# Database Management System

# Semester – III

**Compiled, reviewed and edited by:**

*TCET, Mumbai*

**STUDY MATERIAL FOR SE SEM-III**
**Under the guidance of Dr. B. K. Mishra, Principal**

1st Edition of revised syllabus, 2021



**Published by:**
Thakur College of Engineering and Technology

## PREFACE

This Resource book is a structured and guided learning materials to be used effectively by students to learn the contents of syllabus based on TCET Autonomy scheme. Top priority is given to the students need and contents are presented lecture-wise in order to help student keep track of their learning after every lecture. The resource book emphasizes on the fundamental definitions with a continuity of details easy to follow finally rendering to the practical applications of the topic. The module is written with sequentially starting from prerequisite to theoretical background eventually highlighting the concepts and covering all the topics in detail. Large number of short answer questions, long answer questions and practice questions are included which will be helpful in preparing for examinations.

This is a comprehensive (structured and guided) resource book covering the Database Management System syllabus as per TCET Autonomy scheme. The detail of content is presented module wise below:

**Module-1:** It primarily deals with introduction to Database Concepts. It covers Database structure, database users, and database languages.

**Module-2:** It deals with Entity Relationship Model (ER). It emphasises on different aspects of ER model.

**Module-3:** This module covers Structured Query Language (SQL). It covers basic of SQL like DDL, DML statements.

**Module-4:** This module covers advanced Structured Query Language (SQL) focusing on Integrity, Security and Authorization of database.

**Module-5:** This module primarily includes different types of Relational Database Design covering pitfalls and insights of every normal form.

**Module-6:** It Covers basics of Transaction Management, Concurrency Control and Recovery.

This book will be used extensively for conducting day to day teaching learning process. Sufficient number of Solved and exercise problems is included to develop the required analytical thinking and problem-solving skill.

At the end of each lecture, it provides the learners an opportunity to check their learning from lecture by using sections like takeaway from the lecture, self -assessment and self-evaluation.

**General Guidelines for Learners:**

1. Resource book is for structured and guided teaching learning process and therefore learners are recommended to come with the same in every lecture.
2. While conducting teaching learning process this resource book will be extensively used.
3. Resource book is framed to improve the understanding of subject matter at depth and therefore the learners are recommended to take up all the module contents, home assignments and exercise seriously.
4. A separate notebook should be maintained for every subject.
5. Lectures should be attended regularly. In case of absence, topic done in the class should be referred from the module before attending the next lecture.
6. Motivation, weightage, and pre-requisite in every chapter have been included in order to maintain continuity and improve the understanding of the content to clarify topic requirement from exam point of view.
7. For any other additional point related to the topic instructions will be given by the subject teacher from time to time.

**Subject Related Guidelines:**

1. The subject is pure as well as applied in nature and it requires thorough understanding of subject matter like knowledge of database structure, Query languages.
2. Questions are expected from all modules and learners are instructed not to leave any module in option.
3. Theory paper will be of 100 marks; practical examination will be of 25 marks and continuous and systematic work during the term (Term work) will be of 25 Marks. Importance should be given to term work and all internal assessment/continuous assessment for improving overall percentage in examination.
4. Practice questions should be solved sincerely to enhance confidence level and to excel in end semester examination.

## Exam Specific Guidelines:

1. All modules are equally important from examination point of view.
2. Wherever applicable chemical reaction, mechanism and neat labelled diagram must be provided for writing effective answer.
3. Proper time management is required for completing the question paper within time frame.
4. Read the question paper thoroughly first then choose the questions. Attempt the one that you know the best first but do not change the internal sequence of the sub questions.
5. For further subject clarification/ doubt in the subject, learners can contact the subject teacher.

## Guidelines for Writing Quality Answer:

1. Write content as per marks distribution.
2. Highlight the main points.
3. Write necessary content related to the point.
4. Draw neat and labelled diagrams wherever necessary.
5. While writing distinguishing points, write double the number of points as per the marks given, excluding the example.

**S.E. Semester –III**

<table>
<tr><td colspan="7">B.E. ( Computer Engineering )</td><td colspan="4">S.E. SEM : III</td></tr>
<tr><td colspan="7">Course Name : Database Management System</td><td colspan="4">Course Code : PCC- CS302</td></tr>
<tr><td colspan="5">Teaching Scheme (Program Specific)</td><td colspan="6">Examination scheme</td></tr>
<tr><td colspan="5">Modes of Teaching / Learning / Weightage</td><td colspan="6">Modes of Continuous Assessment / Evaluation</td></tr>
<tr><td colspan="5">Hours Per Week</td><td colspan="2">Theory (100)</td><td>Practical/Oral (25)</td><td>Term Work (25)</td><td>Total</td></tr>
<tr><td>Theory</td><td>Tutorial</td><td>Practical</td><td>Contact Hours</td><td>Credits</td><td>IA</td><td>ESE</td><td>PR/OR</td><td>TW</td><td rowspan="2">150</td></tr>
<tr><td>3</td><td></td><td>2</td><td></td><td>4</td><td>25</td><td>75</td><td>25</td><td>25</td></tr>
<tr><td colspan="10">IA: In-Semester Assessment - Paper Duration – 1.5 Hours<br>ESE: End Semester Examination - Paper Duration - 3 Hours<br>The weightage of marks for continuous evaluation of Term work/Report: Formative (40%), Timely completion of practical (40%) and Attendance/Learning Attitude (20%)</td></tr>
<tr><td colspan="10">Prerequisite: Computer Basics</td></tr>
</table>

**Course Objective:** The course intends to deliver the fundamental knowledge of database management system and apply this knowledge for implementing and analyzing real world problems.

**Course Outcomes: Upon completion of the course students will be able to:**

| SN | Course Objectives | Cognitive levels of attainment as per Bloom's Taxonomy |
|---|---|---|
| 1 | Demonstrate the fundamental elements of relational database Management Systems | L1, L2 |
| 2 | Outline ER and EER diagram for the real life problem and convert it to Relational Database. | L1, L2,L3 |
| 3 | Solve and build basic SQL Queries on given Data. | L1, L2, L3 |
| 4 | Solve and build Advanced SQL Queries on given Data. | L1, L2, L3 |
| 5 | Develop a relational database using concept of functional dependencies. | L1, L2, L3 |
| 6 | Interpret the concepts of transaction, concurrency and recovery | L1, L2 |

## Detailed Syllabus:

| Module No. | Topics | Hrs. | Cognitive levels of attainment as per Bloom's Taxonomy |
|---|---|---|---|
| 1 | **Introduction to Database Concepts** | 3 | L1, L2 |
| | Basic Concepts of Data, Database and DBMS, Applications of Databases, Advantages of DBMS over File Processing System, Three Level Architecture of Database System, Data Abstraction and Data Independence, Database Languages( DDL, DML, TCL, DCL), Database Users, Database Administrator and its roles, Overall System structure. | | |
| 2 | **Entity Relationship Model(ER), Relational Model and Extended ER Model** | 6 | |

| | | | |
|---|---|---|---|
| | **The Entity-Relationship (ER) Model:** Entity with its types Attributes with its types, Relationships with its Types. Real life Examples of ER Diagram.<br>**Relational Model**: Structure of Relational Databases, Keys with its Types , Codd's rules.<br>**Extended ER Model (EER):** Concept of Specialization**,** Generalization and Aggregation, Mapping of ER and EER to Relational Model. | | L1, L2,L3 |
| 3 | **Introduction to Structured Query Language (SQL)**<br>Overview of SQL, Data Definition Language Commands, Data Manipulation Language Commands, Data Control Language Commands, Transaction Control Language Commands, Constraints and its types, Integrity constraints, Set and String Operations, Aggregate Functions, Group by and Having Clause. | 9 | L1, L2, L3 |
| 4 | **Advanced SQL with Integrity, Security and Authorization**<br>Nested Sub queries, Referential Integrity in SQL, Joins, Views, Assertion, Trigger, Database Security and Authorization, Granting of Privileges, Revoking of Authorization in SQL<br>Relational Algebra Operations and concept of tuple relational calculus and domain relational calculus | 11 | L1, L2, L3 |
| 5 | **Relational Database Design**<br>Pitfalls in Relational Database Design, Concept of Normalization, Functional Dependencies and its types, 1 NF, 2 NF, 3 NF, BCNF, 4 NF | 8 | L1, L2, L3 |
| 6 | **Transaction, Recovery and Concurrency Control**<br>**Transaction Management:** Transaction Concept, Transaction States, ACID Properties of Transaction, Serial and Concurrent Executions, Conflict and View Serializability.<br>**Concurrency Control:** Lock Based Protocols, Timestamp based protocols, Deadlock Handling<br>**Recovery:** Failure Classification, Log based recovery, Checkpoint, Shadow Paging. | 8 | L1, L2 |
| | **Total Hours** | **45** | |

## Books and References:

| Sr. No. | Title | Authors | Publisher | Edition | Year |
|---|---|---|---|---|---|
| 1 | Database System Concepts | Korth, Slberchatz,Sudarshan | McGraw Hill | Seventh Edition | 2019 |
| 2 | Fundamentals of Database Systems | Elmasri and Navathe | Pearson education | Seventh Edition | 2016 |
| 3 | Database Management Systems | Raghu Ramkrishnan and Johannes Gehrke | McGraw Hill | Third Edition | 2014 |

## Online References:

| Sr. No. | Website Name | URL | Modules Covered |
|---|---|---|---|
| 1 | www.guru99.com | https://www.guru99.com/dbms-tutorial.html | M1,M2,M6 |
| 2 | www.javatpoint.com | https://www.javatpoint.com/dbms-tutorial | M1-M6 |
| 3 | www.studytonight.com | https://www.studytonight.com/dbms/ | M1 to M3,M5 |
| 4 | www.w3schools.in | https://www.w3schools.in/dbms/<br>https://www.w3schools.com/sql/default.asp | M1,M2,M5,M6<br>M3,M4 |
| 5 | www.geeksforgeeks.org | https://www.geeksforgeeks.org/dbms/ | M1- M6 |
| 6 | www.tutorialcup.com | https://www.tutorialcup.com/dbms | M1, M2, M5,M6 |

## List of Practical/ Experiments:

| Practical Number | Type of Experiment | Practical/ Experiment Topic | Hrs. | Cognitive levels of attainment as per Bloom's Taxonomy |
|---|---|---|---|---|
| 1 | **Basic Experiments** | To study any two open source and commercial Relational database management system(Open source- MySQL, PostgreSQL, Commercial - SQL Server, Oracle) | 2 | L1, L2 |
| 2 | **Design Experiments** | Develop an Entity-Relationship (ER) diagram for the problem definition Identified and convert it into Relational Database. | 2 | L1, L2, L3 |
| 3 | | Apply DDL Commands to Specified System | 2 | L1, L2, L3 |
| 4 | | Apply DML Commands to Specified System | 2 | L1, L2, L3 |
| 5 | | Apply Constraints for the Specified system. | 2 | L1, L2, L3 |
| 6 | | Apply Set , String and Aggregate Operations to Specified System | 2 | L1, L2, L3 |
| 7 | | SQL Program to perform Advanced DML operations.(Group by, having) | 2 | L1, L2, L3 |
| 8 | | SQL program to study sub-query in SQL. | 2 | L1, L2, L3 |
| 9 | | SQL program to study views and Triggers in SQL. | 2 | L1, L2, L3 |
| 10 | | SQL program to study joins in SQL. | 2 | L1, L2, L3 |
| 11 | **Mini/Minor Projects/ Seminar/ Case Studies** | 1. Student Management System<br>2. Library Management System<br>3. Airline Reservation System<br>4. Hospital Management System<br>5. Hotel Management System<br>6. Billing System | 10 | L1, L2, L3 |
| | | **Total Hours** | **30** | |

## List of Tutorials:

| Tutorial Number | Topic | Hrs. | Cognitive levels of attainment as per Bloom's Taxonomy |
|---|---|---|---|
| 1 | Solve and Build SQL Queries on DDL Commands. | 1 | L1, L2, L3 |
| 2 | Solve and Build SQL Queries on DML Commands | 1 | L1, L2, L3 |
| 3 | Solve and Build SQL Queries on Constraints in SQL | 1 | L1, L2, L3 |
| 4 | Solve and Build SQL Queries on Aggregate Functions. | 1 | L1, L2, L3 |
| 5 | Solve and Build SQL Queries on Set Functions. | 1 | L1, L2, L3 |
| 6 | Solve and Build SQL Queries on String Operations | 1 | L1, L2, L3 |
| 7 | Solve and Build SQL Queries on Group by and Having Clause | 1 | L1, L2, L3 |
| 8 | Outline ER Diagram for given real life problem and convert it into relational Database. | 1 | L1, L2, L3 |
| 9 | Solve and Build SQL Nested Queries | 1 | L1, L2, L3 |
| 10 | Solve and Build SQL Queries on Referential Integrity | 1 | L1, L2, L3 |
| 11 | Solve and Build SQL Queries on Joins | 1 | L1, L2, L3 |
| 12 | Solve Build SQL Queries on Real Time Management Systems. | 1 | L1, L2, L3 |
| 13 | Develop Database design by applying concept of Normalization to Student Management System | 1 | L1, L2, L3 |
| 14 | Develop Database design by applying concept of Normalization to Hospital Management System | 1 | L1, L2, L3 |
| 15 | Develop Database design by applying concept of Normalization to Airlines Reservation System | 1 | L1, L2, L3 |
| | **Total Hours** | **15** | |

# INDEX

# Module: 01
# Introduction to Database Concepts

## Lecture : 1

### Motivation:

To acquire the details of database basics with the structure of database.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Basic Concepts of Data, Database and DBMS, Applications of Databases, Advantages of Databases over File Processing System | 1 | 1 |
| 2 | 3 Level Architecture of Database System, Data Abstraction and Data Independence, Database Languages | 1 | 1 |
| 3 | Database Users, Database Administrator and its functions, Overall System Structure | 1 | 1 |

### Learning Objective:

- Learner shall be aware of need of database.
- Learner should be aware of structure of database.
- Learners should know database basics.
- Learners shall be able to explain the mechanism of corrosion.
- Learners should know architecture of database.

### Theoretical Background:

Students can know what database is and what are the disadvantages of file processing system due to which databases came into existence. They can distinguish among different database users and are able to know functions of database administrator.

## Key Definitions:

1. **Schema –** the overall design of the database is called the database schema
2. **Database** -A logically coherent collection of data with some inherent meaning,
3. **An entity set**-A set of entities of the same type that share the same properties. Example: set of all persons, companies, trees, holidays
4. **Instance –** The actual content of the database at a particular point in time is called an instance. The concept of instance is analogous to the value of a variable
5. **Physical Data Independence –** The ability to modify the physical schema without changing the logical schema is called the physical data independence.
6. **Database Administrator**- A database administrator (DBA) is a person who has central control over the system.

## Course Content

**Basic Concepts of Data, Database and DBMS, Applications of Databases**

**Advantages of Databases over File Processing System**

### Basic Concepts of Data

Data is simply defined as raw information. Data can be any collection of characters, numbers, videos, images, documents etc.

### Database and DBMS

The collection of data, usually referred to as the database, contains information relevant to an enterprise.

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. Management of data includes storage, access mechanism, security etc. of data.

The primary goal of DBMS is to provide an environment that is both convenient and efficient to retrieve and store database information.

### Applications of Databases

Databases are widely used. Here are some representative applications:

1. **Enterprise Information**

Sales: For customer, product, and purchase information.

Accounting: For payments, receipts, account balances, assets, and other accounting information.

Human resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of pay checks.

Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.

Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

2. **Banking and Finance**

Banking: For customer information, accounts, loans, and banking transactions.

Credit card transactions: For purchases on credit cards and generation of monthly statements.

Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also, for storing real-time market data to enable online trading by customers and automated trading by the firm.

3. **Universities**: For student information, course registrations, and grades

4. **Airlines**: For reservations and schedule information.

5. **Telecommunication**: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

**Advantages of Databases over File Processing System**

1. Data redundancy and inconsistency :-

Redundancy :-

Redundancy is nothing but replication of same data in number of files. As a result of redundancy there is wastage of space as well as access cost increases.

E.g :- In above savings bank Enterprise example address and mobile number of particular customer can occur in savings account file as well as checking's account file.

Inconsistency:-

Inconsistency is nothing but mismatching of various copies of same data. As a result of inconsistency there is confusion in selection of data as the details of same data do not match.

E.g :- In above savings bank Enterprise example Address and Mobile Nos. Of customers are saved in savings account file as well as checking's account file. If address of particular customer changes and programmer makes the changes in savings account file but forgets to make change in the checking's account file then address stored in both files for a particular customer will differ.

2. Difficulty in accessing data :-

Many times some reports are required for analysis purpose. As this requirement was not known to the programmer when program was written either the programmer as to write new application program or to sort manually the data which is required from existing application programs. Thus it becomes very difficult to access the required data.

E.g :-

In above savings bank Enterprise examples suppose bank manager wants to see the details of customers residing in Mumbai city. In the existing application there is facility to view the details of all customers. To get required information there are 2 options either sort the required data manually form list of all customers or to say to the programmer to write another application.

If the programmer writes another application program and after some days the bank manager needs the details of all customers of Mumbai who has balance in account more than Rs.5000/- then again the same above situation arises.

3 . Data isolation:-

In file processing system data is scattered in various files and files are in various file formats. Due to this it is very difficult to write new application programs to retrieve appropriate data.

4. Integrity Problem:-

It is important that data must satisfy some predefined set of rules, as determined by the database administrator or application developer. But as time passes there can be change in rule or new rule can come into existence. Hence there is a need to write new application program.

E.g :- In above savings bank Enterprise example suppose bank Manager sets a rule that every customer who has issued a cheque book should have balance of Rs.1000/-Developers enforce this constraint by writing the code in various application programs.

But suppose now the Manager sets new constraints that person can withdraw only Rs.10000/- per day then it is difficult to change the program to enforce the new constraint.

5. Atomicity Problem :-

Atomicity states that file processing system must follow an “all or nothing” rule. Each transaction is said to be “atomic.” If one part of the transaction fails, the entire transaction fails. A Computer system like any other mechanical device is subject to failure. In many applications it is difficult to detect that data restored to the consistent state that existed before failure.

E.g :- In above savings bank Enterprise example consider that there are 2 accounts A and B. Suppose A wants to transfer Rs.100/-to B’s account. After deduction of Rs.100/- from A’s account if the system fails then database goes to inconsistent state as A was debited Rs.100/- but B was not credited it.

Thus either both debit and credit should occur or none should occur.

6. Concurrent Access Anomalies:-

To improve the performance many systems allow multiple users to simultaneously update the data. But many times while updating the data simultaneously system goes into inconsistent state. To avoid above situation system must provide some form of supervision which is not provided by file processing system.

E.g :-In above savings bank Enterprise example consider bank account A having balance of Rs.1000/-

If 2 customers P & Q withdraws Rs.50/- and Rs.100/- simultaneously. Consider following scenario. Initially both customers read account balance as Rs.1000/- in its own memory.

When Customer P withdraws Rs.50/- from the account A he overwrites the balance Of A as Rs.950/-(1000 – 50 ).When Customer Q withdraws Rs.100/- from the account A he overwrites the balance Of A as Rs.900/-(1000 – 100 ).

Now after completion of both actions balance of A is 900 which is not a correct value as the correct value of account A is Rs.850/-(1000-(50+100)).

7. Security Problems :-

The Security of a system is a combination of its ability to support: System Availability, Data Integrity , Data Confidentiality

A failure of a system to protect any of these characteristics amounts to a Security violation or weakness.

File Processing System does not provide security to data.

E.g :- In savings bank Enterprise example one customer should not be allowed to change the data of other customer.

Due to above disadvantages of file processing system Data base management system came into existence which provided functionalities to avoid above disadvantages.

## *Let's check the take away from this lecture*

1) **Which is complex among below?**

a) File processing system c) Both a) & b)

b) **Database System** d) can't say

2) **Data and information are same.**

a) True **c) False**

Exercise

Q.1 What is difference between database and database management system?

Q.2 What are advantages of database system?

Questions/Problems for practice:

Q.3 List different databases available in market.

**Learning from this lecture**: Learners will be able to understand basics of database and need of database.

# Lecture: 2

## 3 Level Architecture of Database System, Data Abstraction and Data Independence

**Learning objective:** In this lecture learners will be able to understand the data abstraction level from architectural view.

### 3 Level Architecture of Database System

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

A 3-level/tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

**Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

**Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

**User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

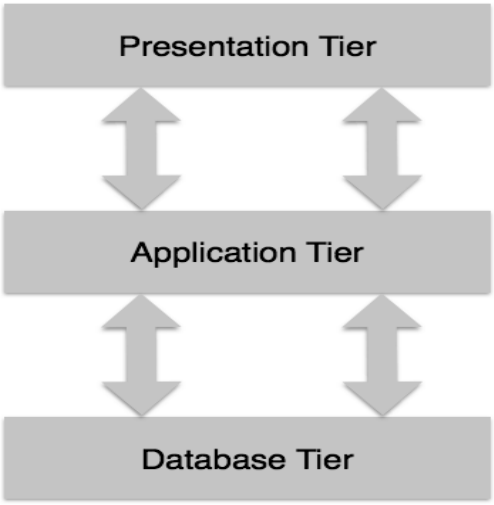



Fig 1.1 Three level/tier architecture of DBMS

**Data Abstraction and Data Independence**

Data Abstraction: Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Physical level: The lowest level of abstraction describes how the data are actually stored.

Eg. complex low-level data structures

Logical level: Describes what data are stored in the database, and what relationships exist among those data.

View level: Describes only part of the entire database. The system may provide many views for the same database.

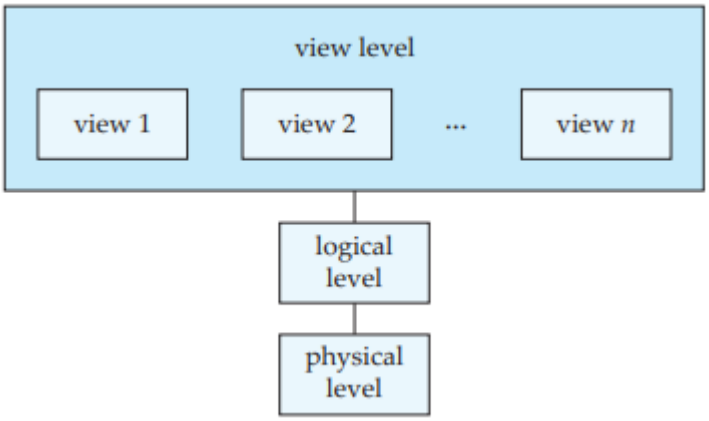



Fig 1.2 The three levels of data abstraction

Above figure shows the relationship among the three levels of abstraction.

