to prefix the table name with the name of the user who created it followed by a period. Creating a synonym eliminates the need to qualify the object name with the schema and provides you with an alternative name for a table, view, sequence, procedure, or other objects. This method can be especially useful with lengthy object names, such as views.
In the syntax:

PUBLIC : creates a synonym accessible to all users
synonym : is the name of the synonym to be created
object : identifies the object for which the synonym is created

Syntax :

```
CREATE [PUBLIC] SYNONYM synonym FOR object;
```

Example :

```
CREATE SYNONYM d_sum FOR dept_sum_vu;
```

Database Sql Commands

Types of SQL Commands

The following sections discuss the basic categories of commands used in SQL to perform various functions. These functions include building database objects, manipulating objects, populating database tables with data, updating existing data in tables, deleting data, performing database queries, controlling database access, and overall database administration.

The main categories are:-

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- DQL (Data Query Language)
- DCL (Data Control Language)
- Data administration commands
- Transactional control commands

Database Sql Commands - DDL

Data Definition Language, DDL:- It is the part of SQL that allows a database user to create and restructure database objects, such as the creation or the deletion of a table.

Some of the most fundamental DDL commands discussed during following hours include the following:

CREATE TABLE:-

Syntax:

Following is a generic syntax for creating a MySQL table in the database.

```
CREATE TABLE table_name (column_name column_type...);
```

Example:

Here, we will create a table named "cus_tbl" in the database "customers".

```
CREATE TABLE cus_tbl(
   cus_id INT NOT NULL AUTO_INCREMENT,
   cus_firstname VARCHAR(100) NOT NULL,
   cus_surname VARCHAR(100) NOT NULL,
   PRIMARY KEY ( cus_id )
);
```

Database Sql Commands - DDL

ALTER TABLE:-

Syntax:

```
1.ALTER TABLE table_name
2. ADD new_column_name column_definition
3. [ FIRST | AFTER column_name ],
4.ADD new_column_name column_definition
5.[ FIRST | AFTER column_name ],
6.  ...
7.;
```

Example:

```
1.ALTER TABLE cus_tbl
2.ADD cus_address varchar(100) NOT NULL
3.AFTER cus_surname,
4.ADD cus_salary int(100) NOT NULL
5.AFTER cus_age ;
```

Database Sql Commands - DDL

DROP TABLE:-

Syntax:

```
1.DROP TABLE  table_name;
```

Example:

```
DROP TABLE  cus_tbl;
```

CREATE INDEX

Syntax:-

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Example:-

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

ALTER INDEX & DROP INDEX :-

```
ALTER TABLE table_name
DROP INDEX index_name;
```

CREATE VIEW:-

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = "Brazil";
```

DROP VIEW:-

```
DROP VIEW view_name;
```

Example

```
DROP VIEW [Brazil Customers];
```

Database Sql Commands - DML

Data Manipulation Language, DML is the part of SQL used to manipulate data within objects of a relational database.

There are three basic DML commands:

INSERT

The **INSERT command creates a new row** in the table to store data.

Syntax:-

```
INSERT INTO `table_name`(column_1,column_2,...) VALUES (value_1,value_2,...);
```

Example:-

```
INSERT INTO `members` (`full_names`,`gender`,`physical_address`,`contact_number`) VALUES ('Leonard
Hofstadter','Male','Woodcrest',0845738767);
```

UPDATE

```
UPDATE `table_name` SET `column_name` = `new_value' [WHERE condition];
```

Example:-

```
SELECT * FROM `members` WHERE `membership_number` = 1;
```

Database Sql Commands - DML

DELETE

syntax

DELETE FROM `table_name` [WHERE condition];

Example

DELETE FROM `movies` WHERE `movie_id` = 18;

Data Query Language (DQL) is the most concentrated focus of SQL for modern relational database users. The base command is as follows:

SQL SELECT syntax:-

SELECT [DISTINCT|ALL] { * | [fieldExpression [AS newName]} FROM tableName [alias] [WHERE condition][GROUP BY fieldName(s)] [HAVING condition] ORDER BY fieldName(s)

Some of the attributes of Select statement that are used to retrieve the data from data base:

•**SELECT** is the SQL keyword that lets the database know that you want to retrieve data.

•**[DISTINCT | ALL]** are optional keywords that can be used to fine tune the results returned from the SQL SELECT statement. If nothing is specified then ALL is assumed as the default.

•**{*| [fieldExpression [AS newName]}** at least one part must be specified, "*" selected all the fields from the specified table name, fieldExpression performs some computations on the specified fields such as adding numbers or putting together two string fields into one.

•**FROM** tableName is mandatory and must contain at least one table, multiple tables must be separated using commas or joined using the JOIN keyword.

Database Sql Commands - DML

- **WHERE** condition is optional, it can be used to specify criteria in the result set returned from the query.
- **GROUP BY** is used to put together records that have the same field values.
- **HAVING** condition is used to specify criteria when working using the GROUP BY keyword.
- **ORDER BY** is used to specify the sort order of the result set.

Examples:-

```
SELECT * FROM `members`;

SELECT `full_names`,`gender`,`physical_address`, `email` FROM `members`;

SELECT Concat(`title`, ' (', `director`, ')') , `year_released` FROM `movies`;

SELECT `membership_number`,`full_names`,LEFT(`date_of_birth`,4) AS `year_of_birth` FROM members;
```

Database Sql Commands - DCL

Data Control Language

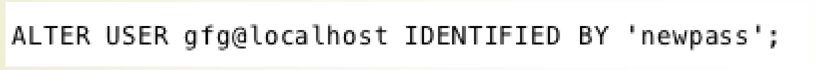
Data control commands in SQL allow you to control access to data within the database. These DCL commands are normally used to create objects related to user access and also control the distribution of privileges among users.

Some data control commands are as follows:

ALTER PASSWORD

To change the password for the user account “gfguser1” to “newpass” using the Alter User statement, syntax is as shown below:

Syntax:



