UNIT I - DATABASE MANAGEMENT SYSTEM

Database:

Database is the collection of interrelated data which are stored in the database systematically. In database, data are stored in the tables. On database, User can perform operations such as Insertion, Deletion, Updation and Selection.

Data Base Management System:

The mechanism for managing Database is called Database Management System (DBMS).

Database System Applications:

Telecom: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.

Industry: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.

Banking System: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.

Sales: To store customer information, production information and invoice details.

Airlines: To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.

Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.

Online shopping: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system. I have mentioned very few applications; this list is never going to end if we start mentioning all the DBMS applications.

Database Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics –

- **Real-world entity** – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- **Relation-based tables** – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application** – A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works

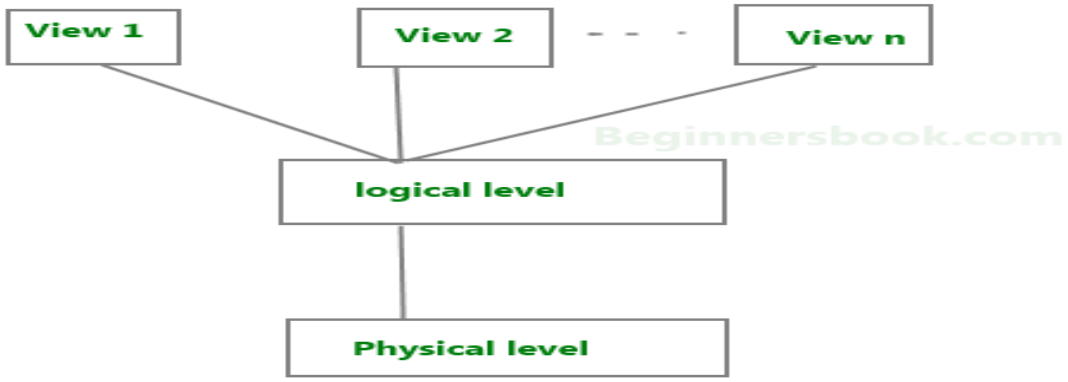
and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- **Less redundancy** – DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency** – Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- **Query Language** – DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties** – DBMS follows the concepts of Atomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

- **Multiuser and Concurrent Access** – DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- **Multiple views** – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security** – Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

DATA ABSTRACTION OR VIEW OF DATA IN DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction



Three Levels of data abstraction

LEVELS OF ABSTRACTION

Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level: Highest level of data abstraction. This level describes the user interaction with database system.

Example: We are storing customer information in a customer table. At **physical level,** these records can be described as blocks of storage

(bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers. At the **logical level,** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems. At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instance and schema in DBMS

DBMS SCHEMA

Definition of schema: The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level. Design of database at logical level is called **logical schema**, programmers and database administrator's work at this level. At this level, data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level). Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

DBMS INSTANCE

Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, let's say we have a single table student in the database, today the table has 100 records, and so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow. So the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance that changes over time when we add or delete data from the database.

DATA BASE USERS

There are different types of users who define, work with a Database. The important users are listed below:

1. Application Programmer
2. Specialized User
3. Sophisticated Users
4. Naive User or Native Users
5. DBA

1. **Application Programmer**
   They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include an embedded SQL query in the C Program.

2. **Specialized Users**
   They are responsible for developer specialized applications which is used to perform transactions/ operations in specialized DB where user can store video files, images, GiFs and raw files.

3. **Sophisticated Users**
   They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.

4. **Naïve Users:**These are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.

5. **Database Administrator**The life cycle of database starts from designing, implementing to administration of it. A database for any kind of requirement needs to be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes challenge. There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by database Administrator – DBA.
A DBA has many responsibilities. A good performing database is in the hands of DBA.

- **Installing and upgrading the DBMS Servers: -** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions comes in the market or requirement. If there is any failure in up gradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.

- **Design and implementation: -** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc for the database.