Eg: At the physical level, an instructor, department, or student record can be described as a block of consecutive storage locations.

At the logical level, each such record is described by a type definition. Programmers using a programming language work at this level of abstraction.

At the view level, several views of the database are defined, and a database user sees some or all of these views.

Data Independence

--User of the logical level does not need to be aware of complexity of physical level. This is referred to as physical data independence.

--Similarly, User of the view level does not need to be aware of complexity of logical level. This is referred to as logical data independence.

--Application programs are said to exhibit physical data independence if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

**Database Languages**

A database system provides 2 different types of languages : Data Definition Language (to specify database schema) and Data Manipulation Language (to express database queries and updates)

**Data Definition Language(DDL)**

It is used to specify a database scheme as a set of definitions expressed in a DDL.

DDL statements are compiled, resulting in a set of tables stored in a special file called a data dictionary or data directory.

The data directory contains metadata (data about data)

The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called a data storage and definition language

basic idea of DDL is to hide implementation details of the database schemes from the users

**Data Manipulation Language(DML)**

Data Manipulation is:

- Retrieval of information from the database
- Insertion of new information into the database
- Deletion of information in the database
- Modification of information in the database

A DML is a language which enables users to access and manipulate data.

The goal is to provide efficient human interaction with the system.

There are two types of DML:

1) Procedural: the user specifies what data is needed and how to get it

2) Nonprocedural: the user only specifies what data is needed

Nonprocedural DML's are easier for user and may not generate code as efficient as that produced by procedural languages

A query language is a portion of a DML involving information retrieval only. The terms DML and query language are often used synonymously.

## *Let's check the take away from this lecture*

**1) DML**

a) Creates database
b) secures data
c) verifies data
**d) Updates data**

**2) Programmers work in following level:**

a) Physical level
**b) Logical level**
c) View Level
d) All of the above

Exercise

Q.1 Discuss the database languages.

Q.2 What do you mean by data abstraction?

Questions/Problems for practice:

Q.3 Explain 3 level architecture of database with example.

**Learning from this lecture:** Learners will be able to understand different levels at which data gets stored and database languages.

# Lecture: 3

**Database Users, Database Administrator and its functions, Overall**

**System Structure**

**Learning objective**: In this lecture learners will be able to understand system architecture of database

**Database Users**

**Application programmers** are computer professionals interacting with the system through DML calls embedded in a program written in a host language (e.g. C, PL/1, Pascal).

- These programs are called **application programs**.
- The **DML precompiler** converts DML calls (prefaced by a special character like $, #, etc.) to normal procedure calls in a host language.
- The host language compiler then generates the object code.
- Some special types of programming languages combine Pascal-like control structures with control structures for the manipulation of a database.
- These are sometimes called **fourth-generation languages**.
- They often include features to help generate forms and display data.

**Sophisticated users** interact with the system without writing programs.

- They form requests by writing queries in a database query language.
- These are submitted to a **query processor** that breaks a DML statement down into instructions for the database manager module.

**Specialized users** are sophisticated users writing special database application programs. These may be CADD systems, knowledge-based and expert systems, complex data systems (audio/video), etc.

**Naive users** are unsophisticated users who interact with the system by using permanent application programs (e.g. automated teller machine).

**Database Administrator and its functions:**

A database administrator (DBA) is a person who has central control over the system.

- **Installation of new software** — It is primarily the job of the DBA to install new versions of DBMS software, application software, and other software related to DBMS administration. It is important that the DBA or other IS staff members test this new software before it is moved into a production environment.

- **Configuration of hardware and software with the system administrator —** In many cases the system software can only be accessed by the system administrator. In this case, the DBA must work closely with the system administrator to perform software installations, and to configure hardware and software so that it functions optimally with the DBMS.
- **Security administration —** Ones of the main duties of the DBA is to monitor and administer DBMS security. This involves adding and removing users, administering quotas, auditing, and checking for security problems.
- **Data analysis —** The DBA will frequently be called on to analyze the data stored in the database and to make recommendations relating to performance and efficiency of that data storage. This might relate to the more effective use of indexes, enabling "Parallel Query" execution, or other DBMS specific features.
- **Database design (preliminary) —** The DBA is often involved at the preliminary database-design stages. Through the involvement of the DBA, many problems that might occur can be eliminated. The DBA knows the DBMS and system, can point out potential problems, and can help the development team with special performance considerations.

**Database System Architecture**

Database systems are partitioned into modules for different functions. Some functions (e.g. file systems) may be provided by the operating system.

Components include:

- **File manager** manages allocation of disk space and data structures used to represent information on disk.
- **Database manager**: The interface between low-level data and application programs and queries.

  The data manager is the central software component of the DBMS. It is sometimes referred to as the database control system. One of the functions of the data manager is to convert operations in the user's queries coming directly via the query processor or\ indirectly via an application program from the user's logical view to a physical file system. The data manager is responsible for interfacing with the file system as show. In addition, the tasks of enforcing constraints to maintain the consistency and integrity of the data, as well as its security, are also performed by the data manager. It is also the responsibility of the Data. Manager to provide the synchronization in the simultaneous operations performed by concurrent users and to maintain the backup and recovery operations.
- **Query processor** translates statements in a query language into low-level instructions the database manager understands. (May also attempt to find an equivalent but more efficient form.) A Database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and query processor components. The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in of data. A gigabyte is 1000 megabytes (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes), since the main memory of computers cannot

store this much information, the information is stored on disks. Data are moved between disk storage and main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the central processing unit, it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory. The Query Processor is important because it helps the database system simplify and facilitate access to data. High level views help to achieve this goal; with them, users of the system are not be burdened unnecessarily with the physical details of the implementation of the system. However, quick processing of updates and queries is important. It is the job of the database system to translate updates and queries is important. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

- **DML precompiler** converts DML statements embedded in an application program to normal procedure calls in a host language. The precompiler interacts with the query processor.
- **DDL compiler** converts DDL statements to a set of tables containing metadata stored in a data dictionary.

In addition, several data structures are required for physical system implementation:

- **Data files:** store the database itself.
- **Data dictionary:** stores information about the structure of the database. It is used **heavily**. Great emphasis should be placed on developing a good design and efficient implementation of the dictionary.
- **Indices:** provide fast access to data items holding values.

Below Figure shows these components.

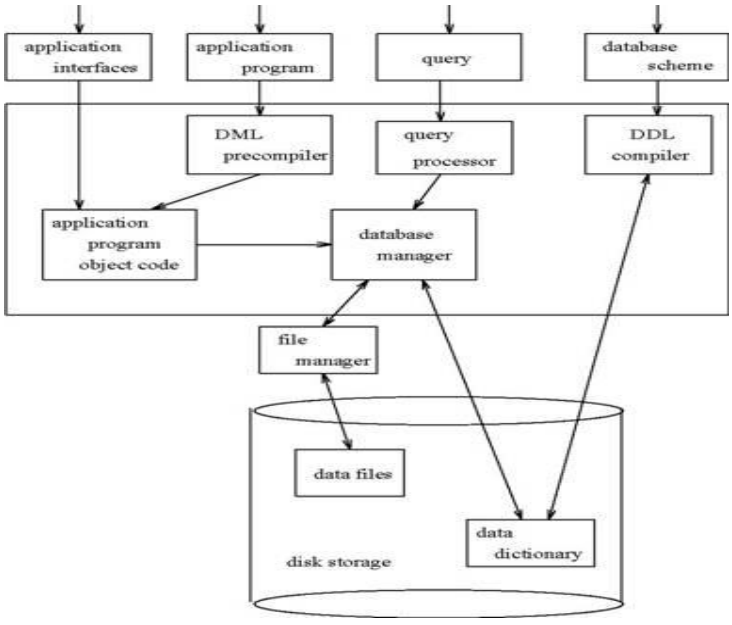



Fig 1.3 System Architecture

## *Let's check the take away from this lecture*

1) **Which of the following is database user?**

a) Programmers c) Database admin

**b)** Website Users **d) All of the above**

Exercise

Q.1 List various database users.

Q.2 Explain role of database administrator.

Questions/Problems for practice:

Q.3 Illustrate database system architecture.

**Learning from this lecture:** Learners will be able to explain role of database administrator and system architecture of database.

**Conclusion**

The study of system architecture illustrates different components of architecture. This study gives overall idea of roles of database administrators and users of database.

**Short Answer Questions:**

1. What is database?

Ans) Database is collection of data relevant to particular enterprise.

2. What is DBMS?

Ans) DBMS is collection of data and programs to access those data.

3. List any 4 DBMS softwares.

Ans) Oracle, Teradata, IBM DB2, Maria DB

4. What are database languages?

Ans) DML, DDL

5. What is query language?

Ans) It is a language used to access data from database.

**Long Answer Questions:**

1. Explain database administrator and its functions.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 1

2. Explain data independance in detail.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 1.

3. Discuss overall architecture of database

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter

4. What are different database users?

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 1.

5. Explain applications of database.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 1.

6. Explain drawback of file processing system.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 1.

**Set of Questions for FA/CE/IA/ESE**

1. Explain different database users.
2. Explain Functions of Database Administrator.
3. Explain physical level data independence.
4. Explain view of data.
5. Data structure of stored data is maintained by which level of data view? Explain in detail.

References:

1) Database Management System by Korth, Sudarshan, Siverchatz

**Practice for Module-01**

Q1. Explain Disadvantages of file processing System.

Q2. Explain Overall architecture of DBMS with diagram.

Q3. Explain Different Database Users.

Q4. Explain Functions of Database Administrator.

Q5. Explain Instances and Schema.

Q6. Explain Different Database Languages.

Q7. Define Database and database management System.

**Self-assessment**

Q.1) Define data abstraction. Explain with example.

Q. 2) Explain types of database users.

Q. 3) Explain physical and logical data independence.

**Self-evaluation**

| Name of Student | | |
|---|---|---|
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick<br>Your choice |
| 1. | Do you understand the meaning of database and DBMS? | o Yes<br>o No |
| 2. | Do you understand need of database? | o Yes<br>o No |
| 3. | Do you know the various views of data? | o Yes<br>o No |
| 4. | Do you understand architecture of database? | o Yes<br>o No |
| 5. | Do you understand module ? | o Yes, Completely.<br>o Partialy.<br>o No, Not at all. |

# Module:02

# Entity Relationship Model(ER), Relational Model and Extended ER Model

## Lecture : 1

### Motivation:

Detail database design of whole firm or even part of them are very complex and therefor difficult to understand small part of the whole design which would be easy to read, cannot give the overall understanding of the system. ER design is possibility to present a complex system as an understandable way.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Entity with its types, Attributes with its types | 1 | 1 |
| 2 | Relationships with its Types | 1 | 1 |
| 3 | Real life Examples of ER Diagram. | 1 | 1 |
| 4 | Relational Model: Structure of Relational Databases, Keys with its Types | 1 | 1 |
| 5 | Extended ER Model (EER): Concept of Specialization, Generalization and Aggregation | 1 | 1 |
| 6 | Mapping of ER and EER to Relational Model | 1 | 2 |

### Learning Objective:

- Learner should know types of data models for database.
- Learners shall be able to learn the significance of the ER-model for DB design.
- Learners shall be able to explain the basic constructs of the ER-model.
- Learners shall be able to develop ER-diagrams (schema in the ER-model) for a given (small) application

### Theoretical Background:

One must be motivated about building relational model of database. To buil this model all basic concepts are introduced in this module. Using this concepts, one can model E-R diagram which is then converted into relational model.

## Key Definitions:

1. **Data Model**- is a Conceptual tool used to describe Data, Data relationships, Data semantics

   Entity- Is an object that exists and is distinguishable from other objects.

   Example: specific person, company, event, plant

2. **Attribute** - Is property of entity

   Example: people have names and addresses

3. **Entity set** - Is a set of entities of the same type that share the same properties.

   Example: set of all persons, companies, trees, holidays

4. **Domain –** The set of permitted values for each attribute

5. **Relationship** - Is an association among several entities

6. **Enhanced Entity-Relationships (EE-R)** model -is a revised E-R model that extends the original E-R model and supports additional semantic concepts by providing new modeling constructs.

7. **Superclass** - is an entity type that has one or more distinct sub groups with unique attributes.

8. **Subclass** - is an entity type that shares common attributes or relationships distinct from other Subclasses.

9. **Attribute Inheritance** - is the property by which subclass entities inherit attributes of the superclass. The subclass entity type is an entity type in itself with a set of attributes and relationships.

10. **Subclass Relationship** - each subclass possesses attributes and relationships of its superclass, It can have its own attributes and relationships that distinguish one subclass from another the relationship between a superclass and subclass is always a one-to-one relationship.

11. **Specialization**- is the process of defining one or more subclasses of a superclass by identifying its distinguishing characteristics. Unlike generalization, specialization is thus a top-down approach.It starts with the general entity (superclass) and forms specialized entity types (subclasses) based on specialized attributes or relationships specific to a subclass.

12. **Generalization**- is the process of defining general entity types from a set of specialized entity types by identifying their common characteristics. Generalization is a bottom-up approach as it starts with the specialized entity types (subclasses) and forms a generalized entity type (superclass).

13. **Participation constraints** - dictate whether each instance (member) of a superclass must participate as an instance (member) of a subclass. A participation of superclass instance may be mandatory or optional in one or more subclasses
14. **Total Participation Rule** - In total participation, membership is mandatory. Each instance of a superclass must be an instance of at least one subclass.
15. **Partial Participation Rule** - Membership is optional in a partial participation. An instance of a superclass does not have to be an instance of any of the subclasses.
16. **Disjoint constraints**- define whether it is possible for an instance of a superclass to simultaneously be a member of one or more subclasses. Disjoint constraints indicate whether a superclass instance can be disjointed or overlap more than one subclass.
17. **Disjoint Rule**- The disjoint rule states that if an instance of a superclass is a member of any subclass, then it cannot be a member of more than one subtype
18. **Overlap Rule**- The overlap rule states that if an instance of a superclass is a member of any subclass, then it can be a member (overlap) of more than one subtype.
19. **Subclass Discriminator**- The participation and disjoint constraints allow an instance of a superclass to be a member of one or more subclasses. We keep track of membership using a special attribute of a superclass, a subclass discriminator.

## Course Content

**Entity with its types**

An entity is an object that exists and is distinguishable from other objects.

Example: specific person, company, event, plant.

Entity sets of a relationship need not be distinct.

The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works-for relationship set.

Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.

Role labels are optional, and are used to clarify semantics of the relationship.

**Attributes with its types**

Entities have attributes.

Example: people have names and addresses

An entity set is a set of entities of the same type that share the same properties.

Example: set of all persons, companies, trees, holidays.

Attribute types: Simple and composite attributes.

E.g. Simple: Name, Composite: Address

Single-valued and multi-valued attributes

E.g.Single-valued attribute: Roll_No, multi valued attribute: phone-numbers

Derived attributes can be computed from other attributes

E.g. age, given date of birth

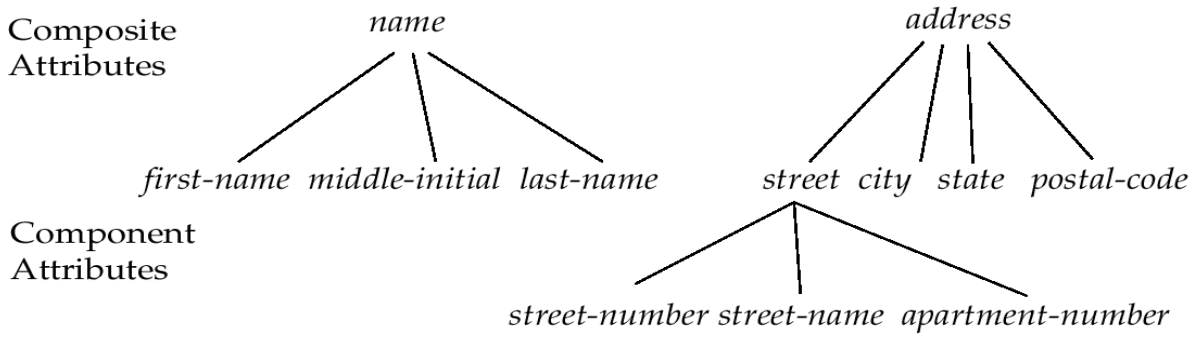



Fig. 2.1 Attribute types

**Notations:**

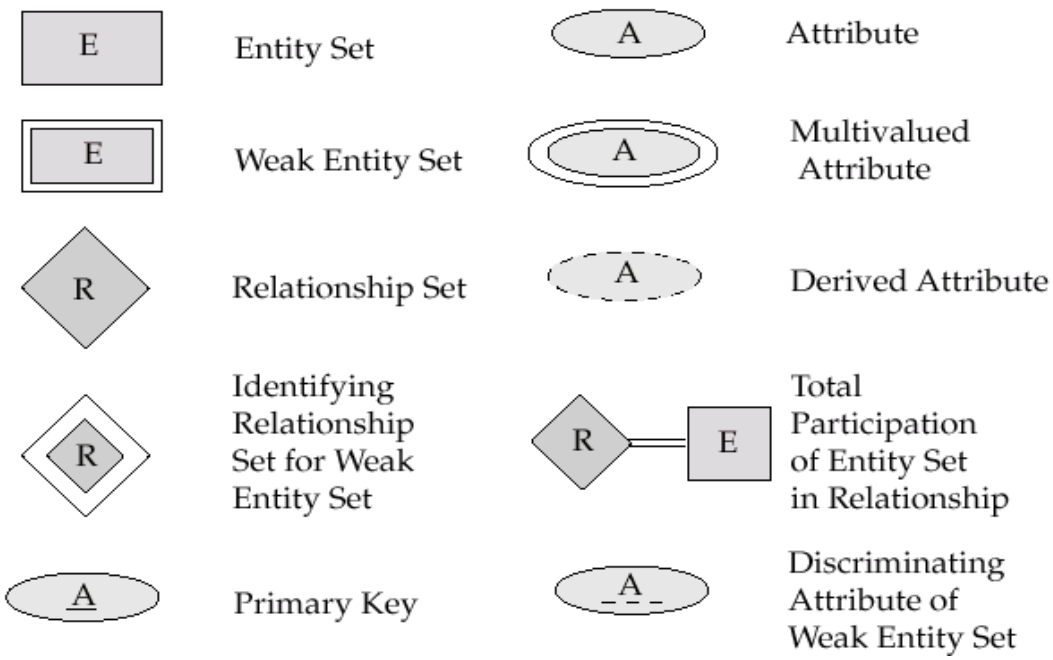



Fig 2.2 Notations

## *Let's check the take away from this lecture*

**1) Notation used to represent relationship is _____.**

**a) Diamond** b) rectangle
c) Ellipse d) None

**2) Rows are also called as tuples.**

**a) True** b) False

Exercise
Q.1 What are different types of attributes?
Q.2 What are the notations used in E-R diagram .Explain with suitable example.
Questions/Problems for practice:
Q.3 We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?

**Learning from this lecture:** Learners will be able to understand basic constructs of ER model.

# Lecture: 2

**Relationships with its Types**

**Learning objective:** In this lecture learners will be able to understand how to represent relationship between entity sets.

