MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 1

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Database System Applications - Purpose of Database Systems

Introduction : (Maximum 5 sentences)

- A database management system (DBMS) refers to the technology for creating and managing databases.
- DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database.
- The main aim of a DBMS is to supply a way to store up and retrieve **database** information that is both convenient and efficient.
- The ultimate purpose of a database management system is to store and transform data into information to support making decisions.
- Through DBMS data can be accessed by multiple users, from multiple locations in a controlled manner

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Data
- Information
- Database
- Tuple
- Attribute
- Integrity
- Recovery
- Backup

Detailed content of the Lecture:

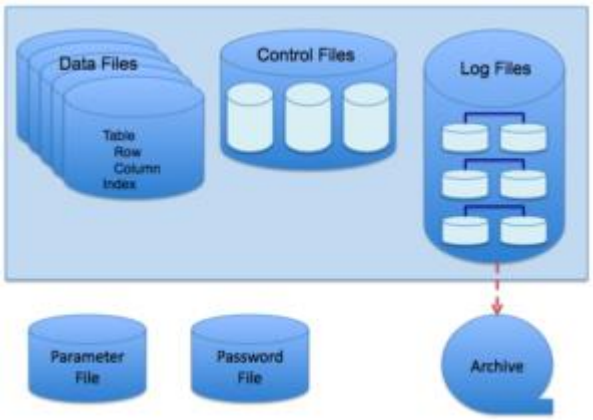
DBMS stands for Database Management System. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Application or Function of DBMS

Following fields are where the main Database Applications lie:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades

- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Database: Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.



These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query

1. Data redundancy
2. Data integrity
3. Inconsistency can be removed
4. data can be shared
5. Provide backup and recovery
6. Restriction for unauthorized access

The following are the main disadvantages of DBMS in File Processing:

- Data redundancy and inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

Video Content / Details of website for further learning (if any):
https://beginnersbook.com/2015/04/database-applications/

https://www.youtube.com/watch?v=hr0FLoWUY8E

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No: 1 - 5

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 2

CSE

II/IV

Course Name with Code **:Database Management System/16CSD03**

Course Teacher **:R.Vinupriya**

Unit **: I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: View of data - Database Languages - Database System Architecture

Introduction : (Maximum 5 sentences)

- Abstraction is one of the main features of database systems.
- The top level of that architecture is "view level".
- The view level provides the "view of data" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.
- Database languages are used to read, update and store data in a database.
- Database management systems architecture will help us understand the components of database system and the relation among them

Prerequisite knowledge for Complete understanding and learning of Topic:

- Abstraction
- Instances
- Schemas
- Storage Manager
- Query Processor

Detailed content of the Lecture:

Abstraction is one of the main features of database systems.

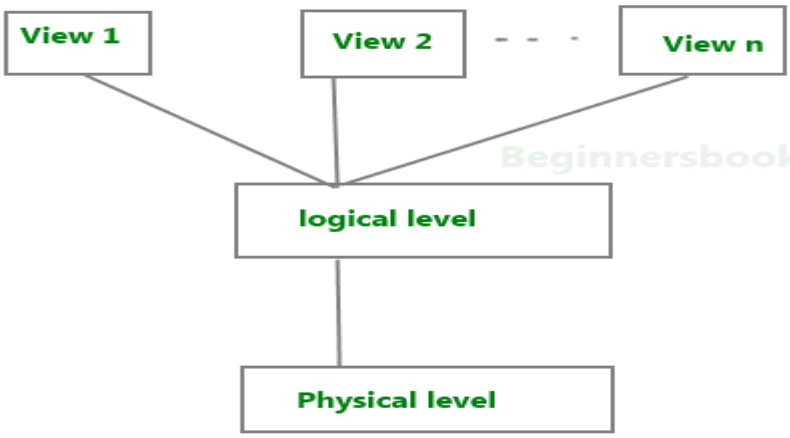
Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient user-database interaction.

The top level of that architecture is "view level". The view level provides the "view of data" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.

Data Abstraction

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.



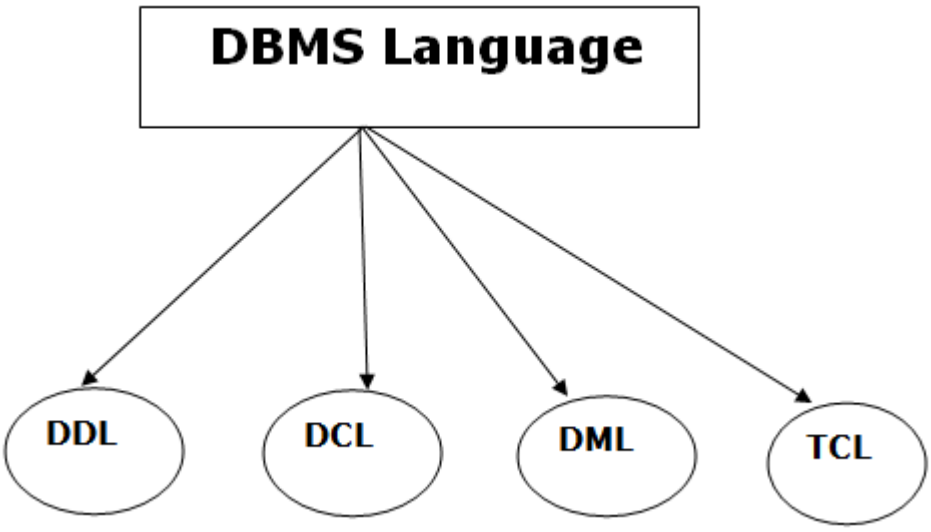
Three Levels of data abstraction

DBMS Schema

Definition of schema: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

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.



Database architecture

Database architecture uses programming languages to design a particular type of software for businesses or organizations.

The tiers are classified as follows :

1. Single tier architecture
2. Two tier architecture
3. Three tier architecture

Data User

- Application programmers – interact with system through DML calls
- Sophisticated users – form requests in a database query language

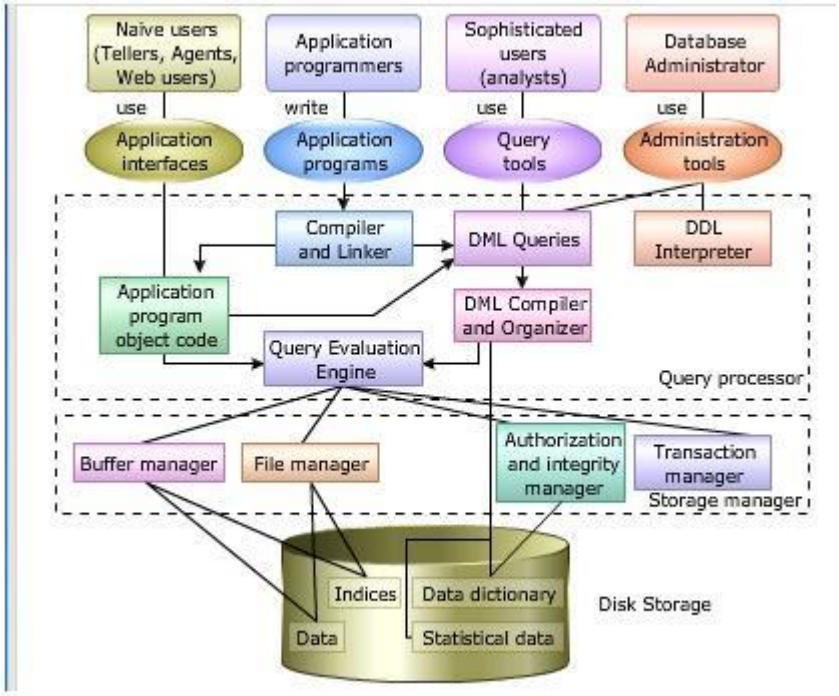
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
- Naïve users – invoke one of the permanent application programs that have been written previously
- Database Administrator
    - Functions of DBA
        - Schema definition
        - Schema and Physical Organization modification
        - Granting of authorization for data access
        - Routine maintenance

Storage management

Query processing

Issues

- Storage access
- File organization
- Indexing and hashing



Video Content / Details of website for further learning (if any):

https://www.slideshare.net/RahulNarang6/view-of-data-dbms
https://www.youtube.com/watch?v=OTFUNdxqIaY
https://www.youtube.com/watch?v=zdABU1bVZDg

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 6 - 8

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 3

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Data models - Entity-Relationship model

Introduction : (Maximum 5 sentences)

- Data models define how the logical structure of a database is modeled.
- Data Models are fundamental entities to introduce abstraction in a DBMS.
- Data models define how data is connected to each other and how they are processed and stored inside the system.
- Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them.
- While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Object
- Entity
- Attribute
- Primary Key
- Relationship

Detailed content of the Lecture:

Data Model is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constrains etc.

Types of Data Models

Object based logical Models – Describe data at the conceptual and view levels.

1. E-R Model
2. Object oriented Model

Record based logical Models – Like Object based model, they also describe data at the conceptual and view levels. These models specify logical structure of database with records, fields and attributes.

1. Relational Model
2. Hierarchical Model
3. Network Model –

Network Model is same as hierarchical model except that it has graph-like structure rather than a tree-based structure. Unlike hierarchical model, this model allows each record to have more than one parent record.

Physical Data Models – These models describe data at the lowest level of abstraction.

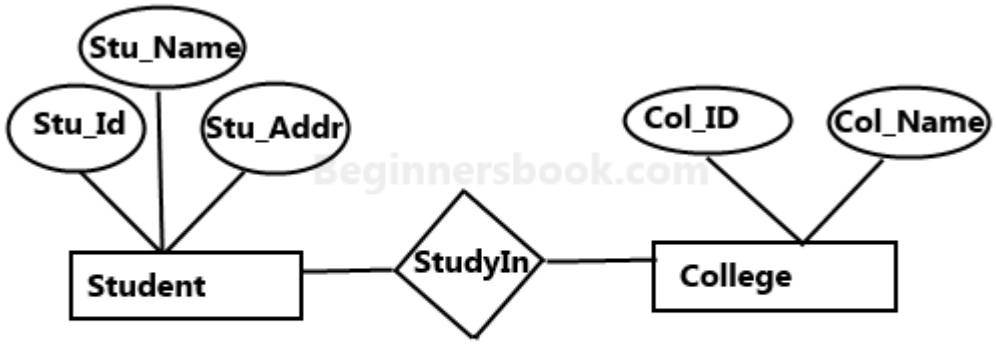
Relational model:

Stu Id	Stu Name	Stu age
111	Ashish	23
169	Lester	24
234	Lou	26

- Hierarchical model

In **hierarchical model**, data is organized into a tree like structure with each record is having one parent record and many children.
The main drawback of this model is that, it can have only one to many relationships between nodes.
Simple ER Diagram:

- In the following diagram we have two entities Student and College and their relationship.
- The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time.



Sample E-R Diagram

Here are the geometric shapes and their meaning in an E-R Diagram. We
Rectangle: Represents Entity sets.
Ellipses: Attributes
Diamonds: Relationship Set
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set

Relationship

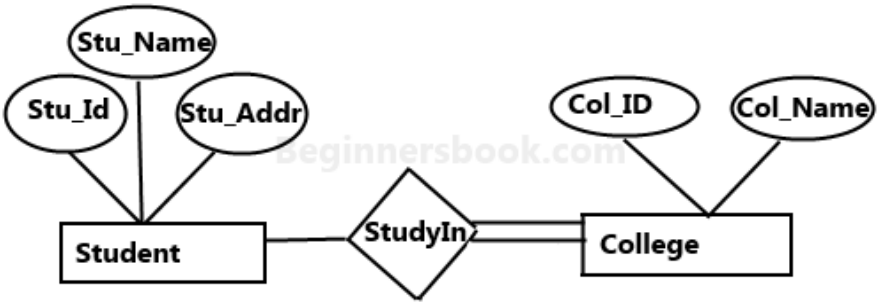
A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:
1. One to One
2. One to Many

3. Many to One
4. Many to Many

Total Participation of an Entity set

A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set.

For example: In the below diagram each college must have at-least one associated Student.



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/dbms/dbms_data_models.html
https://www.geeksforgeeks.org/introduction-of-er-model/
https://www.youtube.com/watch?v=obb7SlUmKQE

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 9 - 11

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 4

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Extended E-R Features - Introduction to relational databases

Introduction : (Maximum 5 sentences)

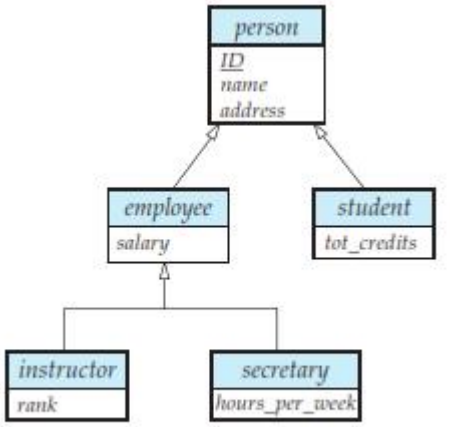
- An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set.
- For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
- Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes.
- The process of designating sub-groupings within an entity set is called specialization.
- Relational databases are a common and powerful approach to storing information.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Instance
- Entity
- Relationship
- Aggregation

Detailed content of the Lecture:

An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set.
For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
As an example, the entity set person may be further classified as one of the following: employee, student.
Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes.
The process of designating sub-groupings within an entity set is called specialization.
An entity set may be specialized by more than one distinguishing feature.
A distinguishing feature among employee entities is the job the employee performs.
In terms of an E-R diagram, specialization is depicted by a hollow arrow-head pointing from the specialized entity to the other entity.
This relationship is the ISA relationship, which stands for “is a” and represents, for example, that an instructor “is a” employee.
Types of specialization :
Overlapping Specialization : An entity may belong to multiple specialized entity sets.
Disjoint specialization : An entity may belong to at most one specialized entity sets.



Specialization and generalization.

For an overlapping specialization (as is the case for student and employee as specializations of person), two separate arrows are used. For a disjoint specialization (as is the case for instructor and secretary as specializations of employee), a single arrow is used.

- The specialization relationship may also be referred to as a **superclass - subclass** relationship.

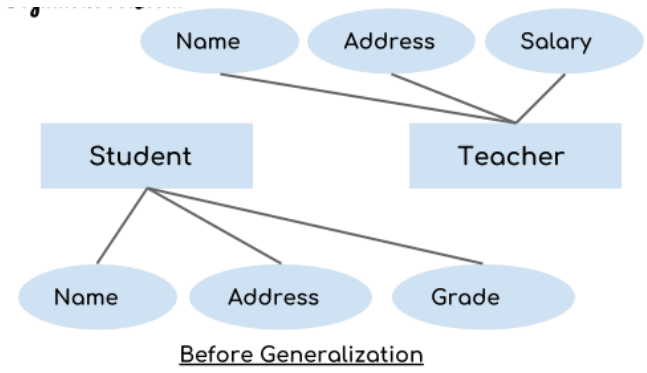
Generalization

Generalization is a process in which the common attributes of more than one entities form a new entity. This newly formed entity is called generalized entity

Generalization Example

- Lets
- There are two entities Student and Teacher.
  Attributes of Entity Student are: Name, Address & Grade
  Attributes of Entity Teacher are: Name, Address & Salary

The ER diagram before generalization looks like this:



Before Generalization

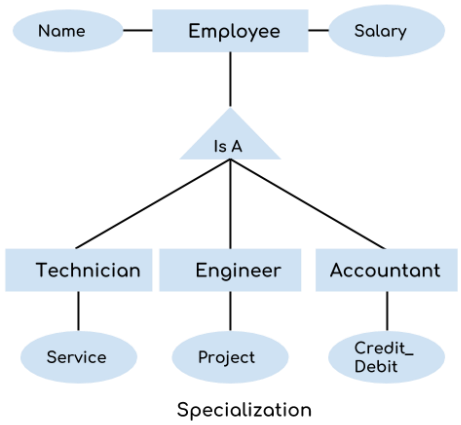
These two entities have two common attributes:
Name and Address, we can make a generalized entity with these common attributes. Lets have a look at the ER model after generalization.
1.Generalization uses bottom-up approach where two or more lower level entities combine together to form a higher level new entity.
2. The new generalized entity can further combine together with lower level entity to create a further higher level generalized entity.

Specialization
Specialization is a process in which an entity is divided into sub-entities.

Specialization is a top-down process.

For example – Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes.

Specialization Example

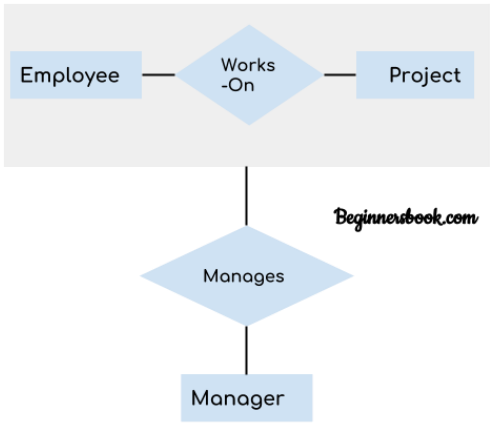


Specialization

Aggregration

Aggregation is a process in which a single entity alone is not able to make sense in a relationship so the relationship of two entities acts as one entity.

Aggregration Example



Relational Database

Relational Database Management System (RDBMS) consists of:

- ✓ A set of tables
- ✓ A schema

Video Content / Details of website for further learning (if any):

http://upscfever.com/upsc-fever/en/gatecse/en-gatecse-chp43.html
https://www.cis.uni-muenchen.de/~hs/teach/14s/ir/rdbms.pdf
https://www.youtube.com/watch?v=PDEIYp6iUlM

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 12 - 15

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 5

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Keys - Integrity Constraints

Introduction : (Maximum 5 sentences)

- A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table)
- They allow you to find the relation between two tables.
- Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
- Integrity constraints are rules that are to be applied on database columns to ensure the validity of data.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Primary key
- Candidate key
- Super key
- Foreign key

Detailed content of the Lecture:

A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table).

They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson

DBMS has following seven types of Keys each have their different functionality:

- Super Key
- Primary Key

- Candidate Key
- Foreign Key

Super key
A superkey is a group of single or multiple keys which identifies rows in a table.
A Super key may have additional attributes that are not needed for unique identification.

Primary Key

- ✓ A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key.
- ✓ This DBMS can't be a duplicate.
- ✓ The same value can't appear more than once in the table.

Candidate Key
A super key with no repeated attribute is called candidate key.

The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key.

Foreign key
A foreign key is a column which is added to create a relationship with another table.

Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity. Every relationship in the model needs to be supported by a foreign key.

Integrity Constraint:

- Integrity constraints are rules that are to be applied on database columns to ensure the validity of data. Every time data is entered into that particular column, it is evaluated against the constraint and only if the result comes out to be true, then the data is inserted into the column.
- Thus these constraints help to maintain the integrity of the data.

Integrity constraints can be divided into the following:

- ✓ Entity integrity constraint
- ✓ Domain Integrity constraint
- ✓ Referential Integrity constraint

Video Content / Details of website for further learning (if any):
https://www.guru99.com/dbms-keys.html#1
https://www.indiastudychannel.com/resources/150128-Integrity-Constraints-DBMS.aspxhttps://www.youtube.com/watch?v=zS2SmH8AlKM

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No: 45 - 47

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 6

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Relational Algebra - Fundamental Operations

Introduction : (Maximum 5 sentences)

- Relational algebra is a widely used procedural query language.
- It collects instances of relations as input and gives occurrences of relations as output.
- It uses various operation to perform this action.
- Relational algebra operations are performed recursively on a relation.
- The output of these operations is a new relation, which might be formed from one or more input relations.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Algebra
- Union
- Intersection
- Set Difference

Detailed content of the Lecture:

Basic Relational Algebra Operations:
Relational Algebra divided in various groups

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol:)

Relational Algebra Operations From Set Theory

- UNION (υ)
- INTERSECTION ( )
- DIFFERENCE (-)

- CARTESIAN PRODUCT (x)

Binary Relational Operations

- JOIN
- DIVISION

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it.

It is used as an expression to choose tuples which meet the selection condition. Select operation selects tuples that satisfy a given predicate.

$\sigma_p(r)$

σ is the predicate

r stands for relation which is the name of the table

p is prepositional logic

Example :

$\sigma_{\text{topic = "Database"}}$

Output - Selects tuples from Tutorials where topic = 'Database'.

Projection(π)

- ✓ The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.
- ✓ This helps to extract the values of specified attributes to eliminates duplicate values. (pi) The symbol used to choose attributes from a relation.
- ✓ This operation helps you to keep specific columns from a relation and discards the other columns.