```
ALTER USER gfg@localhost IDENTIFIED BY 'newpass';
```

GRANT

```
GRANT privilege,[privilege],.. ON privilege_level
TO user [IDENTIFIED BY password]
[REQUIRE tsl_option]
[WITH [GRANT_OPTION | resource_option]];
```

```
SHOW GRANTS FOR super@localhost;
```

Database Sql Commands - DCL

REVOKE

REVOKE privileges ON object FROM user;

Revoking SELECT Privilege to a User in a Table:

```
REVOKE SELECT ON users TO 'Amit'@localhost';
```

Revoking more than Privilege to a User in a Table:

```
REVOKE SELECT, INSERT, DELETE, UPDATE ON Users TO
'Amit'@'localhost;
```

Revoking All the Privilege to a User in a Table:

```
REVOKE ALL ON Users TO 'Amit'@'localhost;
```

Revoking a Privilege to all Users in a Table:

```
REVOKE SELECT ON Users TO '*'@'localhost;
```

Sql - Data Types – Numeric

Data types define the nature of the data that can be stored in a particular column of a table

MySQL has **3** main categories of data types namely

1. Numeric,
2. Text
3. Date/time.

Numeric Data types

Numeric data types are used to store numeric values.

TINYINT( )	-128 to 127 normal 0 to 255 UNSIGNED.
SMALLINT( )	-32768 to 32767 normal 0 to 65535 UNSIGNED.
MEDIUMINT( )	-8388608 to 8388607 normal 0 to 16777215 UNSIGNED.
INT( )	-2147483648 to 2147483647 normal 0 to 4294967295 UNSIGNED.
BIGINT( )	-9223372036854775808 to 9223372036854775807 normal 0 to 18446744073709551615 UNSIGNED.
FLOAT	A small approximate number with a floating decimal point.
DOUBLE(,)	A large number with a floating decimal point.
DECIMAL(,)	A DOUBLE stored as a string , allowing for a fixed decimal point. Choice for storing currency values.

Sql - Data Types - Text

Text Data Types

As data type category name implies these are used to store text values. Always make sure you length of your textual data do not exceed maximum lengths

CHAR( )	A fixed section from 0 to 255 characters long.
VARCHAR( )	A variable section from 0 to 255 characters long.
TINYTEXT	A string with a maximum length of 255 characters.
TEXT	A string with a maximum length of 65535 characters.
BLOB	A string with a maximum length of 65535 characters.
MEDIUMTEXT	A string with a maximum length of 16777215 characters.
MEDIUMBLOB	A string with a maximum length of 16777215 characters.
LONGTEXT	A string with a maximum length of 4294967295 characters.
LONGBLOB	A string with a maximum length of 4294967295 characters.

Date / Time

DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYYMMDDHHMMSS
TIME	HH:MM:SS

Sql - Data Types - Others

Apart from above there are some other data types in MySQL

ENUM	To store text value chosen from a list of predefined text values
SET	This is also used for storing text values chosen from a list of predefined text values. It can have multiple values.
BOOL	Synonym for TINYINT(1), used to store Boolean values
BINARY	Similar to CHAR, difference is texts are stored in binary format.
VARBINARY	Similar to VARCHAR, difference is texts are stored in binary format.

Now let's see a sample SQL query for creating a table which has data of all data types. Study it and identify how each data type is defined.