### Relationship:

- A relationship is an association among several entities

  Example:

  Sam admits CSE

  student entity relationship set department entity

- A relationship set is a mathematical relation among n ≥ 2 entities, each taken from entity sets

  $\{(e1, e2, \ldots en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$

  where (e1, e2, …, en) is a relationship

Example:

(Sam, CSE) ∈ admits

**Roles:**

- Entity sets of a relationship need not be distinct
- The labels "manager" and "worker" are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship

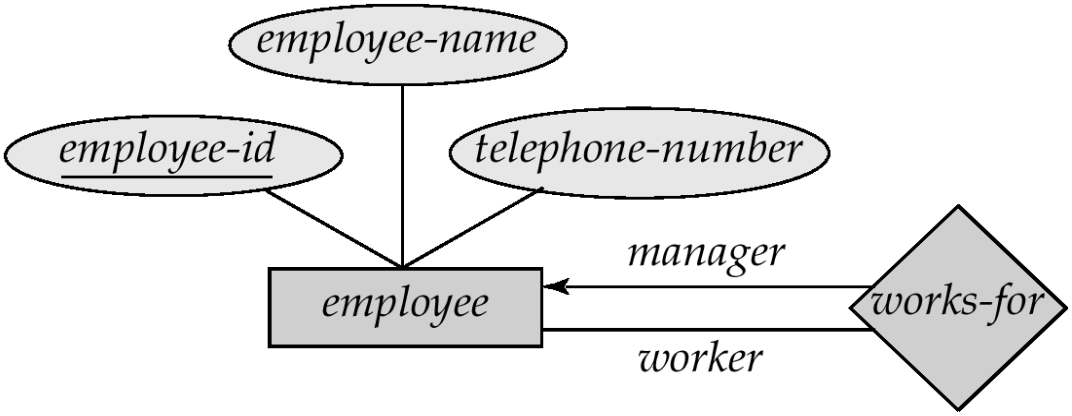



Fig. 2.3 Roles

**Mapping Cardinalities:**

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
    - One to one
    - One to many
    - Many to one
    - Many to many

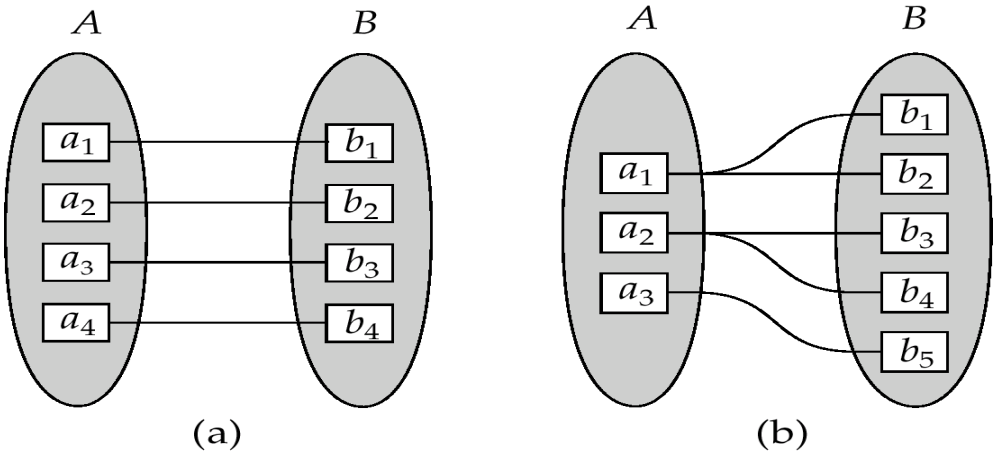



Fig 2.4 One to one & One to Many

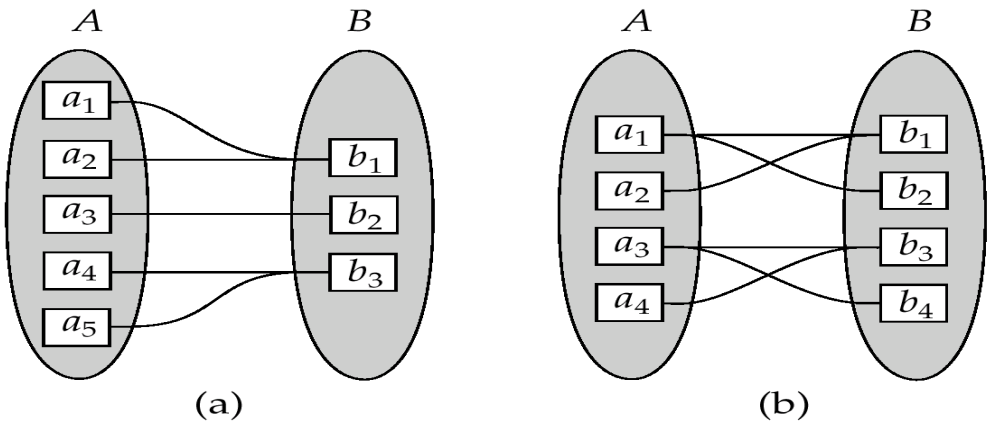



Fig 2.5 Many to one & Many to many

**Ternary Relationship**

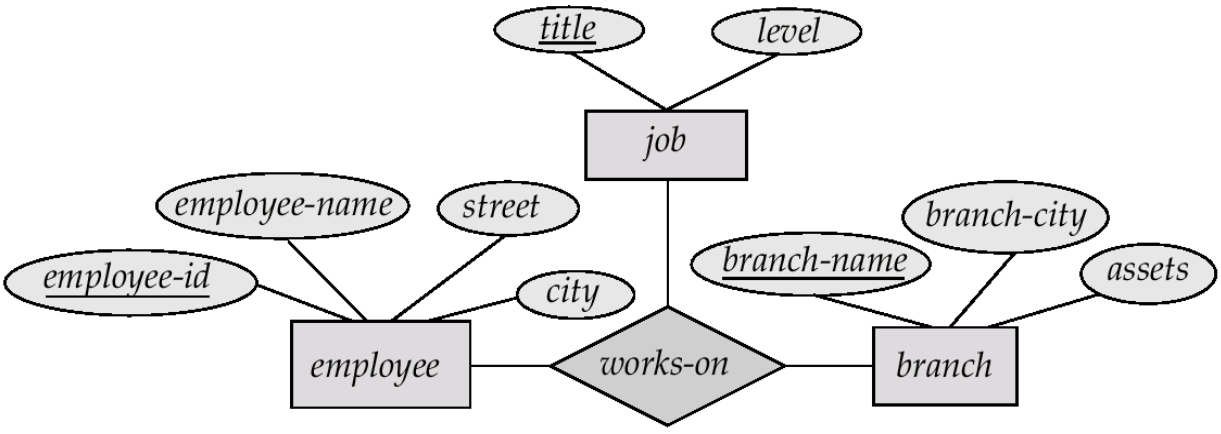



Fig. 2.6 E-R Diagram with a Ternary Relationship

**Cardinality Constraints on Ternary Relationship**

- At most one arrow is allowed out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g. an arrow from works-on to job indicates each employee works on at most one job at any branch.
- If there is more than one arrow, there are two ways of defining the meaning.

E.g a ternary relationship R between A, B and C with arrows to B and C could mean

1. Each A entity is associated with a unique entity from B and C or
2. Each pair of entities from (A, B) is associated with a unique C entity, and each pair (A, C) is associated with a unique B.

**Binary Vs. Non-Binary Relationships**

- Some relationships that appear to be non-binary may be better represented using binary relationships
- E.g. A ternary relationship parents, relating a child to his/her father and mother, is best replaced by two binary relationships, father and mother
- Using two binary relationships allows partial information (e.g. only mother being know)
- But there are some relationships that are naturally non-binary
- E.g. works-on

*Let's check the take away from this lecture*

1) **A relationship is _____**

a) An item in an application c) collection of related entities

b) **A meaningful dependency between entities** d) related data

2) **Notation used to represent relationship is _____.**

a) **Diamond** c) Ellipse

b) Rectangle d) Arrow

Exercise

Q.1 Explain ternary relationship with example.

Q.2 What is the need of relationship?

Questions/Problems for practice:

Q.3 Illustrate roles with example.

**Learning from this lecture**: Learners will be able to understand basic constructs of ER model.

# Lecture: 3

## Real life Examples of ER Diagram

**Learning objective**: In this lecture learners will be able to draw E-R diagram.

Here we are going to design an Entity Relationship (ER) model for a college database. Say we have the following statements:

1. A college contains many departments
2. Each department can offer any number of courses
3. Many instructors can work in a department
4. An instructor can work only in one department
5. For each department there is a Head
6. An instructor can be head of only one department
7. Each instructor can take any number of courses
8. A course can be taken by only one instructor
9. A student can enroll for any number of courses
10. Each course can have any number of students

Step 1: Identify the Entities and attributes

1. From the statements given, the entities are: Department ,Course,Instructor,Student

Stem 2: Identify the relationships

1. One department offers many courses. But one particular course can be offered by only one department. hence the cardinality between department and course is One to Many (1:N)
2. One department has multiple instructors. But instructor belongs to only one department. Hence the cardinality between department and instructor is One to Many (1:N)
3. One department has only one head and one head can be the head of only one department. Hence the cardinality is one to one. (1:1)
4. One course can be enrolled by many students and one student can enroll for many courses. Hence the cardinality between course and student is Many to Many (M:N)
5. One course is taught by only one instructor. But one instructor teaches many courses. Hence the cardinality between course and instructor is Many to One (N :1)

Step 3: Identify the key attributes

1. Course_ID is the key attribute for "Course" Entity.
2. "Department_Name" can identify a department uniquely.

   Hence Department_Name is the key attribute for the Entity "Department".
3. Student_ID is the key attribute for "Student" Entity.
4. Instructor_ID is the key attribute for "Instructor" Entity

Step 4: Identify other relevant attributes

1. For the department entity, other attributes are location
2. For course entity, other attributes are course_name,duration
3. For instructor entity, other attributes are first_name, last_name, phone
4. For student entity, first_name, last_name, phone

Step 5: Draw complete ER diagram by using all these details:

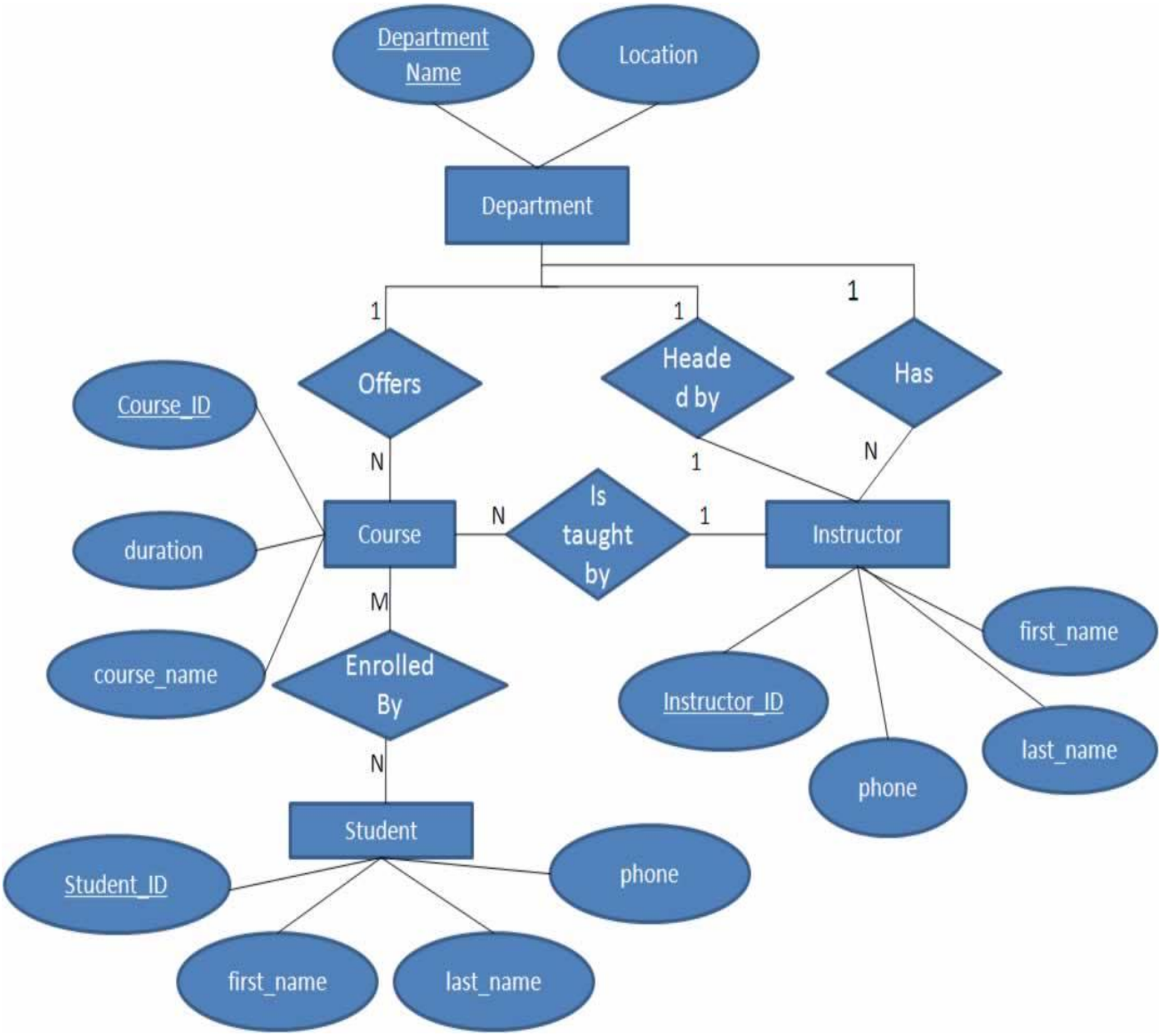



Fig. 2.7 E-R Diagram

## *Let's check the take away from this lecture*

1) **A table can have more than one primary key**

**a) True** b) False

**2) Table can exist without having primary key.**

a) **True** b) False

Exercise

Q.1 Draw an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examination conducted.

Q.2 Construct E-R diagram for customer branch and account relationship.

Questions/Problems for practice:

Q.3 Draw E-R diagram for banking enterprise.

**Learning from this lecture:** Learners will be able to draw E-R diagram for enterprise application.

# Lecture: 4

## Relational Model: Structure of Relational Databases, Keys with its Types

**Learning objective:** In this lecture learners will able to understand relational database and different types of keys.

### Structure of Relational Databases

- The whole data is conceptually represented as an orderly arrangement of data into rows and columns, called a relation or table.
- All values are scalar. That is, at any given row/column position in the relation there is one and only one value.
- All operations are performed on an entire relation and result is an entire relation, a concept known as closure.

### Relational View of Sample database

- Let us take an example of a sample database consisting of supplier, parts and shipments tables. The table structure and some sample records for supplier, parts and shipments tables are given as Tables as shown below:

Table 2.1: Relation View of database

The Supplier Records

| Sno | Name | Status | Status |
|---|---|---|---|
| S1 | Suneet | 20 | Qadian |
| S2 | Ankit | 10 | Amritsar |
| S3 | Amit | 10 | Qadian |

The Part Records

| Pno | Name | Status | Weight | City |
|---|---|---|---|---|
| P1 | Nut | Red | 12 | Qadian |
| P2 | Bolt | Green | 17 | Amritsar |
| P3 | Screw | Blue | 17 | Jalandhar |
| P4 | Screw | Red | 14 | Qadian |

The Shipment Records

| Sno | Name | Qty |
|---|---|---|
| S1 | P1 | 250 |
| S1 | P2 | 300 |
| S1 | P3 | 500 |
| S2 | P1 | 250 |
| S2 | P2 | 500 |
| S3 | P2 | 300 |

- Assume that each row in Supplier table is identified by a unique SNo (Supplier Number), which uniquely identifies the entire row of the table. Likewise each part has a unique PNo (Part Number). Also, we assume that no more than one shipment exists for a given supplier/part combination_in the shipments table.
- Note that the relations Parts and Shipments have PNo (Part Number) in common and Supplier and Shipments relations have SNo (Supplier Number) in common. The Supplier and Parts relations have City in common. For example, the fact that supplier S3 and part P2 are located in the same city is represented by the appearance of the same value, Amritsar, in the city column of the two tuples in relations.

**Keys with its Types:** A key is used to uniquely identify the set of attributes of tuple.

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

  Eg: Roll No. of student relation is key

- A **candidate key** of an entity set is a minimal super key.

  Customer-id is candidate key of customer

  account-number is candidate key of account

- One of the candidate key is chosen as **primary key** of a relation.

- Referential integrity constraints are used to specify the relationships between two relations in a database.
- Consider a referencing relation, R1, and a referenced relation, R2. Tuples in the referencing relation, R1, have attributed FK (called foreign key attributes) that reference the primary key attributes of the referenced relation, R2. A tuple, t1, in R1 is said to reference a tuple, t2, in R2 if t1[FK] = t2[PK].

  Eg: Branch(Branch_Name, City)
  Account(Account_No, Branch_Name, Balance)

  In above schema, Branch_Name of Account schema is foreign schema. It is referencing Branch_Name which is primary key of Branch schema.

## *Let's check the take away from this lecture*

**1) Tuple is _____**

a) Column
b) Collection of Rows and Column
**c) Row**
d) Relation

**2) A table can have more than one candidate key.**

**a) True**
b) False

Exercise:

Q.1 What is difference between Super key and Candidate key?

Q.2 Explain relational database with example.

Questions/Problems for practice:

Q.3 Explain types of keys with example.

**Learning from this lecture:** Learners will be able to understand the relational database structure and types of keys of relation.

# Lecture : 5

## Concept of Specialization, Generalization and Aggregation

**Learning objective:** In this lecture, learners will be able to understand features of Enhanced ER model.

### Concept of Specialization

Top-down design process; we designate sub groupings within an entity set that are distinctive from other entities in the set.

These sub groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.

Depicted by a triangle component labelled ISA (E.g., customer "is a" person).

Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

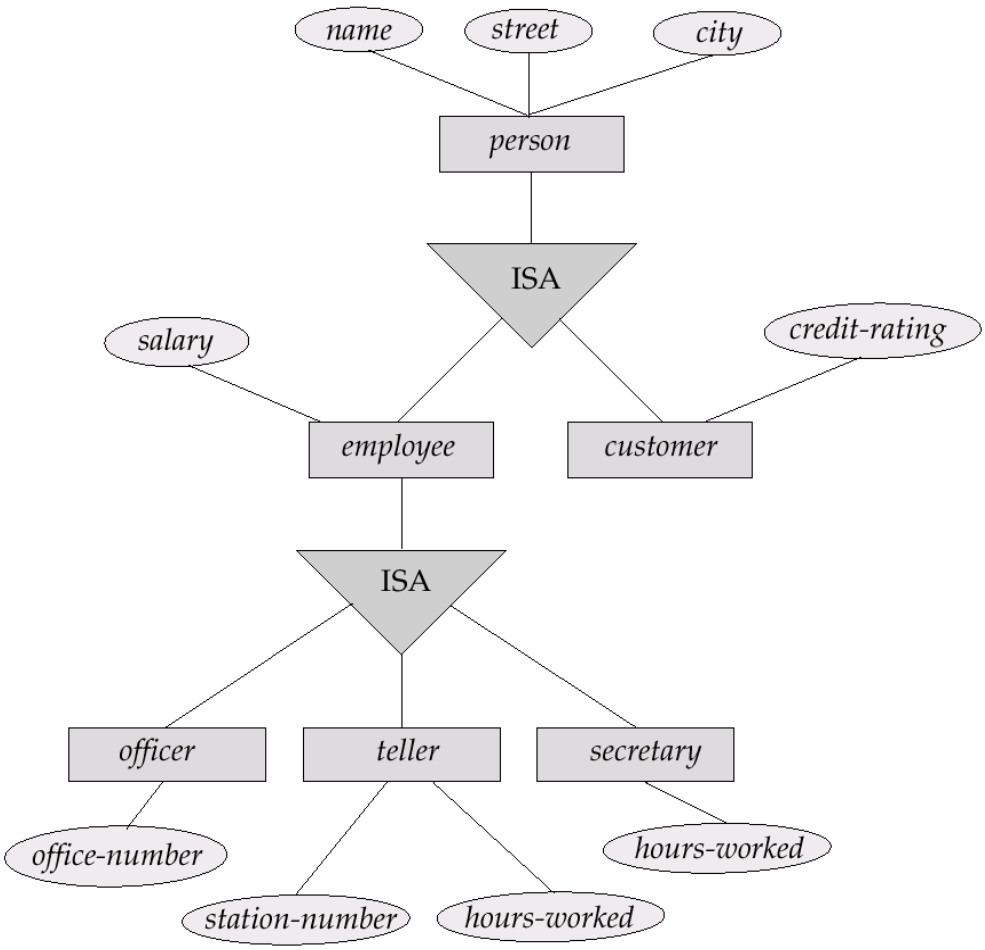



Fig. 2.8 Specialization & Generalization

**Concept of Generalization**

- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.
- Can have multiple specializations of an entity set based on different features.
- E.g., permanent-employee vs. temporary-employee, in addition to officer vs. secretary vs. Teller
- Each employee would be a member of one of permanent-employee or temporary-employee, and a member of one of officer, secretary, or teller
- The ISA relationship also referred to as super class - subclass relationship.

**Aggregation**

- Consider the ternary relationship works-on, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch.
- Relationship sets works-on and manages represent overlapping information. Every manages relationship corresponds to a works-on relationship. However, some works-on relationships may not correspond to any manages relationships. So we can't discard the works-on relationship.
- Eliminate this redundancy via aggregation
- Treat relationship as an abstract entity
- Allows relationships between relationships
- Abstraction of relationship into new entity
- Without introducing redundancy, the following diagram represents:
- An employee works on a particular job at a particular branch
- An employee, branch, job combination may have an associated manager

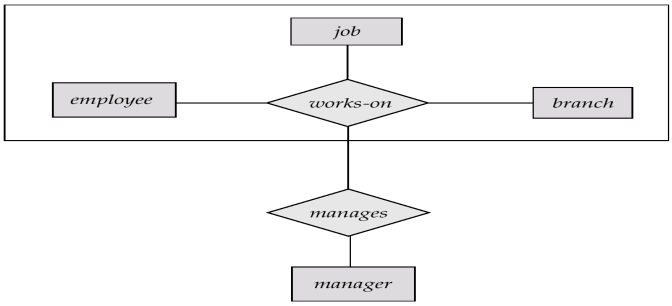



Fig. 2.9 Aggregation

## *Let's check the take away from this lecture*

1) **Specialization is _____**
   a) Top-Down approach  c) Both
   **b) Bottom-Up approach**  d) None of the above

2) **Teachers, Lab assistants, Admin People are all Employees of Organization. This is example of**
   a) Specialization  **c) Generalization**
   b) Both a) & b)  d) None of the above

Exercise:

Q.1 Explain Generalization of ER model with suitable example.

Questions/problems for practice:

Q.2 Explain aggregation of ER model with suitable example.

**Learning from the lecture:** Learners will be able to know different types of Extended E-R model.

# Lecture : 6

### Mapping of ER and EER to Relational Model

**Learning objective:** In this lecture learners will be able to know process of converting ER and EER to relational model.

- Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has several columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.
- A strong entity set reduces to a table with the same attributes.

| *customer-id* | *customer-name* | *customer-street* | *customer-city* |
|---|---|---|---|
| 019-28-3746 | Smith | North | Rye |
| 182-73-6091 | Turner | Putnam | Stamford |
| 192-83-7465 | Johnson | Alma | Palo Alto |
| 244-66-8800 | Curry | North | Rye |
| 321-12-3123 | Jones | Main | Harrison |
| 335-57-7991 | Adams | Spring | Pittsfield |
| 336-66-9999 | Lindsay | Park | Pittsfield |
| 677-89-9011 | Hayes | Main | Harrison |
| 963-96-3963 | Williams | Nassau | Princeton |

**Composite and Multivalued Attributes**

- Composite attributes are flattened out by creating a separate attribute for each component attribute.
- E.g. given entity set customer with composite attribute name with component first-name and last-name, the table corresponding to the entity set has two attributes, first- name and last- name
- A multivalued attribute M of an entity E is represented by a separate table EM
- Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multi valued attribute M
- E.g. Multi valued attribute dependent-names of employee is represented by a table

  employee-dependent-names( employee-id, dname)
- Each value of the multi valued attribute maps to a separate row of the table EM
- E.g., an employee entity with primary key John and dependents Johnson and Johndotir maps to two row: (John, Johnson) and (John, Johndotir)

**We can classify attributes along these dimensions:**

- simple/atomic vs. composite
- single-valued vs. multi-valued (or set-valued)
- A composite attribute is one that is composed of smaller parts. An atomic attribute is indivisible.

- Example 1: A BirthDate attribute can be viewed as being composed of attributes month, day, and year (each of which would probably be viewed as being atomic).
- Example 2: An Address attribute can be viewed as being composed of (sub-)attributes for street address, city, state, and zip code. A street address can itself be viewed as being composed of a number, street name, and apartment number.
- Single vs. multi-valued attribute: Consider a PERSON entity. The person it represents has (one) SSN, (one) date of birth, (one, although composite) name, etc. But that person may have zero or more academic degrees, dependents. How can we model this via attributes Academic Degrees, Dependents, and Spouses? One way is to allow such attributes to be multi-valued, which is to say that we assign to them a (possibly empty) set of values rather than a single value.