- **Performance tuning: -** Since database is huge and it will have lots of tables, data, constraints and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in fraction of seconds.

- **Migrate database servers: -** Sometimes, users using oracle would like to shift to SQL server or Netezza. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.

- **Backup and Recovery: -** Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA. Data/objects should be backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.

- **Security: -** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.

- **Documentation: -** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should

be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.

DATA MODEL

Logical structure of the database is called a Data Model. There are many data model existing in the world. The important data models are listed below:

1. E-R Model
2. Relational Model
3. Object Oriented Data Model
4. Network Model
5. Hierarchical Model
6. Object – Relational Model

Here, we are discussing the ER model and Relational model.

E-R MODEL

Concept:

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.

ER modelling helps you to analyze the data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modelling before implementing your database.

ER diagrams are a visual tool which is helpful to represent the ER model. It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network. He aimed to use an ER model as a conceptual modelling approach

Definition:

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

TYPES OF ATTRIBUTES

Simple attribute – Simple attributes are atomic values, which cannot be divided further. For example, Sno, Eno…

Composite attribute – those attribute which allows the user to divide into many sub parts. Example: Student's Name (First Name, Middle Name, Last Name).

Derived attribute – Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

Single-value attribute – Single-value attributes contain single value. For example – Social_Security_Number.

Multi-value attribute – Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email address, etc.

The expansion of ER Model is Entity Relationship Model. Here Entity refers to a simple object.

For example: Every Student in the class. Every customer in a bank.

The collection of entities of the same type is called as Entity Set.

For example: Collection of students in a class.

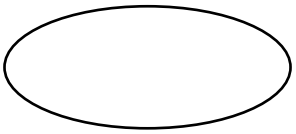
Relationship Set:

It refers to the link between entities. The collection of relationship of same type is referred to as a relationship set.

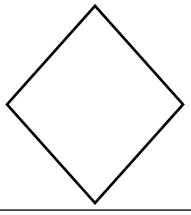
Symbols for ER Diagram:

Represents Entity Set

Represents Attributes

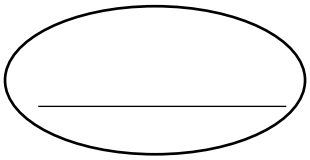


Represents Relationships

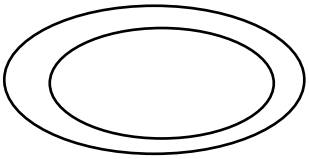


Links 2 Entities or 2 Attributes

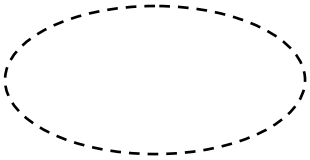
Represents Primary Key Attributes



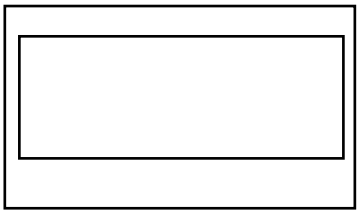
Represents Multivalued Attributes



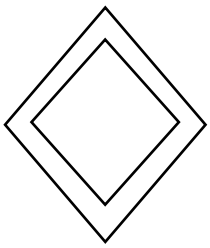
Represents Derived Attributes



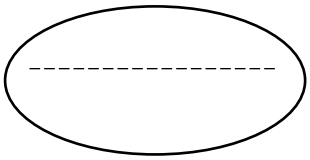
Represents Weak Entity Set



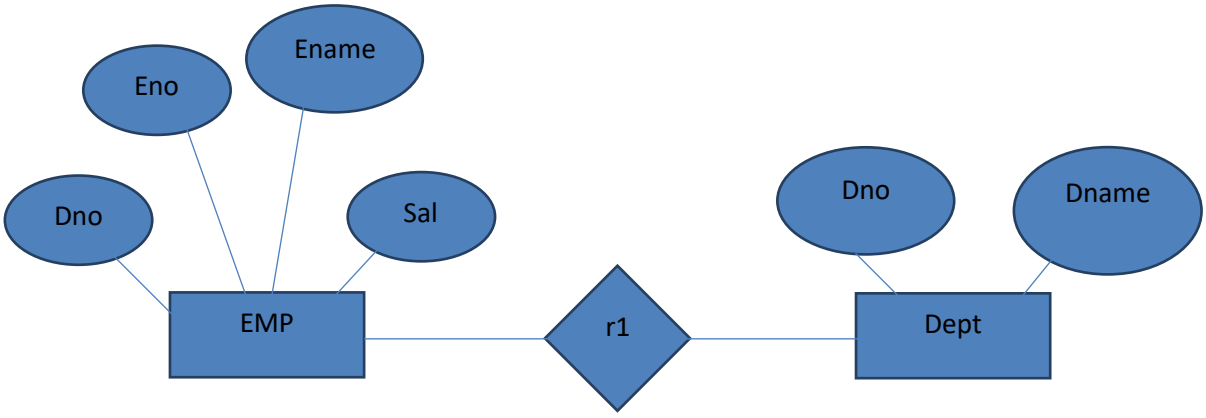
Represents Weak Relationships



Represents Discriminator



Example:

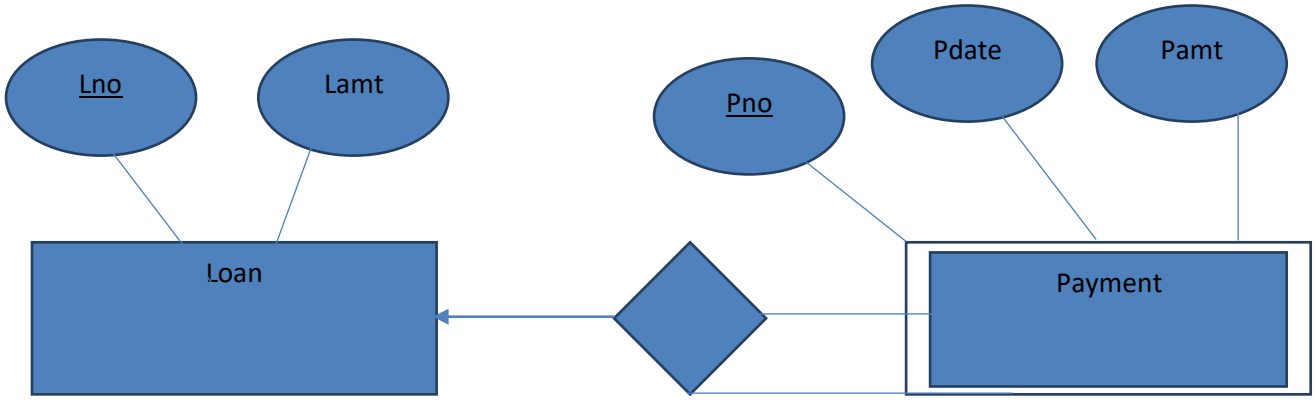


WEAK ENTITY SET

Concept:

The Entities which has the discriminate are called Weak Entity Sets. It means those entities fail to have strong primary key which is used to identify the remaining record/data. User unable to know the remaining record/data of the particular data of the primary key of the entity set.

Example: We take the payment entity set in which we have 3 attributes: pno, pdate, pamt. Using those attributes, it's very difficult to identify the payment made towards particular loan number, that is why, we need the assistance of 'loan' entity set to solve this problem.



Here, the combination of attributes such as lno, pno act as a primary key. Using this primary key {loan-no, p-no}, user can know that the payment done towards particular loan no. In the above pic/diagram, double line represents that the user can make many payments towards a particular loan.