Example:

Consider the following table

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Here, the projection of CustomerName and status will give

$\Pi_{\text{CustomerName, Status}}$ (Customers)

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive

Alibaba	Active

Union operation (υ)

UNION is symbolized by ∪ symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as:

The result <- A ∪ B

For a union operation to be valid, the following conditions must hold -

- R and S must be the same number of attributes.
- Attribute domains need to be compatible.
- Duplicate tuples should be automatically removed.

Example

Consider the following tables.

Table A		**Table B**	
column 1	**column 2**	**column 1**	**column 2**
1	1	1	1
1	2	1	3

A ∪ B gives

Table A ∪ B	
column 1	**column 2**
1	1
1	2
1	3

Set Difference (-)

- Symbol denotes it. The result of A - B, is a relation which includes all tuples that are in A but not in B.

- The attribute name of A has to match with the attribute name in B.
- The two-operand relations A and B should be either compatible or Union compatible.
- It should be defined relation consisting of the tuples that are in relation A, but not in B.

Example

A-B

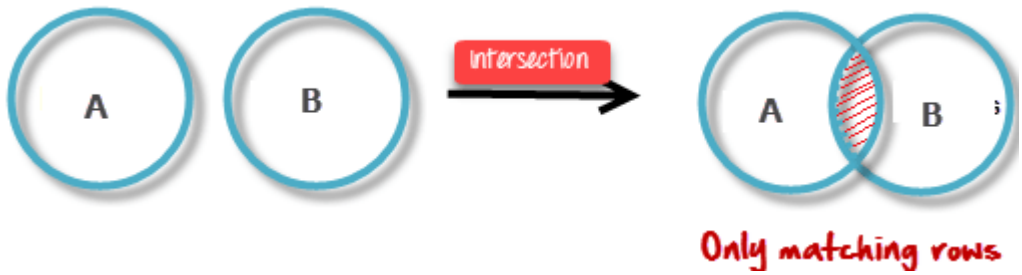
Table A – B	
column 1	**column 2**
1	2

Intersection

An intersection is defined by the symbol ∩

A ∩ B

Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union-compatible.



Example:

A ∩ B

Table A ∩ B	
column 1	**column 2**
1	1

Cartesian product(X)

- ✓ This type of operation is helpful to merge columns from two relations.
- ✓ A Cartesian product is never a meaningful operation when it performs alone. However, it becomes meaningful when it is followed by other operations.

Example – Cartesian product

$\sigma_{\text{column 2} = \text{'1'}}(A \times B)$

Output – The above example shows all rows from relation A and B whose column 2 has value 1

σ column 2 = '1' (A X B)	
column 1	**column 2**
1	1
1	1

Join Operations

- ✓ Join operation is essentially a cartesian product followed by a selection criterion.
- ✓ Join operation denoted by ⋈.
- ✓ JOIN operation also allows joining variously related tuples from different relations.

Types of JOIN:
Various forms of join operation are:
Inner Joins:

- Theta join
- EQUI join
- Natural join

Outer join:

- Left Outer Join
- Right Outer Join
- Full Outer Join

Inner Join:
In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded.

Theta Join:
The general case of JOIN operation is called a Theta join. It is denoted by symbol **θ**
Example

$A \bowtie_\theta B$

Theta join can use any conditions in the selection criteria.

For example:

A ⋈ A.column 2 > B.column 2 (B)	
column 1	**column 2**
1	2

EQUI join:
When a theta join uses only equivalence condition, it becomes a equi join.

For example:

A ⋈ A.column 2 = B.column 2 (B)	
column 1	**column 2**
1	1

EQUI join is the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMS have essential performance problems.

NATURAL JOIN (⋈)
Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

Example

Consider the following two tables

C	
Num	**Square**
2	4
3	9

D	
Num	**Cube**
2	8
3	18

C ⋈ D

C ⋈ D		
Num	**Square**	**Cube**
2	4	4

3	9	9

OUTER JOIN

- ✓ In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.

Left Outer Join(A ⟕ B)

- ✓ In the left outer join, operation allows keeping all tuple in the left relation.
- ✓ However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.

Consider the following 2 Tables

A	
Num	**Square**
2	4
3	9
4	16

B	
Num	**Cube**
2	8
3	18
5	75

A ⟕ B

A ⋈ B		
Num	**Square**	**Cube**
2	4	4
3	9	9
4	16	-

Right Outer Join: (A ⟖ B)

- ✓ In the right outer join, operation allows keeping all tuple in the right relation.
- ✓ However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.

A ⟕ B

A ⋈ B		
Num	**Cube**	**Square**
2	8	4
3	18	9
5	75	-

Full Outer Join: (A ⟗ B)
In a full outer join, all tuples from both relations are included in the result, irrespective of the matching condition.

A ⟗ B

A ⋈ B		
Num	**Cube**	**Square**
2	4	8
3	9	18
4	16	-
5	-	75

Video Content / Details of website for further learning (if any):
https://www.guru99.com/relational-algebra-dbms.html
https://www2.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/node10.html.
https://www.youtube.com/watch?v=4YilEjkNPrQ

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 48 - 51

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 7

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Additional Operations

Introduction : (Maximum 5 sentences)

- Additional operations are defined in terms of the fundamental operations. They do not add power to the algebra, but are useful to simplify common queries.
- The Set Intersection Operation is denoted by ∩, and returns a relation that contains tuples that are in both of its argument relations
- The Natural Join Operation is to simplify queries on a Cartesian product.
- Division, denoted ÷, is suited to queries that include the phrase ``for all"
- The assignment operation, denoted ←, works like assignment in a programming language

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- ✓ Relational Algebra
- ✓ The Select Operation
- ✓ The Project Operation
- ✓ The Cartesian Product Operation
- ✓ The Rename Operation
- ✓ The Union Operation
- ✓ The Set Difference Operation

Detailed content of the Lecture:

Additional Operations

1. Additional operations are defined in terms of the fundamental operations. They do not add power to the algebra, but are useful to simplify common queries.
2. **The Set Intersection Operation**

   Set intersection is denoted by ∩, and returns a relation that contains tuples that are in **both** of its argument relations.

It does not add any power as

$$r \cap s = r - (r - s)$$

To find all customers having both a loan and an account at the SFU branch, we write

$$\Pi_{cname}(\sigma_{bname="SFU"}(borrow)) \cap \Pi_{cname}(\sigma_{bname="SFU"}(deposit))$$

3. The Natural Join Operation

Often we want to simplify queries on a cartesian product.

For example, to find all customers having a loan at the bank and the cities in which they live, we need *borrow* and *customer* relations:

$$\Pi_{borrow.cname,ccity}(\sigma_{borrow.cname=customer.cname}(borrow \times customer))$$

Our selection predicate obtains only those tuples pertaining to only one *cname*.

- This type of operation is very common, so we have the **natural join**, denoted by a $\bowtie$ sign. Natural join combines a Cartesian product and a selection into one operation.
- It performs a selection forcing equality on those attributes that appear in both relation schemes. Duplicates are removed as in all relation operations.

Formally,

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.A_1=s.A_1 \wedge r.A_2=s.A_2 \wedge \ldots \wedge r.A_n=s.A_n}(r \times s))$$

where $R \cap S = \{A_1, A_2, \ldots, A_n\}$.

To find the assets and names of all branches which have depositors living in Stamford, we need *customer*, *deposit* and *branch* relations:

$$\Pi_{bname,assets}(\sigma_{ccity="Stamford"}(customer \bowtie deposit \bowtie branch))$$

Note that $\bowtie$ is associative.

To find all customers who have both an account and a loan at the SFU branch:

$$\Pi_{cname}(\sigma_{bname="SFU"}(borrow \bowtie deposit))$$

This is equivalent to the set intersection version we wrote earlier. We see now that there can be several ways to write a query in the relational algebra.

If two relations $r(R)$ and $s(S)$ have no attributes in common, then $R \cap S = \emptyset$, and $r \bowtie s = r \times s$.

4. The Division Operation

Division, denoted $\div$, is suited to queries that include the phrase ``for all".

Suppose we want to find all the customers who have an account at **all** branches located in Brooklyn.

Strategy: think of it as three steps.

We can obtain the names of all branches located in Brooklyn by

$$r_1 = \Pi_{bname}(\sigma_{bcity="Brooklyn"}(branch))$$

We can also find all *cname, bname* pairs for which the customer has an account by

$$r_2 = \Pi_{cname,bname}(deposit)$$

Now we need to find all customers who appear in r_2 with **every** branch name in r_1.

The divide operation provides exactly those customers:

$$\Pi_{cname,bname}(deposit) \div \Pi_{bname}(\sigma_{bcity="Brooklyn"}(branch))$$

which is simply $r_2 \div r_1$.

5. **The Assignment Operation**

Sometimes it is useful to be able to write a relational algebra expression in parts using a temporary relation variable (as we did with r_1 and r_2 in the division example).

The assignment operation, denoted $\leftarrow$, works like assignment in a programming language.

- No extra relation is added to the database, but the relation variable created can be used in subsequent expressions.
- Assignment to a permanent relation would constitute a modification to the database.

Formal Definition of Relational Algebra

1. A basic expression consists of either
   a. A relation in the database.
   b. A constant relation.
2. General expressions are formed out of smaller sub expressions using
   a. $\sigma_p(E_1)$ select (p a predicate)
   b. $\Pi_s(E_1)$ project (s a list of attributes)
   c. $\rho_x(E_1)$ rename (x a relation name)
   d. $E_1 \cup E_2$ union
   e. $E_1 - E_2$ set difference
   f. $E_1 \times E_2$ Cartesian product

Video Content / Details of website for further learning (if any):
https://www2.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/node10.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 74 - 78

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 8

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Domain Relational Calculus

Introduction : (Maximum 5 sentences)

- Relational Calculus in non-procedural query language and has no description about how the query will work or the data will b fetched.
- It only focuses on what to do, and not on how to do it.
- Relational Calculus exists in two forms:

  1. Tuple Relational Calculus (TRC)
  2. Domain Relational Calculus (DRC)

- Contrary to Relational Algebra which is a procedural query language to fetch data

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Integrity Constraints
- Relational Algebra
- Fundamental Operations
- Additional Operations

Detailed content of the Lecture:

Domain Relational Calculus is a non-procedural query language equivalent in power to Tuple Relational Calculus. Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it. In Domain Relational Calculus, a query is expressed as,

$\{ < x_1, x_2, x_3, ..., x_n > | P (x_1, x_2, x_3, ..., x_n) \}$

where, $< x_1, x_2, x_3, \ldots, x_n >$ represents resulting domains variables and $P (x_1, x_2, x_3, \ldots, x_n)$ represents the condition or formula equivalent to the Predicate calculus.

Predicate Calculus Formula:

1. Set of all comparison operators
2. Set of connectives like and, or, not
3. Set of quantifiers

Query-1: Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$\{\prec l, b, a\succ \mid \prec l, b, a\succ \in loan \wedge (a \geq 100)\}$

Resulting relation:

LOAN NUMBER	BRANCH NAME	AMOUNT
L01	Main	200
L03	Main	150
L10	Sub	90

Query-2: Find the loan number for each loan of an amount greater or equal to 150.

$\{\prec l\succ \mid \exists\ b, a\ (\prec l, b, a\succ \in loan \wedge (a \geq 150)\}$

Resulting relation:

LOAN NUMBER
L01
L03

Query-3: Find the names of all customers having a loan at the "Main" branch and find the loan amount.

$\{\prec c, a\succ \mid \exists\ l\ (\prec c, l\succ \in borrower \wedge \exists\ b\ (\prec l, b, a\succ \in loan \wedge (b = \text{"Main"})))\}$

Resulting relation:

CUSTOMER NAME	AMOUNT
Ritu	200
Debomit	60
Soumya	150

Video Content / Details of website for further learning (if any):

https://www.studytonight.com/dbms/relational-calculus.php
https://www.geeksforgeeks.org/domain-relational-calculus-in-dbms/

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 245 - 247

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 9

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**I Introduction to DBMS** **Date of Lecture:**

Topic of Lecture: Tuple Relational Calculus

Introduction : (Maximum 5 sentences)

Relational Calculus in non-procedural query language and has no description about how the query will work or the data will b fetched. It only focuses on what to do, and not on how to do it.

Relational Calculus exists in two forms:

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)
3. In Tuple relational calculus filtering tuples based on the given condition.

Syntax: { T | Condition }

A tuple variable is nothing but a name, can be anything, generally we use a single alphabet for this, so let's say T is a tuple variable.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Relational Algebra
- Fundamental Operations
- Additional Operations
- Domain relational calculus

Detailed content of the Lecture:

- Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra.
- Tuple Calculus provides only the description of the query but it does not provide the methods to solve it. Thus, it explains what to do but not how to do.
  In Tuple Calculus, a query is expressed as

{t| P(t)}

- where t = resulting tuples,
  P(t) = known as Predicate and these are the conditions that are used to fetch t

- Thus, it generates set of all tuples t, such that Predicate P(t) is true for t.
- P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT(¬).
  It also uses quantifiers:
  ∃ t ∈ r (Q(t)) = ”there exists” a tuple in t in relation r such that predicate Q(t) is true.
  ∀ t ∈ r (Q(t)) = Q(t) is true “for all” tuples in relation r.

Queries-1: Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

$\{t \mid t \in loan \ \wedge t[amount] >= 10000\}$

Resulting relation:

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L98	DEF	65000

In the above query, t[amount] is known as tuple variable.

Queries-2: Find the loan number for each loan of an amount greater or equal to 10000.

$\{t \mid \exists\ s \in loan(t[loan\ number] = s[loan\ number] \wedge s[amount] >= 10000)\}$

Queries-3: Find the names of all customers who have a loan and an account at the bank.

$\{t \mid \exists\ s \in borrower(\ t[customer\text{-}name] = s[customer\text{-}name]) \wedge \exists\ u \in depositor(\ t[customer\text{-}name] = u[customer\text{-}name])\}$

Queries-4: Find the names of all customers having a loan at the “ABC” branch.

$\{t \mid \exists\ s \in borrower(t[customer\text{-}name] = s[customer\text{-}name] \wedge \exists\ u \in loan(u[branch\text{-}name] = \text{“ABC”} \wedge u[loan\text{-}number\ ] = s[loan\text{-}number]))\}$

Video Content / Details of website for further learning (if any):
https://www.studytonight.com/dbms/relational-calculus.php
https://www.geeksforgeeks.org/tuple-relational-calculus-trc-in-dbms/

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No: 239 - 244

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 10

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: SQL Standards - Data types

Introduction : (Maximum 5 sentences)

- Structure Query Language (SQL) is a database query language used for storing and managing data in Relational DBMS.
- SQL was the first commercial language introduced for E.F Codd's Relational model of database.
- SQL is used to perform all types of data operations in RDBMS.
- SQL Datatype is used to define the values that a column can contain.
- Every column is required to have a name and data type in the database table.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- Relational Algebra
- Fundamental Operations
- Additional Operations
- SQL
- Basic Data types

Detailed content of the Lecture:

- Structure Query Language (SQL) is a database query language used for storing and managing data in Relational DBMS.
- SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language.
- SQL is used to perform all types of data operations in RDBMS.
- SQL is the standard language for Relational Database System.

Why SQL?

SQL is widely popular because it offers the following advantages −

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

SQL Process

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

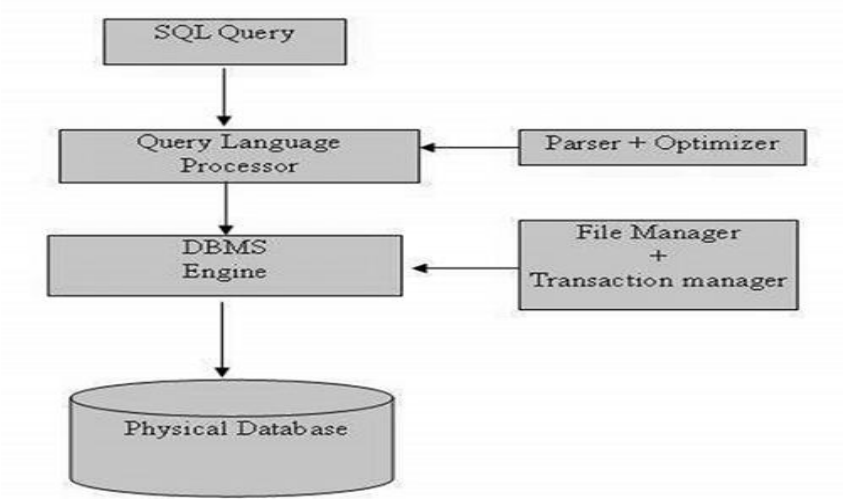
There are various components included in this process.

These components are −

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture −



SQL Command

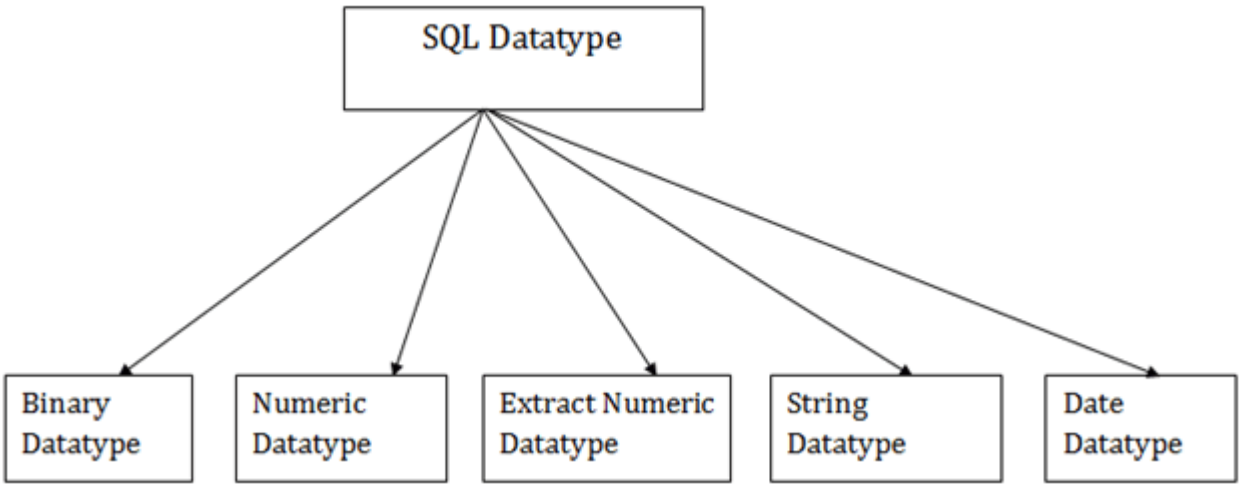
SQL defines following ways to manipulate data stored in an RDBMS.

DDL: Data Definition Language

This includes changes to the structure of the table like creation of table, altering table, deleting a table etc.
DML: Data Manipulation Language
DML commands are used for manipulating the data stored in the table and not the table itself.
TCL: Transaction Control Language
These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling the data back to its original state. It can also make any temporary change permanent.
DCL: Data Control Language
Data control language are the commands to grant and take back authority from any database user.
DQL: Data Query Language
Data query language is used to fetch data from tables based on conditions that we can easily apply

SQL Datatype

- ✓ SQL Datatype is used to define the values that a column can contain.
- ✓ Every column is required to have a name and data type in the database table.



Video Content / Details of website for further learning (if any):
https://www.studytonight.com/dbms/introduction-to-sql.php
https://www.tutorialspoint.com/sql/sql-overview.html
https://www.javatpoint.com/dbms-sql-datatype

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 57 - 62

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 11

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Basic Structure of SQL Queries

Introduction : (Maximum 5 sentences)

- Structured Query Language is a standard Database language which is used to create, maintain and retrieve the relational database
- QL is case insensitive like SELECT, UPDATE, CREATE, etc in capital letters and use user defined things liked table name, column name, etc in small letters.
- We can write comments in SQL using "–" (double hyphen) at the beginning of any line.
- SQL is the programming language for relational databases (explained below) like MySQL, Oracle, Sybase, SQL Server, Postgre, etc.
- Other non-relational databases (also called NoSQL) databases like MongoDB, DynamoDB, etc do not use SQL
- Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Relational Database
- Keys
- Data types
- SQL

Detailed content of the Lecture:

- Structured Query Language is a standard Database language which is used to create, maintain and retrieve the relational database
- QL is case insensitive like SELECT, UPDATE, CREATE, etc in capital letters and use user defined things liked table name, column name, etc in small letters.
- We can write comments in SQL using "–" (double hyphen) at the beginning of any line.
- SQL is the programming language for relational databases (explained below) like MySQL, Oracle, Sybase, SQL Server, Postgre, etc.