**Representing Weak Entity Sets:**

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

Table 2.2 Payment table includes loan-number column of identifying strong entity set Loan

| *loan-number* | *payment-number* | *payment-date* | *payment-amount* |
|---|---|---|---|
| L-11 | 53 | 7 June 2001 | 125 |
| L-14 | 69 | 28 May 2001 | 500 |
| L-15 | 22 | 23 May 2001 | 300 |
| L-16 | 58 | 18 June 2001 | 135 |
| L-17 | 5 | 10 May 2001 | 50 |
| L-17 | 6 | 7 June 2001 | 50 |
| L-17 | 7 | 17 June 2001 | 100 |
| L-23 | 11 | 17 May 2001 | 75 |
| L-93 | 103 | 3 June 2001 | 900 |
| L-93 | 104 | 13 June 2001 | 200 |

**Representing Relationship Sets as Tables:**

- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set borrower having columns from Customer and Loan entity

| *customer-id* | *loan-number* |
|---|---|
| 019-28-3746 | L-11 |
| 019-28-3746 | L-23 |
| 244-66-8800 | L-93 |
| 321-12-3123 | L-17 |
| 335-57-7991 | L-16 |
| 555-55-5555 | L-14 |
| 677-89-9011 | L-15 |
| 963-96-3963 | L-17 |

**Redundancy of Tables:**

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the other side.
- E.g.: Instead of creating a table for relationship account-branch, add an attribute branch_name to the entity set account

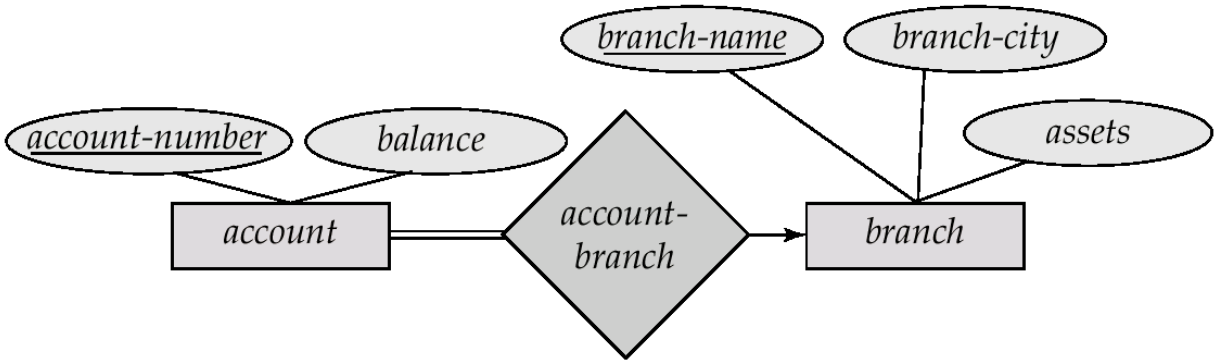



Fig. 2.10 account-branch relationship

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
- That is, extra attribute can be added to either of the tables corresponding to the two entity sets .
- If participation is partial on the many side, replacing a table by an extra attribute in the relation corresponding to the "many" side could result in null values
- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- E.g. The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-number).

**Representing Specialization as Tables**

**Method 1:**

- Form a table for the higher level entity
- Form a table for each lower level entity set, include primary key of higher level entity set and local attributes

| table | table attributes |
|---|---|
| person | name, street, city |
| customer | name, credit-rating |
| employee | name, salary |

- Drawback: getting information about, e.g., employee requires accessing two tables

**Method 2:**

- Form a table for each entity set with all local and inherited attributes

| table | table attributes |
|---|---|
| person | name, street, city |
| customer | name, street, city, credit-rating |
| employee | name, street, city, salary |

- If specialization is total, table for generalized entity (person) not required storing information can be defined as a "view" relation containing union of specialization tables. But explicit table may still be needed for foreign key constraints.
- Drawback: street and city may be stored redundantly for persons who are both customers and employees.

**Relations Corresponding to Aggregation:**

- To represent aggregation, create a table containing primary key of the aggregated relationship,the primary key of the associated entity set, Any descriptive attributes
- E.g. to represent aggregation manages between relationship works-on and entity set manager, create a table manages(employee-id, branch-name, title, manager-name)
- Table works-on is redundant provided we are willing to store null values for attribute manager-name in table manages

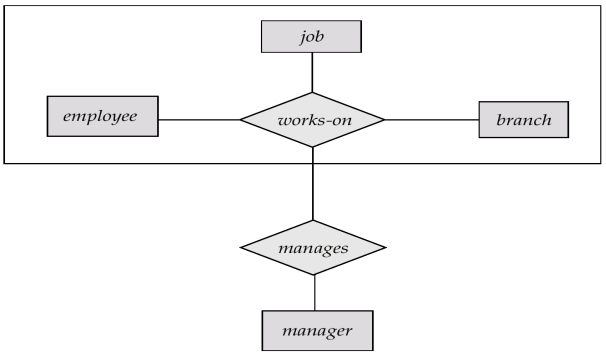



Fig 2.11 Aggregation

## *Let's check the take away from this lecture*

1) **Which type of entity set is mapped as it is in form of table?**

a) Weak Entity set
b) All types of Entity Sets
c) Strong Entity set
d) none of the above

2) **Relation type is checked while mapping relation to table to avoid**

a) Complexity
b) Redundancy of table
c) Space Complexity
d) All of the above

Exercise:

Q.1 Draw an ER diagram for a Hospital with a set of patients and a set of medical doctors. Associate with each patient a log of various tests and examination conducted. Also, convert it to a relational model.

Learning from the lecture: In this lecture learners will be able to convert ER model to relational model.

**Conclusion**

The study of E-R model shows abstract design of a database. With help of E-R diagram, Relational model can be designed which is further used to store data.

**Short Answer Questions:**

1. What is entity?

Ans) It is a thing of real life which is distinguishable from each other.

2. What is domain?

Ans) It is allowed set of candidates in that particular attribute.

3. What is cardinality?

Ans) Number of rows present in table/relation is called as cardinality.

4. What is key?

Ans) A unique attribute of table.

5. What is strong entity?

Ans) Entity having primary key is known as strong entity.

**Long Answer Questions:**

1. Construct an E-R diagram for a car-insurance company that has a set of customers, each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents and convert any weak entity set to a strong entity set by simply adding appropriate attributes.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7.

2. Create E-R diagrams for following description:

   Each company operates four departments, and each department belongs to one company. Each department employs one or more employees and each employee Works for one department. Each employee may or may not have one or more Dependents and each dependent belong to one employee. Each employee may or may not have an Employment history.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7.

3. What are the notations used in E-R diagram? Explain with suitable example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7.

4. What are the different types of attributes? Explain with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7

5. Draw E-R diagram for banking enterprise.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7

6. What do you mean by E-R Diagram? Draw an E-R Diagram for University.

   Database consisting of four entities: Student, Department, Class and faculty. Student has a unique id; student can enroll for multiple classes. Faculty must belong to department and

faculty can teach multiple classes. Each class is taught by only one faculty. Every student will get grade for the class he/she has enrolled.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7

7. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors.

   Associate with each patient a log of the various tests and examination conducted.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7

8. Consider a database used to record the marks that students get in different exams of different course offerings. Construct an E-R diagram that models exams as entities, and uses a ternary Relationship , for the above database.

   Construct an alternative E-R diagram that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7.

9. Design an E-R diagram for keeping track of the exploits of your favorite sports team. You should store the matches played, the scores in each match,the players in each match, and individual player statistics for each match.Summary statistics should be modeled as derived attributes.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 7.

**Set of Questions for FA/CE/IA/ESE**

Q.1) Explain the difference between a weak and a strong entity set.

Q.2) Illustrate different attributes with examples.

Q.3) Describe steps in converting ER model to Relational model.

Q. 4) Draw ER diagram of medical shop management. The medical shop should maintain the data about drugs, vendors, customers.

Q.5) Explain generalization of ER model with suitable example.

Q.6) Design a generalization–specialization hierarchy for a motor vehicle sales company. The company sells motorcycles, passenger cars, vans, and buses.Justify your placement of attributes at each level of the hierarchy. Explain why they should not be placed at a higher or lower level.

References:

1) Database System Concepts by Korth, Silberschatz, Sudarshan

**Practice for Module-02**

Q.1) Construct ER model for customer branch and account relationship.

Q.2) Explain aggregation with example.

Q.3) List various symbols used to represent ER model.

Q.4) Differentiate between different types of keys.

Q.5) Show how candidate key and primary key are related?

**Self-assessment**

Q.1) We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?

Q. 2) Explain the distinctions among the terms primary key, candidate key, and superkey.

**Self-evaluation**

| | | |
|---|---|---|
| Name of Student | | |
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick<br>Your choice |
| 6. | Do you understand the various symbols used to represent ER diagram? | o Yes<br>o No |
| 7. | Do you understand the attribute and its types? | o Yes<br>o No |

| | | |
|---|---|---|
| 8. | Do you know the steps to convert ER model to relational model? | ○ Yes<br>○ No |
| 9. | Do you understand different types of keys? | ○ Yes<br>○ No |
| 10. | Do you understand module ? | ○ Yes, Completely.<br>○ Partialy.<br>○ No, Not at all. |

# Module: 03

# Introduction to Structured Query Language (SQL)

## Lecture: 1

### Motivation:

Concept of SQL gives ability to perform operations on database. As database has advantage over file processing system, SQL provides better way to manage the data. SQL has different clauses that provides way to ensure security, transaction control over database.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Overview of SQL | 1 | 1 |
| 2 | Data Definition Language Commands | 1 | 2 |
| 3 | Data Manipulation Language Commands | 1 | 2 |
| 4 | Data Control Language Commands | 1 | 1 |
| 5 | Transaction Control Language Commands | 1 | 1 |
| 6 | Constraints | 1 | 1 |
| 7 | Set and String Operations | 1 | 2 |
| 8 | Aggregate Functions | 1 | 2 |
| 9 | Group by and Having Clause | 1 | 2 |

### Learning Objective:

- Learner will be able to write SQL queries.
- Learners will be able to design queries for data manipulation.
- Learners will be able to implement queries using various functions.
- Learners will be able to write queries to provide security to database objects.

### Theoretical Background:

One must have a very good knowledge of SQL to deal with database. Database provides a way to store, manipulate and control the access of the data. Many SQL constructs allows to do this with database. User can access required data without searching full database with the help of SQL queries.

## Key Definitions:

1. **SQL** is structured query language.
2. **DDL** is data definition language that creates the tables of relational model.
3. **DML** is data manipulation language that manipulates data of database.
4. **DCL** is data control language that controls access mechanism of data.
5. **TCL** is transaction control language that controls transaction states.

## Course Content

### Overview of SQL

SQL is a structured query language. Query language is nothing but language used to deal with database.

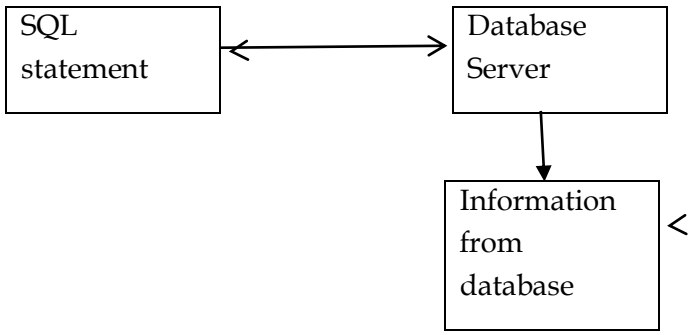



Fig. 3.1 Communication with Database Server using SQL

Following are the different SQL statements by which one can communicate with database.

1. Data Manipulation Language (DML): Fetches data from database. In short, manipulates contents of database.

SELECT

INSERT

UPDATE

DELETE

2. Data Definition Language (DDL): Deals with structure of database.

CREATE

ALTER

DROP

RENAME

TRUNCATE

3. Data Control Language (DCL): Deals with access rights of database.

GRANT

REVOKE

4. Transaction Control Language (TCL): Deals with changes made by DML statements.

COMMIT

ROLLBACK

Following are several SQL databases:

1. MySQL
2. Oracle
3. MSSQL
4. PostgresSQL
5. MariaDB

**Applications of SQL**

As mentioned before, SQL is one of the most widely used query language over the databases. I'm going to list few of them here:

- 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 Databases**

- **MS SQL Server**

  MS SQL Server is a Relational Database Management System developed by Microsoft Inc. Its primary query languages are −

  - T-SQL
  - ANSI SQL

- **MySQL**

  MySQL comes with a very fast, multi-threaded, multi-user and robust SQL database server.

- **ORACLE**

  It is a very large multi-user based database management system. Oracle is a relational database management system developed by 'Oracle Corporation'.

- **MS ACCESS**

  This is one of the most popular Microsoft products. Microsoft Access is an entry-level database management software. MS Access database is not only inexpensive but also a powerful database for small-scale projects.

## *Let's check the take away from this lecture*

1) **SQL is**

**a) Query language** b) programming language

c) machine language d) Oracle language

2) **Which of the following is not SQL statement?**

a) DDL c) DML

b) DCL **d) Oracle**

Exercise

Q.1 List different SQL statements.

Q.2 What is difference between DCL and TCL?

Questions/Problems for practice:

Q.2 How the SQL statements are executed?

**Learning from this lecture**: Learners will be able to understand SQL basics.

# Lecture: 2

## Data Definition Language Commands

**Learning objective:** In this lecture learners will be able to understand DDL commands.

Following are the DDL commands.

CREATE: creates structure of table.

ALTER: changes structure of table

DROP: removes data, structure of table from database

RENAME: renames table name

TRUNCATE: removes data of table but structure remains as it is.

**CREATE Command:**

In this command, specify the name of table, column names and their data type with size.

Syntax:

```
CREATE TABLE table_name
( column_name data_type);
```

Eg.

```
create table student
( name varchar2(10),
  Id number(2));
```

Above syntax creates student table with two columns, name and Id. Varchar2 and number are datatypes. Number in bracket indicates size of datatype. This syntax creates empty table.

You can check created table by using following command:

```
describe student;
```

Above command will show structure of created database.

Following are the different available datatypes:

1. Varchar2(size): variable-length character data

2. Char(size): fixed-length character data

3. Number(p,s): variable-length numeric data

4. Date: date and time values

5. LONG: variable-length character data upto 2GB

**Alter Command**

It is used to

1. Add new column

2. Modify existing column

3. Drop column

**Add new column**

Syntax: ALTER TABLE *table_name*
ADD *column_name datatype*;

Example: alter table student

add (department varchar2(10));

**Modify existing column**

Syntax: ALTER TABLE *table_name*
ALTER COLUMN *column_name datatype*;

Example: alter table student

ALTER COLUMN name varchar(30);

**Drop column**

Syntax: ALTER TABLE *table_name*
DROP COLUMN *column_name*;

Example: ALTER TABLE Customers
DROP COLUMN Email;

*Let's check the take away from this lecture*

1) **Which of following command keeps structure of table?**

a) Drop **b) Truncate**

c) Create d) All of the above.

2) **Which of following is efficient in terms of space?**

a) char **c) varchar2**

b) varchar d) None of the above

Exercise

Q.1 Explain different DDL commands.

Q.2 What are different datatypes of SQL?

Questions/Problems for practice:

Q.3 Write down basic syntax of an ALTER TABLE command to change the DATA TYPE of a column in a table

**Learning from this lecture:** Learners will be able to write DDL commands.

# Lecture: 3

## Data Manipulation Language Commands

**Learning objective**: In this lecture learners will be able to understand DML commands to manipulates database.

- SELECT
- INSERT
- UPDATE
- DELETE

**SELECT clause:**

Table 3.1 Student

| **Name** | **Id** | **Address** | **Marks** |
|---|---|---|---|
| Sam | 1 | Mumbai | 50 |
| Rohit | 2 | Pune | 60 |
| Harish | 3 | Sangli | 70 |

1. To select all the data from table:

```
select *
from student;
OR
Select name, id, address
From student
```

2. To select whole column:

```
Select name
From student;
```

3. To select data with condition:

```
Select name,address
From student
Where name='Sam';
```

4. To select data with comparison conditions:

```
Select name,address
From student
Where id>=1;
```

5.As Clause

AS is used to assign temporarily a new name to a table column.

```
Select address as ad
From student
```

6.Distinct

The SQL DISTINCT command is used with SELECT key word to retrieve only distinct or unique data.

```
Select distinct address
From student
```

**Comparison conditions**: >, <, <>, >=, <=, between, in, like, is null

```
Select name, marks
From student
Where marks between 70 and 90;

Select name,address
From student
Where id in (1,2)

Select name,address
From student
Where name like 'S%';
```

**Like condition:** It is pattern matching format.

_ : represents exactly 1 character

%: 0 or more character

**Logical conditions: AND, OR, NOT**

```
Select name,address
From student
Where id>=1
And name like '%E%';

Select name,address
From student
Where id not in (1,2);
```

**INSERT Clause:**

Syntax:

```
INSERT INTO table_name
VALUES (value1, value2, value3....);
```

Example:

```
Insert into student
Values ( 'Pankaj', 5, 'Satara');
```

**UPDATE Clause:**

Syntax:

```
UPDATE table_name
SET column_name = expression
```

WHERE conditions

Example: Update student
Set id=4
Where name='Pankaj';

Update student
Set id=4, Address='Pune'
Where name='Pankaj';

**DELETE Clause:**

Syntax: DELETE FROM table_name [WHERE condition];

Example: Delete from student;

Delete from student
Where id='3';

*Let's check the take away from this lecture*

1) *** in select statement indicates** _____

a) **all columns of table**

b) all rows of table

c) all data of table

d) None of the above

2) **< > is** _____ **operator.**

a) **not equal to** c) less than
b) greater than d) invalid

3) The language used by application programs to request data from the DBMS is referred to as _____

a. DML
b. DDL

c**. Query language**
d. All of the Mentioned

Exercise
Q.1 Explain DML commands with example.
Q.2 Write Update syntax and explain with example.
Q.3 Explain difference between delete and drop.
Questions/Problems for practice:
Q.1 Explain like operator with example.

**Learning from this lecture:** Learners will be able to write DML commands to manipulate data.

# Lecture: 4

## Data Control Language Commands

**Learning objective:** In this lecture learners will be able to apply DCL commands to provide access rights to user.

**Following are DCL commands.**

GRANT

REVOKE

Database creates user with CREATE USER statement.

Create user name

Identified by password;

Eg: create user admin

identified by admin123;

After creating user, DBA gives privileges to user.

Grant priviliges

To user;

Eg. Grant craete table, create view

To admin;

Following are some system privileges: create table, create session, create view, create procedure, create sequence

**Role:** It is group of related privileges. It becomes easier to grant and revoke privileges in bulk.

Crate role HOD;

Grant create table, create view

to HOD;

Grant HOD to admin, clerk;

Now, admin, clerk have create table, create view privileges.

**Object Privileges:**

Grant select

On student

To emp1;

Grant update (marks)

On student

To emp1;

**With grant option:** This option allows to pass on privileges.

Grant select

On student

To emp1

With grant option;

Above example shows that emp1 can pass select privilege to others.

**Revoke statement:**

Revoke select

On student

From emp1;

Above example shows that select privilege is revoked from emp1.

## *Let's check the take away from this lecture*

**1) To pass on the privileges to other user, _____ is used.**
a) Grant c**) With grant option**
b) Revoke d) All of the above

2) ___ **is not object privilege.**
a) select c) update
b) create **d) create user**

Exercise:

Q.1 Explain DML commands with example.
Q.2 What is role? What is its use in DCL?
Questions/problems for practice:
Q. 3 Illustrate difference between GRANT and REVOKE command.

**Learning from this lecture:** Learners will be able to understand the mechanism of grant and revoke privileges.

# Lecture: 5

### Transaction Control Language Commands

**Learning objective:** In this lecture, learners will be able to understand use of TCL commands to control transactions.

**COMMIT**

1. Data changes are reflected in database.
2. Earlier state of data will be no longer available.
3. Automatic commit occurs after DDL and DCL statements are executed.

```
Delete from student
Where id =3;

Commit;
```

Above query deletes row from table and makes it permanent by issuing commit statement.

**ROLLBACK**

1. Database changes are undone.
2. Data is restored to previous state.

```
Delete from student;

ROLLBACK;

Delete from student
Where id=3;

Select name
From student
Where id=3;

Commit;
```

Above query will return zero rows.

## *Let's check the take away from this lecture*

**1) Automatic commit occurs in _____**

a) DML **b) DDL** c) TCL d) All of the above

**2) A transaction completes its execution is said to be**

a) **Committed** b) Aborted
b) Failed d) Rolled Back

Exercise:

Q.1 Explain TCL commands.

Questions/problems for practice:

Q.2 Illustrate ROLLBACK command with example.

Learning from the lecture: Learners will be able to know about controlling transactions.

# Lecture: 6

## Constraints

**Learning objective:** In this lecture learners will be able to apply conditions on database.

Constraints are used to apply rules at table level.

Different Types of Constraints:

1. **NOT NULL**: Does not allow to have null value of column.

Eg. Roll no

2. **UNIQUE**: every value must be unique that is not same

Eg. Two persons cant have same email_id

3. **PRIMARY KEY**: unique identifier of table

Eg. Roll_No

4. **FOREIGN KEY**: refernces primary key of another table

Eg. Student(id,name,departmet_id)

Departments(department_id,dept_name)

Here, department_id of Student table is foreign key, as it refers primary key of Department table.

**5. CHECK:** checks for particular stated condition

Eg. Salary should not be less than 40000.

```
create table employee
( emp_id number(4),
Constraint emp_cons primary key,
Name varchar2(10),
  Job_id  number(2)  NOT NULL,
  Email varchar2(25)
        Constraint email_cons unique)
```

Above query creates table having emp_id as primary key. It will not allow job_id to be null.

Also, no two tuples will have same email_id.

emp_cons and email_cons are names given to constraints. These constraints are saved in database by these names.

```
create table employee
( emp_id number(4),
Constraint emp_cons primary key,
Name varchar2(10),
 Job_id  number(2)  NOT NULL,
 Email varchar2(25)
        Constraint email_cons unique
Department_id number(2),
        Constraint emp_dept FOREIGN KEY (Department_id) REFERENCES
        Departments(Department_id));
```

Above query creates table having department_id as reference key. You can enter department_id in employee table which does not exist in Departments table. In Departments table, Department_id is primary key.

```
create table employee
( emp_id number(4),
 Salary number(10)
        Constraint sal_cons CHECK(salary>40000));
```

Above query creates table having condition for salary attribute. While inserting data into this table, database will not allow you to enter salary value which is less than 40000.