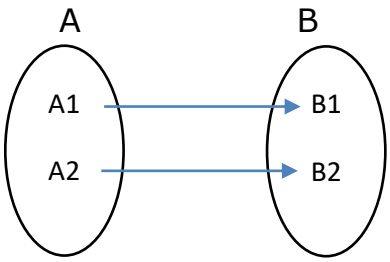
In the above diagram, the arrow head represents that every payment must be made towards a single loan at a time.

MAPPING CONSTRAINTS/CARDINALITIES

It is used to understand the way entity sets are connected. There are 4 types of Mapping Constraints.

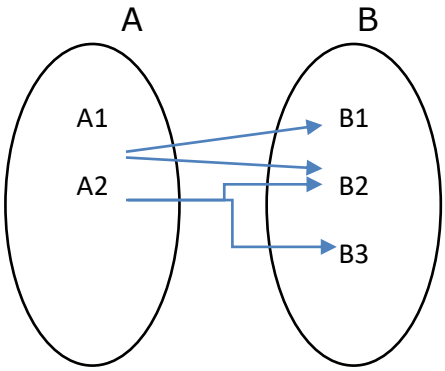
1. One-One:

Every entity in entity set A is associated with at most one entity in B & vice-versa.

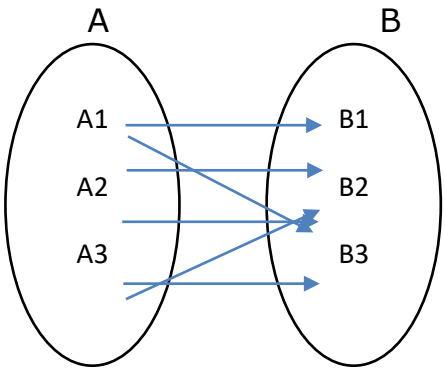


2. <u>One to Many:</u>

   Entities in A can be associated with 0 or more entities in B whereas entities in B can be associated with at most one entity in A.



3. <u>Many To Many:</u>
   Entities in A can be associated with 0 or more entities in B and vice versa.

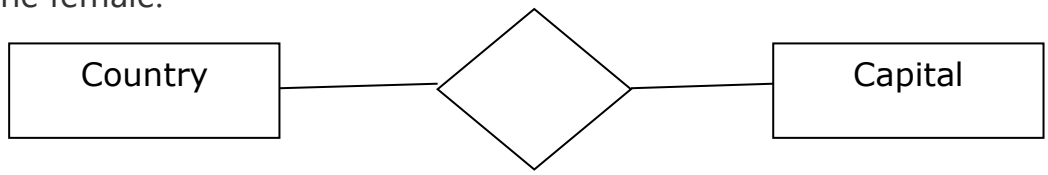


4. <u>Many To One:</u>

   It is vice versa of many to one constraint.

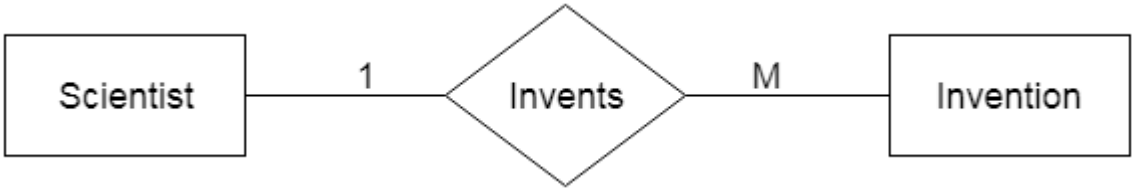
One-to-One Relationship

For example, A female can marry to one male, and a male can marry to one female.