- Other non-relational databases (also called NoSQL) databases like MongoDB, DynamoDB, etc do not use SQL
- Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax.

Relational Database?

Relational database means the data is stored as well as retrieved in the form of relations (tables). Table 1 shows the relational database with only one relation called student which stores roll_no, name, address, phone and age of students.

student

Roll_no	Name	Address	Phone	Age
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

TABLE 1

These are some important terminologies that are used in terms of relation.

Attribute:

Attributes are the properties that define a relation. e.g.; Roll_No, Name etc.

Tuple:

Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

Degree:

- The number of attributes in the relation is known as degree of the relation.
- The student relation defined above has degree 5.

Cardinality:

The number of tuples in a relation is known as cardinality.

The Student relation defined above has cardinality 4.

Column:

Column represents the set of values for a particular attribute.

The column Roll_No is extracted from relation Student.

ROLL_NO
1
2

3
4

The queries to deal with relational database can be categories as:

Data Definition Language:
It is used to define the structure of the database.
e.g; CREATE TABLE, ADD COLUMN, DROP COLUMN and so on.

Data Manipulation Language:
It is used to manipulate data in the relations.
e.g.; INSERT, DELETE, UPDATE and so on.

Data Query Language:
It is used to extract the data from the relations.
e.g.; SELECT

1. **SELECT** [**DISTINCT**] Attribute_List **FROM** R1,R2….RM
2. [**WHERE** condition]
3. [**GROUP BY** (Attributes)[**HAVING** condition]]
4. [**ORDER BY**(Attributes)[**DESC**]];

Case 1: If we want to retrieve attributes Roll_No And Name of all students, the query will be:

```
SELECT Roll_No, Name FROM Student;
```

Roll_no	Name
1	RAM
2	RAMESH
3	SUJIT
4	SURESH

Case 2:
If we want to retrieve Roll_No And Name of the students whose Roll_No is greater than 2, the query will be:

```
SELECT ROLL_NO, NAME FROM STUDENT
WHERE ROLL_NO>2;
```

Roll_no	Name
3	SUJIT
4	SURESH

CASE 3:
If we want to retrieve all attributes of students, we can write * in place of writing all attributes as:

```
SELECT * FROM STUDENT
WHERE ROLL_NO>2;
```

Roll_no	Name	Address	Phone	Age
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

CASE 4:

If we want to represent the relation in ascending order by **AGE**, we can use ORDER BY clause as:

```
SELECT * FROM Student ORDER BY Age;
```

Roll_no	Name	Address	Phone	Age
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
4	SURESH	DELHI	9156768971	18
3	SUJIT	ROHTAK	9156253131	20

Note:

ORDER BY **AGE** is equivalent to ORDER BY **AGE** ASC.

If we want to retrieve the results in descending order of **AGE**, we can use ORDER BY **AGE** DESC.

CASE 5:

If we want to retrieve distinct values of an attribute or group of attribute, DISTINCT is used as in:

```
SELECT DISTINCT Address FROM Student;
```

If DISTINCT is not used, DELHI will be repeated twice in result set.

Before understanding GROUP BY and HAVING, we need to understand aggregations functions in SQL.

AGGRATION FUNCTIONS:

Aggregation functions are used to perform mathematical operations on data values of a relation. Some of the common aggregation functions used in SQL are:

- ✓ **COUNT:** Count function is used to count the number of rows in a relation.

e.g;

```
SELECT COUNT (PHONE) FROM STUDENT;
```

COUNT(PHONE)
4

- ✓ **SUM:** SUM function is used to add the values of an attribute in a relation. e.g;

```
SELECT SUM (Age) FROM Student;
```

SUM(Age)

In the same way, MIN, MAX and AVG can be used. As we have seen above, all aggregation functions return only 1 row.

GROUP BY:

- ✓ Group by is used to group the tuples of a relation based on an attribute or group of attribute.
- ✓ It is always combined with aggregation function which is computed on group.

e.g.

```
SELECT Address, SUM(Age) FROM Student
GROUP BY (Address);
```

In this query, SUM(**Age**) will be computed but not for entire table but for each address.
i.e. sum of AGE for address DELHI(18+18=36) and similarly for other address as well.
The output is:

ADDRESS	SUM(AGE)
DELHI	36
GURGAON	18
ROHTAK	20

it will result in error because although we have computed SUM(AGE) for each address, there are more than 1 ROLL_NO for each address we have grouped.

So it can't be displayed in result set.

We need to use aggregate functions on columns after SELECT statement to make sense of the resulting set whenever we are using GROUP BY.

```
SELECT Roll_No, Address, SUM(Age) FROM Student
GROUP BY (Address);
```

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/structured-query-language/
https://www.slideshare.net/SunitaAher1/4-basic-structure-of-sql-queries

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 63 – 78

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 12

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: DDL - DML

Introduction : (Maximum 5 sentences)

- Data Definition Language can be defined as a standard for commands through which data structures are defined.
- It is a computer language that used for creating and modifying the structure of the database objects, such as schemas, tables, views, indexes, etc.
- A data manipulation language (DML) is a family of computer languages including commands permitting users to manipulate data in a database.
- This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data.
- DML is mostly incorporated in SQL databases.

Prerequisite knowledge for Complete understanding and learning of Topic:

- SQL
- Database
- Keys
- Schemas
- Data types

Detailed content of the Lecture:

Data Definition Language can be defined as a standard for commands through which data structures are defined. It is a computer language that used for creating and modifying the structure of the database objects, such as schemas, tables, views, indexes, etc. Additionally, it assists in storing the metadata details in the database.

Data Definition language(DDL) with Examples

Some of the common Data Definition Language commands are:

- CREATE

- ALTER
- DROP

1. **CREATE- Data Definition language(DDL)**

✓ The main use of the create command is to build a new table and it comes with a predefined syntax.
✓ It creates a component in a relational database management system.
✓ There are many implementations that extend the syntax of the command to create the additional elements, like user profiles and indexes.

syntax :

CREATE TABLE tablename (Column1 DATATYPE, Column2 DATATYPE, Column3 DATATYPE, …….. ColumnN DATATYPE)

Example:

CREATE TABLE PUPIL (PUPIL_ID CHAR (10), STUDENT_Name Char (10);

2. **ALTER- Data Definition language(DDL)**

✓ An existing database object can be modified by the ALTER statement.
✓ Using this command, the users can add up some additional column and drop existing columns.
✓ the data type of columns involved in a database table can be changed by the ALTER command.

syntax ,

ALTER TABLE table_name ADD column_name (for adding a new column)
ALTER TABLE table_name RENAME To new_table_name (for renaming a table)
ALTER TABLE table_name MODIFY column_name data type (for modifying a column)
ALTER TABLE table_name DROP COLUMN column_name (for deleting a column)

✓ A **data manipulation language (DML)** is a family of computer languages including commands permitting users to manipulate data in a database.
✓ This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data.
✓ DML is mostly incorporated in SQL databases.

Examples of DML:

- INSERT – is used to insert data into a table.
- UPDATE – is used to update existing data within a table.
- DELETE – is used to delete records from a database table.

1. SELECT COMMAND

✓ SELECT command is used to retrieve data from the database.
✓ This command allows database users to retrieve the specific information they desire from an operational database.

- ✓ It returns a result set of records from one or more tables.

SELECT Command has many optional clauses are as stated below:

Clause	Description
WHERE	It specifies which rows to retrieve.
GROUP BY	It is used to arrange the data into groups.
HAVING	It selects among the groups defined by the GROUP BY clause.
ORDER BY	It specifies an order in which to return the rows.
AS	It provides an alias which can be used to temporarily rename tables or columns.

Syntax:

SELECT*FROM<table name>;

Example : SELECT Command

SELECT * FROM employee;

2. INSERT COMMAND

- ✓ INSERT command is used for inserting a data into a table.
- ✓ Using this command, you can add one or more records to any single table in a database.
- ✓ It is also used to add records to an existing code.

Syntax:

INSERT INTO <table_name> (`column_name1` <datatype>, `column_name2` <datatype>, . . . , `column_name_n` <database>) VALUES (`value1`, `value2`, . . . , `value n`);

Example:

INSERT INTO employee (`eid` int, `ename` varchar(20), `city` varchar(20))
VALUES (`1`, `ABC`, `PUNE`);

3. UPDATE COMMAND

- ✓ **UPDATE command** is used to modify the records present in existing table.
- ✓ This command updates existing data within a table.
- ✓ It changes the data of one or more records in a table.

Syntax:

UPDATE <table_name>

SET <column_name = value>

WHERE condition;

4. DELETE COMMAND

DELETE command is used to delete some or all records from the existing table.

It deletes all the records from a table.

Syntax:

DELETE FROM<table name> WHERE <condition>;

Example : DELETE Command

DELETE FROM employee

WHERE emp_id = '001';

If we does not write the WHERE condition, then all rows will get deleted.

Video Content / Details of website for further learning (if any):

https://whatisdbms.com/data-definition-language-ddl-in-dbms-with-examples/https://www.techopedia.com/definition/1179/data-manipulation-language-dml

https://www.tutorialride.com/dbms/sql-data-control-language-dcl.html

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 60 - 63

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 13

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: DCL - TCL

Introduction : (Maximum 5 sentences)

- Data Control Language(DCL) is used to control privileges in Database.
- To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges.

Privileges are of two types,

- **System:** This includes permissions for creating session, table, etc and all types of other system privileges.
- **Object:** This includes permissions for any command or query to perform any operation on the database tables.
- Transaction Control Language(**TCL**) commands are used to manage transactions in the database

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data manipulation Language
- Data Definition Language
- SQL Basic Structures
- Keys
- Data types

Detailed content of the Lecture:

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

In DCL we have two commands,

- GRANT: Used to provide any user access privileges or other priviliges for the database.
- REVOKE: Used to take back permissions from any user.

- The operations for which privileges may be granted to or revoked from a user or role apply to both the Data definition language (DDL) and
- the Data manipulation language (DML), and may include CONNECT, SELECT, INSERT, UPDATE, DELETE, EXECUTE, and USAGE.

In the Oracle database, executing a DCL command issues an implicit commit

- ✓ GRANT-gives user's access privileges to database.
- ✓ REVOKE-withdraw user's access privileges given by using the GRANT command.

Provide user with space on tablespace to store table

Allowing a user to create table is not enough to start storing data in that table. We also must provide the user with priviliges to use the available tablespace for their table and data.

ALTER USER username QUOTA UNLIMITED ON SYSTEM;
The above command will alter the user details and will provide it access to unlimited tablespace on system.

Grant all privilege to a User

GRANT sysdba TO username;

Grant permission to create any table

Sometimes user is restricted from creating come tables with names which are reserved for system tables.,

GRANT CREATE ANY TABLE TO username;

Grant permission to drop any table

privilege to the user,

GRANT DROP ANY TABLE TO username

To take back Permissions
And, if you want to take back the privileges from any user, use the REVOKE command.

REVOKE CREATE TABLE FROM username.

Commit Command,

- COMMIT command is used to permanently save any transaction into the database.
- When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by

these commands can be rolled back.

- To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's syntax,

COMMIT;

ROLLBACK command

- This command restores the database to last commited state.
- It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.
- If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

Following is rollback command's syntax,

ROLLBACK TO savepoint_name;

SAVEPOINT command

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

SAVEPOINT savepoint_name;
this command we can name the different states of our data in any table and then rollback to that state using the ROLLBACK command whenever required.

Video Content / Details of website for further learning (if any):
https://www.studytonight.com/dbms/dcl-command.php
https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/
https://www.studytonight.com/dbms/tcl-command.php

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No:120 - 128

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 14

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Views - Advanced SQL

Introduction : (Maximum 5 sentences)

- A database view is a searchable object in a database that is defined by a query.
- Though a view doesn't store data, some refer to a views as "virtual tables," you can query a view like you can a table.
- A view can combine data from two or more table, using joins, and also just contain a subset of information.
- Basic SQL statements for storing, retrieving, and manipulating data in a relational database.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Data manipulation language
- Data definition language
- Transaction Control language
- Data control Language
- Views

Detailed content of the Lecture:

- ✓ **V**iews in SQL are considered as a virtual table. A view also contains rows and columns.
- ✓ To create the view, we can select the fields from one or more tables present in the database.
- ✓ A view can either have specific rows based on certain condition or all the rows of a table.

1. Creating view

- ✓ A view can be created using the **CREATE VIEW** statement.
- ✓ We can create a view from a single table or multiple tables.

Syntax:

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

2. Creating View from a single table

we create a View named Details View from the table Student_Detail.

Query:

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM Student_Details
WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

- ✓ View from multiple tables can be created by simply include multiple tables in the SELECT statement.
- ✓ A view is created named Marks View from two tables Student_Detail and Student_Marks.

Query:

```
CREATE VIEW MarksView AS
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS
FROM Student_Detail, Student_Mark
```

WHERE Student_Detail.NAME = Student_Marks.NAME;

To display data of View MarksView:

SELECT * FROM MarksView;

4. Deleting View

A view can be deleted using the Drop View statement.

SQL > Advanced SQL

- sql union
- sql union all
- sql inline view
- sql intersect
- sql minus
- sql limit
- sql top
- sql subquery
- Sql exists
- sql case
- sql decode
- sql auto increment
- sql identity
- sql sequence and nextval

Video Content / Details of website for further learning (if any):
https://www.javatpoint.com/dbms-sql-view
https://www.1keydata.com/sql/advanced.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 157 - 170

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L 15

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code :**Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit :**II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Embedded SQL

Introduction : (Maximum 5 sentences)

- Embedded SQL is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL.
- Embedded SQL statements are SQL statements written inline with the program source code, of the host language

Prerequisite knowledge for Complete understanding and learning of Topic:

- ✓ SQL Queries
- ✓ Views
- ✓ Advanced SQL

Detailed content of the Lecture:

- ✓ Embedded SQL is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL.
- ✓ Embedded SQL statements are SQL statements written inline with the program source code, of the host language
- ✓ QL stands for Structured Query Language, it provides as a declarative query language. However, a general-purpose programming language requires to get access to the database because

- SQL is not as powerful as any of the general purpose language available today.
- There are many declarative actions such as interacting with the user sending the result to a GUI or printing a report which we cannot do using SQL.
- There are many queries that we can express in C, Pascal, Cobol and many more but we cannot express in SQL.
- It allows the application languages to communicate with DB and get requested result.
- The high level languages which supports embedding SQLs within it are also known as host language.

- There are different host languages which support embedding SQL within it like C, C++, ADA, Pascal, FORTRAN, Java etc.
- When SQL is embedded within C or C++, then it is known as Pro*C/C++ or simply Pro*C language. Pro*C is the most commonly used embedded SQL.

Structure of Embedded SQL

Structure of embedded SQL defines step by step process of establishing a connection with DB and executing the code in the DB within the high level language.

Connection to DB

- ✓ This is the first step while writing a query in high level languages.
- ✓ First connection to the DB that we are accessing needs to be established.
- ✓ This can be done using the keyword CONNECT. But it has to precede with 'EXEC SQL' to indicate that it is a SQL statement.

EXEC SQL CONNECT db_name;

EXEC SQL CONNECT HR_USER; //connects to DB HR_USER

Declaration Section

- ✓ Once connection is established with DB, we can perform DB transactions.
- ✓ These DB transactions are dependent on the values and variables of the host language. Depending on their values, query will be written and executed.
- ✓ Similarly, results of DB query will be returned to the host language which will be captured by the variables of host language.
- ✓ Hence we need to declare the variables to pass the value to the query and get the values from query.

There are two types of variables used in the host language.

Host variable :

These are the variables of host language used to pass the value to the query as well as to capture the values returned by the query EXEC SQL BEGIN DECLARE SECTION;

```
int STD_ID;
char STD_NAME [15];
```

```
        char ADDRESS[20];
EXEC SQL END DECLARE SECTION;
```

Indicator Variable :

- ✓ These variables are also host variables but are of 2 byte short type always.
- ✓ These variables are used to capture the NULL values that a query returns or to INSERT/ UPDATE any NULL values to the tables

```
EXEC SQL SELECT STD_NAME INTO :SNAME :IND_SNAME
                        FROM STUDENT WHERE STUDENT_ID =:STD_ID;
```

Error Handling

Like any other programming language, in embedded SQL also we need to handle errors.

Error handling method would be based on the host language.

```
struct sqlca {
   /* ub1 */ char sqlcaid [8];
   /* b4 */ long sqlabc;
   /* b4 */ long sqlcode;
   struct {
      /* ub2 */ unsigned short sqlerrml;
      /* ub1 */ char sqlerrmc[70];
   } sqlerrm;
….
long sqlcode; //returns the error code
…
char sqlstate [6]; //returns predefined error statements
….
}
```

Video Content / Details of website for further learning (if any):

https://data-flair.training/blogs/embedded-sql/
https://www.tutorialcup.com/dbms/embedded-sql.htm

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No:169 -178

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 16

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Static Vs Dynamic SQL

Introduction : (Maximum 5 sentences)
Static or Embedded SQL are SQL statements in an application that do not change at runtime and, therefore, can be hard-coded into the application.
Dynamic SQL is SQL statements that are constructed at runtime; for example, the application may allow users to enter their own queries.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ✓ **SQL query Language**

Detailed content of the Lecture:

- ✓ Static or Embedded SQL are SQL statements in an application that do not change at runtime and, therefore, can be hard-coded into the application.
- ✓ Dynamic SQL is SQL statements that are constructed at runtime; for example, the application may allow users to enter their own queries

- Dynamic SQL queries are prepared at program execution time, not compilation time.
- This means that the compiler cannot check for errors at compilation time and preprocessor macros cannot be used within Dynamic SQL.
- It also means that executing programs can create specialized Dynamic SQL queries in response to user or other input.
- Dynamic SQL can issue a CREATE TABLE or CREATE VIEW and perform an INSERT or SELECT on that table or view in the same routine.

- Embedded SQL, because it is compiled, cannot do this.
- Dynamic SQL executes slightly less efficiently than Embedded SQL, because it does not generate in-line code for queries.
- However, re-execution of a Dynamic SQL query is substantially faster than the first execution of the query because Dynamic SQL supports cached queries.
- Dynamic SQL can accept a literal value input to a query in two ways: input parameters specified using the "?" character, and input host variables (for example, :var). Embedded SQL uses input and output host variables (for example, :var).

- Dynamic SQL output values are retrieved using the API of the result set object (that is, the Data property).
- Embedded SQL uses host variables (for example, :var) with the INTO clause of a SELECT statement to output values.
- Dynamic SQL sets the %SQLCODE, %Message, %ROWCOUNT, and %ROWID object properties.
- Embedded SQL sets the corresponding SQLCODE, %msg, %ROWCOUNT, and %ROWID local variables. Dynamic SQL does not set %ROWID for a SELECT query;
- Embedded SQL sets %ROWID for a cursor-based SELECT query.
- Dynamic SQL provides an easy way to find query metadata (such as quantity and names of columns).
- Queries prepared by Dynamic SQL are maintained within the query cache so that subsequent calls to prepare the same query can reuse previously generated code.
- Embedded SQL generates in-line code at compilation time and does not need to use the query cache.
- Dynamic SQL does not cache most non-query SQL statements, because these statements are commonly only used once. Refer to the "Cached Queries" chapter of the SQL Optimization Guide for further details.

- Dynamic SQL performs SQL privilege checking; you must have the appropriate privileges to access or modify a table, field, etc.
- Embedded SQL does not perform SQL privilege checking. Refer to the SQL %CHECKPRIV statement for further details.
- Dynamic SQL cannot access a private class method.
- To access an existing class method, the method must be made public.
- This is a general SQL limitation.
- However, Embedded SQL gets around this limitation because the Embedded SQL operation itself is a method of the same class.

Video Content / Details of website for further learning (if any):
https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQL_dynsql_vs_esql

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 158

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 17

CSE

II/IV

Course Name with Code **:Database Management System/16CSD03**

Course Teacher **: R.Vinupriya**

Unit **: II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Query Processing - Query Optimization

Introduction : (Maximum 5 sentences)

- ✓ Query Processing would mean the entire process or activity which involves query translation into low level instructions,
- ✓ Query optimization to save resources, cost estimation or evaluation of query, and extraction of data from the database

Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- SQL Queries
- Parser
- Query Processing

Detailed content of the Lecture:

- ➢ Query Processing is a translation of high-level queries into low-level expression.
- ➢ It is a step wise process that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result.
- ➢ It refers to the range of activities that are involved in extracting data from the database.
- ➢ It includes translation of queries in high-level database languages into expressions that can be implemented at the physical level of the file system.

- In query processing, we will actually understand how these queries are processed and how they are optimized.