## *Let's check the take away from this lecture*

1) **Constraints are like**
a**) Condition to data** c) DML statement
b) DDL statement d) none of the above

2) **Which of the following constraint does not enforce uniqueness?**

a) Unique b) Foreign Key
c) Primary Key d) **None of the above**

**3) Constraints can be applied on**
a) column b) table
c) field **d) all of the above**

Exercise:

Q.1 Explain difference between Unique and Primary key with the help of a suitable example.
Q.2 Explain Constraints with example.

Questions/problems for practice:
Q. 3 Create table Department having all the constraints included in it. Use Foreign key with help of another table.

**Learning from the lecture:** In this lecture learners will be able to understand and apply constraints on relation.

# Lecture: 7

## Set and String Operations

**Learning objective:** In this lecture learners will able to use set operations on tables and string operations on characters.

### Set Operations

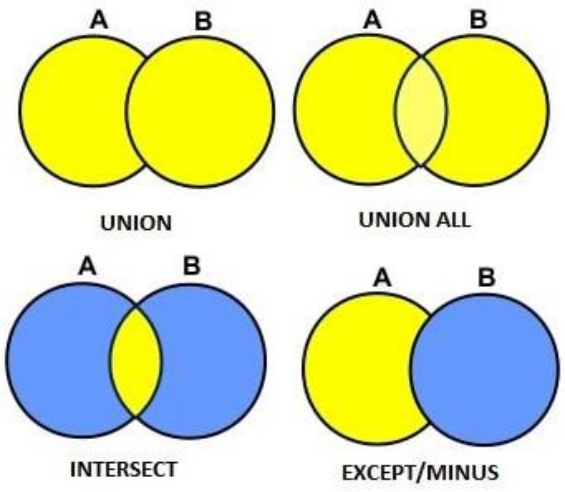



Fig. 3.2 Set Operations

**Union Operator:**

1. Returns result from both queries without duplicates.
2. The number of columns and their data types should be same.
3. UNION ALL operator returns result from both queries including duplicates.

Eg.
```
Select emp_id, job_id
From employee
        UNION
Select emp_id, job_id
From departments;
```

**Intersect Operator:**

```
Select emp_id, job_id
From employee
        INTERSECT
Select emp_id, job_id
From departments;
```

Above query returns emp_id, job_id values that are present in both tables.

**Minus Operator:**

```
Select emp_id
From employee
   MINUS
Select emp_id
From history;
```

Above query returns emp_id from employee table which are not present in history table.

**String Operations:**

1. **Case Manipulation Functions:** LOWER, UPPER, INITCAP
2. **Character Manipulation Functions:** CONCAT, SUBSTR, LENGTH, INSTR, LPAD/RPAD, TRIM, REPLACE

LOWER: converts char to lowercase
UPPER: converts char to uppercase
INITCAP: converts char to uppercase for each word

lower('DBMS Course') --> dbms course
upper('DBMS Course') --> DBMS COURSE
initcap('DBMS Course') --> Dbms Course

Concat('Hi', 'Students') --> HiStudents
Substr('HelloWorld',1,5) --> Hello
Length('Hi')-->2
Instr('HelloWorld','e')-->2
Lpad(salary,10,'*') --> ****240000
Rpad(salary,10,'*') --> 240000****
Replace('Jack and Jue','J','Bl') --> Black and Blue
Trim( 'H' from 'Hello') --> ello

## *Let's check the take away from this lecture*

1) **Duplicate tuples must be eliminated when the query uses the _____**
a) select clause
b) where clause
**c) select distinct**
d) from distinct

2) **The string function that returns the index of the first occurrence of substring is _____**
a) INSERT()
b) INSTR()
c) INSTRING()
d) INFSTR()

Exercise:

Q.1 Explain the string operations with suitable example.
Q.2 Explain set operators with help of exam

Questions/problems for practice:
Q. 3 Explain difference between UNION and UNION ALL operators.

**Learning from the above lecture:** Learners will be able to know various types of set operators and string functions.

# Lecture: 8

### Aggregate Functions

**Learning objective:** In this lecture learners will be able to apply aggregate functions on group of data.

1. Aggregate functions work on set of rows.
2. It gives one result per group.

3. Types of group functions: avg, count,max,min, stddev, sum,variance
4. Order by clause arranges output of query either in ascending order or descending order.

**Avg**: It averages all values of column. It ignores null values

**Count**: Results sum of specified column value.

**Max**: Gives maximum value among specified column.

**Min**: Gives minimum value among specified column.

**Stddev**: Gives standard deviation of specified value.

**Sum**: gives sum value of specified column values.

**Variance**: Gives variance of specified value.

Syntax:

```
Select column_name, group_function
From table_name
Where condition
Group by column
Order by column;
```

Eg.
```
Select name, avg(salary), min(salary),max(salary)
From employee
Where id=1;

Select count(*)
From employee;

Select student_id, avg(marks)
From students
```

Group by student_id;

Select avg(salary)

From departments

Group by department_id;

Column name which is present in select statement without aggregate function should be present in group by clause.

In above query, student_id is present in select clause with aggregate function average of marks.

So, it should be present in group by clause, otherwise query will give error.

## *Let's check the take away from this lecture*

1) **Aggregate function can be used in from clause of SQL**
   **a)** True **b) False**

**2) Which of the following is not a built in aggregate function in SQL?**
a)avg
b)max
**c)total**
d) count

Exercise:

Q.1 Explain different aggregate functions in detail.

Questions/problems for practice:

Q. 2 What will be output of following query: Select count(emp_name) From employee

**Learning from the lecture:** Learners will be able to group data of tables and can perform operations on groups.

# Lecture: 9

### Group by and Having Clause

**Learning objective:** In this lecture learners will be able to apply conditions on aggregate groups.

Group by clause with having clause:

1. Rows are grouped.
2. Aggregate function is applied.
3. Having condition is applied on groups.

```
Select column_name, group_function
From table_name
Where condition
Group by column
Having condition
Order by column;
```

In above syntax, you can see that condition is applied at where clause and having clause. Where clause applies condition before forming groups of data, i.e. to all rows of table. Excluded rows are not included in forming groups of data.

Having clause applies condition on groups of data. Groups that does not satisfy condition mentioned in having clause, are excluded from group.

Eg.

```
Select dept_id, max(salary)
From employee
Where dept_id<>1
Group by dept_id
Having max(salary) > 40000;
```

Above query excludes dept_id 1 from groups. Then groups are formed dept wise. On these groups, condition of having clause is applied. If you want result of above query in ascending order:

```
Select dept_id, max(salary)
From employee
Where dept_id<>1
Group by dept_id
Having max(salary) > 40000;
```

Order by dept_id;

Order by clause always comes as last clause of the query.

**Nesting of group functions**

Select max(avg(salary))

From departments

Group by department_id;

*Let's check the take away from this lecture*

1) **Where and Having clause are interchangeable.**

a) True **b) False**

2) **Having clause applies condition on all rows of table.**

a) True **c) False**

Exercise

Q.1 Explain use of group by clause.

Q.2 Explain use of order by clause.

Questions/Problems for practice:

Q.3 Illustrate conditions of when to use Having and when to use Where condition.

**Learning from this lecture:** Learners will be able to apply group functions with conditions on groups.

**Conclusion**

The study of SQL shows manipulation, creation of database. It also helps users to control transactions of database. One can give security permissions based on commands to various users. Different clauses of SQL helps to access the database in efficient way.

**Short Answer Questions:**

1. What is SQL?

Ans) SQL (Structured Query Language) is a query language used to access te database.

2.List DML commands.

Ans) SELECT, INSERT, UPDATE, DELETE

3.List DDL commands.

Ans) CREATE, ALTER, DROP, RENAME

4.List DCL commands.

Ans) GRANT, REVOKE

5.List TCL commands.

Ans) COMMIT, ROLLBACK

**Long Answer Questions:**

1. Consider following Insurance database.

person (driver id, name, address)

car (license, model, year)

accident (report number, date, location)

owns (driver id, license)

participated (report number, license, driver id, damage amount)

Construct the following SQL queries for this relational database.

a. Find the total number of people who owned cars that were involved in accidents in 2009.

b. Add a new accident to the database; assume any values for required attributes.

c. Delete the Mazda belonging to "John Smith".

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 3

2. Write the following inserts, deletes or updates in SQL, using the university schema.

a. Increase the salary of each instructor in the Comp. Sci. department by 10%.

b. Delete all courses that have never been offered (that is, do not occur in the section relation).

c. Insert every student whose tot cred attribute is greater than 100 as an instructor in the same department, with a salary of $10,000.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 3

3) Write the following queries in SQL, using the university schema.

a. Find the titles of courses in the Comp. Sci. department that have 3

credits.

b. Find the IDs of all students who were taught by an instructor named

Einstein; make sure there are no duplicates in the result.

c. Find the highest salary of any instructor.

d. Find all instructors earning the highest salary (there may be more

than one with the same salary).

e. Find the enrollment of each section that was offered in Autumn 2009.

f. Find the maximum enrollment, across all sections, in Autumn 2009.

g. Find the sections that had the maximum enrollment in Autumn 2009.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 3

4) The SQL like operator is case sensitive, but the lower() function on strings can be used to perform case insensitive matching. To show how, write a query that fifinds departments whose names contain the string "sci" as a substring, regardless of the case.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 3

**Set of Questions for FA/CE/IA/ESE**

Q. 1) Create table emp with following attributes.

Emp_no, emp_name, job,mgr, joindate, salary, comm, dept_no. Use character datatype for job and number datatype for mgr.

a. Display the list of employees whose salary is in between 20000 and 30000

Q. 2) Write the following queries in SQL, using the university schema.

a. Find the names of all students who have taken at least one Comp. Sci. course; make sure there are no duplicate names in the result.

b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

c. For each department, fifind the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

d. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.

Q. 3) Write SQL DDL corresponding to the schema in Q.1 (Long answer questions). Make any reasonable assumptions about data types, and be sure to declare primary and foreign keys.

Q. 4) List two reasons why null values might be introduced into the database.

Q.5) Show that, in SQL, <> all is identical to not in.

Q.6) Explain datatypes available in SQL.

Q.7) Explain ALTER command with example.

Q.8) Write query to create table with primary key and foreign key.

Q.9) List and write SQL queries for illustrating arithmetic operators.

Q.10) Illustrate pattern matching operator with example.

Q. 11) Compare where and having clause.

Q. 12) Consider the employee database given below. Give an expression on SQL for each of the following queries.

Employee data base :

Employee (emp_name, street, city)

Works (emp_name, company_name, salary)

Company (Company_name, city)

Manager (Company_manager_name)

I) Modify the database so that "Sachin" now Lives in Maharashtra.)

II) Give all employee of "Transtrack Software Solution Pvt. Ltd". a 10% raise.

III) Delete all tuples in the works relation for employee of "ÄBC Pvt Ltd".

IV) Display company name and city from company relation.

Q. 13) Consider following database and solve queries .

emp (empno, ename, ph, sal, dept_no, comm)

i) Change employee name 'Rahul' to 'Ramesh'.

ii) Give increment of 20% in salary to all employees.

Q. 14) Consider following schema :

Depositor (cust_name,acc_no)

Borrower (cust_name, loan_no)

Solve following queries :

i) Find customer name having saving account as well as loan account.

ii) Find customer names having loan account but not the saving account.

References:

1) Database System Concepts By Korth, Sudarshan, Silberchatz

**Practice for Module-03**

Q.1) List following commands: DML, DDL, DCL, TCL (4 Marks)

Q.2) Write create table syntax including following constraints: primary key, unique, not null, foreign key, unique (5 Marks)

Q.3) Write query to illustrate group by and having clause. (5 Marks)

Q.4) Illustrate set operations with example. (5 Marks)

Q.5) List and explain SQL select capabilities. (5 Marks)

**Self-assessment**

Q.1) Explain DDL, DML, TCL and DCL commands with example.

Q.2) Write all possible clauses of SQL with proper sequence.

Q.3) Describe constraints with example.

**Self-evaluation**

| Name of Student | | |
|---|---|---|
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick<br>Your choice |
| 11. | Do you understand the various types of commands? | o Yes<br>o No |
| 12. | Do you understand the basic constructs of SQL? | o Yes<br>o No |
| 13. | Do you know the various types of group functions? | o Yes<br>o No |
| 14. | Do you understand how constraints are applied? | o Yes<br>o No |
| 15. | Do you understand module ? | o Yes, Completely.<br>o Partialy.<br>o No, Not at all. |

# Module: 04

# Advanced SQL with Integrity, Security and Authorization

## Lecture: 1

### Motivation:

The knowledge of advanced SQL is essential for software developer hence it is an important subject in computer programming around the world.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Nested Sub queries | 1 | 1 |
| 2 | Referential Integrity in SQL | 1 | 1 |
| 3,4 | Joins | 2 | 2 |
| 5 | View | 1 | 2 |
| 6 | Assertion | 1 | 2 |
| 7 | Trigger | 1 | 2 |
| 8 | Database Security and Authorization | 1 | 1 |
| 9 | Granting of Privileges, Revoking of Authorization in SQL | 1 | 1 |
| 10,11 | Fundamental Operations in Relational Algebra | 2 | 2 |

### Learning Objective:

- To learn advanced Structured Query Language commands
- To learn how data can be handled with the help of tools (e.g SQL Server or Oracle).
- To learn how to store and retrieve data.

### Theoretical Background:

**Access control:** SQL can be used to restrict a user's ability to retrieve, add, and modify data, protecting stored data against unauthorized access.

**Data sharing**: SQL is used to coordinate data sharing by concurrent users, ensuring that they do not interface with one another.

**SQL:** SQL is a tool for organizing, managing, and retrieving data stored by a computer database. SQL is a computer language that we use to interact with the database. In fact, SQL works with one specific type of database, called a relational data base.

**Relational** Database**:** A relational database is a database that can be perceived as a set of tables and manipulated in accordance with the relational model of data. It contains a set of objects used to store, manage, and access data. Examples of such objects are tables, views, indexes, functions, triggers, and packages

## Key Definitions:

**Data integrity:** SQL defines integrity constraints in the database, protecting in the database, protecting it from corruption due to inconsistent updates or system failures.

## Course Content

**Nested Sub queries**

The inner query (or subquery) returns a value that is used by the outer query.

Subquery Syntax:

```
Select selct_list
From table_name
Where operator    ( Select selct_list
                    From table_name )
```

In above syntax, operator includes a comparison condition:

Single row operators: >, <, = , <>,>=,<=

Multiple row operators: IN,ANY, ALL

**Set Membership**

Use the **in** and **not in** operators for set membership.

```
select distinct cname
from borrower
    where cname in
                (select cname
                 from account
                  where bname=``SFU'')
```

Note that we can write the same query several ways in SQL.

Finding all customers who have a loan but not an account, we can use the **not in** operation.

**Set Comparison**

To compare set elements in terms of inequalities, we can write

```
 select distinct T.bname
     from branch T,branch S
where T.assets > S.asset
and S.bcity=``Burnaby''
```

or we can write

```
select bname
        from branch
        where assets > some (select assets
```

```
                    from branch
                    where bcity=``Burnaby'')
```

To find branches whose assets are greater than some branch in Burnaby.

We can use any of the equality or inequality operators with **some**. If we change **> some** to **> all**, we find branches whose assets are greater than all branches in Burnaby.

Example. Find branches with the highest average balance. We cannot compose aggregate functions in SQL, e.g. we cannot do **max (avg ...))**. Instead, we find the branches for which average balance is greater than or equal to all average balances:

```
select bname
from account
group by bname
having avg (balance) >= all (select avg (balance)
                    from account
                    group by bname)
```

### 3. Test for Empty Relations

The **exists** construct returns **true** if the argument subquery is nonempty.

Find all customers who have a loan **and** an account at the bank.

```
select cname
    from borrower
where exists (select *
        from depositor
            where depositor.cname = borrower.cname)
```

### 4. Test for the Absence of Duplicate Tuples

The **unique** construct returns **true** if the argument subquery contains no duplicate tuples.

Find all customers who have only one account at the SFU branch.

```
select T.cname
from depositor as T
where unique (select R.cname
              from account, depositor as R
              where T.cname = R.cname and
              R.account = account.account and
              account.bname = ``SFU")
```

## *Let's check the take away from this lecture*

1) **Which of the following executes first?**

a) Subquery

b) **Outer Query**

2) **Which of the following is true about the result of a sub-query?**

a) The result of a sub-query is generally ignored when executed.
b) The result of a sub-query doesn't give a result, it is just helpful in speeding up the main query execution
**c) The result of a sub-query is used by the main query.**
d) The result of a sub-query is always NULL

3**) Which of the following clause is mandatorily used in a sub-query?**

**a) SELECT**
b) WHERE
c) ORDER BY
d) GROUP BY

Exercise

Q.1 Define with example nested queries for employee table.

Questions/Problems for practice:

Q.2 Write a query to display the employee last name, job id, and salary of all employees whose salary is equal to the minimum salary.

**Learning from this lecture:** Learners will be able to understand concepts of nested queries.

# Lecture: 2

## Referential Integrity in SQL

**Learning objective:** In this lecture learners will be able to understand integrity constraints used to maintain integrity of data.

There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation (referential integrity).

For example, the department listed for each course must be one that actually exists.

More precisely, the dept_name value in a course record must appear in the dept_name attribute of some record of the department relation.

Database modifications can cause violations of referential integrity. When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

Foreign keys can be specified as part of the SQL create table statement by using the foreign key clause.

```
create table department
(dept name varchar (20),
building varchar (15),
 budget numeric (12,2)  check (budget > 0),
   primary key (dept name))
```

```
create table course
(course id varchar (8),
title varchar (50),
dept name varchar (20),
```

credits numeric (2,0) check (credits > 0),

primary key (course id),

foreign key (dept name) references department)

**On Delete Cascade**

create table course

( ... foreign key (dept name) references department

on delete cascade );

Because of the clause on delete cascade associated with the foreign-key declaration, if a delete of a tuple in department results in this referential-integrity constraint being violated, the system does not reject the delete.

Instead, the delete "cascades" to the course relation, deleting the tuple that refers to the department that was deleted.

Similarly, the system does not reject an update to a field referenced by the constraint if it violates the constraint; instead, the system updates the field dept name in the referencing tuples in course to the new value as well.

*Let's check the take away from this lecture*

1) F**oreign key and Primary key can be of same table.**

a) True **b) False**

2**) Constraints that are applied on individual tuples and are verified whenever any tuple is modified or new tuple is inserted are called**

a. language based constraints
b. tuple based constraints
c. scale based constraints
d. precision based constraints

**3)To include integrity constraint in an existing relation use:**

a. Modify table
b. Drop table
**c. Alter table**
d. Create table

Exercise

Q.1 Discuss the use of on delete cascade clause with example.

Questions/Problems for practice:

Q.2 Give example of how referential integrity constraint is implemented in SQL.

Learning from this lecture: Learners will be able to apply concept of integrity constraints on data.

# Lecture: 3

## Joins

**Learning objective**: In this lecture learners will be able to understand Join concept to apply on one or more tables.

### Join types and conditions

Each variant of the join operations consists of a *join type* and a *join condition*.

Join types: inner join, left outer join, right outer join, full outer join.

The keyword inner and outer are optional since the rest of the join type enables us to deduce whether the join is an inner join or an outer join.

Join conditions: natural, on predicate, using (A1,A2,…….An)

The use of join condition is mandatory for outer joins, but is optional for inner joins (if it is omitted, a Cartesian product results).

**Natural Join**

1. It is based on all columns in the two tables that have same name.

2. It selects rows from the two tables that have equal values in all matched columns.

Consider the following schemas:

Deparments: department_id, department_name, manager_id,location_id

Locations: location_id, street_address, city

Employees: employee_id, manager_id, department_id,last_name, job_id,salary

```
Select department_id, department_name, location_id
From departments
NATURAL JOIN Locations;
```

**Using Clause**

1. Use this clause to match only one column.

```
Select  l.city, d.department_name
        From Locations l JOIN departments d
        USING (location_id)
        Where location_id=40;
```

**On Clause**

1. Use the ON clause to specify random conditions.
2. You can specify columns to join

```
Select e.employee_id, e.last_name,e.department_id,d.department_id,d.location_id
From employees e JOIN departments d
ON e.department_id = d.department_id
```

*Let's check the take away from this lecture*

1) **Table can be joined to itself.**

**a) True** b) False

2) **Which product is returned in a join query where there is no join condition:**

a) Equijoins
b) Cartesian
c) Both Equijoins and Cartesian
d) None of the mentioned

Exercise

Q.1 Define natural join with example.

Questions/Problems for practice:

Q.2 Write a query to describe self-join.

**Learning from this lecture:** Learners will be able to apply join conditions on tables.

# Lecture: 4

## Joins

**Learning objective**: In this lecture learners will be able to understand the Join concept to apply on one or more tables.

**OUTER Joins**

1. If a row does not satisfy a join condition, that row doesn't come as a result.

2. Such rows can be retrieved by using outer joins

3. It is like inner join plus unmatched columns.

**LEFT OUTER Joins**

Select e.last_name, e.department_id, d.department_name
From employees e **LEFT OUTER JOIN** departments d
**ON** (e.department_id = d.department_id);

This query retrieves all the rows that satisfy join condition as well as unmatched rows from left table employees.

**RIGHT OUTER Joins**

Select e.last_name, e.department_id, d.department_name
From employees e **RIGHT OUTER JOIN** departments d
**ON** (e.department_id = d.department_id);

This query retrieves all the rows that satisfy join condition as well as unmatched rows from right table departments.

**FULL OUTER Joins**

Select e.last_name, e.department_id, d.department_name
From employees e **FULL OUTER JOIN** departments d

**ON** (e.department_id = d.department_id);

This query retrieves all the rows that satisfy join condition as well as unmatched rows from right table departments as well as left table employees.

*Let's check the take away from this lecture*

1) **Outer Join gives result of equijoin also.**

**a) True** c) False

2)**Which are the join types in join condition:**
a) Cross join
b) Natural join
c) Join with USING clause
**d) All of the mentioned**

Exercise:

Q.1 List types of joins.