```
CREATE TABLE`all_data_types` (
`varchar` VARCHAR( 20 ) ,
`tinyint` TINYINT , `text` TEXT ,
`date` DATE ,
`smallint` SMALLINT ,
`mediumint` MEDIUMINT ,
`int` INT ,
`bigint` BIGINT ,
`float` FLOAT( 10, 2 ) ,
`double` DOUBLE ,
`decimal` DECIMAL( 10, 2 ) ,
`datetime` DATETIME ,
`timestamp` TIMESTAMP ,
`time` TIME ,
`year` YEAR ,
`char` CHAR( 10 ) ,
`tinyblob` TINYBLOB ,
`tinytext` TINYTEXT ,
`blob` BLOB ,
`mediumblob` MEDIUMBLOB ,
`mediumtext` MEDIUMTEXT ,
`longblob` LONGBLOB ,
`longtext` LONGTEXT ,
`enum` ENUM( '1', '2', '3' ) ,
`set` SET( '1', '2', '3' ) ,
`bool` BOOL ,
`binary` BINARY( 20 ) ,
`varbinary` VARBINARY( 20 )
) ENGINE= MYISAM ;
```

Primary Key

A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records.

A primary key’s main features are:

•It must contain a unique value for each row of data.

•It cannot contain null values.

A primary key is either an existing table column or a column that is specifically generated by the database according to a defined sequence.

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

Primary Key

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

SQL PRIMARY KEY on ALTER TABLE

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

MySQL

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

MySQL

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

Primary Key

DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

MySQL:

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

Database Design

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems
It helps produce database systems
1.That meet the requirements of the users
2.Have high performance.

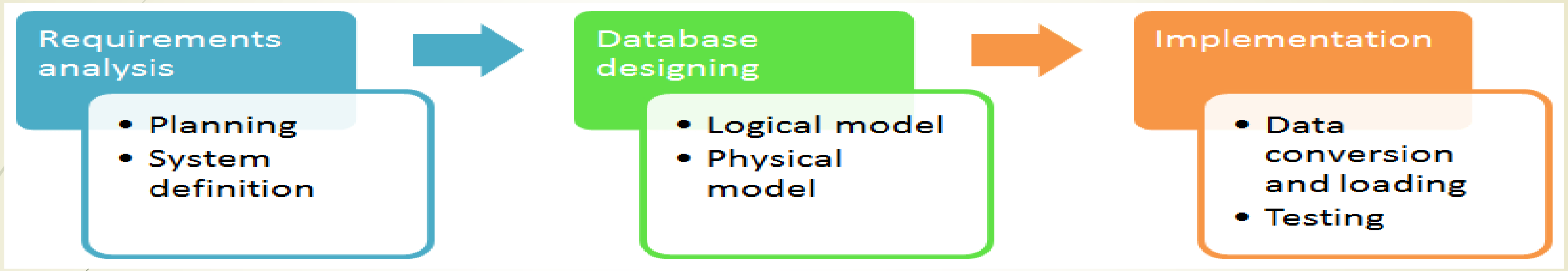
The main objectives of database designing are to produce logical and physical designs models of the proposed database system.
The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Why Database Design is Important ?

Database designing is crucial to **high performance** database system.
Apart from improving the performance, properly designed database are easy to maintain, improve data consistency and are cost effective in terms of disk storage space.
Note , the genius of a database is in its design . Data operations using SQL is relatively simple

Database Design

Database development life cycle



The database development life cycle has a number of stages that are followed when developing database systems.
The steps in the development life cycle do not necessary have to be followed religiously in a sequential manner.
On small database systems, the database system development life cycle is usually very simple and does not involve a lot of steps.
In order to fully appreciate the above diagram, let's look at the individual components listed in each step.