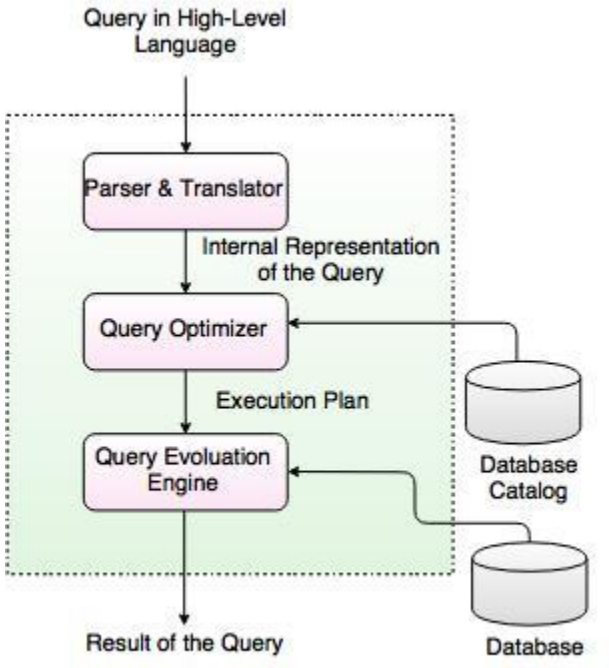


Fig. Query Processing

-
- A query is translated into SQL and into a relational algebraic expression.
- During this process, Parser checks the syntax and verifies the relations and the attributes which are used in the query.
- The second step is Query Optimizer.
- It transforms the query into equivalent expressions that are more efficient to execute.
- The third step is Query evaluation. It executes the above query execution plan and returns the result.

Translating SQL Queries into Relational Algebra

Example

```
SELECT Ename FROM Employee
  WHERE Salary > 5000;
```

Translated into Relational Algebra Expression

σ Salary > 5000 (π Ename (Employee))

OR

π Ename (σ Salary > 5000 (Employee))

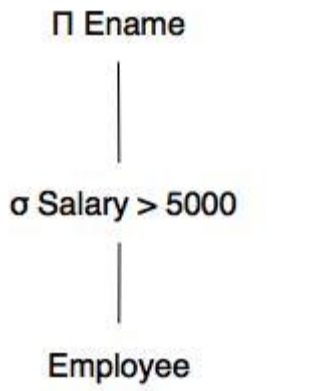


Fig. Query Execution Plan

- A sequence of primitive operations that can be used to evaluate a query is a Query Execution Plan or Query Evaluation Plan.
- It indicates that the query execution engine takes a query execution plan and returns the answers to the query.
- Query Execution Plan minimizes the cost of query evaluation.

Importance of Query Optimization

- Query optimization provides faster query processing.
- It requires less cost per query.
- It gives less stress to the database.
- It provides high performance of the system.
- It consumes less memory.

Video Content / Details of website for further learning (if any):
https://www.tutorialride.com/dbms/sql-query-processing.html
https://www.tutorialcup.com/dbms/query-optimization.htm

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 537 – 568, 579

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 18

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **II SQL & Query Optimization** **Date of Lecture:**

Topic of Lecture: Heuristic and Cost based Query Optimization

Introduction : (Maximum 5 sentences)

- Query optimization is the part of the query process in which the database system compares different query strategies and chooses the one with the least expected cost.
- The query optimizer, which carries out this function, is a key part of the relational database and determines the most efficient way to access data.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Database
- SQL Queries
- Parser
- Query Processing

Detailed content of the Lecture:

Query optimization is a difficult part of the query processing.
It determines the efficient way to execute a query with different possible query plans.
It cannot be accessed directly by users once the queries are submitted to the database server or parsed by the parser.
A query is passed to the query optimizer where optimization occurs.
Main aim of Query Optimization is to minimize the cost function,
I/O Cost + CPU Cost + Communication Cost
It defines how an RDBMS can improve the performance of the query by re-ordering the operations.

It is the process of selecting the most efficient query evaluation plan from among various strategies if the query is complex.

It computes the same result as per the given expression, but it is a least costly way of generating result.

There are two methods of query optimization.

1. Cost based Optimization (Physical)

This is based on the cost of the query. The query can use different paths based on indexes, constraints, sorting methods etc. This method mainly uses the statistics like record size, number of records, number of records per block, number of blocks, table size, whether whole table fits in a block, organization of tables, uniqueness of column values, size of columns etc.
Suppose, we have series of table joined in a query.
T1 ∞ T2 ∞ T3 ∞ T4∞ T5 ∞ T6

- Dynamic Programming

As we learnt above, the least cost for the joins of any combination of table is generated here. These values are stored in the database and when those tables are used in the query, this combination is selected for evaluating the query.

- Left Deep Trees

This is another method of determining the cost of the joins. Here, the tables and joins are represented in the form of trees. The joins always form the root of the tree and table is kept at the right side of the root. LHS of the root always point to the next join. Hence it gets deeper and deeper on LHS. Hence it is called as left deep tree.

- Interesting Sort Orders

This method is an enhancement to dynamic programming. Here, while calculating the best join order costs, it also considers the sorted tables. It assumes, calculating the join orders on sorted tables would be efficient. i.e.; suppose we have unsorted tables T1 , T2 , T3 .. Tn and we have join on these tables.
(T1 ∞T2)∞ T3 ∞… ∞ Tn

2. Heuristic Optimization (Logical)

This method is also known as rule based optimization. This is based on the equivalence rule on relational expressions; hence the number of combination of queries get reduces here. Hence the cost of the query too reduces.

Video Content / Details of website for further learning (if any):
https://www.tutorialride.com/dbms/sql-quel.htm
https://www.youtube.com/watch?v=RoK_TL4JqlE

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No: 582 - 598

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 19

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Functional Dependencies - Codd's Rule

Introduction : (Maximum 5 sentences)

- Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system.
- Functional dependency helps you to maintain the quality of data in the database.
- A functional dependency is denoted by an arrow →.
- The functional dependency of X on Y is represented by X → Y. Functional Dependency plays a vital role to find the difference between good and bad database design

Prerequisite knowledge for Complete understanding and learning of Topic:

- Multivalued dependency
- Trivial functional dependency
- Non-trivial functional dependency
- Transitive dependency

Detailed content of the Lecture:

- ✓ Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system.
- ✓ Functional dependency helps you to maintain the quality of data in the database.
- ✓ A functional dependency is denoted by an arrow →.
- ✓ The functional dependency of X on Y is represented by X → Y. Functional Dependency plays a vital role to find the difference between good and bad database design.

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

Rules of Functional Dependencies

Below given are the Three most important rules for Functional Dependency:

- Reflexive rule –. If X is a set of attributes and Y is_subset_of X, then X holds a value of Y.
- Augmentation rule: When x -> y holds, and c is attribute set, then ac -> bc also holds. That is adding attributes which do not change the basic dependencies.
- Transitivity rule: This rule is very much similar to the transitive rule in algebra if x -> y holds and y -> z holds, then x -> z also holds. X -> y is called as functionally that determines y.

Types of Functional Dependencies

- Multivalued dependency:
- Trivial functional dependency:
- Non-trivial functional dependency:
- Transitive dependency:

Multivalued dependency in DBMS

- ✓ Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table.
- ✓ A multivalued dependency is a complete constraint between two sets of attributes in a relation. It requires that certain tuples be present in a relation.

Example:

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue

- ➢ In this example, maf_year and color are independent of each other but dependent on car_model. In this example, these two columns are said to be multivalue dependent on car_model.
- ➢ This dependence can be represented like this:
- ➢ car_model -> maf_year
- ➢ car_model-> colour

Trivial Functional dependency:

- ✓ The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.
- ✓ So, X -> Y is a trivial functional dependency if Y is a subset of X.

For example:

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Consider this table with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

Non trivial functional dependency in DBMS

- ✓ Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A.
- ✓ In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

Example:
(Company} -> {CEO} (if we know the Company, we knows the CEO name)

But CEO is not a subset of Company, and hence it's non-trivial functional dependency.

CODD'S RULES:

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Rule 1: Information Rule

- ✓ The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

- ✓ Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

- ✓ The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

- ✓ The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users.
- ✓ Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

- ✓ A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations.
- ✓ This language can be used directly or by means of some application. If the database allows

access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

- ✓ A database must support high-level insertion, updation, and deletion.
- ✓ This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

- ✓ The data stored in a database must be independent of the applications that access the database.
- ✓ Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

- ✓ The logical data in a database must be independent of its user's view (application).
- ✓ Any change in logical data must not affect the applications using it.
- ✓ For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

- ✓ A database must be independent of the application that uses it.
- ✓ All its integrity constraints can be independently modified without the need of any change in the application.
- ✓ This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

- ✓ The end-user must not be able to see that the data is distributed over various locations.
- ✓ Users should always get the impression that the data is located at one site only.
- ✓ This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

- ✓ If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Video Content / Details of website for further learning (if any):

https://www.guru99.com/dbms-functional-dependency.html

https://www.tutorialspoint.com/dbms/dbms_codds_rules.htm

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 260 - 270

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L - 20

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Normalization

Introduction : (Maximum 5 sentences)

- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.
- Managing a database with anomalies is next to impossible.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ✓ Functional Dependencies
- ✓ Codd's Rule
- ✓ Normalization

Detailed content of the Lecture:

- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.
- Managing a database with anomalies is next to impossible.

Normalization:

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

First Normal Form:

- ✓ First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.
- ✓ We re-arrange the relation (table) as below, to convert it to First Normal Form.
- ✓ Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form:

- ✓ Before we learn about the second normal form, we need to understand the following −

- ✓ **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
- ✓ **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.
That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.

Third Normal Form:

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
  - ➢ X is a superkey or,
  - ➢ A is prime attribute.

Boyce-Codd Normal Form:

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,
Stu_ID → Stu_Name, Zip
and
Zip → City
Which confirms that both the relations are in BCNF.

Fourth Normal Form (4NF)

- Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce- Codd Normal Form (BCNF).

It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency

Fifth Normal Form (5NF)

A database is said to be in 5NF, if and only if,

- It's in 4NF.
- If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise.
- In simple words, joining two or more decomposed table should not lose records nor create new records.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/dbms/database_normalization.htm

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 257 - 296

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L - 21

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Non-loss decomposition

Introduction : (Maximum 5 sentences)

- ✓ Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ✓ Functional Dependencies
- ✓ Codd's Rule
- ✓ Normalization

Detailed content of the Lecture:

Non-loss decomposition:

Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

The following are the types:

Lossless Decomposition:

- ✓ Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins.. The information will not lose from the relation when decomposed.
- ✓ The join would result in the same original relation.

Let us see an example:

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Decompose the above table into two tables:

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama

E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	Emp_ID	Dept_Name
Dpt1	E001	Operations
Dpt2	E002	HR
Dpt3	E003	Finance

Now, Natural Join is applied on the above two tables:

The result will be:

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Therefore, the above relation had lossless decomposition i.e. no loss of information.

Lossy Decomposition:

As the name suggests, when a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.

Let us see an example:

<EmpInfo>

Emp_ID	**Emp_Name**	**Emp_Age**	**Emp_Location**	**Dept_ID**	**Dept_Name**
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Decompose the above table into two tables:

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	Dept_Name
Dpt1	Operations
Dpt2	HR
Dpt3	Finance

Now, you won't be able to join the above tables, since Emp_ID isn't part of the Dept Details relation. Therefore, the above relation has lossy decomposition.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/Lossless-and-Lossy-Decomposition-in-DBMS

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 271 - 276

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 22

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher :**R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: 1NF to 5NF

Introduction : (Maximum 5 sentences)

- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.
- Managing a database with anomalies is next to impossible.

Normalization:

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.

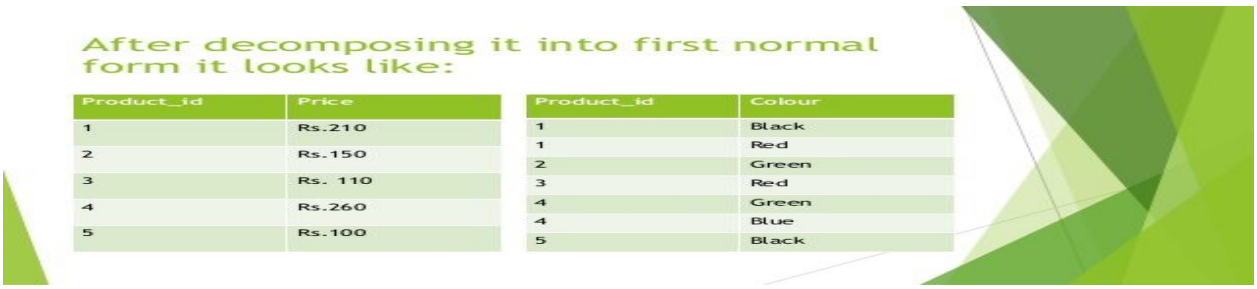
Prerequisite knowledge for Complete understanding and learning of Topic:

- ✓ Functional Dependencies
- ✓ Codd's Rule
- ✓ Normalization

Detailed content of the Lecture:

First Normal Form:

- ✓ First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.
- ✓ We re-arrange the relation (table) as below, to convert it to First Normal Form.
- ✓ Each attribute must contain only a single value from its pre-defined domain.

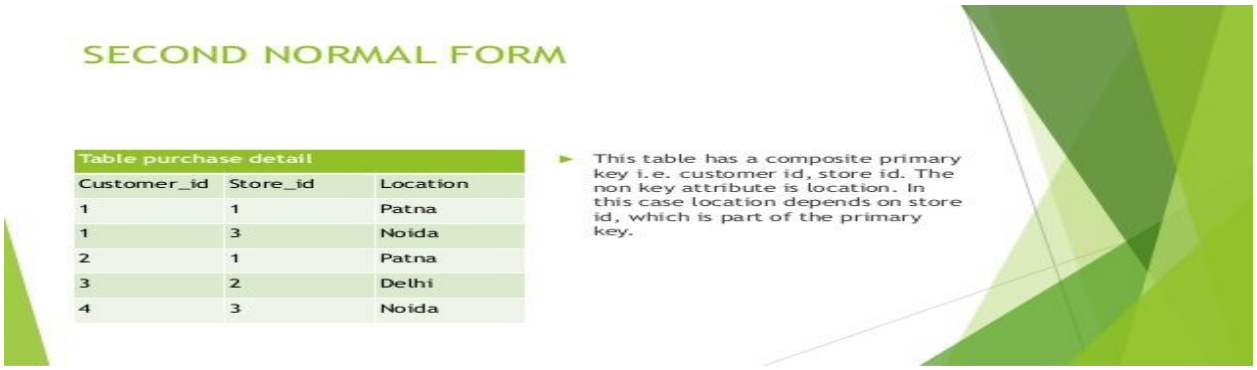


Second Normal Form:

- ✓ Before we learn about the second normal form, we need to understand the following −
- ✓ **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
- ✓ **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.

That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.

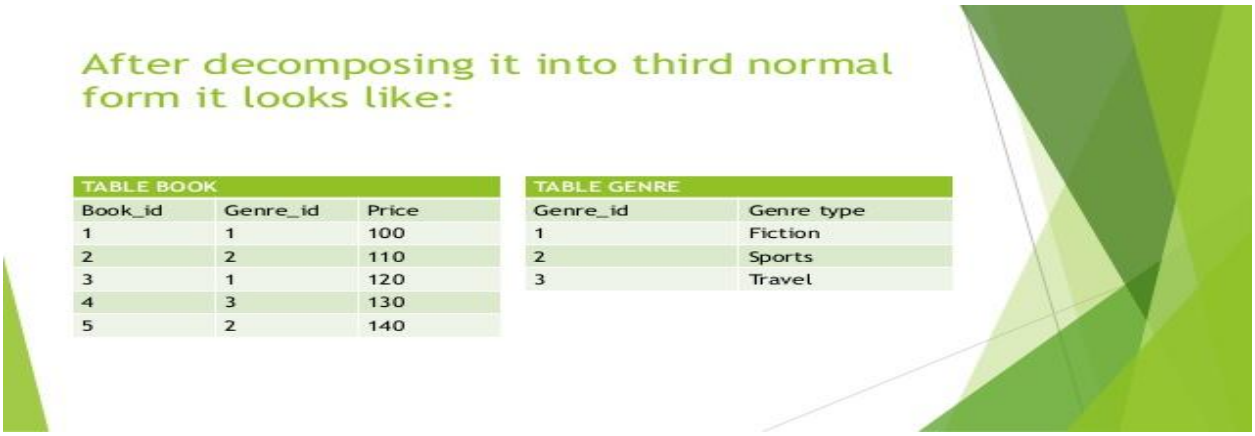


Third Normal Form:

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
  - ➢ X is a superkey or,

- A is prime attribute.



Boyce-Codd Normal Form:

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

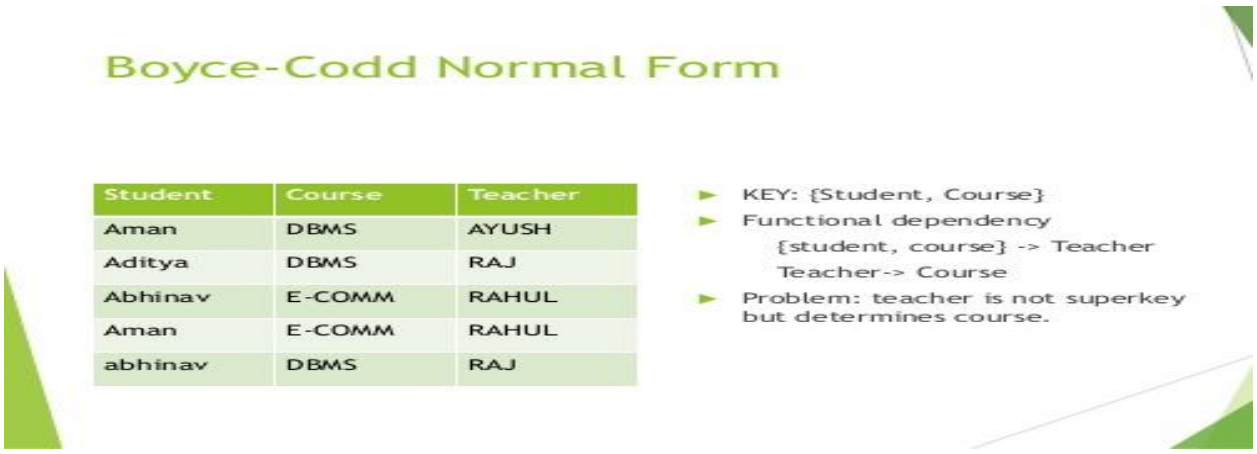
In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,

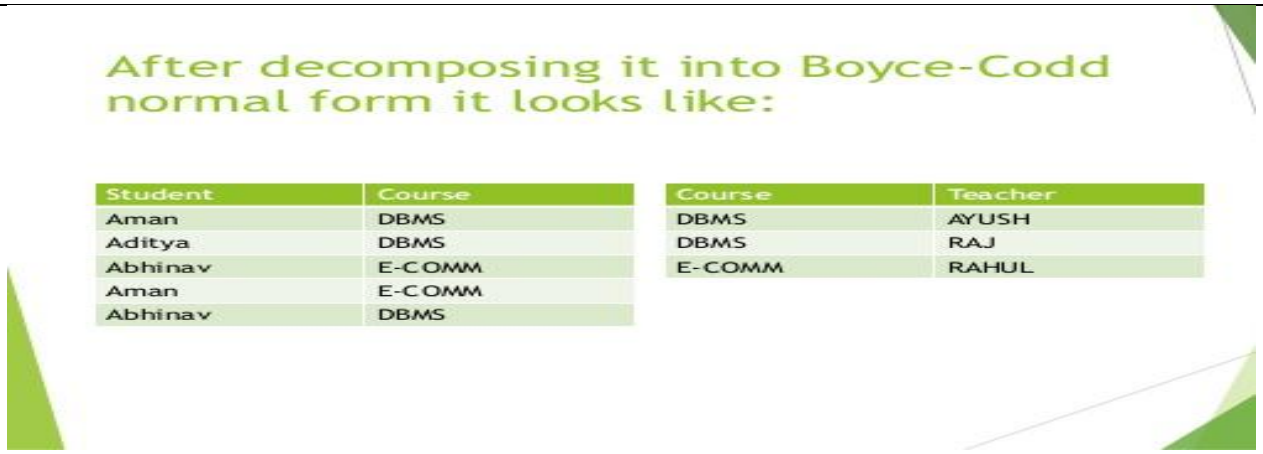
Stu_ID → Stu_Name, Zip

and

Zip → City

Which confirms that both the relations are in BCNF.