Q.2 Explain the use of table alias with example.

Questions/problems for practice:

Q. 2 Write a query to illustrate outer join.

**Learning from this lecture:** Learners will be able to apply join conditions on tables

# Lecture : 5

## View

**Learning objective:** In this lecture, learners will able to understand view concept.

### Views

A view in SQL is defined using the create view command:

**create view** *v* **as** query

Here, query is any legal query expression.

The view created is given the name v.

To create a view *all-customer* of all branches and their customers:

```
create view all-customer as    (select bname, cname
                                from depositor, account
                                where depositor.account = account.account)
                                        union
                               (select bname, cname
                               from borrower, loan
                               where borrower.loan = loan.loan)
```

Having defined a view, we can now use it to refer to the virtual relation it creates. View names can appear anywhere a relation name can.

We can now find all customers of the SFU branch by writing

```
                        select cname
                from all-customer
                where bname=``SFU'
```

**Update of a view**

Consider a clerk who needs to see all information in the *loan* relation except *amount*.

Let the view *branch-loan* be given to the clerk:

```
                create view branch-loan as   (select bname, loan
                                                from loan )
```

Since SQL allows a view name to appear anywhere a relation name may appear, the clerk can write:

```
                insert into branch-loan values (``SFU'', ``L-307'')
```

This insertion is represented by an insertion into the actual relation *loan*, from which the view is constructed. However, we have no value for *amount*.

This insertion results in (``*SFU*'', ``*L-307*'', null) being inserted into the *loan* relation.

When a view is defined in terms of several relations, serious problems can result. As a result, many SQL-based systems impose the constraint that a modification is permitted through a view only if the view in question is defined in terms of one relation in the database.

## *Let's check the take away from this lecture*

**1)Which operation are allowed in a join view:**
a) UPDATE
b) INSERT
c) DELETE
d**) All of the mentioned**

**2)Evaluate this SQL statement:**

SELECT e.EMPLOYEE_ID,e.LAST_NAME,e.DEPARTMENT_ID, d.DEPARTMENT_NAME
FROM EMP e, DEPARTMENT d WHERE e.DEPARTMENT_ID = d.DEPARTMENT_ID;

a) Selection, projection, join
b) Difference, projection, join
c) Selection, intersection, join
d) Intersection, projection, join
e) Difference, projection, product

Exercise:

Q.1 What is view?

Questions/problems for practice:

Q.2 Explain use of view with example.

**Learning from the lecture**: Learners will be able to understand view concept to work on part of table.

# Lecture: 6

### Assertion

**Learning objective:** In this lecture learners will be able to know assertion concepts.

An assertion is a predicate expressing a condition that we wish the database always to satisfy.

Domain constraints and referential-integrity constraints are special forms of assertions.

1. Domain constraints and referential-integrity constraints are special forms of assertions.

2. However, there are many constraints that we cannot express by using only these special forms.

Eg.

For each tuple in the student relation, the value of the attribute tot cred must equal the sum of credits of courses that the student has completed successfully

An assertion in SQL takes the form:

create assertion <assertion-name> check <predicate>;

```
create assertion credits earned constraint check
(not exists (select ID
      from student
       where tot_cred <> (select sum(credits)
    from takes natural join course
     where student.ID= takes.ID
      and grade is not null and grade<> 'F' )
```

*Let's check the take away from this lecture*

1) **Constraints are the same as assertions.**

a) True b) False

2) **Domain constraints, functional dependency and referential integrity are special forms of** _____

a. Foreign key
b. Primary key
c. Assertion
d. Referential constraint

Exercise:

Q.1 Explain assertion with suitable example.

Learning from the lecture: In this lecture learners will be able to know assertion concept.

# Lecture : 7

## Trigger

**Learning objective:** In this lecture learners will be able to understand trigger on database.

A trigger is a statement that the system executes automatically as a side effect of a modification to the database.

To design a trigger mechanism, we must meet two requirements:

1. Specify when a trigger is to be executed. This is broken up into an event that causes the trigger to be checked and a condition that must be satisfied for trigger execution to proceed.

2. Specify the actions to be taken when the trigger executes.

Once we enter a trigger into the database, the database system takes on the responsibility of executing it whenever the specified event occurs and the corresponding condition is satisfied.

```
create trigger timeslot_check1 after insert on section
referencing new row as nrow
for each row when (nrow.time slot id not in ( select time slot id
      from time slot)

    begin
rollback
    end;
```

1. Trigger is initiated after any insert on the relation section.

2. It ensures that the time slot id value being inserted is valid

3. An SQL insert statement could insert multiple tuples of the relation, and the **for each row** clause in the trigger code would then explicitly iterate over each inserted row.

4. The **referencing new row as** clause creates a variable nrow(called atransition variable) that stores the value of an inserted row after the insertion.

5. The **when** statement specifies a condition. The system executes the rest of the trigger body only for tuples that satisfy the condition.

6. Thus any transaction that violates the referential integrity constraint gets rolled back, ensuring the data in the database satisfies the constraint.

**Updates on table**

The trigger can specify attributes whose update causes the trigger to execute.

Updates to other attributes would not cause it to be executed.

For example : to specify that a trigger executes after an update to the grade attribute of the takes relation, we write:

after **update** of takes on grade

**referencing old row as** : used to create a variable storing the old value of an updated or deleted row.

**referencing new row as** : clause can be used with updates in addition to inserts.

create trigger credits earned after update of takes on (grade)

referencing new row as nrow

referencing old row as orow

for each row

when nrow.grade <> 'F' and nrow.grade is not null

and (orow.grade = 'F' or orow.grade is null)

begin atomic

update student

set tot cred= tot cred+ (select credits

```
from course

where course.course id = nrow.course id)

where student.id = nrow.id;

end;
```

**for each statement:**

```
create trigger setnull before update on takes

referencing new row as nrow

for each row

when (nrow.grade = ' ')

begin atomic

set nrow.grade = null;

end;
```

We can carry out a single action for the entire SQL statement that caused the insert, delete, or update.

## *Let's check the take away from this lecture*

1) **Triggers can't be used in**

a) View c) Insert

b) Update d) All of the above

Exercise:

Q.1 Write a query to illustrate row trigger and statement trigger.

Q.2 Explain the term trigger with example.

Questions/problems for practice:

Q. 3 Write a query to illustrate trigger for delete operation.

**Learning from the above lecture:** Learners will be able to apply trigger concept on tables to maintain integrity of data.

# Lecture : 8

## Database Security and Authorization

**Learning objective:** In this lecture learners will be able to understand concept of security and authorization of database.

**1. Authentication**

User authentication is to make sure that the person accessing the database is who he claims to be. Authentication can be done at the operating system level or even the database level itself. Many authentication systems such as retina scanners or biometrics are used to make sure unauthorized people cannot access the database.

2. **Authorization**

Authorization is a privilege provided by the Database Administer. Users of the database can only view the contents they are authorized to view. The rest of the database is out of bounds to them.

The different permissions for authorizations available are:

Primary Permission - This is granted to users publicly and directly.

Secondary Permission - This is granted to groups and automatically awarded to a user if he is a member of the group.

Public Permission - This is publicly granted to all the users.

Context sensitive permission -This is related to sensitive content and only granted to a select user.

The categories of authorization that can be given to users are:

System Administrator -This is the highest administrative authorization for a user. Users with this authorization can also execute some database administrator commands such as restore or upgrade a database.

System Control - This is the highest control authorization for a user. This allows maintenance operations on the database but not direct access to data.

System Maintenance - This is the lower level of system control authority. It also allows users to maintain the database but within a database manager instance.

System Monitor - Using this authority, the user can monitor the database and take snapshots of it.

## 3.Database Integrity

Data integrity in the database is the correctness, consistency, and completeness of data. Data integrity is enforced using the following three integrity constraints:

**Entity Integrity -** This is related to the concept of primary keys. All tables should have their own primary keys which should uniquely identify a row and not be NULL.

**Referential Integrity -** This is related to the concept of foreign keys. A foreign key is a key of a relation that is referred in another relation.

**Domain Integrity -** This means that there should be a defined domain for all the columns in a database.

## *Let's check the take away from this lecture*

1)Which of the following applies for: Name of person should be in characters only.

a)Entity integrity **c) Domain integrity**

b) Authentication d) Database security

Exercise:

Q.1 What is authorization?

Q.2 Explain difference between authentication and authorization.

Questions/problems for practice:

Q. 2 Explain how database integrity can be maintained?

**Learning from the lecture:** Learners will be able to explain how the security and authorization of database can be maintained.

# Lecture : 9

### Granting of Privileges, Revoking of Authorization in SQL

**Learning objective:** In this lecture learners will be able to understand how Granting and Revoking is done in SQL.

A user who has been granted some form of authorization may be allowed to pass on this authorization to other users.

By default, a user/role that is granted a privilege is not authorized to grant that privilege to another user/role.

If we wish to grant a privilege and to allow the recipient to pass the privilege on to other users, we append the with grant option clause to the appropriate grant command.

For example, if we wish to allow Amit the select privilege on department and allow

Amit to grant this privilege to others, we write:

Grant select on department to Amit with grant option;

The creator of an object (relation/view/role) holds all privileges on the object, including the privilege to grant privileges to others.

**Granting of update authorization**

Initially, the database administrator grants update authorization on teaches to users U1, U2, and U3, who may in turn pass on this authorization to other users.

The passing of a specific authorization from one user to another can be represented by an authorization graph.

The nodes of this graph are the users.

Consider the graph for update authorization on teaches. The graph includes an edge Ui → Uj if user Ui grants update authorization on teaches to Uj . The root of the graph is the database administrator.

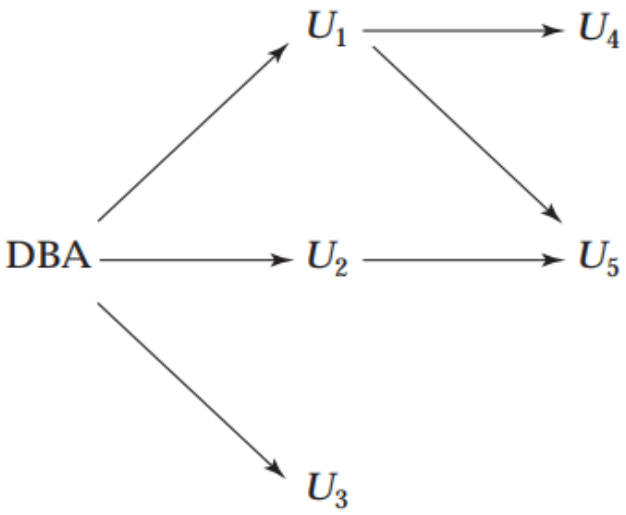



Fig. 4.1 Authorization Grant Graph

U5 is granted authorization by both U1 and U2; U4 is granted authorization by only U1.

**Revoking of Privileges**

Suppose that the database administrator decides to revoke the authorization of user U1.

Since U4 has authorization from U1, that authorization should be revoked

as well.

However, U5 was granted authorization by both U1 and U2.

Since the database administrator did not revoke update authorization on teaches from U2,

U5 retains update authorization on teaches.

If U2 eventually revokes authorization from U5, then U5 loses the authorization.

**Cascading revocation**

A pair of devious users might attempt to defeat the rules for revocation of authorization by granting authorization to each other.

For example, if U2 is initially granted an authorization by the database administrator, and U2 further grants it to U3.

Suppose U3 now grants the privilege back to U2.

If the database administrator revokes authorization from U2, it might appear that U2 retains

authorization through U3.

However, note that once the administrator revokes authorization from U2, there is no path in the authorization graph from the root to either U2 or to U3.

Thus, SQL ensures that the authorization is revoked from both the users.

The revoke statement may specify restrict in order to prevent cascading revocation:

```
revoke select on department from Amit, Satoshi restrict;
```

The following revoke statement revokes only the grant option, rather than the actual select privilege:

```
revoke grant option for select on department from Amit;
```

## *Let's check the take away from this lecture*

1) **Cascading revocation can be denied by ____ clause.**

a) With Grant Option
b) **Restrict**
c) Revoke All
d) Revoke

Exercise

Q.1 Distinguish between granting and revoking privileges.

Questions/Problems for practice:

Q.3 What are different privileges available for database users.

**Learning from this lecture:** Learners will able be to know the apply SQL constructs to grant and revoke privileges from database users.

# Lecture: 10

### Fundamental Operations in Relational Algebra

**Learning Objective:** In this lecture, learners will be able to understand relational Algebra.

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

**Unary operations**: select, project, and rename operations are called unary operations, because they operate on one relation.

**Select Operation:**

The select operation selects tuples that satisfy a given predicate.
Use the lowercase Greek letter sigma (σ) to denote selection.
The predicate appears as a subscript to σ.
The argument relation is in parentheses after the σ.

Example:

Select tuples from books where subject is 'database'.

σsubject = "database"(Books)

Select tuples from books where subject is 'database' and 'price' is 450.

σsubject = "database" and price = "450"(Books)

Select tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

$\sigma_{\text{subject = "database" and price = "450" or year > "2010"}}(\text{Books})$

**Project Operation**

The project operation is a unary operation that returns its argument relation, with certain attributes left out.
Any duplicate rows are eliminated.
Projection is denoted by the uppercase Greek letter pi (∏)

Example:
To list all instructors' ID, name, and salary, but do not care about the dept name.

$$\prod_{\text{id,name,salary}} (\text{Instructor})$$

Select subject and author from the relation Books

$$\prod_{\text{subject, author}} (\text{Books})$$

**Rename Operation**

Results of relational-algebra expressions do not have a name that we can use to refer to them.
Rename operator, denoted by the lowercase Greek letter rho does this renaming the expressions.

$\rho_x$( E ) : Returns the result of expression E under the name x

$\rho_{x(A1, A2, A3,\ldots,An)}$ ( E ) : Returns the result of expression E under the name x, and with the attributes renamed to A1, A2,..., An.

Example:

Rename Department attributes id,name to dept_id, dept_name.

$\rho_{(dept\_id, dept\_name)}$ ( Department)

Rename Department relation to dept and its attributes id,name to dept_id, dept_name.

$\rho_{dept\ (dept\_id, dept\_name)}$ ( Department )

*Let's check the take away from this lecture*

1) **Rename operation is denoted by __.**

a) ρ c) ∏

b) σ d)none of the above

Exercise

Q.1 Explain Rename operation of relational algebra.

**Learning from the Lecture:** Learners will be able to learn Unary relational algebra operations.

# Lecture : 11

## Fundamental Operations in Relational Algebra

**Learning Objective:** In this lecture, learners will be able to understand relational Algebra.

**Binary operations:**

- Union
- Set different
- Cartesian product

**Union Operation**

It performs binary union between two given relations and is defined as :

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation − r U s

Where r and s are either database relations or relation result set (temporary relation).
For a union operation to be valid, the following conditions must hold −
r, and s must have the same number of attributes.
Attribute domains must be compatible.
Duplicate tuples are automatically eliminated.

Example: Projects the names of the authors who have either written a book or an article or both.

$$\prod_{author} (Books) \cup \prod_{author} (Articles)$$

**Set difference Operation**

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.
Notation − r − s
Example: Provides the name of authors who have written books but not articles.

$$\prod_{author} (Books) - \prod_{author} (Articles)$$

**Cartesian product Operation**

Combines information of two different relations into one.
Notation − r X s
Where r and s are relations and their output will be defined as :
r X s = { q t | q ∈ r and t ∈ s}
Example: Obtain all the books and articles written by author Korth.

$$\sigma_{author = 'Korth'}(Books \times Articles)$$

*Let's check the take away from this lecture*

**Cartesion product of two relations have 3 and 4 tuples respectively. How many tuples will be in result ?**

a) 3 c) 7
b) 4 **d)12**

Exercise
Q.1 Explain Cartesian operation of relational algebra.
Q.2 Explain Set difference operation in detail.

**Learning from the Lecture:** Learners will be able to learn Binary relational algebra operations.

**Conclusion**

The study of advanced SQL gives additional functionalities like access control, security, trigger constraint etc. to perform on database. This allows efficient handling of data. Relational algebra gives idea of how procedural language works. It helps in designing efficient query design.

**Short Answer Questions:**

1.What is relational Algebra?

Ans) Relational algebra is a procedural query language.

2.What is Unary Operation?

Ans) Operation that operate on one relation.

3.What is Grant?

Ans) Grant keyword is used to give privileges to user.

4.What is Revoke?

Ans) Revoke keyword is used to to take back privileges from user.

5.What do you mean by authorization?

Ans) Authorization is a privilege provided by the Database Administer.

6.What is authentication ?

Ans) Authentication is to make sure that the person accessing the database is who he claims to be

7.What do you mean by trigger?

Ans) Trigger is a statement that the system executes automatically as a side effect of a modification to the database.

**Long Answer Questions:**

1. Write a query to apply after update trigger with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 5

2. Explain trigger with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz, 6th Edition, Chapter 5

3. Discuss the use of on delete cascade clause with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 5

4. Discuss database security and authorization.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 5

5. Illustrate granting of privileges with suitable diagram.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 5

6. Illustrate revoking of privileges with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 5

7. Explain fundamental relation algebra operations.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 6

**Set of Questions for FA/CE/IA/ESE**

Q. 1) Explain unary relation algebra operations.

Q. 2) Explain difference between grant and revoke.

Q. 3) Discuss the various factors affecting the trigger.

Q. 4) Explain trigger types.

Q. 5) Discuss the Cartesian product of relations.

Q. 6) Explain difference between row trigger and statement trigger.

References:

1) Database System Concept – by Korth, Sudershan, Schilberchatz

**Practice for Module-04**

Q.1) Explain trigger in detail. (10 Marks)

Q. 2) Explain statement trigger and row trigger in detail. (10 Marks)

Q.3) Explain relational algebra operations in detail. (10 Marks)

Q. 4) Discuss database security and authorization. (5 Marks)

Q. 5) Apply referntial integrity constraint on any example relation. (5 marks)

**Self-assessment**

Q.1) Illustrate trigger on any relation.

Q. 2) Apply trigger type on any example relation.

Q. 3) Give example of grant and revoke operations.

**Self-evaluation**

| | | |
|---|---|---|
| Name of Student | | |
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick<br>Your choice |
| | Do you understand the advanced SQL constructs? | Yes<br>No |
| | Do you understand the database security and authorization? | Yes<br>No |
| | Do you know the use of trigger? | Yes<br>No |
| | Do you understand how triggers can be applied? | Yes<br>No |
| | Do you understand module ? | Yes, Completely.<br>Partialy.<br>No, Not at all. |

# Module:05
# Relational Database Design

## Lecture: 1

### Motivation:

In general, the goal of relational database design is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily. This is accomplished by designing schemas that are in an appropriate normal form. To determine whether a relation schema is in one of the desirable normal forms, we need information about the real-world enterprise that we are modelling with the database. Some of this information exists in a well-designed E-R diagram, but additional information about the enterprise may be needed as well.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Pitfalls in Relational Database Design | 1 | 1 |
| 2 | Concept of Normalization | 1 | 1 |
| 3 | Functional Dependencies | 1 | 1 |
| 4 | 1NF | 1 | 1 |
| 5 | 2NF | 1 | 1 |
| 6 | 3NF | 1 | 1 |
| 7 | BCNF | 1 | 1 |
| 8 | 4NF | 1 | 1 |

### Learning Objective:

- Learner will be able to know functional dependencies.
- Learners will be to know advantage of normalized database.
- Learners will be able to know types of normalization.

### Theoretical Background:

One must have very good knowledge of creating E-R model design. This design gives overall representation of schema.

## Key Definitions:

1. **Normalization** refers to organizing the database in such a manner that the data can be retrieved or updated in the most efficient manner.

## Course Content

**Pitfalls in Relational Database Design**

Informal design guidelines for relational schema:

1. Semantics of the Attributes
2. Reducing the Redundant Value in Tuples.
3. Reducing Null values in Tuples.
4. Disallowing spurious Tuples.

1. Semantics of the Attributes:

Whenever we are going to form relational schema there should be some meaning among the attributes. This meaning is called semantics. This semantics relates one attribute to another with some relation.

Eg:

| USN No | Student name | Sem |
|---|---|---|

2. Reducing the Redundant Value in Tuples:

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies

  Insertion anomalies

  Deletion anomalies

  Modification anomalies

The main goal of the schema diagram is to minimize the storage space that the base memory occupies. Grouping attributes information relations has a significant effect on storage space.

Eg;

| USN No | Student name | Sem |
|---|---|---|

| Dept No | Dept Name |
|---|---|

If we integrate these two and is used as a single table i.e Student Table

| USN No | Student name | Sem | Dept No | Dept Name |
|---|---|---|---|---|

Here whenever if we insert the tuples there may be 'N' students in one department, so Dept No, Dept Name values are repeated 'N' times which leads to data redundancy.

Another problem is update anomalies I.e. if we insert new dept that has no students.

If we delete the last student of a dept, then whole information about that department will be deleted

If we change the value of one of the attributes of a particular table that we must update the tuples of all the students belonging to that dept else Database will become inconsistent.

Note: Design in such a way that no insertion, deletion, modification anomalies will occur.

3. Reducing Null values in Tuples.

Relations should be designed such that their tuples will have as few NULL values as possible.

Attributes that are NULL frequently could be placed in separate relations (with the primary key)

4. Disallowing spurious Tuples:

Bad designs for a relational database may result in erroneous results for certain JOIN operations

The "lossless join" property is used to guarantee meaningful results for join operations.

The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

**Pitfalls in Relational DB Design**

A bad design may have several properties, including:

- Repetition of information.
- Inability to represent certain information.
- Loss of information.

*Let's check the take away from this lecture*

Exercise

Q.1 Discuss properties of good relational database design.

**Learning from this lecture**: Learners will be able to understand characteristics that relational database should possess.

# Lecture: 2

## Concept of Normalization

**Learning objective:** In this lecture learners will be able to understand Normalization concept.

The normalization process, as first proposed by Codd (1972), takes a relation schema through a series of tests to "certify" whether or not it belongs to a certain NORMAL FORM. Initially, Codd proposed three normal forms, which he called first, second, and third form. A stronger definition of 3NF was proposed later by Boyce and Codd and known as Boyce-Codd normal form. All these normal forms are based on the functional dependencies among the attributes of a relation. Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed, based on the concepts of multivalued dependencies and join dependencies, respectively.