Database Design

Requirements analysis

•**Planning -** This stages concerns with planning of entire Database Development Life Cycle It takes into consideration the Information Systems strategy of the organization.

•**System definition -** This stage defines the scope and boundaries of the proposed database system.

Database designing

•**Logical model -** This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.

•**Physical model -** This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

•**Data conversion and loading -** this stage is concerned with importing and converting data from the old system into the new database.

•**Testing -** this stage is concerned with the identification of errors in the newly implemented system .It checks the database against requirement specifications.

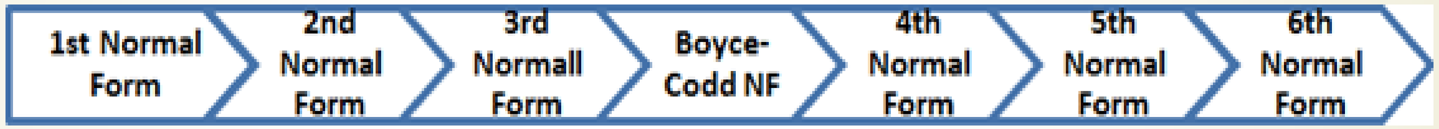
Two Types of Database Techniques

1. **ER Modeling**
2. **Normalization**

Normalization

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.
It divides larger tables to smaller tables and links them using relationships.

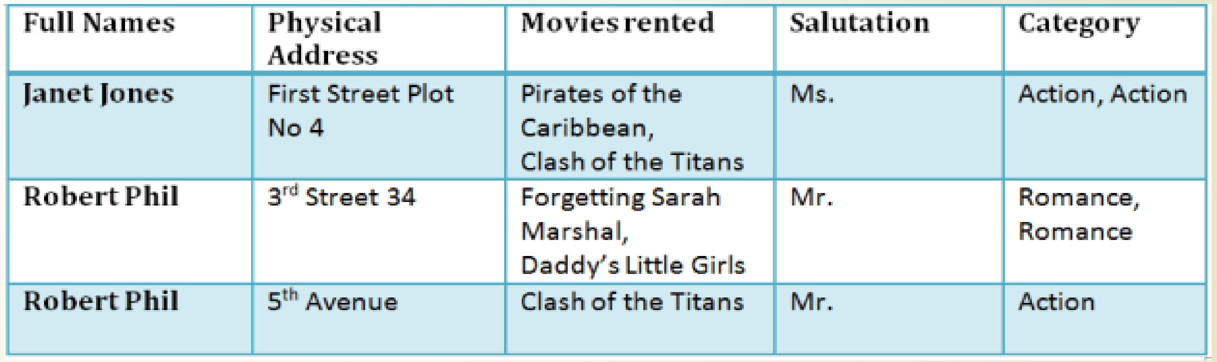
Theory of Data Normalization in SQL is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form**. The evolution of Normalization theories is illustrated below-



Database Normalization Examples -

Assume a video library maintains a database of movies rented out. Without any normalization, all information is stored in one table as shown below.

Here you see **Movies Rented column has multiple values**.



Full Names	Physical Address	Movies rented	Salutation	Category
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.	Action, Action
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.	Romance, Romance
Robert Phil	5th Avenue	Clash of the Titans	Mr.	Action

Normalization – 1NF

Database Normal Forms

Now let's move into 1st Normal Forms

1NF (First Normal Form) Rules

•Each table cell should contain a single value.

•Each record needs to be unique.

The above table in 1NF-

Before we proceed let's understand a few things --

What is a KEY?

A KEY is a value used to identify a record in a table uniquely. A KEY could be a single column or combination of multiple columns

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

What is a Primary Key?

A primary is a single column value used to identify a database record uniquely.
It has following attributes

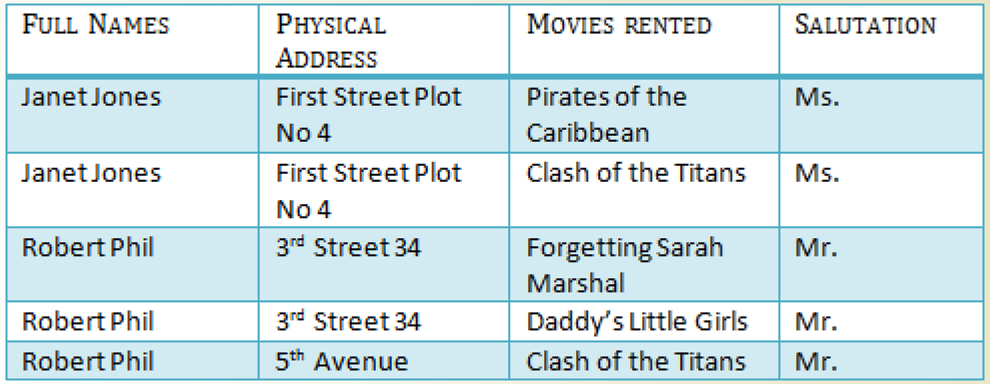
•A primary key cannot be NULL

•A primary key value must be unique

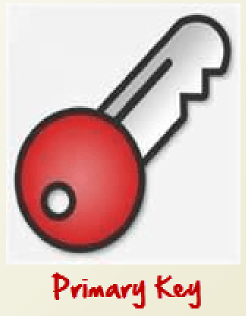
•The primary key values should rarely be changed

•The primary key must be given a value when a new record is inserted.

1NF Example



Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

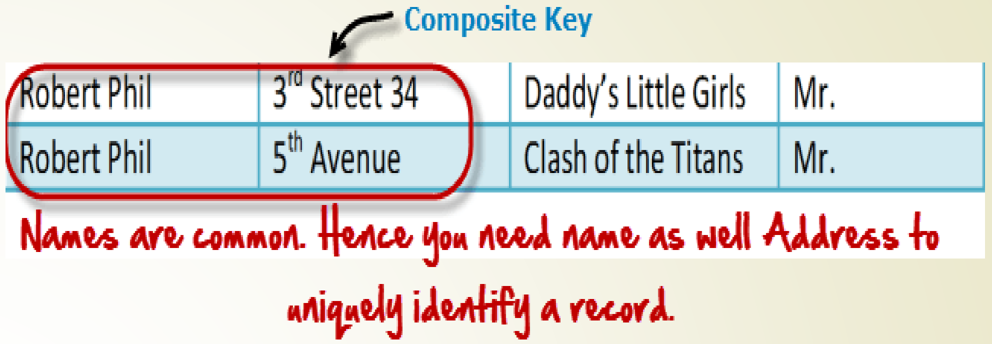


Normalization – 2NF

What is Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely
In our database, we have two people with the same name Robert Phil, but they live in different places.

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

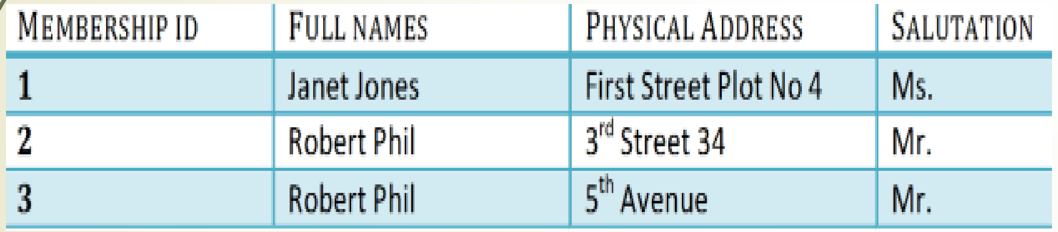

Composite Key

Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Names are common. Hence you need name as well Address to uniquely identify a record.

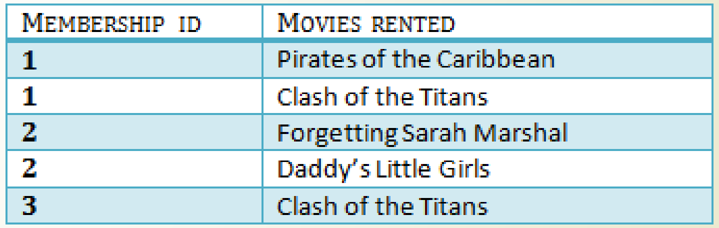
2NF (Second Normal Form) Rules

•Rule 1- Be in 1NF
•Rule 2- Single Column Primary Key
It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.



MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Table 1



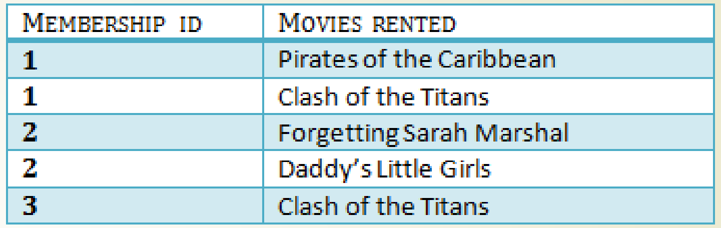
MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2

We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.
We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

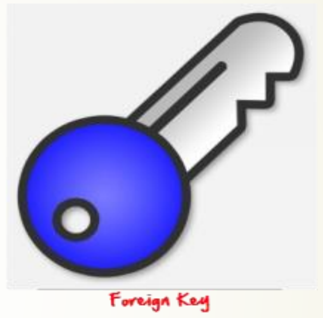
Normalization - Foreign Key

Database - Foreign Key
In Table 2, Membership_ID is the Foreign Key



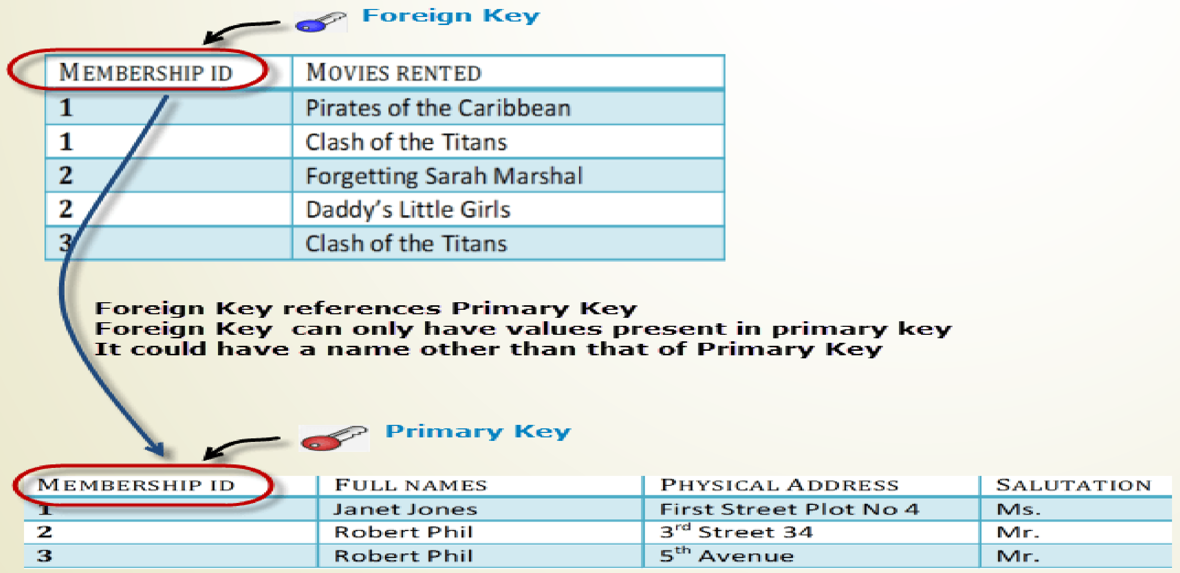
Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2



Foreign Key references the primary key of another Table! It helps connect your Tables

•A foreign key can have a different name from its primary key
•It ensures rows in one table have corresponding rows in another
•Unlike the Primary key, they do not have to be unique. Most often they aren't
•Foreign keys can be null even though primary keys can not



Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Foreign Key references Primary Key
Foreign Key can only have values present in primary key
It could have a name other than that of Primary Key

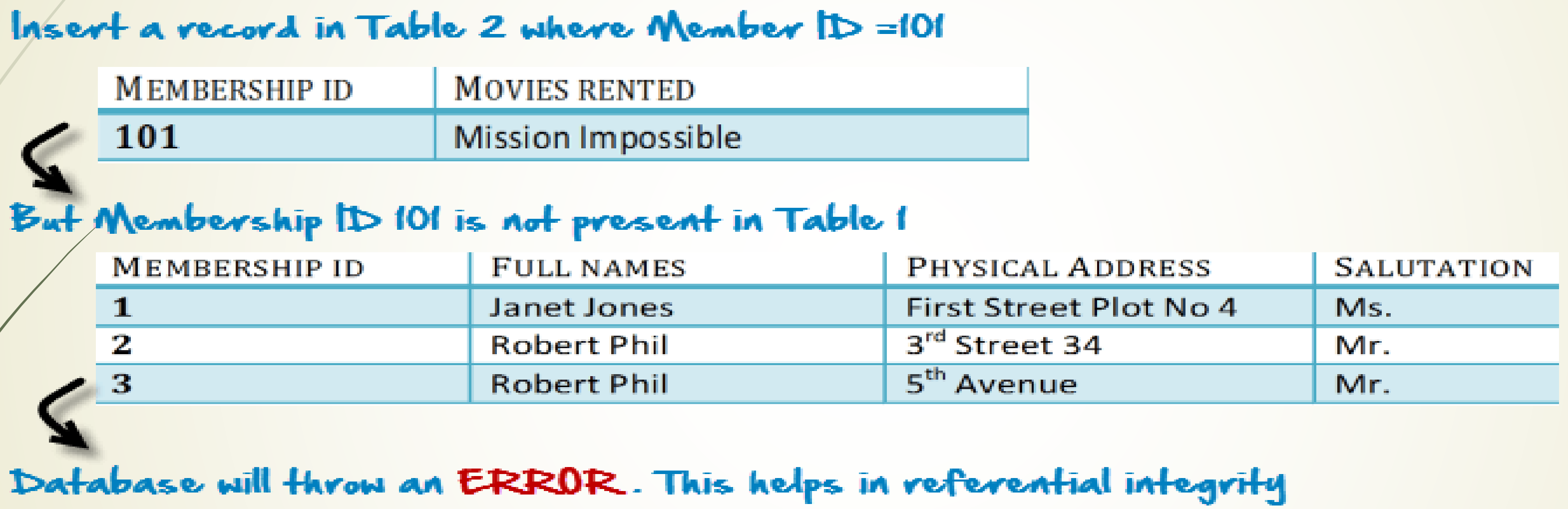
Membership ID	Full Names	Physical Address	Salutation
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Normalization – Foreign Key

Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.



Insert a record in Table 2 where Member ID =101

Membership ID	Movies rented
101	Mission Impossible

But Membership ID 101 is not present in Table 1

Membership ID	Full Names	Physical Address	Salutation
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Database will throw an ERROR. This helps in referential integrity

The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1

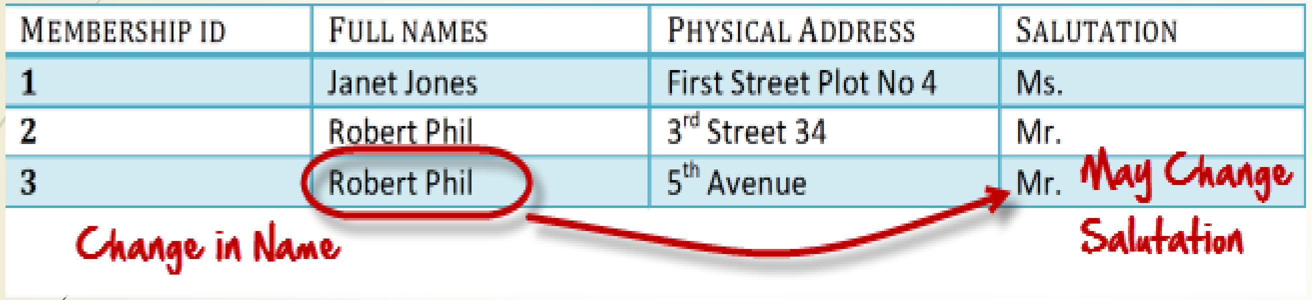
Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

Normalization – 3NF

What are transitive functional dependencies?

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

Consider the table 1. Changing the non-key column Full Name may change Salutation.


May Change Salutation
Change in Name