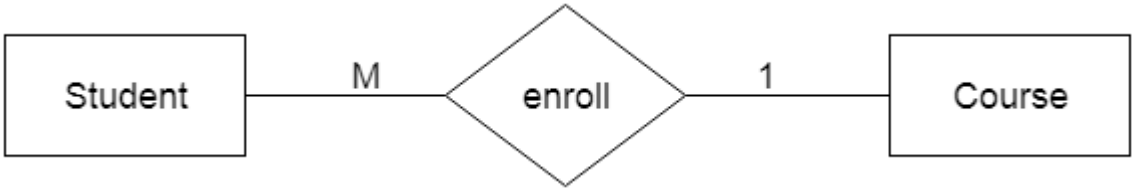
One-to-many relationship

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



Many-to-one relationship

For example, Student enrols for only one course, but a course can have many students.



Many-to-many relationship

For example, Employee can assign by many projects and project can have many employees.

RELATIONAL MODEL

The **relational model** (RM) for **database** management is first described in 1969 by English computer scientist Edgar F. Codd, where all data is represented in terms of records, grouped into relations.

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

Concepts

Tables: In relational data model, data are saved in the Table format. A table has rows and columns in which rows represents records and columns represent the attributes.

Record – Every row in the table is called record.

Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in SQL.

- NOT NULL Constraint – Ensures that a column cannot have NULL value.
- DEFAULT Constraint – Provides a default value for a column when none is specified.
- UNIQUE Constraint – Ensures that all values in a column are different.

- PRIMARY Key – uniquely identifies each row/record in a database table.
- FOREIGN Key – uniquely identifies a row/record in any of the given database table.
- CHECK Constraint – The CHECK constraint ensures that all the values in a column satisfies certain conditions.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

Dropping Constraints

Any constraint that you have defined can be dropped using the ALTER TABLE command with the DROP CONSTRAINT option.

For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command.

```
ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES_PK;
```

Some implementations may provide shortcuts for dropping certain constraints. For example, to drop the primary key constraint for a table in Oracle, you can use the following command.

```
ALTER TABLE EMPLOYEES DROP PRIMARY KEY;
```

Some implementations allow you to disable constraints. Instead of permanently dropping a constraint from the database, you may want to temporarily disable the constraint and then enable it later.

Integrity Constraints

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in **Referential Integrity (RI)**. These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints which are mentioned above.

Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

RELATIONAL ALGEBRA

It is a procedural Query language which is used to perform operations on relations. We are permitted to perform operations on relation. Performing operations on one relation is called as Unary Operation and performing operations on more than one relation is called as Binary Operation. The collections of these two operations are referred as Fundamental Operations in Relational Algebra.

Example:

Select Operation

This is used to fetch the tuples from the relation. We are permitted to fetch one, set of or all tuple(s). The select operation in relational algebra are referred using the symbol "σ" Sigma.

Eno	Ename	Sal	Dno
1	Ganesh	10000	10
2	Ramesh	5000	10
3	Suresh	3000	20
4	Dinesh	1500	30
5	Kamalesh	100000	30
6	Vignesh	25000	20

σ (emp): Returns all tuples.
$\sigma_{eno=1}$ (emp): Returns the tuples whose eno is 1.
$\sigma_{dno=10}$ (emp):Returns all the tuples who are working in dno 10

Project Operation:

It is used to fetch the values of particular columns or set of columns from the relation. It is denoted by π.

Example:

$\pi_{eno,\ ename}$ (emp), π_{eno} (emp)

We can also perform the transactions using project and select operations.

Union Operation

It is used to combine tuples of both relations.

Example:

Borrower

Cname	Loan No
Y	L1
Z	L2

Depositor

Cname	A/c No
X	256
Y	324
Z	123
A	L3
B	L4

Π_{Cname} (Depositor U Borrower) returns the following:

Cname
X
Y
Z
A
B

NOTE: Union operations avoid Duplicate Values.

Set Intersection Operator

This operator returns only common tuples from both relations. It is denoted by ∩.

Example: Π_{Cname} (Depositor ∩ Borrower)

Cname
Y
Z

Set Difference Operator:

It is denoted by - , it returns the tuples from the first relation which does not exist in the second relation.