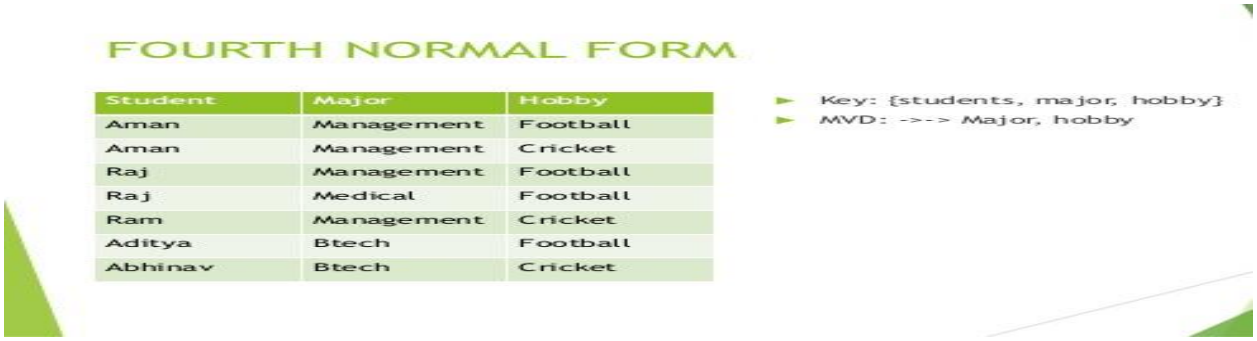




Fourth Normal Form (4NF)

- Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce- Codd Normal Form (BCNF).

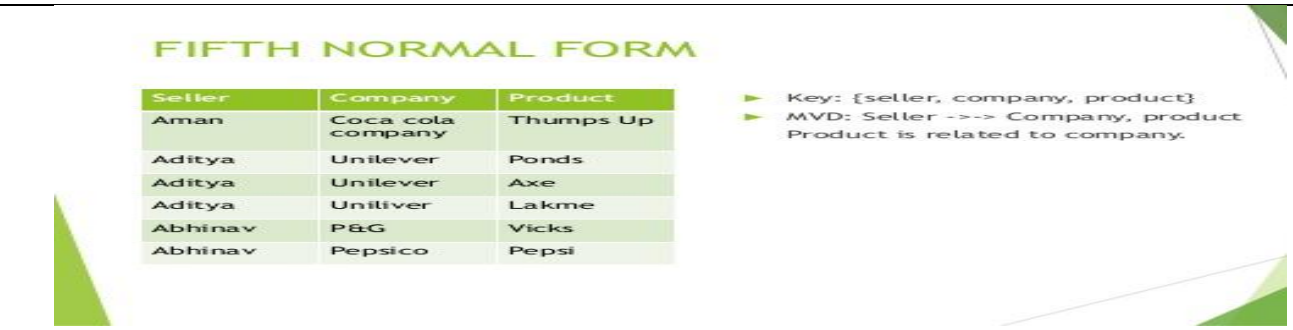
It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency

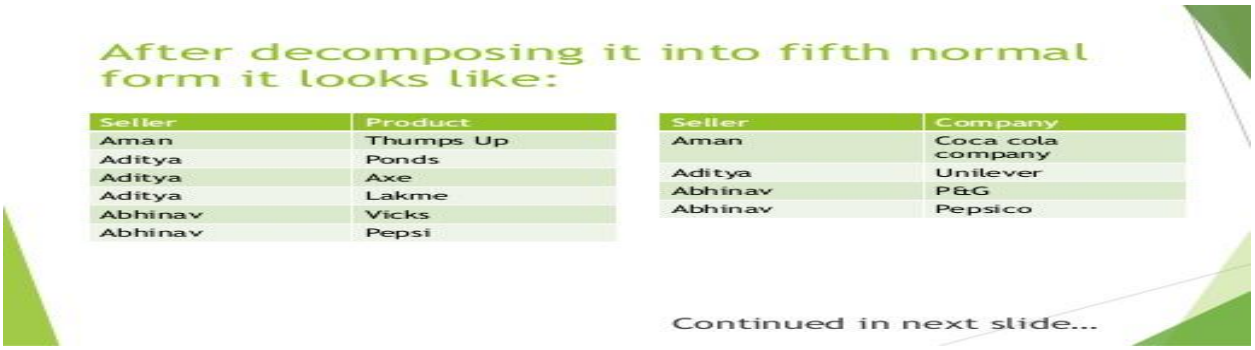


Fifth Normal Form (5NF)

A database is said to be in 5NF, if and only if,

- It's in 4NF.
- If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise.
- In simple words, joining two or more decomposed table should not lose records nor create new records.





Video Content / Details of website for further learning (if any):

https://www.studytonight.com/dbms/database-normalization.php

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 257 - 293

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 23

LECTURE HANDOUTS

CSE **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Domain Key Normal Form- Denormalization

Introduction : (Maximum 5 sentences)

- ✓ A relation is in DKNF when insertion or delete anomalies are not present in the database.
- ✓ Domain-Key Normal Form is the highest form of Normalization.
- ✓ The reason is that the insertion and updation anomalies are removed.
- ✓ The constraints are verified by the domain and key constraints.
- ✓ A table is in Domain-Key normal form only if it is in 4NF, 3NF and other normal forms

Prerequisite knowledge for Complete understanding and learning of Topic:

- Functional Dependencies
- Codd's Rule
- Normalization
- Non-loss decomposition

Detailed content of the Lecture:

DOMIN KEY NORMAL FORM:

- ✓ A relation is in DKNF when insertion or delete anomalies are not present in the database.
- ✓ Domain-Key Normal Form is the highest form of Normalization.
- ✓ The reason is that the insertion and updation anomalies are removed.
- ✓ The constraints are verified by the domain and key constraints.

Domain Constraint:

Values of an attribute had some set of values, for example, EmployeeID should be four digits long:

EmpID	**EmpName**	**EmpAge**
0921	Tom	33
0922	Jack	31

Key Constraint:

An attribute or its combination is a candidate key

General Constraint:

Predicate on the set of all relations.

Every constraint should be a logical sequence of the domain constraints and key constraints applied to

the relation. The practical utility of DKNF is less.

- ✓ A table is in Domain-Key normal form only if it is in 4NF, 3NF and other normal forms

Denormalization

- ➢ It is a strategy used on a previously-normalized database to increase performance.
- ➢ In computing, denormalization is the process of trying to improve the read performance of a database, at the expense of losing some write performance, by adding redundant copies of data or by grouping data.
    - o A normalized design will often "store" different but related pieces of information in separate logical tables
- ➢ If these relations are stored physically as separate disk files, completing a database query that draws information from several relations (a join operation) can be slow.
- ➢ If many relations are joined, it may be prohibitively slow. There are two strategies for dealing with this.
    - o A denormalized data model is not the same as a data model that has not been normalized, and denormalization should only take place after a satisfactory level of normalization has taken place and that any required constraints and/or rules have been created to deal with the inherent anomalies in the design.
    - o For example, all the relations are in third normal form and any relations with join and multi-valued dependencies are handled appropriately.
- ➢ Examples of denormalization techniques include:
- ➢ "Storing" the count of the "many" elements in a one-to-many relationship as an attribute of the "one" relation
- ➢ Adding attributes to a relation from another relation with which it will be joined
- ➢ Star schemas, which are also known as fact-dimension models and have been extended to snowflake schemas
- ➢ Prebuilt summarization or OLAP cubes.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/Domain-Key-Normal-Form

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
Page No: 293 - 296

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 24

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Transaction Concepts - ACID Properties

Introduction : (Maximum 5 sentences)

- A transaction is a very small unit of a program and it may contain several lowlevel tasks.
- transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Transaction State
- ACID Properties

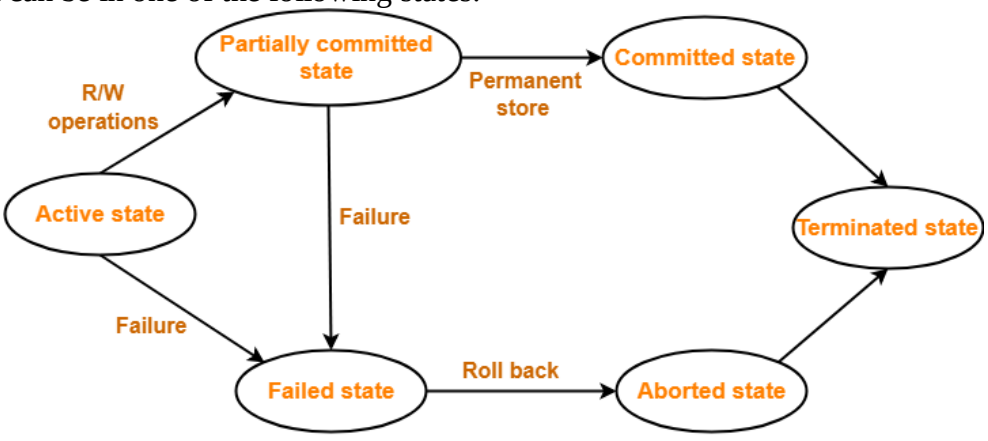
Detailed content of the Lecture:

Transaction:

A transaction is a very small unit of a program and it may contain several lowlevel tasks.

A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

A transaction can be in one of the following states:



Transaction States in DBMS

ACID Properties:

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

Atomicity

- ✓ By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially.

- ✓ Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.
  —**Abort**: If a transaction aborts, changes made to database are not visible.
  —**Commit**: If a transaction commits, changes made are visible.
  Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.
If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

Consistency

- ✓ This means that integrity constraints must be maintained so that the database is consistent before and after the transaction.
- ✓ It refers to the correctness of a database. Referring to the example above,
  The total amount before and after the transaction must be maintained.
  Total **before T** occurs = **500 + 200 = 700**.
  Total **after T occurs** = **400 + 300 = 700**.
  Therefore, database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result T is incomplete.

Isolation

- This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state.
- Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.

This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.
Let **X**= 500, **Y** = 500.
Consider two transactions **T** and **T".**
Suppose **T** has been executed till **Read (Y)** and then **T''** starts. As a result , interleaving of operations takes place due to which **T''** reads correct value of **X** but incorrect value of **Y** and sum computed by
T'': (X+Y = 50, 000+500=50, 500)
is thus not consistent with the sum at end of transaction:
T: (X+Y = 50, 000 + 450 = 50, 450).
This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Durability:

- ✓ This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs.
- ✓ These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

Video Content / Details of website for further learning (if any):
https://tutorialink.com/dbms/introduction-to-transaction-concepts.dbms
https://www.geeksforgeeks.org/acid-properties-in-dbms/

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 565 - 570

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 25

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Serializability-Concurrency Control

Introduction : (Maximum 5 sentences)

- ✓ Serializability is a concurrency scheme where the concurrent transaction is equivalent to one that executes the transactions serially.
- ✓ A schedule is a list of transactions. Serial schedule defines each transaction is executed consecutively without any interference from other transactions.
- ✓ Non-serial schedule defines the operations from a group of concurrent transactions that are interleaved.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Normalization
- Domain Key Normal Form
- Denormalization
- Transaction Concepts
- ACID Properties

Detailed content of the Lecture:

SERIALIZABILITY:

- ✓ Serializability is a concurrency scheme where the concurrent transaction is equivalent to one that executes the transactions serially.
- ✓ A schedule is a list of transactions.Serial schedule defines each transaction is executed consecutively without any interference from other transactions.
- ✓ Non-serial schedule defines the operations from a group of concurrent transactions that are interleaved.

Conflict serializability:

- ✓ Conflict serializability defines two instructions of two different transactions accessing the same data item to perform a read/write operation.
- ✓ It deals with detecting the instructions that are conflicting in any way and specifying the order in which the instructions should execute in case there is any conflict.
- ✓ A conflict serializability arises when one of the instruction is a write operation.

The following rules are important in Conflict Serializability,

1. If two transactions are both read operation, then they are not in conflict.
2. If one transaction wants to perform a read operation and other transaction wants to perform a write operation, then they are in conflict and cannot be swapped.
3. If both the transactions are for write operation, then they are in conflict, but can be allowed to take place in any order, because the transactions do not read the value updated by each other.

View Serializability

1. View serializability is the another type of serializability.
2. It can be derived by creating another schedule out of an existing schedule and involves the same set of transactions.

If a concurrent schedule is view equivalent to a serial schedule of same transaction then it is said to be View serializable.

concurrency control:

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions.

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it.

Locks are of two kinds −

- Binary Locks − A lock on a data item can be in two states; it is either locked or unlocked.
- Shared/exclusive − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

There are four types of lock protocols available −

Simplistic Lock Protocol

Simplistic lock-based protocols allow transactions to obtain a lock on every object before a 'write' operation is performed. Transactions may unlock the data item after completing the ‘write’ operation.

Pre-claiming Lock Protocol

- ✓ Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks.
- ✓ Before initiating an execution, the transaction requests the system for all the locks it needs beforehand.
- ✓ If all the locks are granted, the transaction executes and releases all the locks when all its operations are over.
- ✓ If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.

Two-Phase Locking 2PL:

- ✓ This locking protocol divides the execution phase of a transaction into three parts.
- ✓ In the first part, when the transaction starts executing, it seeks permission for the locks it requires.
- ✓ The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.
- ✓ Two-phase locking has two phases, one is **growing**, where all the locks are being acquired by the transaction; and the second phase is shrinking, where the locks held by the transaction are being released.
- ✓ To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking

- ✓ The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally.
- ✓ But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.
- ✓ Strict-2PL does not have cascading abort as 2PL does.

Timestamp-based Protocols

- The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.
- Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.
- Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction.
- A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.
- In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Timestamp Ordering Protocol

- The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations.
- This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.
- The timestamp of transaction T_i is denoted as $TS(T_i)$.
- Read time-stamp of data-item X is denoted by R-timestamp(X).
- Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows −

- **If a transaction Ti issues a read(X) operation −**
  - If TS(Ti) < W-timestamp(X)
    - Operation rejected.
  - If TS(Ti) >= W-timestamp(X)
    - Operation executed.
  - All data-item timestamps updated.
- **If a transaction Ti issues a write(X) operation −**
  - If TS(Ti) < R-timestamp(X)
    - Operation rejected.
  - If TS(Ti) < W-timestamp(X)
    - Operation rejected and Ti rolled back.
  - Otherwise, operation executed.

Thomas' Write Rule

This rule states if TS(Ti) < W-timestamp(X), then the operation is rejected and T_i is rolled back.

Time-stamp ordering rules can be modified to make the schedule view serializable.

Instead of making T_i rolled back, the 'write' operation itself is ignored.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/dbms/dbms_concurrency_control.htm

https://www.tutorialride.com/dbms/serializability-in-transaction-control.htm

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 576 - 584

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L - 26

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Locking Mechanisms

Introduction : (Maximum 5 sentences)

- ✓ Locking mechanisms are a way for databases to produce sequential data output without the sequential steps.
- ✓ The locks provide a method for securing the data that is being used so no anomalies can occur like lost data or additional data that can be added because of the loss of a transaction.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Serializability
- ➢ Concurrency Control

Detailed content of the Lecture:

Locking mechanism:

- ✓ Locking mechanisms are a way for databases to produce sequential data output without the sequential steps.
- ✓ The locks provide a method for securing the data that is being used so no anomalies can occur like lost data or additional data that can be added because of the loss of a transaction.
- ✓ Problems that Locks Solve Lost Update Problem- An update can get lost if two or more transactions try to update the same data.
- ✓ The two transactions are unaware of each other, and data can be overwritten. Temporary Update Problem- If one transaction updates the database and then fails, another transaction can read the incorrect value which lowers the integrity of the database.
- ✓ Incorrect Summary Problem- If one transaction is calculating an aggregate while another transaction is u same data the integrity would be compromised.
- ✓ Phantom Reads- When an insert or delete is performed on a row that is being used by another transaction the integrity of the data is compromised.

Different Types of Locks :

There are three primary types of locks that are used in a database.

Read Locks

- ✓ These types of locks make it so that data can only be read.
- ✓ Depending on the database system and restrictions on the read lock, this can make it so only one user can read the data to allowing every user access to reading the data but not being able to modify anything.
- ✓ The read lock can be applied to a single row, a section of rows, or an entire table.
- ✓ This can also be dependent on the type of database system that is being used that could limit the amount of data that can be locked for reading.

Write Locks

- ✓ This type of lock is used for updates in the database system.
- ✓ When this lock is applied it prevents any other transactions from changing the records that are being accessed.
- ✓ This does allow transactions to read the data before it is modified and the changes are made permanent.

Exclusive Write Locks

- ✓ This type of lock is similar to a write lock.
- ✓ The only difference is that with an exclusive write lock, the only things that can look at the data or modify the data is the original transaction.
- ✓ No other transaction can read the data while this lock is applied.
- ✓ There are transactions what types of locks the transaction has throughout the hierarchy of data levels.

Multi-Level-Locks Update Lock-

Signals intention to request an exclusive lock in the near future.

- ✓ IS Lock- Signals intent to request shared (read) lock in the near future.
- ✓ IX Lock- Signals intent to request an exclusive (write) lock in the near future.

Two-Phase Locking Protocol:

- ✓ The Two Phase Commit is designed to coordinate the transactions of the requests to the system.
- ✓ The idea behind the protocol is to produce serialized results from a non-serialized system.
- ✓ This protocol requires that each transaction issues lock and unlock requests in two phases: the shrinking phase and the growing phase.
- ✓ During the growing phase transactions may obtain locks, but cannot release any.
- ✓ During the shrinking phase transactions may release locks but may not obtain any new locks.
- ✓ By following this protocol any update problems with the transaction can be detected and one transaction gets rolled back.
- ✓ It also can raise the priority of the affected transaction to prevent a repeat of the problem.

Video Content / Details of website for further learning (if any):
https://databasemanagement.fandom.com/wiki/Locking_Mechanisms

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 591 - 606

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L – 27

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **III Relational Database Design and Transactions**

Date of Lecture:

Topic of Lecture: Two Phase Commit Protocol - Dead lock

Introduction : (Maximum 5 sentences)

- ✓ The two phase commit protocol is a distributed algorithm which lets all sites in a distributed system agree to commit a transaction.
- ✓ The protocol results in either all nodes committing the transaction or aborting, even in the case of site failures and message losses.
- ✓ Deadlocks are not healthy for a system.
- ✓ In case a system is stuck in a deadlock, the transactions involved in the deadlock are either rolled back or restarted

Prerequisite knowledge for Complete understanding and learning of Topic:

Serializability
Concurrency Control
Locking Mechanisms

Detailed content of the Lecture:

two phase commit protocol:

- ✓ The two phase commit protocol is a distributed algorithm which lets all sites in a distributed system agree to commit a transaction.
- ✓ The protocol results in either all nodes committing the transaction or aborting, even in the case of site failures and message losses.
- ✓ However, due to the work by Skeen and Stone braker, the protocol will not handle more than one random site failure at a time.
- ✓ The two phases of the algorithm are broken into the COMMIT-REQUEST phase, where the COORDINATOR attempts to prepare all the COHORTS, and the COMMIT phase, where the COORDINATOR completes the transactions at all COHORTS.

Assumptions

- ✓ The protocol works in the following manner: One node is designated the coordinator, which is the master site, and the rest of the nodes in the network are called cohorts.
- ✓ The protocol include stable storage at each site and use of a write ahead log by each node.
- ✓ Also, the protocol assumes that no node crashes forever, and eventually any two nodes can communicate with each other.

Disadvantages

- ✓ The greatest disadvantage of the two phase commit protocol is the fact that it is a blocking protocol.
- ✓ A node will block while it is waiting for a message.

- ✓ This means that other processes competing for resource locks held by the blocked processes will have to wait for the locks to be released.
- ✓ A single node will continue to wait even if all other sites have failed. If the coordinator fails permanently, some cohorts will never resolve their transactions.
- ✓ This has the effect that resources are tied up forever.

Dead lock:

- ✓ In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.
- ✓ For example, assume a set of transactions $\{T_0, T_1, T_2, ...,T_n\}$. T_0 needs a resource X to complete its task. Resource X is held by T_1, and T_1 is waiting for a resource Y, which is held by T_2. T_2 is waiting for resource Z, which is held by T_0. Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.
- ✓ Deadlocks are not healthy for a system. In case a system is stuck in a deadlock, the transactions involved in the deadlock are either rolled back or restarted.

Deadlock Prevention

- ✓ To prevent any deadlock situation in the system, the DBMS aggressively inspects all the operations, where transactions are about to execute.
- ✓ The DBMS inspects the operations and analyzes if they can create a deadlock situation.
- ✓ If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed.
- ✓ There are deadlock prevention schemes that use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation.