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

3NF (Third Normal Form) Rules

•Rule 1- Be in 2NF

•Rule 2- Has no transitive functional dependencies

To move our 2NF table into 3NF, we again need to again divide our table.

3NF Example

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

Table 1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Table 3

Normalization

We have again divided our tables and created a new table which stores Salutations.
There are no transitive functional dependencies, and hence our table is in 3NF
In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Boyce-Codd Normal Form (BCNF)
Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.
Sometimes is BCNF is also referred as **3.5 Normal Form.**

4NF (Fourth Normal Form) Rules
If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4th Normal Form.

5NF (Fifth Normal Form) Rules
A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

6NF (Sixth Normal Form) Proposed
6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future...

Understanding Relationship – One-to-One

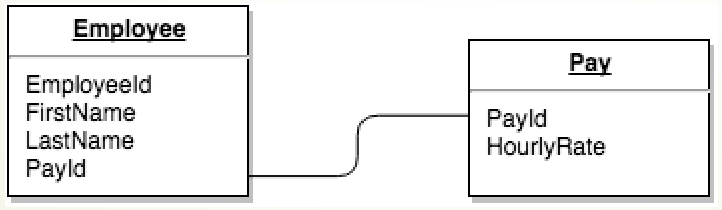
A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items.

There are 3 types of relationships in relational database design.
They are:

•One-to-One
•One-to-Many (or Many-to-One)
•Many-to-Many

One-to-One

A row in table A can have only one matching row in table B, and vice versa.

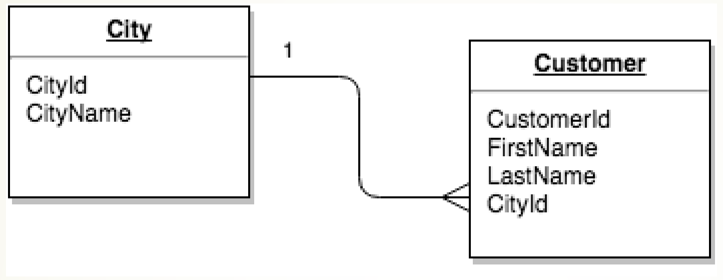


This is not a common relationship type, as the data stored in table B could just have easily been stored in table A. However, there are some valid reasons for using this relationship type. A one-to-one relationship can be used for security purposes, to divide a large table, and various other specific purposes.

Understanding Relationship – One-to-Many

One-to-Many (or Many-to-One)

This is the most common relationship type. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.

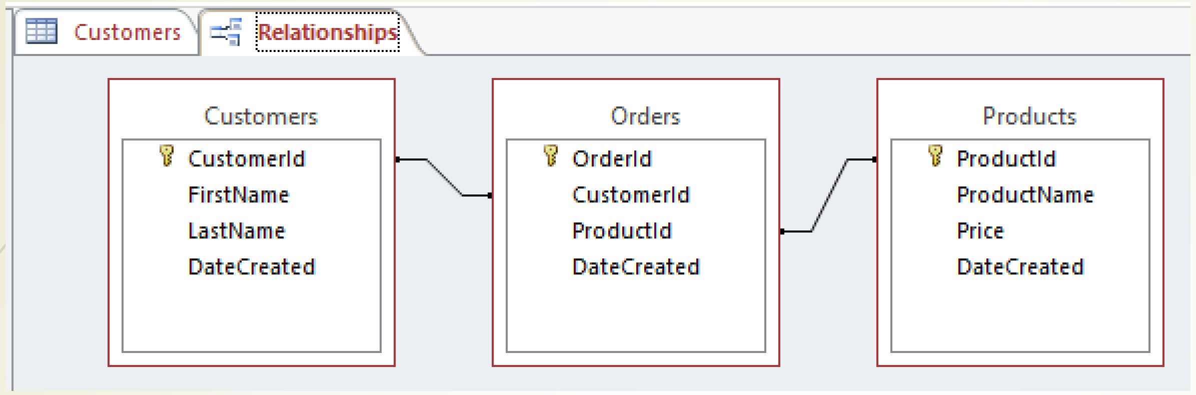


One-to-Many relationships can also be viewed as Many-to-One relationships, depending on which way you look at it. In the above example, the Customer table is the “many” and the City table is the “one”. Each customer can only be assigned one city,. One city can be assigned to many customers.

Many-to-Many

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa.
A many-to-many relationship could be thought of as two one-to-many relationships, linked by an intermediary table.
The intermediary table is typically referred to as a “junction table” (also as a “cross-reference table”). This table is used to link the other two tables together. It does this by having two fields that reference the primary key of each of the other two tables.