Example: Π_{Cname} (Depositor - Borrower)

Cname
X

Cartesian product operation

The Cartesian product operation, denoted by the symbol ×, takes two relations as input and produces a new relation that contains all possible combinations of tuples from the first and second relations. Here's an

Example:

Consider the following `employee` table:

EmpID	**Name**
1	John
2	Jane

And the following `department` table:

DeptID	DeptName
101	Sales
102	Marketing

To find the Cartesian product of these two tables, we can use the following command: `employee × department`

The output would be:

EmpID	Name	DeptID	DeptName
1	John	101	Sales
1	John	102	Marketing
2	Jane	101	Sales
2	Jane	102	Marketing

Rename Operation

The rename operation, denoted by the Greek letter rho (ρ), is a unary operation that renames the attributes of a relation. It is used to rename the attributes of a relation to avoid ambiguity when the same attribute name appears in two different relations. Here's an example:
Consider the following `employee` table:

EmpID	Name	Salary

EmpID	Name	Salary
1	John	5000
2	Jane	6000
3	Bob	7000

To rename the `Name` attribute to `EmployeeName`, we can use the following command: `ρEmployeeName/Name(employee)`

The output would be:

EmpID	EmployeeName	Salary
1	John	5000
2	Jane	6000
3	Bob	7000

Extended Relational Algebra

Concept:

Extended operators in relational algebra are operators that can be derived from basic operators. There are mainly three types of extended operators in relational algebra: Aggregate Functions, Join, Intersection, and Divide. These operators allow for more complex queries to be performed on relations

Aggregate Functions

Aggregate functions are a type of extended operator in relational algebra that apply a function to a collection of values to generate a single result. Some common aggregate functions include SUM, AVG, COUNT, MIN, and MAX. These functions work on multisets, not sets, meaning that a value can appear in the input multiple times2. Aggregate functions can be used to reduce the amount of data that needs to be processed and displayed, making them useful for data reduction.

Suppose we have a relation orders that contains information about customer orders, including the total cost of each order. We can use the SUM aggregate function to find the total revenue from all orders by summing the cost of all orders:

G_{sum}(cost) (orders)

This expression will return the sum of the cost attribute for all tuples in the orders relation'

Suppose we have a relation employee that contains information about employees, including their salary. We can use the MAX aggregate function to find the highest salary among all employees by finding the maximum value of the salary attribute:

G_{max} (salary) (employees)

This expression will return the maximum value of the salary attribute for all tuples in the employee's relation.
Suppose we have a relation employee that contains information about employees, including their salary. We can use the MIN aggregate function to find the lowest salary among all employees by finding the minimum value of the salary attribute.

G_{min} (salary) (employees)

This expression will return the maximum value of the salary attribute for all tuples in the employee's relation.

Suppose we have a relation employee that contains information about employees. We can use the COUNT aggregate function to find the number of employees working in that organization.

G_{count} (eno) (employees)

This expression will return the number of employees working in that organization.

Suppose we have a relation employee that contains information about employees, including their salary. We can use the AVG aggregate function to find the average salary of all employees.

G_{AVG} (salary) (employees)

This expression will return the average salary of the salary attribute for all tuples in the employee's relation.

Divide: The Divide operator is used to find tuples in one relation that are associated with all tuples in another relation.

Suppose we have two relations, student_courses and math_courses, here student_courses contains information about the courses each student is taking, including their student ID and course name, and math_courses contains a list of all math courses.
student_courses relation:

student_id	course_name
1	Calculus
1	Physics
2	Algebra

student_id	course_name
2	Calculus
3	Algebra
3	Calculus

math_courses relation:

course_name
Algebra
Calculus

We can use the Divide operator to find the students who are taking all math courses by dividing the student_courses relation by the math_courses relation:

πstudent_id, course_name (student_courses) $\div$ πcourse_name (math_courses)

This expression will return a new relation that contains the student_id attribute for all students who are taking all courses listed in the math_courses relation.

Result:

student_id
2
3