Wait-Die Scheme

In this scheme, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur −

- If $TS(T_i) < TS(T_j)$ − that is T_i, which is requesting a conflicting lock, is older than T_j − then T_i is allowed to wait until the data-item is available.
- If $TS(T_i) > TS(t_j)$ − that is T_i is younger than T_j − then T_i dies. T_i is restarted later with a random delay but with the same timestamp.

This scheme allows the older transaction to wait but kills the younger one.

Wound-Wait Scheme

In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur −

- If $TS(T_i) < TS(T_j)$, then T_i forces T_j to be rolled back − that is T_i wounds T_j. T_j is restarted later with a random delay but with the same timestamp.
- If $TS(T_i) > TS(T_j)$, then T_i is forced to wait until the resource is available.

This scheme, allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item.

In both the cases, the transaction that enters the system at a later stage is aborted.

Deadlock Avoidance

Aborting a transaction is not always a practical approach. Instead, deadlock avoidance mechanisms can be used to detect any deadlock situation in advance. Methods like "wait-for graph" are available

but they are suitable for only those systems where transactions are lightweight having fewer instances of resource.

Wait-for Graph

- ✓ This is a simple method available to track if any deadlock situation may arise. For each transaction entering into the system, a node is created.
- ✓ When a transaction T_i requests for a lock on an item, say X, which is held by some other transaction T_j, a directed edge is created from T_i to T_j. If T_j releases item X, the edge between them is dropped and T_i locks the data item.
- ✓ The system maintains this wait-for graph for every transaction waiting for some data items held by others. The system keeps checking if there's any cycle in the graph.

Video Content / Details of website for further learning (if any):
http://courses.cs.vt.edu/~cs5204/fall00/distributedDBMS/duckett/tpcp.html
https://www.tutorialspoint.com/dbms/dbms_deadlock.htm

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 591-- 615

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 28

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Overview of Physical Storage Media

Introduction : (Maximum 5 sentences)

- ✓ Several types of data storage exist in most computer systems. They vary in speed of access, cost per unit of data, and reliability.
- ✓ Cache: most costly and fastest form of storage. Usually very small, and managed by the operating system.
- ✓ Main Memory (MM): the storage area for data available to be operated on.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Cache
- ➢ Main Memory
- ➢ Flash memory
- ➢ Magnetic-disk storage
- ➢ Optical storage

Detailed content of the Lecture:

Over view of physical storage media:

- ➢ Several types of data storage exist in most computer systems. They vary in speed of access, cost per unit of data, and reliability.

- Cache: most costly and fastest form of storage. Usually very small, and managed by the operating system.
- Main Memory (MM): the storage area for data available to be operated on.
  - General-purpose machine instructions operate on main memory.
  - Contents of main memory are usually lost in a power failure or ``crash".
  - Usually too small (even with megabytes) and too expensive to store the entire database.
- Flash memory: EEPROM (electrically erasable programmable read-only memory).
  - Data in flash memory survive from power failure.
  - Reading data from flash memory takes about 10 nano-secs (roughly as fast as from main memory), and writing data into flash memory is more complicated: write-once takes about 4-10 microsecs.

- To overwrite what has been written, one has to first erase the entire bank of the memory. It may support only a limited number of erase cycles (to).
- It has found its popularity as a replacement for disks for storing small volumes of data (5-10 megabytes).

- Magnetic-disk storage: primary medium for long-term storage.
  - Typically the entire database is stored on disk.
  - Data must be moved from disk to main memory in order for the data to be operated on.
  - After operations are performed, data must be copied back to disk if any changes were made.
  - Disk storage is called direct access storage as it is possible to read data on the disk in any order (unlike sequential access).
  - Disk storage usually survives power failures and system crashes.
- Optical storage: CD-ROM (compact-disk read-only memory), WORM (write-once read-many) disk (for archival storage of data), and Juke box (containing a few drives and numerous disks loaded on demand).
- Tape Storage: used primarily for backup and archival data.

  ✓ Cheaper, but much slower access, since tape must be read sequentially from the beginning.
  ✓ Used as protection from disk failures!

➢ The storage device hierarchy is presented, where the higher levels are expensive (cost per bit), fast (access time), but the capacity is smaller in Storage-device hierarchy

➢ Another classification: Primary, secondary, and tertiary storage.

1. Primary storage: the fastest storage media, such as cash and main memory.
2. Secondary (or on-line) storage: the next level of the hierarchy, e.g., magnetic disks.
3. Tertiary (or off-line) storage: magnetic tapes and optical disk juke boxes.

➢ Volatility of storage. Volatile storage loses its contents when the power is removed. Without power backup, data in the volatile storage (the part of the hierarchy from main memory up) must be written to nonvolatile storage for safekeeping.

Video Content / Details of website for further learning (if any):
https://www2.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter10/node2.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 393-396

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 29

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: RAID - Tertiary storage - File Organization

Introduction : (Maximum 5 sentences)

- ✓ Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device.
- ✓ The memory storage that is directly accessible to the CPU comes under this category
- ✓ Secondary storage devices are used to store data for future use or as backup
- ✓ Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Cache
- ➢ Main Memory
- ➢ Flash memory
- ➢ Magnetic-disk storage
- ➢ Optical storage

Detailed content of the Lecture:

These storage devices can be broadly categorized into three types −

- **Primary Storage** − The memory storage that is directly accessible to the CPU comes under this category.
- **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.
- **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic **Magnetic Disks**

➢ Hard disk drives are the most common secondary storage devices in present computer systems.

➢ These are called magnetic disks because they use the concept of magnetization to store information. Hard disks consist of metal disks coated with magnetizable material.

- These disks are placed vertically on a spindle.
- A read/write head moves in between the disks and is used to magnetize or de-magnetize the spot under it. A magnetized spot can be recognized as 0 (zero) or 1 (one).
- Hard disks are formatted in a well-defined order to store data efficiently. A hard disk plate has many concentric circles on it, called **tracks**.
- Every track is further divided into **sectors**. A sector on a hard disk typically stores 512 bytes of data.
- Redundant Array of Independent Disks

RAID or **R**edundant **A**rray of **I**ndependent **D**isks, is a technology to connect multiple secondary storage devices and use them as a single storage media.

RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

RAID 0

- In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks.
- Each disk receives a block of data to write/read in parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.

RAID 1

- ✓ RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array.
- ✓ RAID level 1 is also called **mirroring** and provides 100% redundancy in case of a failure.

RAID 2

- ✓ RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks.
- ✓ Due to its complex structure and high cost, RAID 2 is not commercially available.

RAID 3

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.

RAID 4

- ✓ In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk.
- ✓ Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.

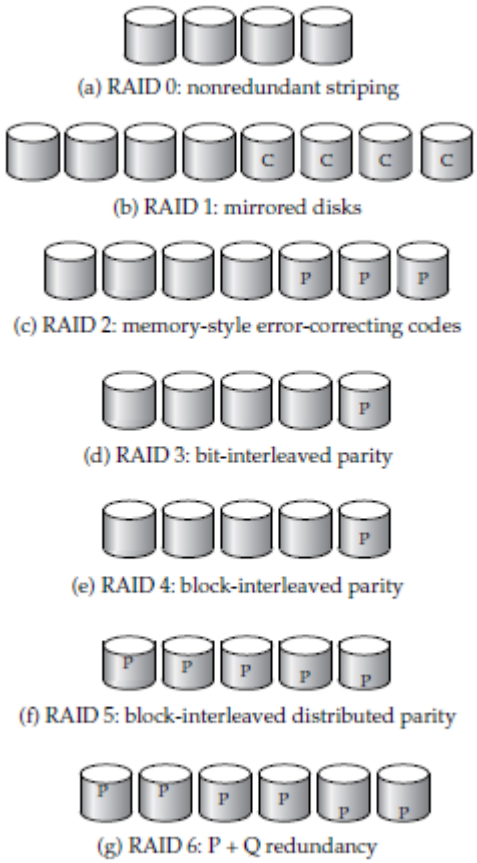
RAID 5

RAID 5 writes whole data blocks onto different disks, but the parity bits generated for data block

stripe are distributed among all the data disks rather than storing them on a different dedicated disk.

RAID 6

- ✓ RAID 6 is an extension of level 5.
- ✓ In this level, two independent parities are generated and stored in distributed fashion among multiple disks. Two parities provide additional fault tolerance.
- ✓ This level requires at least four disk drives to implement RAID.



(a) RAID 0: nonredundant striping

(b) RAID 1: mirrored disks

(c) RAID 2: memory-style error-correcting codes

(d) RAID 3: bit-interleaved parity

(e) RAID 4: block-interleaved parity

(f) RAID 5: block-interleaved distributed parity

(g) RAID 6: P + Q redundancy

File Organization:

Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

File Organization:

File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records −

Heap File Organization:

- ✓ When a file is created using Heap File Organization, the Operating System allocates memory area to that file without any further accounting details.
- ✓ File records can be placed anywhere in that memory area. It is the responsibility of the software to manage the records.

- ✓ Heap File does not support any ordering, sequencing, or indexing on its own.

Sequential File Organization:

- ✓ `Every file record contains a data field (attribute) to uniquely identify that record.
- ✓ In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key.
- ✓ Practically, it is not possible to store all the records sequentially in physical form.

Hash File Organization:

- ✓ Hash File Organization uses Hash function computation on some fields of the records.
- ✓ The output of the hash function determines the location of disk block where the records are to be placed.

Clustered File Organization:

- ✓ Clustered file organization is not considered good for large databases.
- ✓ In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

Video Content / Details of website for further learning (if any):
https://en.wikipedia.org/wiki/Computer_data_storage.html

https://www.tutorialspoint.com/dbms/dbms_storage_system.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 402-412

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 30

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Organization of Records in Files

Introduction : (Maximum 5 sentences)

- ✓ Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format.
- ✓ A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks

Prerequisite knowledge for Complete understanding and learning of Topic:

- Physical Storage Media
- RAID
- Tertiary storage

Detailed content of the Lecture:

File Operations

Operations on database files can be broadly classified into two categories −

- **Update Operations**
- **Retrieval Operations**

Update operations change the data values by insertion, deletion, or update. Retrieval operations, on the other hand, do not alter the data but retrieve them after optional conditional filtering. In both types of operations, selection plays a significant role. Other than creation and deletion of a file, there could be several operations, which can be done on files.

- **Open** − A file can be opened in one of the two modes, **read mode** or **write mode**. In read mode, the operating system does not allow anyone to alter data.
- **Locate** − Every file has a file pointer, which tells the current position where the data is to be read or written. This pointer can be adjusted accordingly. Using find (seek) operation, it can be moved forward or backward.
- **Read** − By default, when files are opened in read mode, the file pointer points to the beginning of the file. There are options where the user can tell the operating system where to locate the file pointer at the time of opening a file. The very next data to the file pointer is read.
- **Write** − User can select to open a file in write mode, which enables them to edit its contents. It can be deletion, insertion, or modification. The file pointer can be located at the time of opening or can be dynamically changed if the operating system allows to do so.
- **Close** − This is the most important operation from the operating system's point of view. When a request to close a file is generated, the operating system
  - removes all the locks (if in shared mode),
  - saves the data (if altered) to the secondary storage media, and
  - releases all the buffers and file handlers associated with the file.

The organization of data inside a file plays a major role here. The process to locate the file pointer to a desired record inside a file various based on whether the records are arranged sequentially or clustered.

Organization of records in files:

There are several ways of organizing records in files.

- **heap file organization**. Any record can be placed anywhere in the file where there is space for the record. There is no ordering of records.
- **sequential file organization**. Records are stored in sequential order, based on the value of the search key of each record.
- **hashing file organization**. A hash function is computed on some attribute of each record. The result of the function specifies in which block of the file the record should be placed -- to be discussed in chapter 11 since it is closely related to the indexing structure.
- **clustering file organization**. Records of several different relations can be stored in the same file. Related records of the different relations are stored on the same block so that one I/O operation fetches related records from all the relations..

Clustering File Organization

1. One relation per file, with fixed-length record, is good for small databases, which also reduces the code size.
2. Many large-scale DB systems do not rely directly on the underlying operating system for file management. One large OS file is allocated to DB system and all relations are stored in one file.
3. The depositor tuple for each cname near the customer tuple for the corresponding cname.
4. This structure mixes together tuples from two relations, but allows for efficient processing of the join.
5. If the customer has many accounts which cannot fit in one block, the remaining records appear on nearby blocks. This file structure, called clustering, allows us to read many of the required records using one block read.
6. Our use of clustering enhances the processing of a particular join but may result in slow processing of other types of queries, such as selection on customer.
   now requires more block accesses as our customer relation is now interspersed with the deposit relation.
7. Thus it is a trade-off, depending on the types of query that the database designer believes to be most frequent. Careful use of clustering may produce significant performance gain.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/dbms/dbms_file_structure.html

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 415-424

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 31

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Indexing and Hashing - Ordered Indices

Introduction : (Maximum 5 sentences)

- ✓ Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ File Organization
- ➢ Organization of Records in Files

Detailed content of the Lecture:

Indexing and hashing:

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing is defined based on its indexing attributes.

Indexing can be of the following types −

- **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types −

- Dense Index
- Sparse Index

Dense Index

- ✓ In dense index, there is an index record for every search key value in the database. This makes

searching faster but requires more space to store index records itself.

- ✓ Index records contain search key value and a pointer to the actual record on the disk.

Sparse Index

- ✓ In sparse index, index records are not created for every search key.
- ✓ An index record here contains a search key and an actual pointer to the data on the disk.

Multilevel Index

- ✓ Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files.
- ✓ Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

Ordered indices:

- ✓ In order to allow fast random access, an index structure may be used.
- ✓ A file may have several indices on different search keys.
- ✓ If the file containing the records is sequentially ordered, the index whose search key specifies the sequential order of the file is the primary index, or clustering index. Note: The search key
- ✓ of a primary index is usually the primary key, but it is not necessarily so.
- ✓ Indices whose search key specifies an order different from the sequential order of the file are called the secondary indices, or non clustering indices.
- ➢ The main goal of designing the database is faster access to any data in the database and quicker insert/delete/update to any data. This is because no one likes waiting.
- ➢ When a database is very huge, even a smallest transaction will take time to perform the action. In order to reduce the time spent in transactions,
- ➢ Indexes are used. Indexes are similar to book catalogues in library or even like an index in a book. What it does? It makes our search simpler and quicker
- ➢ When records are stored in the primary memory like RAM, accessing them is very easy and quick. But records are not limited in numbers to store in RAM.
- ➢ They are very huge and we have to store it in the secondary memories like hard disk. As we have seen already, in memory we cannot store records like we see – tables.
- ➢ They are stored in the form of files in different data blocks. Each block is capable of storing one or more records depending on its size.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/dbms/dbms_hashing.html

https://www.tutorialcup.com/dbms/indexing.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 445-452

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L - 32

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: B+ Tree Index Files - B Tree Index Files

Introduction : (Maximum 5 sentences)

- . B+ tree has one root, any number of intermediary nodes (usually one) and a leaf node.
- Here all leaf nodes will have the actual records stored.
- Intermediary nodes will have only pointers to the leaf nodes; it not has any data.
- Any node will have only two leaves. This is the basic of any B+ tree.

Prerequisite knowledge for Complete understanding and learning of Topic:

- File Organization
- Organization of Records in Files
- Indexing and Hashing
- Ordered Indices

Detailed content of the Lecture:

B^+ Tree

A B^+ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B^+ tree denote actual data pointers.

B^+ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B^+ tree can support random access as well as sequential access.

Structure of B^+ Tree

Every leaf node is at equal distance from the root node. A B^+ tree is of the order **n** where **n** is fixed for every B^+ tree.

Internal nodes −

- Internal (non-leaf) nodes contain at least ⌈n/2⌉ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes −

- Leaf nodes contain at least ⌈n/2⌉ record pointers and ⌈n/2⌉ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B^+ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows −
  - Split node into two parts.
  - Partition at **i = $\lfloor (m+1)/2 \rfloor$.**
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.

- i^{th} key is duplicated at the parent of the leaf.
- If a non-leaf node overflows −
  - Split node into two parts.
  - Partition the node at $\mathbf{i = \lceil (m+1)/2 \rceil}$.
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

B^+ Tree Deletion

- B^+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
  - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
  - Merge the node with left and right to it.

B Tree Index Files

B-Tree is a self-balancing search tree. In most of the other self-balancing search trees (like AVL and Red-Black Trees), it is assumed that everything is in main memory. To understand the use of B-Trees, we must think of the huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses. Most of the tree operations (search, insert, delete, max, min, ..etc) require O(h) disk accesses where h is the height of the tree. B-tree is a fat tree. The height of B-Trees is kept low by putting maximum possible keys in a B-Tree node. Generally, a B-Tree node size is kept equal to the disk block size. Since h is low for B-Tree, total disk accesses for most of the operations are reduced significantly compared to balanced Binary Search Trees like AVL Tree, Red-Black Tree, ..etc.

Properties of B-Tree

1) All leaves are at same level.
2) A B-Tree is defined by the term minimum degree 't'. The value of t depends upon disk block size.
3) Every node except root must contain at least t-1 keys. Root may contain minimum 1 key.
4) All nodes (including root) may contain at most 2t – 1 keys.
5) Number of children of a node is equal to the number of keys in it plus 1.
6) All keys of a node are sorted in increasing order. The child between two keys k1 and k2 contains all keys in the range from k1 and k2.
7) B-Tree grows and shrinks from the root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink from downward.
8) Like other balanced Binary Search Trees, time complexity to search, insert and delete is O(Logn).

Video Content / Details of website for further learning (if any):

https://www.tutorialcup.com/dbms/b-tree.html
https://www.javatpoint.com/dbms-b-plus-tree.html

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 453-464

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 33

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Static Hashing - Dynamic Hashing

Introduction : (Maximum 5 sentences)

- In static hashing, when a search-key value is provided, the hash function always computes the same address.
- For example, if mod-4 hash function is used, then it shall generate only 5 values.
- The output address shall always be same for that function.
- The number of buckets provided remains unchanged at all times.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Indexing and Hashing
- Ordered Indices
- B+ Tree Index Files
- B Tree Index Files

Detailed content of the Lecture:

Static hashing:

- In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if mod-4 hash function is used, then it shall generate only 5 values.
- The output address shall always be same for that function. The number of buckets provided remains unchanged at all times.

Operation:

- **Insertion** − When a record is required to be entered using static hash, the hash function **h** computes the bucket address for search key **K**, where the record will be stored. Bucket address = h(K)
- **Search** − When a record needs to be retrieved, the same hash function can be used to retrieve the address of the bucket where the data is stored.
- **Delete** − This is simply a search followed by a deletion operation.

Bucket Overflow:

The condition of bucket-overflow is known as **collision**. This is a fatal state for any static hash function. In this case, overflow chaining can be used.

- **Overflow Chaining** − When buckets are full, a new bucket is allocated for the same hash result and is linked after the previous one. This mechanism is called **Closed Hashing**.
- **Linear Probing** − When a hash function generates an address at which data is already stored, the next free bucket is allocated to it. This mechanism is called **Open Hashing**.

Dynamic Hashing:

The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks. Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand. Dynamic hashing is also known as extended hashing.

Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially.

Organization:

The prefix of an entire hash value is taken as a hash index. Only a portion of the hash value is used for computing bucket addresses. Every hash index has a depth value to signify how many bits are used for computing a hash function. These bits can address 2n buckets. When all these bits are consumed − that is, when all the buckets are full − then the depth value is increased linearly and twice the buckets are allocated.

Operation:

- **Querying** − Look at the depth value of the hash index and use those bits to compute the bucket address.
- **Update** − Perform a query as above and update the data.
- **Deletion** − Perform a query to locate the desired data and delete the same.
- **Insertion** − Compute the address of the bucket

Hashing is not favorable when the data is organized in some ordering and the queries require a range of data. When data is discrete and random, hash performs the best.

Hashing algorithms have high complexity than indexing. All hash operations are done in constant time.

Dynamic Hashing

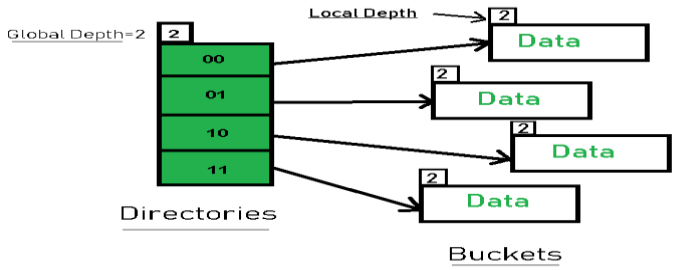
Extendible Hashing is a dynamic hashing method wherein directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.