Understanding Relationship – Many-to-Many



So in order to create a many-to-many relationship between the Customers table and the Products table, we created a new table called Orders.

In the Orders table, we have a field called CustomerId and another called ProductId. The values that these fields contain should correspond with a value in the corresponding field in the referenced table. So any given value in Orders.CustomerId should also exist in the Customer.CustomerId field. If this wasn't the case then we could have orders for customers that don't actually exist. We could also have orders for products that don't exist. Not good referential integrity.

Most database systems allow you to specify whether the database should enforce referential integrity. So, when a user (or a process) attempts to insert a foreign key value that doesn't exist in the primary key field, an error will occur.

In our example, Orders.CustomerId field is a foreign key to the Customers.CustomerId (which is the primary key of that table). And the Orders.ProductId field is a foreign key to the Products.ProductId field (which is the primary key of that table).

Data Integrity

The term *data integrity* refers to the accuracy and consistency of data.
When creating databases, attention needs to be given to data integrity and how to maintain it. A good database will enforce data integrity whenever possible.
For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

Risks to Data Integrity

Some more examples of where data integrity is at risk:
•A user tries to enter a date outside an acceptable range.
•A user tries to enter a phone number in the wrong format.
•A bug in an application attempts to delete the wrong record.
•While transferring data between two databases, the developer accidentally tries to insert the data into the wrong table.
•While transferring data between two databases, the network went down.
•A user tries to delete a record in a table, but another table is referencing that record as part of a relationship.
•A user tries to update a primary key value when there's already a foreign key in a related table pointing to that value.
•A developer forgets that he's on a production system and starts entering test data directly into the database.
•A hacker manages to steal all user passwords from the database.
•A hacker hacks into the network and drops the database (i.e. deletes it and all its data).
•A fires sweeps through the building, burning the database computer to a cinder.
•The regular backups of the database has been failing for the past two months…

Data Integrity

4 Types of Data Integrity

In the database world, data integrity is often placed into the following types:

•Entity integrity
•Referential integrity