NORMALIZATION of DATA can be looked on as a process during which unsatisfactory relation schemas are decomposed by breaking up their attributes into smaller relation schemas that possess desirable properties. One objective of the original normalization process is to ensure that the update anomalies (discussed earlier) do not occur.

Normal forms, when considered in isolation from other factors, do not guarantee a good database design.

The database designers may not need to normalize to the highest possible normal form. Relations may be left in lower normal forms for performance reasons.

Normalization refers to organizing the database in such a manner that the data can be retrieved or updated in the most efficient manner.

1 NF : Remove multivalued attribute

2 NF : Remove Partial dependancy

3 NF: Remove transitive dependancy

BCNF: Remove anomalies resulting from functional dependancy

4 NF: Remove multivalued dependancy

## *Let's check the take away from this lecture*

**1) Normalization increases redundancy.**
**a)** True **b) False**

2) **Anomalies are avoided by splitting the offending relation into multiple relations, is also known as**

a) Accupressure

b) **Decomposition**

c) Precomposition

d) Both Decomposition and Precomposition

Exercise
Q.1 What is normalization.
Questions/Problems for practice:
Q.2 List types of normalization.

Learning from this lecture: Learners will be able to understand normalization and its need.

# Lecture: 3

## Functional Dependencies

**Learning objective**: In this lecture learners will be able to understand Functional Dependencies among relations.

The single most important concept in relational schema design is that of a functional dependencies.

Definition: A functional dependency is a constraint between two sets of attributes from the database.

X functionally determines Y in a relation schema R if and only if, whenever two tuples of R(X,Y) agree on their X-value, they must necessarily agree on Y-value.

There are following types of on functional dependencies:

**1. Trivial Functional Dependency:**

A Trivial Functional Dependency occurs when we describe a functional dependency of an attribute on a collection of attributes that contains original attribute.

X --> Y

Where Y is subset of X

Eg. AB --> B

**2. Full Functional Dependency**

X --> Y is Full Functional Dependency iff all the attributes of X constitute candidate key .

Eg. A--> B

A--> C

Both are Full Functional Dependency because A is key and every attribute is determined by A.

**3. Partial Functional Dependency**

X --> Y is partial Functional Dependency if X is not fully a key but is a part of key.

Eg. AB --> C

C --> D

A --> E

Here, AB is a key.

So, A --> E is a Partial Functional Dependency.

**4. Transitive Functional Dependency**

Transitive Functional Dependency occurs when there is an indirect relationship that causes a functional dependency.

Eg. A-->C Transitive Functional Dependency is true only because both A-->B and B-->C are true and C does not belongs to candidate key.

*Let's check the take away from this lecture*

1) **AB --> C is trivial FD.**

a) True. **b) False**

2) **A _____ is an indirect functional dependency, one in which X->Z only by virtue of X->Y and Y->Z.**

a) Multivalued Dependencies

b) Join Dependency

c) Trivial Functional Dependency

**d) Transitive Dependencies**

Exercise

Q.1 What is Functional Dependency?

Q.2 Explain types of Functional Dependency.

Questions/Problems for practice:

Q.3 Explain partial dependency with suitable example.

**Learning from this lecture:** Learners will be able to state various types of functional dependencies.

# Lecture: 4

## 1NF

**Learning objective:** In this lecture learners will be able to understand first normal form of database design.

### 1 NF:

1. A relation is in 1NF if the values in the domain of each attribute of the relation are atomic.
2. It does not allow composite and multivalued attribute.
3. Eg. Student (id, name, department)
4. Here, name is composite attribute consisting of first name and last name.
5. So, remove composite attribute by creating two relations of student.

   Student_fname (id, first_name, department)

   Student_lname (id, last_name, department)

6. To maintain the uniqueness, keep primary key of original relation in both separated relations.
7. In above example, student id (which is primary key) is kept in both newly created relations.

| STUD_NO | STUD_NAME | STUD_PHONE | STUD_STATE | STUD_COUNTRY |
|---|---|---|---|---|
| 1 | RAM | 9716271721, 9871717178 | HARYANA | INDIA |
| 2 | RAM | 9898297281 | PUNJAB | INDIA |
| 3 | SURESH | | PUNJAB | INDIA |

Table 1

Conversion to first normal form

| STUD_NO | STUD_NAME | STUD_PHONE | STUD_STATE | STUD_COUNTRY |
|---|---|---|---|---|
| 1 | RAM | 9716271721 | HARYANA | |
| 1 | RAM | 9871717178 | HARYANA | INDIA |
| 2 | RAM | 9898297281 | PUNJAB | INDIA |
| 3 | SURESH | | PUNJAB | INDIA |

Table 2

## *Let's check the take away from this lecture*

**1) Email id attribute can be_____**

**a) multivalued attribute** c) single attribute

b) composite attribute d) partial attribute

**2) In which normal foam conversion of composite attribute to individual attribute happens,**

**a.first**

b.second

c.third

d.four

e.none of these

**3) Normalization is used to design _____**

a.join dependencies

**b.relational database**

c.multi-valued dependencies

d.cyclic dependencies

e.none of these

Exercise:

Q.1 Explain 1NF with example.

Questions/problems for practice:

Q. 2 Take any relation example and convert it in 1NF.

Learning from this lecture: Learners will be able to understand the mechanism of conversion of relation into 1NF.

# Lecture : 5

**2NF**

**Learning objective:** In this lecture learners will be able to understand Second normal form of database design.

1. A relation is in 2NF if it is in 1NF and

2. Every non key attribute A in R is fully functionally dependant on any key of R.

3. It is based on full functional dependency.

4. If partial dependency is there, then it is not in 2NF.

Example:

A --> BCDEF
BC --> ADEF
B --> F
D --> E

Here, FD No. 3 violates 2NF condition as A and BC are candidate keys and F is non key attribute.

5. So convert it into 2NF by dividing above relation into two relations.

Relation1: A --> BCDEF
BC --> ADE
D --> E

Relation 2: B --> F

## *Let's check the take away from this lecture*

1) **If relation is in 2NF, it has to be in following normal form.**
**i) 1NF** ii) 3NF iii) any normal form iv) not required of any normal form

Exercise:

Q.1 Explain 2NF with example.

Questions/problems for practice:

Q.2 Explain how 2NF ensures that there is no partial dependency.

Learning from the lecture: Learners will be able to understand the mechanism of conversion of relation into 2NF.

# Lecture : 6

**3 NF**

**Learning objective:** In this lecture learners will able to understand Third normal form of database design.

**3 NF**

1. A relation is in 3NF if it is 2NF and
2. When a non trivial functional dependency X--> A holds in R either
   a) X is super key of R or

b) A is a key attribute.

3. There should not be transitive functional dependency.

4. No non key attribute should determine any other non key attribute.

A --> BCDEF
BC --> ADE
D --> E

Here, A and BC are candidate keys. So, A,B and C are key attributes.

5. So, D--> E is transitive functional dependency.

6. To covert this relation in 3 NF:

Relation 1: A --> BCD
BC --> AD

Relation 2: D-->E

## *Let's check the take away from this lecture*

1) **Which one is strict among following.**
**a) 2 NF** c) 1 NF
b) all are same d) none of the above

Exercise:

Q.1 Explain 3 NF with the help of a suitable example.
Questions/problems for practice:
Q. 2 Explain how 3NF ensures that there is no transitive dependency.

Learning from the lecture: Learners will be able to understand the mechanism of conversion of relation into 3NF.

# Lecture : 7

**BCNF**

**Learning objective:** In this lecture learners will be able to understand BCNF of database design.

**BCNF**

1. It is Boycee Codd Normal Form

2. A relation is in BCNF if it is 3NF and

3. When a non trivial functional dependency X--> A holds in then
   a) X is super key of R
   b) A should always be non key.
4. There should not be transitive functional dependency.
5. It is not dependency preserving normal form.

Eg. A --> BCD
BC --> AD
D --> B

Here, FD no. 3 violates the rules of BCNF.

6. So decompose this relation.

Relation 1: A --> CD

Relation 2: D --> B

## *Let's check the take away from this lecture*

1) **Every relation that is in 3NF is also in**
a) BCNF c) **2NF**
b) cant say d) none of the above

**2) Define the function of BCNF .**

a. dependency preserving and lossless join

**b. not dependency preserving and lossless join**

c. dependency preserving and not lossless join

d. none of these

e .all of above

Exercise:

Q.1 Explain the 3NF with example.
Questions/problems for practice:
Q. 2 Discuss the rules that relation should satisfy so as to be in BCNF.

**Learning from the above lecture**: Learners will be able to understand the mechanism of conversion of relation into BCNF.

# Lecture : 8

**4NF**

**Learning objective:** In this lecture learners will be able to understand 4NF of database design.

**4NF**

1. A relation is in 4NF if
   a.It is in BCNF and
   b.For every non trivial multi valued dependancy X-->Y, X is superkey of R.
2. Eg. Employee (name, pname,dname)
   Ename is not superkey of R.
3. So, decompose employee relation :

   Relation 1: name --> pname
   Relation 2: name --> dname

A table is said to have multi-valued dependency, if the following conditions are true,

1. For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
3. And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

## *Let's check the take away from this lecture*

1) **From the following which one is toughest normal form.**
   i) 1NF ii) 2NF iii) BCNF **iv) 4NF**

2) **which of the following is designed to cope with 4NF.**

A.**multi value dependency**
B.join dependency
C.Transitive dependency
D. none of these
E. all of the above

Exercise:

Q.1 Explain 4NF in detail.

**Learning from the lecture:** Learners will able to understand the mechanism of conversion of relation into 4NF.

**Conclusion**

The study of formal approach to relational database design gives the notion of functional dependencies. We then defined normal forms in terms of functional dependencies and other types of data dependencies. Different normal forms give idea about hoe any relation can be converted into other normal forms. This makes design of relational database more accurate and have less redundancy.

**Short Answer Questions:**

1.What is normalization?

Ans) Normalization refers to organizing the database in such a manner that the data can be retrieved or updated in the most efficient manner.

2.What is functional dependency?

Ans) For a given relation R, attribute Y of R is functionally dependant on attribute X iff each X value in R is associated with one Y value of R.

**Long Answer Questions:**

1. What is normalization and why it is needed?

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 8

2. Explain 1st and 2nd normal form.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 8

3. Explain BCNF normal form.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 8

4. State and explain Functional dependency and full functional dependence.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 8

**Set of Questions for FA/CE/IA/ESE**

Q. 1) Explain normalization and state its types.

Q. 2) Illustrate functional dependency with example.

Q. 3) Explain functional dependency with example and illustrate its types..

Q. 4) What is BCNF? How it is related to multivalued dependency?

Q. 5) Discuss the rules of 3 NF and explain it with example.

References:

1) Database System Concept – by Korth, Sudershan, Schilberchatz

**Practice for Module-04**

Q.1) a) Explain pitfalls in relational database design (10 Marks)

Q.2) a) Define functional dependency and its types. (10 Marks)

Q.3) Normalize the relation whose functional dependencies is given as (10 Marks)
R (A, B, C, D, E, F)

A->CE

B->D

C->ADE

BD->F

Q.4) Explain multivalued dependency.

Q.5) Illustrate 3NF and BCNF with example.

**Self-assessment**

Q.1) Decompose the relation to the highest possible normal form.

Consider the relation R (PQRSTU) with following dependencies:

P->Q, ST->PR, S->U.

State R in which normal form? Decompose it to BCNF.

Q. 2) What are the different types of functional dependencies?

Q. 3) Consider following relation.

Employee(emp_id,emp_name,emp_address,emp_designation,emp_salary,dept_no,dept_name,location,joining_date)

Find functional dependancies and normalize upto 2NF.

**Self-evaluation**

| Name of Student | | |
|---|---|---|
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick Your choice |
| 16. | Do you understand the normalization? | o Yes<br>o No |
| 17. | Do you understand the need of normalization? | o Yes<br>o No |
| 18. | Do you know the various types of functional dependencies? | o Yes<br>o No |
| 19. | Do you understand types of normal forms? | o Yes<br>o No |
| 20. | Do you understand module ? | o Yes, Completely.<br>o Partialy.<br>o No, Not at all. |

# Module: 06

# Transaction, Recovery and Concurrency Control

## Lecture : 1

### Motivation:

To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.

### Syllabus:

| Lecture no | Content | Duration (Hr) | Self-Study (Hrs) |
|---|---|---|---|
| 1 | Transaction Concept, Transaction States | 1 | 1 |
| 2 | ACID Properties of Transaction | 1 | 1 |
| 3 | Serial and Concurrent Executions | 1 | 2 |
| 4 | Conflict and View Serializability | 1 | 2 |
| 5 | Lock Based Protocols | 1 | 2 |
| 6 | Deadlock Handling | 1 | 2 |
| 7 | Failure Classification, Log based recovery | 1 | 2 |
| 8 | Checkpoint, Shadow Paging | 1 | 2 |

### Learning Objective:

- Learner should describe DBMS Techniques for concurrency control
- Learner should describe and distinguish locking and deadlock
- Learner should describe locking techniques for concurrency control based on time stamp ordering
- Learner should describe multi version concurrency control techniques

- Learner should describe some of the techniques that can be used for database recovery from failures.
- Learner should describe several recovery concepts, including write ahead logging, in-place versus shadow updates, and the process of rolling back (undoing) the effect of an incomplete or failed transaction.
- Learner should describe the technique known as shadowing or shadow paging
- Learner should describe techniques for recovery from catastrophic failure

## Theoretical Background:

Students can identify how transaction is performed in database by satisfying all the properties of transaction so proper result can be retrieved form DB. Identification of serializable and non-serializable schedule is possible.

## Key Definitions:

1. **Transaction**:- It is a unit of program execution that accesses and possibly updates various data items.
2. **Atomicity** :- Either all operations of the transaction are properly reflected in the database or none are.
3. **Consistency**:-Execution of a transaction in isolation preserves the consistency of the database.
4. **Isolation**:-Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.
5. **Durability**:- After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.
6. **Concurrency**:-Performing Many transactions - at the same time is called Concurrency.
7. **Concurrency control**:-Transactions must be isolated then we need of concurrency control to ensure no interference. Concurrency control is essential for the correctness of transactions executed concurrently in a DBMS, which is the common execution mode for performance reasons. The main concern and goal of concurrency control is isolation.
8. **Locking protocol**:-The protocol to manage the multiple transactions is called Locking Protocol.

9. **Lock types**:-Locks can be shared or exclusive, and can lock out readers and/or writers. Locks can be created implicitly by the DBMS when a transaction performs an operation, or explicitly at the transaction's request.Shared locks allow multiple transactions to lock the same resource. The lock persists until all such transactions complete. Exclusive locks are held by a single transaction and prevent other transactions from locking the same resource.
10. **Lock granularity**:-Locks can be coarse, covering an entire database, fine-grained, covering a single data item, or intermediate covering a collection of data such as all the rows in a RDBMS table.

## Course Content

**Transaction Concept, Transaction States**

A transaction is a unit of program execution that accesses and possibly updates various data items. Usually, a transaction is initiated by a user program written in a high-level data-manipulation language or programming language (for example, SQL,COBOL, C, C++, or Java), where it is delimited by statements (or function calls) of the form begin transaction and end transaction. The transaction consists of all operations executed between the begin transaction and end transaction.

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

**A's Account**

Open_Account(A)

Old_Balance = A.balance

New_Balance = Old_Balance - 500

A.balance = New_Balance

Close_Account(A)

**B's Account**

Open_Account(B)

Old_Balance = B.balance

New_Balance = Old_Balance + 500

B.balance = New_Balance

Close_Account(B)

A transaction is a sequence of read and write operations on data items that logically functions as one unit of work

- It should either be done entirely or not at all
- If it succeeds, the effects of write operations persist (commit); if it fails, no effects of write operations persist (abort)
- These guarantees are made despite concurrent activity in the system, and despite failures that may occur

**Transaction States**

**Active –** the initial state; the transaction stays in this state while it is executing
**Partially committed** – after the final statement has been executed.
**Failed –** after the discovery that normal execution can no longer proceed.
**Aborted –** after the transaction has been rolled back and the database restored to its state prior to the start of the transaction.Two options after it has been aborted:

- restart the transaction:
  can be done only if no internal logical error
- kill the transaction:

**Committed –** after successful completion.

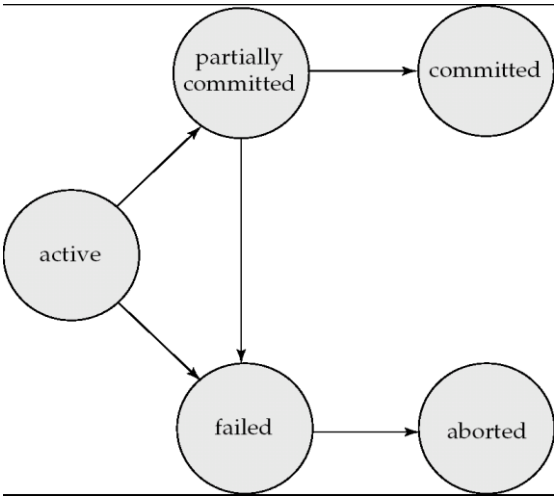



Fig 6.1 State diagram of a transaction.

## *Let's check the take away from this lecture*

1) **Transaction can't be in _____ state.**

a) Active b) Committed

c) Aborted d) **Execution**

2) **Atomicity needs transaction to be completed either in full or not at all.**

a) **True** b) False

Exercise

Q.1 What is transaction?

Questions/Problems for practice:

Q.2 Describe transaction states.

**Learning from this lecture**: Learners will be able to understand transaction concept and various phases of transactions.

# Lecture: 2

## ACID Properties of Transaction

**Learning objective:**

In this lecture learners will be able to understand ACID Properties of Transaction.

**Atomicity:** This means that either all the instructions within the transaction will be reflected in the database, or none of them will be reflected.

Say for example, we have two accounts A and B, each containing Rs 1000/-. We now start a transaction to deposit Rs 100/- from account A to Account B.

```
Read A;
A = A – 100;
Write A;
Read B;
B = B + 100;
Write B;
```

Fine, is not it? The transaction has 6 instructions to extract the amount from A and submit it to B. The AFIM will show Rs 900/- in A and Rs 1100/- in B.

Now, suppose there is a power failure just after instruction 3 (Write A) has been complete. What happens now? After the system recovers the AFIM will show Rs 900/- in A, but the same Rs 1000/- in B. It would be said that Rs 100/- evaporated in thin air for the power failure. Clearly such a situation is not acceptable.

The solution is to keep every value calculated by the instruction of the transaction not in any stable storage (hard disc) but in a volatile storage (RAM), until the transaction completes its last instruction. When we see that there has not been any error we do something known as a COMMIT operation. Its job is to write every temporarily calculated value from the volatile storage on to the stable storage. In this way, even if power fails at instruction 3, the post

recovery image of the database will show accounts A and B both containing Rs 1000/-, as if the failed transaction had never occurred.

**Consistency:** If we execute a particular transaction in isolation or together with other transaction, (i.e. presumably in a multi-programming environment), the transaction will yield the same expected result.

To give better performance, every database management system supports the execution of multiple transactions at the same time, using CPU Time Sharing. Concurrently executing transactions may have to deal with the problem of sharable resources, i.e. resources that multiple transactions are trying to read/write at the same time. For example, we may have a table or a record on which two transaction are trying to read or write at the same time. Careful mechanisms are created in order to prevent mismanagement of these sharable resources, so that there should not be any change in the way a transaction performs. A transaction which deposits Rs 100/- to account A must deposit the same amount whether it is acting alone or in conjunction with another transaction that may be trying to deposit or withdraw some amount at the same time.

**Isolation:** In case multiple transactions are executing concurrently and trying to access a sharable resource at the same time, the system should create an ordering in their execution so that they should not create any anomaly in the value stored at the sharable resource.

There are several ways to achieve this and the most popular one is using some kind of locking mechanism. Again, if you have the concept of Operating Systems, then you should remember the semaphores, how it is used by a process to make a resource busy before starting to use it, and how it is used to release the resource after the usage is over. Other processes intending to access that same resource must wait during this time. Locking is almost similar. It states that a transaction must first lock the data item that it wishes to access, and release the lock when the accessing is no longer required. Once a transaction locks the data item, other transactions wishing to access the same data item must wait until the lock is released.

**Durability:** It states that once a transaction has been complete the changes it has made should be permanent.

As we have seen in the explanation of the Atomicity property, the transaction, if completes successfully, is committed. Once the COMMIT is done, the changes which the transaction has made to the database are immediately written into permanent storage. So, after the transaction has been committed successfully, there is no question of any loss of information even if the power fails. Committing a transaction guarantees that the AFIM has been reached.

There are several ways Atomicity and Durability can be implemented. One of them is called Shadow Copy. In this scheme a database pointer is used to point to the BFIM of the database. During the transaction, all the temporary changes are recorded into a Shadow Copy, which is an exact copy of the original database plus the changes made by the transaction, which is the AFIM. Now, if the transaction is required to COMMIT, then the database pointer is updated to point to the AFIM copy, and the BFIM copy is discarded. On the other hand, if the transaction is not committed, then the database pointer is not updated. It keeps pointing to the BFIM, and the AFIM is discarded. This is a simple scheme, but takes a lot of memory space and time to implement.

If you study carefully, you can understand that Atomicity and Durability is essentially the same thing, just as Consistency and Isolation is essentially the same thing.

Example: Transaction to transfer $50 from account A to account B:

1. read (A)
2. A := A – 50
3. write(A)
4. read(B)
5. B := B + 50
6. write(B)

Two main issues to deal with:

1. Failures of various kinds, such as hardware failures and system crashes
2. Concurrent execution of multiple transactions

Transaction to transfer $50 from account A to account B:

1. read(A)

2. A := A – 50
3. write(A)
4. read(B)
5. B := B + 50
6. write(B)

**Atomicity requirement:-**

1. If the transaction fails after step 3 and before step 6, money will be "lost" leading to an inconsistent database state

2. Failure could be due to software or hardware

3. System should ensure that updates of a partially executed transaction are not reflected in the database

4. Either all operations of the transaction are properly reflected in the database or none are.

**Durability requirement**:

Once the user has been notified that the transaction has completed (i.e., the transfer of the $50 has taken place), the updates to the database by the transaction must persist even if there are software or hardware failures.