Main features of Extendible Hashing:

The main features in this hashing technique are:

- Directories: The directories store addresses of the buckets in pointers. An id is assigned to each directory which may change each time when Directory Expansion takes place.
- Buckets: The buckets are used to hash the actual data.

Basic Structure of Extendible Hashing:



Extendible Hashing

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/dbms/dbms_hashing.htmt

https://www.geeksforgeeks.org/extendible-hashing-dynamic-approach-to-dbms/html

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 465-476

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 34

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Distributed Databases

Introduction : (Maximum 5 sentences)

- In a distributed database, there are a number of databases that may be geographically distributed all over the world.
- A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users

Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- SQL Queries
- Relational algebra

Detailed content of the Lecture:

Distributed Database:

- In a distributed database, there are a number of databases that may be geographically distributed all over the world.
- A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users.

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do not have any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.

- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

The following factors encourage moving over to DDBMS −

- **Distributed Nature of Organizational Units** − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data** − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.
- **Support for Both OLTP and OLAP** − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery** − One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.
- **Support for Multiple Application Software** − Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed Databases

Modular Development − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning.

More Reliable − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

Better Response − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

Lower Communication Cost − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

- **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity** − The need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution** − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_databases.html

Important Books/Journals for further learning including the page nos.:

Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.

Page No: 709

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 35

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: - Distributed Data Storage

Introduction : (Maximum 5 sentences)

- A distributed database is a database in which not all storage devices are attached to a common processor.
- It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- SQL Queries
- Relational algebra
- Distributed Database

Detailed content of the Lecture:

Distributed Data Storage

There are 2 ways in which data can be stored on different sites. These are:

1. Replication

- The entire relation is stored redundantly at 2 or more sites.
- If the entire database is available at all sites, it is a fully redundant database.
- Hence, in replication, systems maintain copies of data.
  This is advantageous as it increases the availability of data at different sites.
- Also, now query requests can be processed in parallel.

However, it has certain disadvantages as well.

- ✓ Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency.
- ✓ This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. Fragmentation

- The relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that
- the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

- Horizontal fragmentation – Splitting by rows – The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- Vertical fragmentation – Splitting by columns – The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure lossless join.

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/distributed-database-system/html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 710-713

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 36

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **IV System Architecture** **Date of Lecture:**

Topic of Lecture: Distributed Transactions

Introduction : (Maximum 5 sentences)

- Distributed transaction is a database transaction in which two or more network hosts are involved. Usually, hosts provide transactional resources, while the transaction manager is responsible for creating and managing a global transaction that encompasses all operations against such resources.

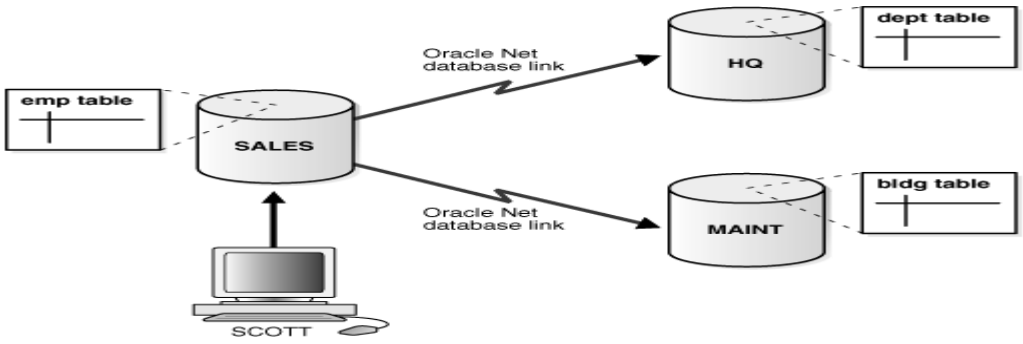
Prerequisite knowledge for Complete understanding and learning of Topic:

- Database
- SQL Queries
- Relational algebra
- Distributed Database

Detailed content of the Lecture:

Distributed Transactions

A **distributed transaction** includes one or more statements that, individually or as a group, update data on two or more distinct nodes of a distributed database. For example, assume the database configuration depicted.



DML and DDL Transactions

The following are the DML and DDL operations supported in a distributed transaction:

- CREATE TABLE AS SELECT
- DELETE
- INSERT (default and direct load)
- LOCK TABLE
- SELECT

- SELECT FOR UPDATE

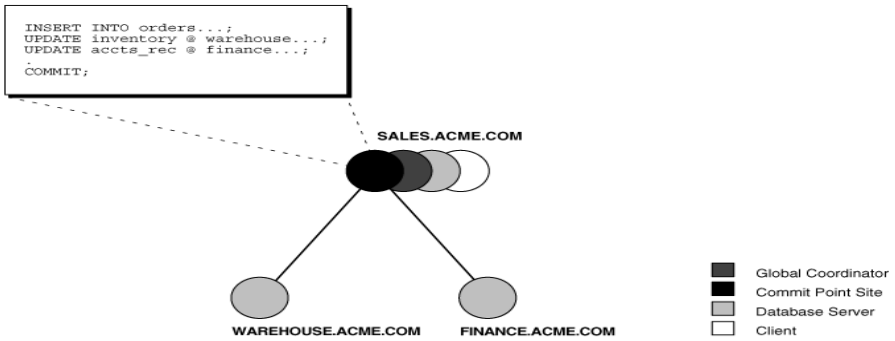
Transaction Control Statements

The following are the supported transaction control statements:

- COMMIT
- ROLLBACK
- SAVEPOINT

Session Trees for Distributed Transactions

As the statements in a distributed transaction are issued, the database defines a **session tree** of all nodes participating in the transaction. A session tree is a hierarchical model that describes the relationships among sessions and their roles.



Clients

A node acts as a client when it references information from a database on another node. The referenced node is a database server

Local Coordinators

A node that must reference data on other nodes to complete its part in the distributed transaction is called a local coordinator

A local coordinator is responsible for coordinating the transaction among the nodes it communicates directly with by:

- Receiving and relaying transaction status information to and from those nodes
- Passing queries to those nodes
- Receiving queries from those nodes and passing them on to other nodes
- Returning the results of queries to the nodes that initiated them

Global Coordinator

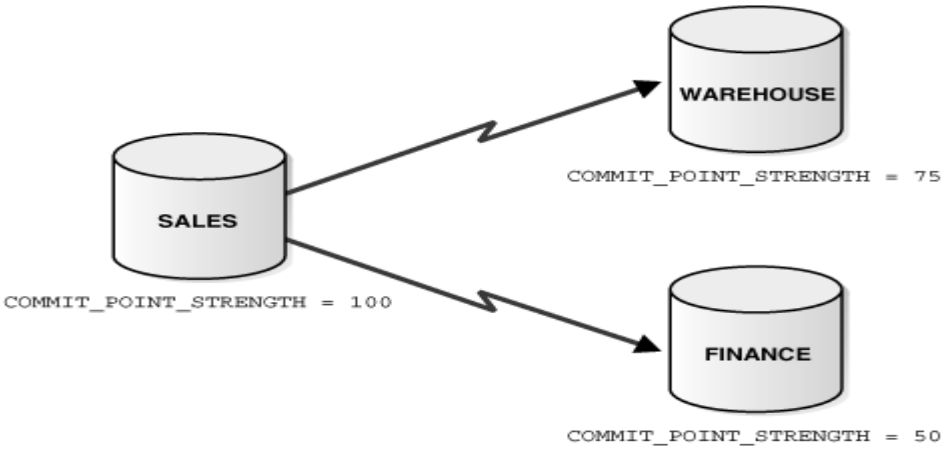
The node where the distributed transaction originates is called the global coordinator. The database application issuing the distributed transaction is directly connected to the node acting as the global coordinator. The global coordinator becomes the parent or root of the session tree. The global coordinator performs the following operations during a distributed transaction:

- Sends all of the distributed transaction SQL statements, remote procedure calls, and so forth to the directly referenced nodes, thus forming the session tree
- Instructs all directly referenced nodes other than the commit point site to prepare the transaction
- Instructs the commit point site to initiate the global commit of the transaction if all nodes prepare successfully

- Instructs all nodes to initiate a global rollback of the transaction if there is an abort response

Commit Point Site

The job of the commit point site is to initiate a commit or roll back operation as instructed by the global coordinator. The system administrator always designates one node to be the commit point site in the session tree by assigning all nodes a commit point strength. The node selected as commit point site should be the node that stores the most critical data.

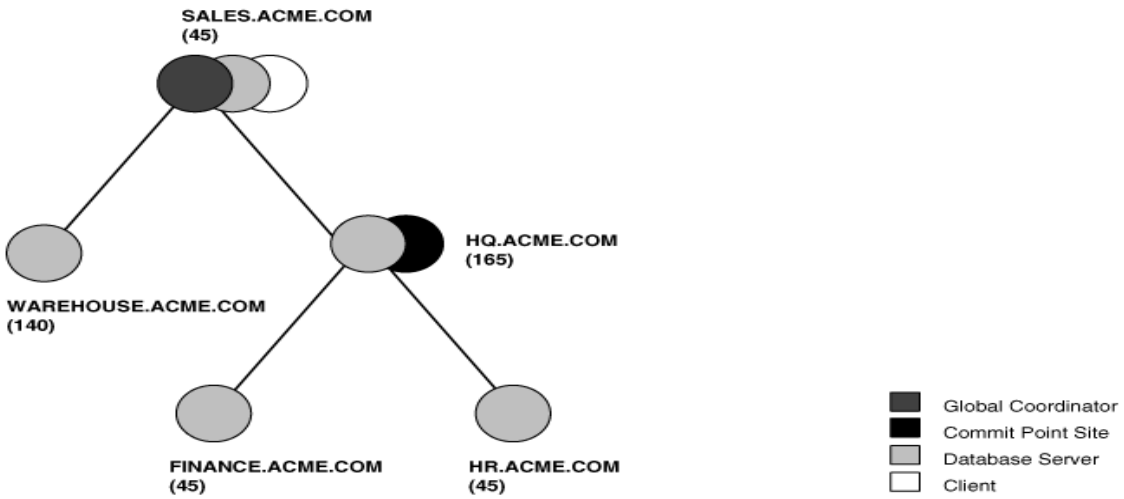


How a Distributed Transaction Commits

- A distributed transaction is considered committed after all non-commit-point sites are prepared, and the transaction has been actually committed at the commit point site.
- The redo log at the commit point site is updated as soon as the distributed transaction is committed at this node.

Commit Point Strength

- Every database server must be assigned a commit point strength.
- If a database server is referenced in a distributed transaction, the value of its commit point strength determines which role it plays in the two-phase commit.
- Specifically, the commit point strength determines whether a given node is the commit point site in the distributed transaction and thus commits before all of the other nodes.
- This value is specified using the initialization parameter COMMIT_POINT_STRENGTH. This section explains how the database determines the commit point site.



Two-Phase Commit Mechanism

- Unlike a transaction on a local database, a distributed transaction involves altering data on multiple databases. Consequently, distributed transaction processing is more complicated, because the database must coordinate the committing or rolling back of the changes in a transaction as a self-contained unit.
- The database ensures the integrity of data in a distributed transaction using the two-phase commit mechanism.
- In the prepare phase, the initiating node in the transaction asks the other participating nodes to promise to commit or roll back the transaction.
- During the commit phase, the initiating node asks all participating nodes to commit the transaction. If this outcome is not possible, then all nodes are asked to roll back.

Video Content / Details of website for further learning (if any):
https://docs.oracle.com/cd/B19306_01/server.102/b14231/ds_txns.html

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 713-738

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 37

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Database Security - Data Classification

Introduction : (Maximum 5 sentences)

- ✓ Database security concerns the use of a broad range of information security controls to protect databases against compromises of their confidentiality, integrity and availability.
- ✓ Databases have been largely secured against hackers through network security measures such as firewalls, and network-based intrusion detection systems

Prerequisite knowledge for Complete understanding and learning of Topic:

- Access control
- Auditing
- Authentication
- Encryption
- Integrity controls
- Backups
- Application security
- Database Security applying Statistical Method

Detailed content of the Lecture:

Database security:

Database security concerns the use of a broad range of information security controls to protect databases against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical.

Security risks to database systems include, for example:

- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services;
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended;
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic breakdowns/equipment failures and obsolescence;
- Design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities (e.g. unauthorized privilege escalation), data loss/corruption, performance degradation etc.;
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc.

- ✓ Databases have been largely secured against hackers through network security measures such as firewalls, and network-based intrusion detection systems.
- ✓ While network security controls remain valuable in this regard, securing the database systems themselves, and the programs/functions and data within them, has arguably become more critical as networks are increasingly opened to wider access, in particular access from the Internet.
- ✓ Furthermore, system, program, function and data access controls, along with the associated user identification, authentication and rights management functions, have always been important to limit and in some cases log the activities of authorized users and administrators.

Many organizations develop their own "baseline" security standards and designs detailing basic security control measures for their database systems.

Data classification:

When implemented it provides a bridge between IT professionals and process or application owners.

- ✓ IT staff are informed about the data value and management (usually application owners) understands better which part of the data centre needs to be invested in to keep operations running effectively.
- ✓ This can be of particular importance in risk management, legal discovery, and compliance with government regulations.
- ✓ Data classification is typically a manual process; however, there are many tools from different vendors that can help gather information about the data.

Data classification needs to take into account the following:

- Regulatory requirements
- Strategic or proprietary worth
- Organization specific policies
- Ethical and privacy considerations
- Contractual agreements

How to start process of data classification:

- Relational or Tabular data (around 15% of non audio/video data)
  - Generally describes proprietary data which can be accessible only through application or application programming interfaces (API)
  - Applications that produce structured data are usually database applications.
  - To ensure adequate quality standards, the classification process has to be monitored by subject matter experts.
- Semi-structured or Poly-structured data (all other non audio/video data that does not conform to a system or platform defined Relational or Tabular form).
  - Generally describes data files that have a dynamic or non-relational semantic structure (e.g. documents,XML,JSON,Device or System Log output,Sensor Output).
  - Relatively simple process of data classification is criteria assignment.
  - Simple process of data migration between assigned segments of predefined storage tiers.

Video Content / Details of website for further learning (if any):
https://www.imperva.com/learn/data-security/data-classification/

Important Books/Journals for further learning including the page nos.:
Web Reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L - 38

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Threats and risks

Introduction : (Maximum 5 sentences)

- ✓ A **threat**, in the context of computer security, refers to anything that has the potential to cause serious harm to a computer system.
- ✓ A **threat** is something that may or may not happen, but has the potential to cause serious damage

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Threat
- ➢ Risks
- ➢ Database Secuity

Detailed content of the Lecture:

- ✓ A threat, in the context of computer security, refers to anything that has the potential to cause serious harm to a computer system.
- ✓ A threat is something that may or may not happen, but has the potential to cause serious damage. Threats can lead to attacks on computer systems, networks and more.

The 3 Biggest Database Threats:

1) Credential Threats
2) Privilege Threats
3) System Threats

Database Security

- **SQL injection attacks:**
  - ➢ Database injection attacks can cause damage to databases through passing malicious SQL instruction into Databases by trusted applications and users.
  - ➢ Data Sunrise is well equipped with a facility to automatically detect and block SQL injection using profile-based assessment of those activities.
- **Malware:**
  - ➢ Malwares always been used to help intruders leak sensitive data via legitimate users connections using bugging servers with their code.
  - ➢ Data Sunrise can help organization flag, detect and automatically block those activities in their very early stages.
- **Storage media exposure:**
  - ➢ Backup file and their storage media are often left unencrypted, unprotected from attack,

the cruciality of those files lies in the fact that a restoration of one them to the wrong person means leaking thousands of confidential records.
    - DataSunrise always advises it customers to keep those backups encrypted and protected from any potential data leakage.
- **Exploitation of vulnerable databases:**
    - Most organization hate change and delay patching their databases longer than they should.
    - Data Sunrise always advise their customer to keep their databases up to date with the latest security patch-sets while it would be protecting those databases in the meanwhile via its comprehensive arsenal of security facilities.
- **Unmanaged sensitive data:**
    1. Maintaining sensitive information in test environments is usually one of the main challenges that face organization keep those systems either behind the up to date data when adopting traditional static masking tools from the production or exposing those databases by having them unprotected and sharing those sensitive information with other department and individuals.
    2. Data Sunrise has the ability to mask data in an obfuscated static format or dynamically while for all ongoing traffic while allowing certain applications and/or users for accessing information if they aren't supposed to be subject for those masking policies.
- **The human factor:**
    - While it has been reported that 30 percent of data breach incidents are result of human negligence, according to the Ponemon Institute Cost of Data Breach Study.
    - DataSunrise can always flag any malicious that could be result of those incidents and help organizations contain and address those issues at very early stages.

Multilayered Security Solutions:

DataSunrise was built to work seamlessly as an integrated layer of security along with the native protection facets wired the databases protected by it suit:

1. DataSunrise can rewrite queries while keeping the database engine free from any additional overhead obfuscating those queries when possible.
2. DataSunrise can audit and consolidate the trails from all the organizations' database vendors that the organization supports homogeneously in one repository using a unified console.
3. DataSunrise can impose additional layer of Access Control policies to complement the conventional databases privileges paradigm.

Video Content / Details of website for further learning (if any):

https://www.datasunrise.com/blog/potential-db-threats/mitigating-top-database-security-threats-using-datasunrise-security-suite/

Important Books/Journals for further learning including the page nos.:
Book: Web Reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

L - 39

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Database Access Control

Introduction : (Maximum 5 sentences)

- Database access control is a method of allowing access to company's sensitive data only to those people (database users) who are allowed to access such data and to restrict access to unauthorized persons.
- It includes two main components:
  - authentication
  - authorization.
- Authentication is a method of verifying the identity of a person who is accessing your database

Prerequisite knowledge for Complete understanding and learning of Topic:

- DDL
- DML
- TCL
- DCL

Detailed content of the Lecture:

Database access control is a method of allowing access to company's sensitive data only to those people (database users) who are allowed to access such data and to restrict access to unauthorized persons.

It includes two main components:

- authentication
- authorization.

Authentication is a method of verifying the identity of a person who is accessing your database. Note that authentication isn't enough to protect data.

An additional layer of security is required, authorization, which determines whether a user should be allowed to access the data or make the transaction he's attempting. Without authentication and authorization, there is no data security.

Any company whose employees connect to the Internet, thus, every company today, needs some level of access control implemented.

Types of Access Control

Obsolete access models include Discretionary Access Control (DAC) and Mandatory Access Control

(MAC). Role Based Access Control (RBAC) is the most common method today, and the most recent model is Attribute Based Access Control (ABAC).

Discretionary Access Control (DAC)

With DAC models, the data owner allows access. DAC is a means of assigning access rights based on user-specified rules.

Mandatory Access Control (MAC)

MAC was developed using a nondiscretionary model, in which people are granted access based on an information clearance. MAC is a policy in which access rights are assigned based on central authority regulations.

Role Based Access Control (RBAC)

RBAC grants access based on a user's role and implements key security principles such as "least privilege" and "separation of privilege." Thus, someone attempting to access information can only access data necessary for their role.

Attribute Based Access Control (ABAC)

In ABAC, each resource and user are assigned a series of attributes. In this dynamic method, a comparative assessment of the user's attributes, including time of day, position and location, are used to make a decision on access to a resource

Video Content / Details of website for further learning (if any):
https://www.datasunrise.com/blog/professional-info/what-is-access-control/

Important Books/Journals for further learning including the page nos.:
Book: Web refernce

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 40

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Types of Privileges

Introduction : (Maximum 5 sentences)

- ✓ There are two types of privileges.

1) System privileges - This allows the user to CREATE, ALTER, or DROP database objects.

2) Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ DDL
- ➢ DML
- ➢ TCL
- ➢ DCL

Detailed content of the Lecture:

Types of privileges in dbms :

- ➢ The account level:

At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.

- ➢ The relation level (or table level):
- ✓ At this level, the DBA can control the privilege to access each individual relation or view in the database.capabilities provided to the account itself and can include the CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation,
- ✓ The CREATE VIEW privilege
- ✓ The ALTER privilege, to apply schema changes such adding or removing attributes from relations;
- ✓ The DROP privilege, to delete relations or views
- ✓ The MODIFY privilege, to insert, delete, or update tuples;

and the SELECT privilege, to retrieve information from the database by using a SELECT query.