•Domain integrity
•User-defined integrity

Entity Integrity

Entity integrity defines each row to be unique within its table. No two rows can be the same.
To achieve this, a primary key can be defined. The primary key field contains a unique identifier – no two rows can contain the same unique identifier.

Referential Integrity

Referential integrity is concerned with relationships. When two or more tables have a relationship, we have to ensure that the foreign key value matches the primary key value at all times. We don't want to have a situation where a foreign key value has no matching primary key value in the primary table. This would result in an orphaned record.
So referential integrity will prevent users from:
•Adding records to a related table if there is no associated record in the primary table.
•Changing values in a primary table that result in orphaned records in a related table.
•Deleting records from a primary table if there are matching related records.

Data Integrity

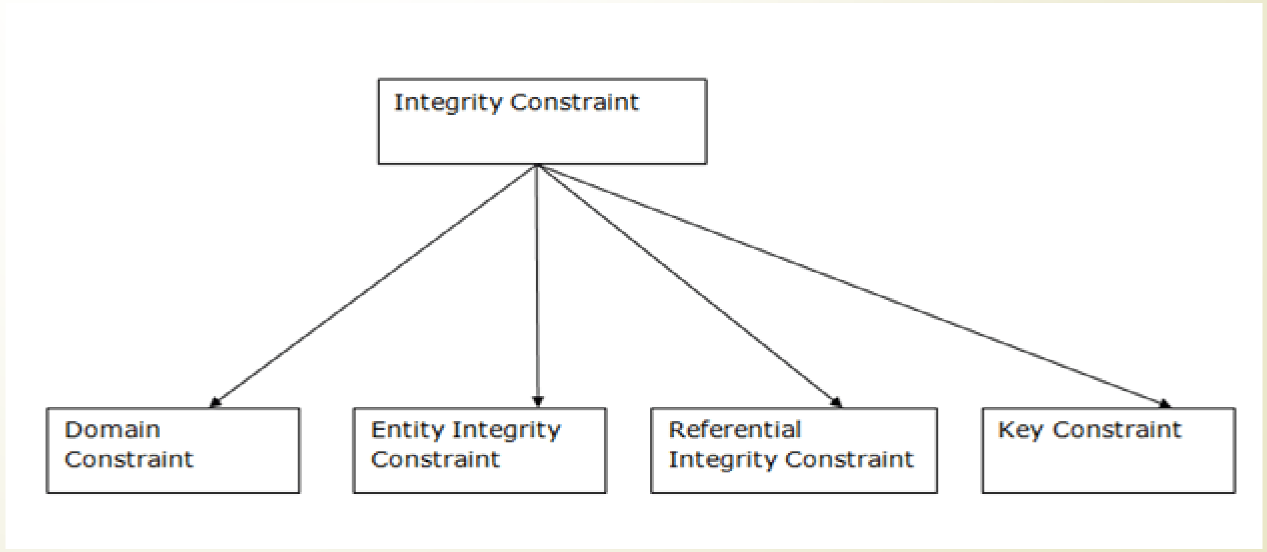
Domain Integrity

Domain integrity concerns the validity of entries for a given column. Selecting the appropriate data type for a column is the first step in maintaining domain integrity. Other steps could include, setting up appropriate constraints and rules to define the data format and/or restricting the range of possible values.

Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint



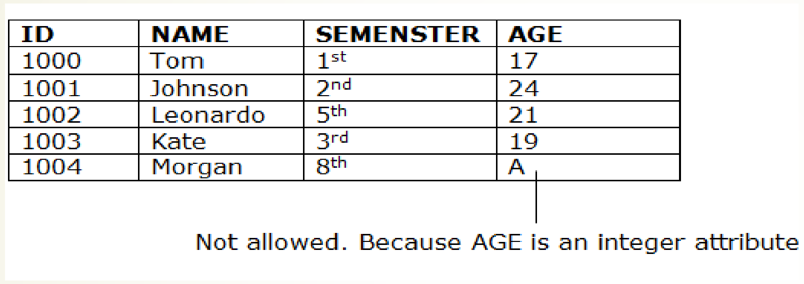
Data Integrity

1. Domain constraints

•Domain constraints can be defined as the definition of a valid set of values for an attribute.

•The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:



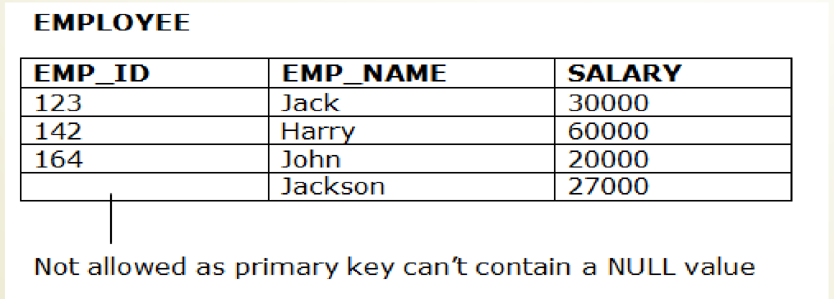
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

•The entity integrity constraint states that primary key value can't be null.
•This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
•A table can contain a null value other than the primary key field.

Example:



EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

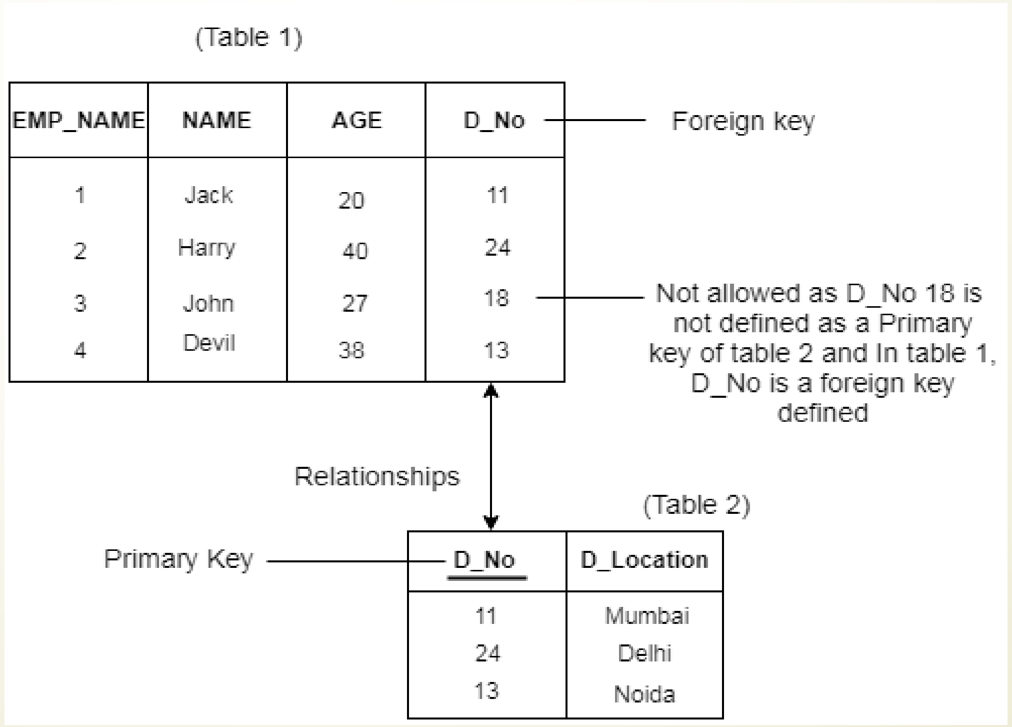
Data Integrity

3. Referential Integrity Constraints

•A referential integrity constraint is specified between two tables.

•In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

(Table 2)

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
Relationships
Primary Key

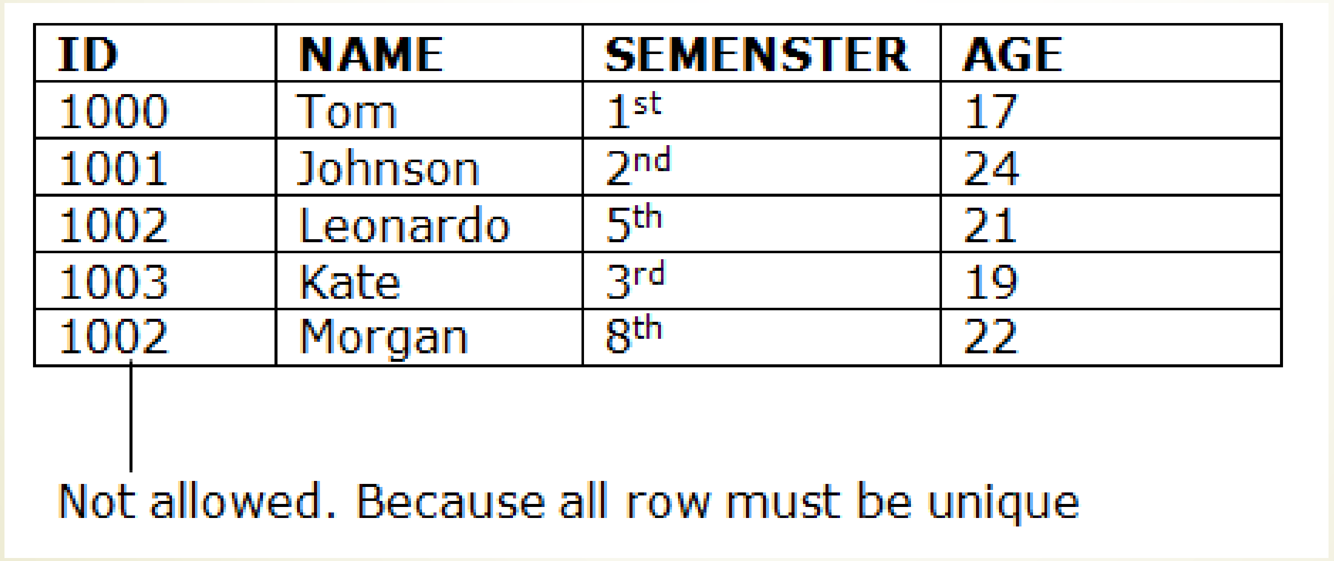
Data Integrity

4. Key constraints

•Keys are the entity set that is used to identify an entity within its entity set uniquely.

•An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:



ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

THANK YOU

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283