The second level of privileges applies to the relation level

- ✓ This includes base relations and virtual (view) relations.
- ✓ The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the access matrix model where
- ✓ The rows of a matrix M represents subjects (users, accounts, programs)
- ✓ The columns represent objects (relations, records, columns, views, operations).
- ✓ Each position M(i,j) in the matrix represents the types of privileges (read, write, update) that subject i holds on object j.
- ✓ To control the granting and revoking of relation privileges, each relation R in a database is assigned and owner account, which is typically the account that was used when the relation was created in the first place.
- ✓ The owner of a relation is given all privileges on that relation.
- ✓ In SQL2, the DBA can assign and owner to a whole schema by creating the schema and associating the appropriate authorization identifier with that schema, using the CREATE SCHEMA command.
- ✓ The owner account holder can pass privileges on any of the owned relation to other users by granting privileges to their accounts.
- ✓ In SQL the following types of privileges can be granted on each individual relation R:
- ✓ SELECT (retrieval or read) privilege on R:
- ✓ Gives the account retrieval privilege.
- ✓ In SQL this gives the account the privilege to use the SELECT statement to retrieve tuples from R.
- ✓ MODIFY privileges on R:
- ✓ This gives the account the capability to modify tuples of R.
- ✓ In SQL this privilege is further divided into UPDATE, DELETE, and INSERT privileges to apply the corresponding SQL command to R.
- ✓ In addition, both the INSERT and UPDATE privileges can specify that only certain

Video Content / Details of website for further learning (if any):
https://www.techglads.com/cse/sem3/types-of-privileges-in-dbms/

Important Books/Journals for further learning including the page nos.:
Book: Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Sixth Edition, Tata McGraw Hill, 2011.
Page No: 135-200

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 41

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management System/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Security of Statistical Databases Parallel Databases

Introduction : (Maximum 5 sentences)

- ✓ Statistical database security system is used to control the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria.
- ✓ A statistical database contains confidential information about individuals or organisations, which is used to answer statistical queries concerning sums, averages, and numbers with certain characteristics.
- ✓ Thus, a statistical database permits queries that derive aggregated (statistical) information, for example, sums, averages, counts, maximums, minimums, standard deviations, means, totals, or a query such as "What is the average salary of Analysts?", etc. They do not permit queries that derive individual information.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Database
- ➢ Database Management System
- ➢ Relational Database

Detailed content of the Lecture:

- ✓ Statistical database security system is used to control the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria.
- ✓ A statistical database contains confidential information about individuals or organisations, which is used to answer statistical queries concerning sums, averages, and numbers with certain characteristics.
- ✓ Thus, a statistical database permits queries that derive aggregated (statistical) information, for example, sums, averages, counts, maximums, minimums, standard deviations, means, totals, or a query such as "What is the average salary of Analysts?", etc. They do not permit queries that derive individual information.Goals of Parallel Databases

Parallel Database

Improve performance:

The performance of the system can be improved by connecting multiple CPU and disks in parallel.

Many small processors can also be connected in parallel.

Improve availability of data:

Data can be copied to multiple locations to improve the availability of data.

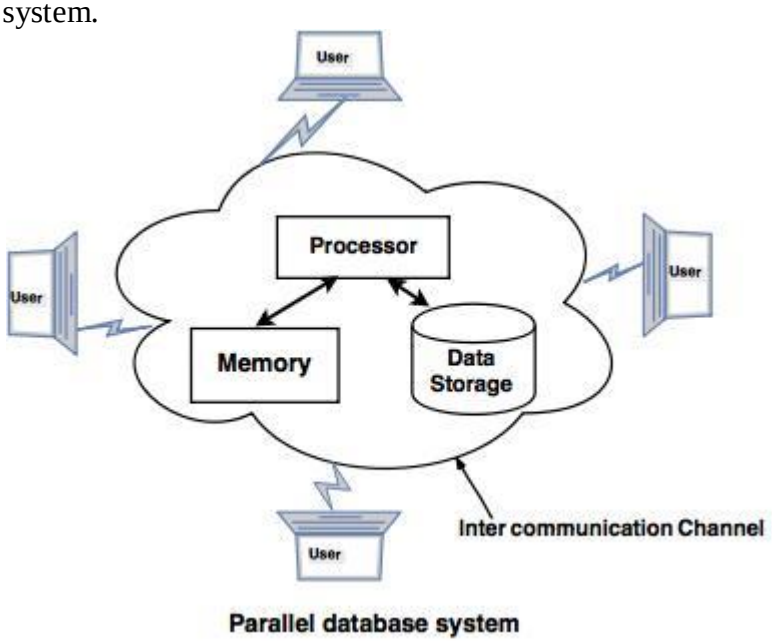
For example: if a module contains a relation (table in database) which is unavailable then it is important to make it available from another module.

Improve reliability:

Reliability of system is improved with completeness, accuracy and availability of data.

Provide distributed access of data:

Companies having many branches in multiple cities can access data with the help of parallel database system.



Parallel database system

Parallel DBMS • Parallelism is natural to DBMS processing

– Pipeline parallelism: many machines each doing one step in a multi-step process.

– Data-partitioned parallelism: many machines doing the same thing to different pieces of data.

Video Content / Details of website for further learning (if any):

https://www.tutorialride.com/parallel-databases/parallel-databases-tutorial.htm
https://www2.cs.duke.edu/courses/fall17/compsci516/Lectures/Lecture-20-Parallel-DB.pdf

Important Books/Journals for further learning including the page nos.:
Book: Web Reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 42

LECTURE HANDOUTS

CSE | **II/IV**

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Spatial and Multimedia database

Introduction : (Maximum 5 sentences)

- ✓ A spatial database is a database that is optimized for storing and querying data that represents objects defined in a geometric space.
- ✓ Most spatial databases allow the representation of simple geometric objects such as points, lines and polygons
- ✓ A Multimedia database (MMDB) is a collection of related for multimedia data. The multimedia data include one or more primary media data types such as text, images, graphic objects (including drawings, sketches and illustrations) animation sequences, audio and video.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Security of Statistical Database
- ➢ Parallel Database
- ➢ Multimedia

Detailed content of the Lecture:

- ➢ Spatial data is associated with geographic locations such as cities, towns etc.
- ➢ A spatial database is optimized to store and query data representing objects.
- ➢ These are the objects which are defined in a geometric space.

Characteristics of Spatial Database

A spatial database system has the following characteristics

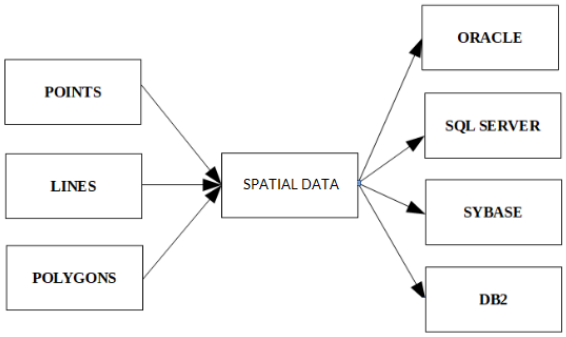
1. It is a database system
2. It offers spatial data types (SDTs) in its data model and query language.
3. It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

Example

- ➢ A road map is a visualization of geographic information.
- ➢ A road map is a 2-dimensional object which contains points, lines, and polygons that can represent cities, roads, and political boundaries such as states or provinces.

In general, spatial data can be of two types:

1. Vector data: This data is represented as discrete points, lines and polygons
2. Rastor data: This data is represented as a matrix of square cells.



Multimedia Database :

- A Multimedia database (MMDB) is a collection of related for multimedia data. The multimedia data include one or more primary media data types such as text, images, graphic objects (including drawings, sketches and illustrations) animation sequences, audio and video.
- The multimedia databases are used to store multimedia data such as images, animation, audio, video along with text.
- This data is stored in the form of multiple file types like .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc.

Contents of the Multimedia Database

The multimedia database stored the multimedia data and information related to it. This is given in detail as follows:

Media data

This is the multimedia data that is stored in the database such as images, videos, audios, animation etc.

Media format data

The Media format data contains the formatting information related to the media data such as sampling rate, frame rate, encoding scheme etc.

Media keyword data

This contains the keyword data related to the media in the database. For an image the keyword data can be date and time of the image, description of the image etc.

Media feature data

Th Media feature data describes the features of the media data. For an image, feature data can be colours of the image, textures in the image etc.

Challenges of Multimedia Database

There are many challenges to implement a multimedia database. Some of these are:

1. Multimedia databases contains data in a large type of formats such as .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc. It is difficult to convert one type of data format to another.
2. The multimedia database requires a large size as the multimedia data is quite large and needs to be stored successfully in the database.
3. It takes a lot of time to process multimedia data so multimedia database is slow.

Video Content / Details of website for further learning (if any):

https://en.wikipedia.org/wiki/Spatial_database
https://www.tutorialspoint.com/Multimedia-Databases

Important Books/Journals for further learning including the page nos.:

Book:

Web Reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 43

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Mobile and web Database

Introduction : (Maximum 5 sentences)

- ✓ Mobile Database is a database that is transportable, portable, and physically separate or detached from the corporate database server but has the capability to communicate with those servers from remote sites allowing the sharing of various kinds of data.
- ✓ Web database is a database application designed to be managed and accessed through the Internet.
- ✓ Website operators can manage this collection of data and present analytical results based on the data in the Web database application.

Prerequisite knowledge for Complete understanding and learning of Topic:

- ➢ Security of Statistical Database
- ➢ Parallel Database
- ➢ Multimedia
- ➢ DBMS Architecture

Detailed content of the Lecture:

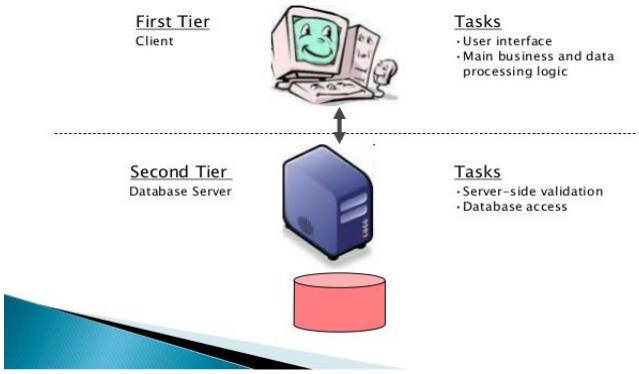
Mobile Database

- ✓ Mobile Database is a database that is transportable, portable, and physically separate or detached from the corporate database server but has the capability to communicate with those servers from remote sites allowing the sharing of various kinds of data.
- ✓ Mobile databases:
- ✓ Physically separate from the central database server.
- ✓ Resided on mobile devices.
- ✓ Capable of communicating with a central database server or other mobile clients from remote sites.
- ✓ Handle local queries without connectivity
- The notebook and laptop Computers are being used increasingly among the Business Community
- The development and availability of a relatively low-cost wireless digital communication infrastructure. This infrastructure is based on wireless local-area networks, cellular digital packet networks, and other technologies

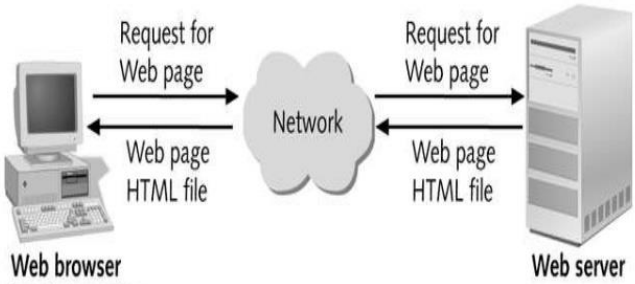
Web Database

A Web database is a database application designed to be managed and accessed through the Internet. Website operators can manage this collection of data and present analytical results based on the data in the Web database application.

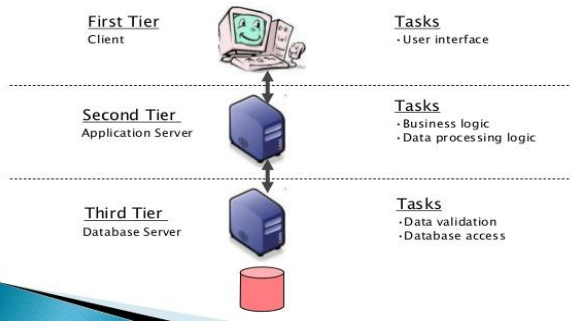
Two-tier Application



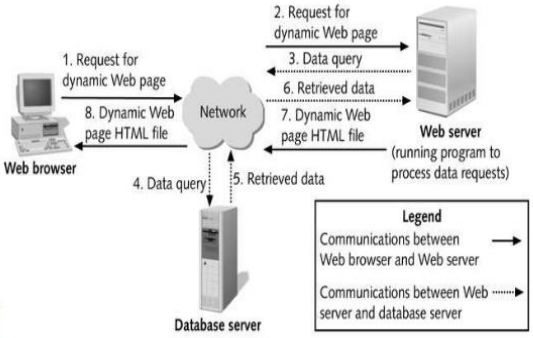
Web client/server architecture



Three-tier Application



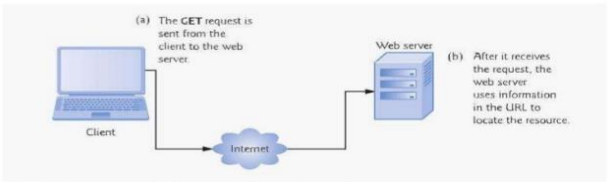
Database-driven Web site Architecture



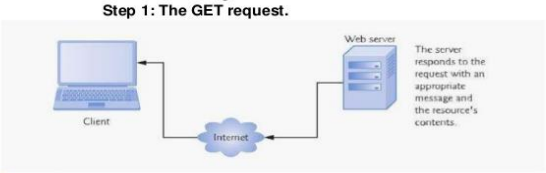
How Web Interface interacts with the Database

Web interface provides attractive, even for database applications that are used only with a single organization.

The HyperText Markup Language(HTML) standard allows text to be neatly formatted , with important information highlighted. Hyperlinks which are links to other documents.Clicking the Hyperlink fetches and displays the linked document.



Client interacting with web server.
Step 1: The GET request.



Client interacting with web server.
Step 2: The HTTP response.

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/data-management-issues-in-mobile-database/
https://webservices.ignou.ac.in/virtualcampus/adit/course/cst302/block2/cst302-bl2-u1.htm
https://www.w3schools.in/dbms/web-based-database-management-system/
https://www.slideshare.net/saddam02/presentations-on-web-database

Important Books/Journals for further learning including the page nos.:

Book: Web reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L - 44

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: Object Oriented Database

Introduction : (Maximum 5 sentences)

- ✓ Object-oriented databases (OODB) are databases that represent data in the form of objects and classes. In object-oriented terminology, an object is a real-world entity, and a class is a collection of objects.
- ✓ Object-oriented databases follow the fundamental principles of object-oriented programming (OOP).

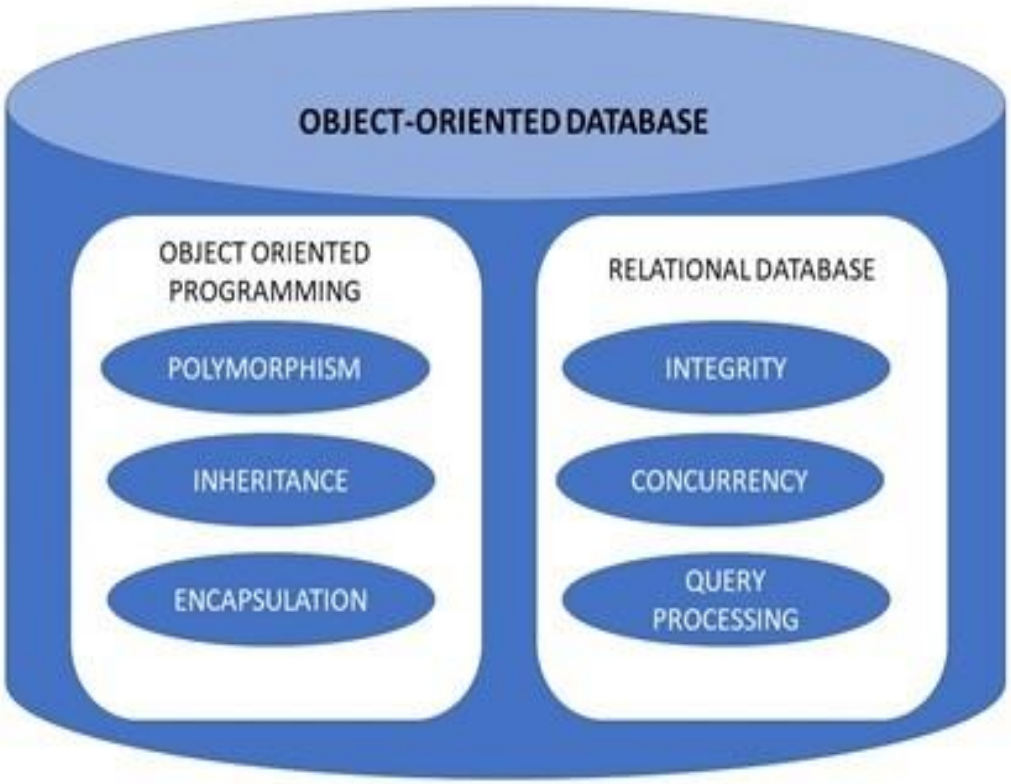
Prerequisite knowledge for Complete understanding and learning of Topic:

- DBMS Architecture
- C
- C++
- Java

Detailed content of the Lecture:

Object-oriented databases

- ✓ Object-oriented databases (OODB) are databases that represent data in the form of objects and classes. In object-oriented terminology, an object is a real-world entity, and a class is a collection of objects.
- ✓ Object-oriented databases follow the fundamental principles of object-oriented programming (OOP).
- ✓ An object-oriented database is a collection of object-oriented programming and relational database. There are various items which are created using object-oriented programming languages like C++, Java which can be stored in relational databases, but object-oriented databases are well-suited for those items.
- ✓ An object-oriented database is organized around objects rather than actions, and data rather than logic. For example, a multimedia record in a relational database can be a definable data object, as opposed to an alphanumeric value.



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/html-dom-dl-object

Important Books/Journals for further learning including the page nos.:
Book: Web Reference

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L - 45

LECTURE HANDOUTS

CSE

II/IV

Course Name with Code : **Database Management Systems/16CSD03**

Course Teacher : **R.Vinupriya**

Unit : **V Database Security** **Date of Lecture:**

Topic of Lecture: XML Database

Introduction : (Maximum 5 sentences)

- ✓ XML Database is used to store huge amount of information in the XML format.
- ✓ As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents.
- ✓ The data stored in the database can be queried using **XQuery**, serialized, and exported into a desired format.

Prerequisite knowledge for Complete understanding and learning of Topic:

- C/C++
- Java
- HTML
- Network Protocols

Detailed content of the Lecture:

- ✓ XML Database is used to store huge amount of information in the XML format.
- ✓ As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents.
- ✓ The data stored in the database can be queried using XQuery, serialized, and exported into a desired format.

XML Database Types

There are two major types of XML databases −

- XML- enabled
- Native XML (NXD)

XML - Enabled Database

- XML enabled database is nothing but the extension provided for the conversion of XML document. This is a relational database, where data is stored in tables consisting of rows and columns.
- The tables contain set of records, which in turn consist of fields.

Native XML Database

- Native XML database is based on the container rather than table format. It can store large amount of XML document and data. Native XML database is queried by the **XPath**-expressions.
- Native XML database has an advantage over the XML-enabled database. It is highly capable to store, query and maintain the XML document than XML-enabled database.

Example

Following example demonstrates XML database −

```
<?xml version = "1.0"?>
<contact-info>
   <contact1>
      <name>Tanmay Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 123-4567</phone>
   </contact1>

   <contact2>
      <name>Manisha Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 789-4567</phone>
   </contact2>
</contact-info>
```

XML Editor is a markup language editor.

The XML documents can be edited or created using existing editors such as Notepad, WordPad, or any similar text editor.

You can also find a professional XML editor online or for downloading, which has more powerful editing features such as −

- It automatically closes the tags that are left open.
- It strictly checks syntax.
- It highlights XML syntax with colour for increased readability.
- It helps you write a valid XML code.
- It provides automatic verification of XML documents against DTDs and Schemas.

Open Source XML Editors

Following are some open source XML editors −

- Online XML Editor − This is a light weight XML editor which you can use online.
- Xerlin − Xerlin is an open source XML editor for Java 2 platform released under an Apache license. It is a Java based XML modelling application, for creating and editing XML files easily.
- CAM - Content Assembly Mechanism − CAM XML Editor tool comes with XML+JSON+SQL Open-XDX sponsored by Oracle.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/xml/xml_databases.htm

Important Books/Journals for further learning including the page nos.:
Book:.
Web Reference

Course Teacher

Verified by HOD