Transaction to transfer $50 from account A to account B:

1. read(A)
2. A := A – 50
3. write(A)
4. read(B)
5. B := B + 50write(B)

Consistency requirement in above example:

the sum of A and B is unchanged by the execution of the transaction

**Consistency requirements:--**

Explicitly specified integrity constraints such as primary keys and foreign keys

e.g. sum of balances of all accounts, minus sum of loan amounts must equal value of cash-in-hand

1. A transaction must see a consistent database.

2. During transaction execution the database may be temporarily inconsistent.

3. When the transaction completes successfully the database must be consistent

4. Erroneous transaction logic can lead to inconsistency

## Isolation requirement:-

If between steps 3 and 6, another transaction T2 is allowed to access the partially updated database, it will see an inconsistent database (the sum A + B will be less than it should be).

| | T1 | T2 |
|---|---|---|
| 1. | read(A) | |
| 2. | A := A – 50 | |
| 3. | write(A) | read(A), read(B), print(A+B) |
| 4. | read(B) | |
| 5. | B := B + 50 | |
| 6. | write(B | |

Isolation can be ensured trivially by running transactions serially

that is, one after the other.

However, executing multiple transactions concurrently has significant benefits.

*Let's check the take away from this lecture*

**1) …….. is Acid Property**

a) **Atomicity**

b) Redundancy

c) Internet

d) Recoverabilit

Exercise

Q.1 Discuss ACID properties in detail.

Q.2 Explain atomicity with suitable example.

Questions/Problems for practice:

Q.3 Explain Isolation in detail.

**Learning from this lecture:** Learners will be able to understand important database properties that one it must maintain.

# Lecture: 3

## Serial and Concurrent Executions

**Learning objective**: In this lecture learners will be able to understand transaction execution types.

A schedule is a collection of many transactions which is implemented as a unit. Depending upon how these transactions are arranged in within a schedule, a schedule can be of two types:

- Serial: The transactions are executed one after another, in a non-preemptive manner.
- Concurrent: The transactions are executed in a preemptive, time shared method.

In **Serial schedule**, there is no question of sharing a single data item among many transactions, because not more than a single transaction is executing at any point of time. However, a serial schedule is inefficient in the sense that the transactions suffer for having a longer waiting time and response time, as well as low amount of resource utilization.

In **concurrent schedule**, CPU time is shared among two or more transactions in order to run them concurrently. However, this creates the possibility that more than one transaction may need to access a single data item for read/write purpose and the database could contain

inconsistent value if such accesses are not handled properly. Let us explain with the help of an example.

Let us consider there are two transactions T1 and T2, whose instruction sets are given as following. T1 is the same as we have seen earlier, while T2 is a new transaction.

T1

Read A;

A = A – 100;

Write A;

Read B;

B = B + 100;

Write B;

T2

Read A;

Temp = A * 0.1;

Read C;

C = C + Temp;

Write C;

T2 is a new transaction which deposits to account C 10% of the amount in account A.

If we prepare a serial schedule, then either T1 will completely finish before T2 can begin, or T2 will completely finish before T1 can begin. However, if we want to create a concurrent schedule, then some Context Switching need to be made, so that some portion of T1 will be executed, then some portion of T2 will be executed and so on. For example say we have prepared the following concurrent schedule.

| T1 | T2 |
|---|---|
| Read A; | |
| A = A - 100; | |
| Write A; | |
| | Read A; |
| | Temp = A * 0.1; |
| | Read C; |
| | C = C + Temp; |
| | Write C; |
| Read B; | |
| B = B + 100; | |
| Write B; | |

No problem here. We have made some Context Switching in this Schedule, the first one after executing the third instruction of T1, and after executing the last statement of T2. T1 first deducts Rs 100/- from A and writes the new value of Rs 900/- into A. T2 reads the value of A, calculates the value of Temp to be Rs 90/- and adds the value to C. The remaining part of T1 is executed and Rs 100/- is added to B.

It is clear that a proper Context Switching is very important in order to maintain the Consistency and Isolation properties of the transactions. But let us take another example where a wrong Context Switching can bring about disaster. Consider the following example involving the same T1 and T2

| T1 | T2 |
|---|---|
| Read A; | |

A = A - 100;

Read A;

Temp = A * 0.1;

Read C;

C = C + Temp;

Write C;

Write A;

Read B;

B = B + 100;

Write B;

This schedule is wrong, because we have made the switching at the second instruction of T1. The result is very confusing. If we consider accounts A and B both containing Rs 1000/- each, then the result of this schedule should have left Rs 900/- in A, Rs 1100/- in B and add Rs 90 in C (as C should be increased by 10% of the amount in A). But in this wrong schedule, the Context Switching is being performed before the new value of Rs 900/- has been updated in A. T2 reads the old value of A, which is still Rs 1000/-, and deposits Rs 100/- in C. C makes an unjust gain of Rs 10/- out of nowhere.

In the above example, we detected the error simple by examining the schedule and applying common sense. But there must be some well formed rules regarding how to arrange instructions of the transactions to create error free concurrent schedules. This brings us to our next topic, the concept of Serializability.

**Advantages of concurrency:**

1. Improved throughput and resource utilization.

(THROUGHPUT = Number of Transactions executed per unit of time.)

The CPU and the Disk can operate in parallel. When a Transaction Read/Write the Disk another Transaction can be running in the CPU. The CPU and Disk utilization also increases.

2. Reduced waiting time.

In a serial processing a short Transaction may have to wait for a long transaction to complete. Concurrent execution reduces the average response time; the average time for a Transaction to be completed.

## *Let's check the take away from this lecture*

**1) ..... helps solve concurrency problem.**

**a. locking** b. transaction monitor

c. transaction serializability d. two phase commit

Exercise

Q.1 Define Concurrent schedule in detail.

Q.2 What is difference between serial and concurrent schedule?

Questions/Problems for practice:

Q.3 Explain serial schedules in detail.

**Learning from this lecture:** Learners will be able to describe serial and concurrent transactions.

# Lecture: 4

### Conflict and View Serializability

**Learning objective:** In this lecture learners will be able to understand serializability concept.

**Schedule** –A sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed. a schedule for a set of transactions must consist of all instructions of those transactions must preserve the order in which the instructions appear in each individual transaction.

**Serializability**

When several concurrent transactions are trying to access the same data item, the instructions within these concurrent transactions must be ordered in some way so as there are no problem in

accessing and releasing the shared data item. There are two aspects of serializability which are described here:

A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule.

Different forms of schedule equivalence give rise to the notions of:

1. **Conflict serializability**
2. **View serializability**

**Conflicting Instructions:**

Instructions li and lj of transactions Ti and Tj respectively, conflict if and only if there exists some item Q accessed by both li and lj, and at least one of these instructions wrote Q.

1. li = read(Q), lj = read(Q). li and lj don't conflict.
2. li = read(Q), lj = write(Q). They conflict.
3. li = write(Q), lj = read(Q). They conflict
4. li = write(Q), lj = write(Q). They conflict

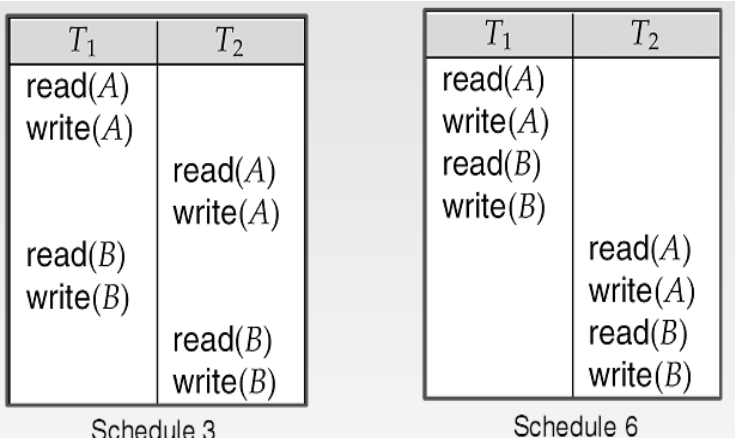
Intuitively, a conflict between li and lj forces a (logical) temporal order between them.

If li and lj are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

**Conflict Serializability:**

If a schedule S can be transformed into a schedule S´ by a series of swaps of non conflicting instructions, we say that S and S´ are conflict equivalent. We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule

Schedule 3 can be transformed into Schedule 6, a serial schedule where T2 follows T1, by series of swaps of non conflicting instructions. Therefore Schedule 3 is conflict serializable.



| $T_1$ | $T_2$ |
|---|---|
| read($A$) | |
| write($A$) | |
| | read($A$) |
| | write($A$) |
| read($B$) | |
| write($B$) | |
| | read($B$) |
| | write($B$) |

Schedule 3

| $T_1$ | $T_2$ |
|---|---|
| read($A$) | |
| write($A$) | |
| read($B$) | |
| write($B$) | |
| | read($A$) |
| | write($A$) |
| | read($B$) |
| | write($B$) |

Schedule 6

Fig. 6.2 Schedules

**View Serializability:**

This is another type of serializability that can be derived by creating another schedule out of an existing schedule, involving the same set of transactions. These two schedules would be called View Serializable if the following rules are followed while creating the second schedule out of the first. Let us consider that the transactions T1 and T2 are being serialized to create two different schedules S1 and S2 which we want to be View Equivalent and both T1 and T2 wants to access the same data item.

1. If in S1, T1 reads the initial value of the data item, then in S2 also, T1 should read the initial value of that same data item.

2. If in S1, T1 writes a value in the data item which is read by T2, then in S2 also, T1 should write the value in the data item before T2 reads it.

3. If in S1, T1 performs the final write operation on that data item, then in S2 also, T1 should perform the final write operation on that data item.

Except in these three cases, any alteration can be possible while creating S2 by modifying S1.

A schedule S is view serializable if it is view equivalent to a serial schedule. Every conflict serializable schedule is also view serializable. Below is a schedule which is view serializable but not conflict serializable.

| $T_3$ | $T_4$ | $T_6$ |
|---|---|---|
| read($Q$) | | |
| | write($Q$) | |
| write($Q$) | | |
| | | write($Q$) |

Fig. 6.3 Schedules

Above schedule is view equivalent to the serial schedule <T3,T4, T6> since the one read(Q) instruction reads the initial value of Q in both schedules and T6 performs the final write of Q in both schedules.

## *Let's check the take away from this lecture*

1) **……. Is sequence of instructions**

a. **Schedule** b. View

c. Conflit d. Centralized

Exercise:

Q.1 Explain view serializability.
Q.2 Explain conflict serializability.
Questions/problems for practice:
Q. 3 Explain conflicting instructions.

**Learning from this lecture:** Learners will be able to understand concept of serializability.

# Lecture : 5

## Lock Based Protocols

**Learning objective:** In this lecture, learners will be able to lock based protocols.

**Locking :**

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories −

1. Lock based protocols
2. 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.

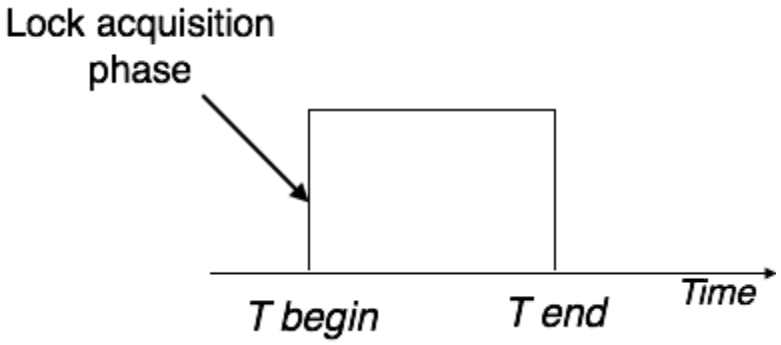



Fig. 6.4 Locks

**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.

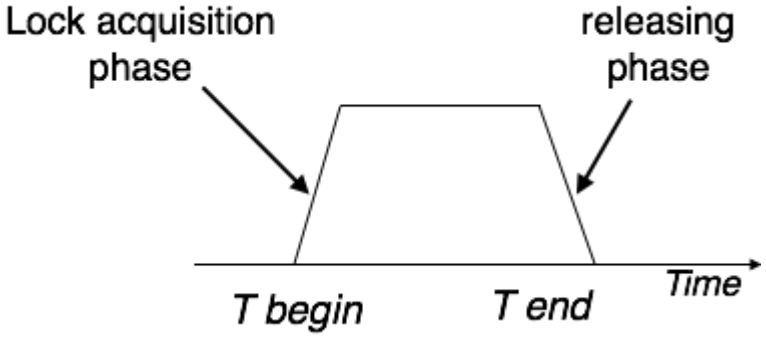



Fig. 6.5 Locks

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.

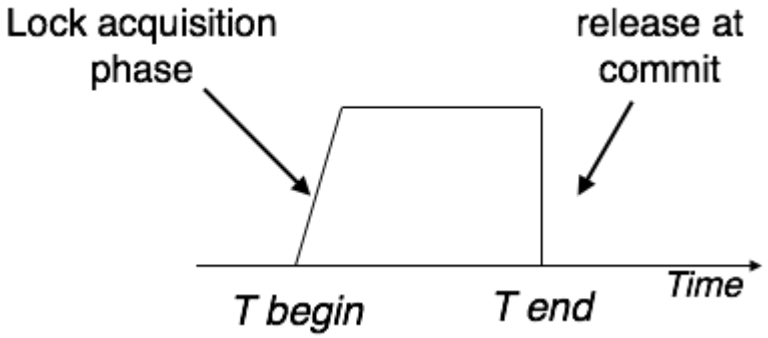



Fig. 6.6 Locks

Strict-2PL does not have cascading abort as 2PL does.

**Locking Rules**

You know there are various locking rules that are applicable when a user reads or writes a data to a database. The various locking rules are -

- Any number of transactions can hold S-locks on an item

• If any transaction holds an X-lock on an item, no other transaction may hold any lock on the item

• A transaction holding an X-lock may issue a write or a read request on the data item

• A transaction holding an S-lock may only issue a read request on the data item

## *Let's check the take away from this lecture*

**1) S lock can be held by only one transaction.**

a) True **b) False**

Exercise:

Q.1 Explain different types of lock protocols.

Questions/problems for practice:

Q.2 What is lock? Explain rules of locking.

**Learning from the lecture:** Learners will be able to know lock-based protocols to ensure database consistency.

# Lecture : 6

### Deadlock Handling

**Learning objective:** In this lecture learners will be able to know the deadlock handling mechanism.

In operating systems or databases, a situation in which two or more processes are prevented from continuing while each waits for resources to be freed by the continuation of the other. Any of several situations where two or more processes cannot proceed because they are both waiting for the other to release some resource.

A situation in which processes of a concurrent processor are waiting on an event which will never occur. A simple version of deadlock for a loosely synchronous environment arises when blocking reads and writes are not correctly matched. For example, if two nodes both execute

blocking writes to each other at the same time, deadlock will occur since neither write can complete until a complementary read is executed in the other node.

**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 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 −

1. If TS(Ti) < TS(Tj) − that is Ti, which is requesting a conflicting lock, is older than Tj − then Ti is allowed to wait until the data-item is available.

2. If TS(Ti) > TS(tj) − that is Ti is younger than Tj − then Ti dies. Ti 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 −

1. If TS(Ti) < TS(Tj), then Ti forces Tj to be rolled back − that is Ti wounds Tj. Tj is restarted later with a random delay but with the same timestamp.

2. If TS(Ti) > TS(Tj), then Ti 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. In a bulky system, deadlock prevention techniques may work well.

**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 Ti requests for a lock on an item, say X, which is held by some other transaction Tj, a directed edge is created from Ti to Tj. If Tj releases item X, the edge between them is dropped and Ti 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.

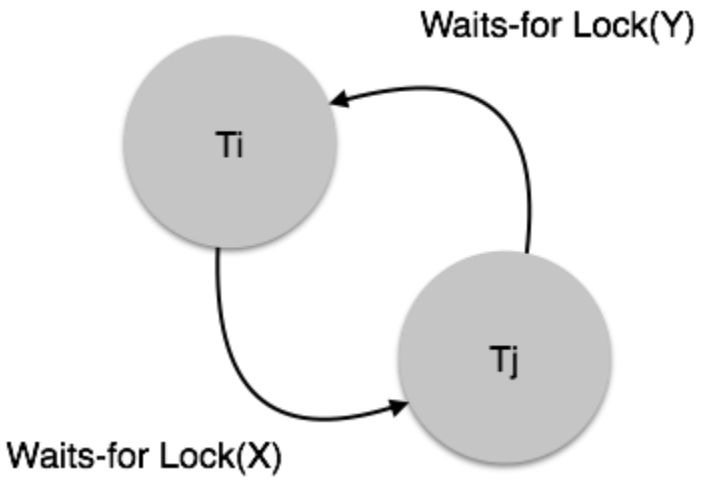



Fig. 6.7 Wait-for Graph

Here, we can use any of the two following approaches −

1. First, do not allow any request for an item, which is already locked by another transaction. This is not always feasible and may cause starvation, where a transaction indefinitely waits for a data item and can never acquire it.

2. The second option is to roll back one of the transactions. It is not always feasible to roll back the younger transaction, as it may be important than the older one. With the help of some relative algorithm, a transaction is chosen, which is to be aborted. This transaction is known as the victim and the process is known as victim selection.

*Let's check the take away from this lecture*

Exercise:

Q.1 Explain Wait-for Graph with example.

Questions/problems for practice:
Q. 2 Explain different deadlock prevention schemes.

**Learning from the lecture:** In this lecture learners will be able to understand different deadlock prevention schemes.

# Lecture : 7

## Failure Classification, Log based recovery

**Learning objective:** In this lecture learners will be able to interpret failure type and recovery type.

There are various types of failure that may occur in a system, each of which needs to be dealt with in a different manner. In this chapter, we shall consider only the following types of failure:

**Transaction failure**.

There are two types of errors that may cause a transaction to fail:

1. Logical error. The transaction can no longer continue with its normal execution because of some internal condition, such as bad input, data not found, overflow, or resource limit exceeded.

2. System error. The system has entered an undesirable state (for example, deadlock), because of which a transaction cannot continue with its normal execution. The transaction, however, can be reexecuted later.

3. System crash. There is a hardware malfunction, or a bug in the database software or the operating system, that causes the loss of the content of volatile storage, and brings transaction processing to a halt. The content of nonvolatile storage remains intact, and is not corrupted.

**Log Based Recovery**

The most widely used structure for recording database modifications is the log.

The log is a sequence of log records, recording all the update activities in the database.

Database Modification

1. Transaction creates a log record prior to modifying the database.

2. We say a transaction modifies the database if it performs an update on a disk buffer, or on the disk itself; updates to the private part of main memory do not count as database modifications.

3. If a transaction does not modify the database until it has committed, it is said to use the deferred-modification technique.

4. If database modifications occur while the transaction is still active, the transaction is said to use the immediate-modification technique.

5. Because all database modifications must be preceded by the creation of a log record, the system has available both the old value prior to the modification of the data item and the new value that is to be written for the data item.

6. This allows the system to perform undo and redo operations as appropriate.

- Undo using a log record sets the data item specified in the log record to the old value.
- Redo using a log record sets the data item specified in the log record to the new value.

Using the Log to Redo and Undo Transactions

1. **redo(Ti )** sets the value of all data items updated by transaction Ti to the new values.

The order in which updates are carried out by redo is important; when recovering from a system crash, if updates to a particular data item are applied in an order different from the order in which they were applied originally, the final state of that data item will have a wrong value.

Most recovery algorithms, do not perform redo of each transaction separately; instead they perform a single scan of the log, during which redo actions are performed for each log record as it is encountered. This approach ensures the order of updates is preserved.

2. **undo(Ti )** restores the value of all data items updated by transaction Ti to the old values.

The undo operation not only restores the data items to their old value, but also writes log records to record the updates performed as part of the undo process.

When the undo operation for transaction Ti completes, it writes a <Ti abort> log record, indicating that the undo has completed.

## *Let's check the take away from this lecture*

1) Due to deadlock, following failure occurs -

a) Logical error c) **system error**

b) system crash d) none of the above

Exercise:

Q.1 Explain deadlock prevention scheme.
Q.2 Illustrate log based recovery.
Questions/problems for practice:
Q. 2 What is deadlock? Explain in detail.

Learning from the above lecture: Learners will be able to know techniques of log based protocol.

# Lecture : 8

## Checkpoint, Shadow Paging

**Learning objective:** In this lecture learners will be able to understand various methods used for decreasing the rate of corrosion, specifically selection of materials and Design

**Checkpoint**

When a system crash occurs, we must consult the log to determine those transactions that need to be redone and those that need to be undone. In principle, we need to search the entire log to determine this information.

There are two major difficulties with this approach:

1. The search process is time-consuming.

2. Most of the transactions that, according to our algorithm, need to be redone have already written their updates into the database. Although redoing them will cause no harm, it will nevertheless cause recovery to take longer.

To reduce these types of overhead, we introduce checkpoints. We describe below a simple checkpoint scheme that

(a) does not permit any updates to be performed while the checkpoint operation is in progress, and (b) outputs all modified buffer blocks to disk when the checkpoint is performed.

A checkpoint is performed as follows:

1. Output onto stable storage all log records currently residing in main memory.

2. Output to the disk all modified buffer blocks.

3. Output onto stable storage a log record of the form <checkpoint L>, where L is a list of transactions active at the time of the checkpoint.

Transactions are not allowed to perform any update actions, such as writing to a buffer block or writing a log record, while a checkpoint is in progress.

The presence of a <checkpoint L> record in the log allows the system to streamline its recovery procedure.

Consider a transaction Ti that completed prior to the checkpoint. For such a transaction, the <Ti commit> record (or < Ti abort> record) appears in the log before the <checkpoint> record. Any database modifications made by Ti must have been written to the database either prior to the

checkpoint or as part of the checkpoint itself.

Thus, at recovery time, there is no need to perform a redo operation on Ti .

After a system crash has occurred, the system examines the log to find the last <checkpoint L> record (this can be done by searching the log backward, from the end of the log, until the fifirst <checkpoint L> record is found).

The redo or undo operations need to be applied only to transactions in L, and to all transactions that started execution after the <checkpoint L> record was written to the log.

Let us denote this set of transactions as T.

• For all transactions Tk in T that have no <Tk commit> record or <Tk abort> record in the log, execute undo(Tk ).

• For all transactions Tk in T such that either the record <Tk commit> or the record <Tk abort> appears in the log, execute redo(Tk ).

**Shadow Paging**

In the shadow-copy scheme, a transaction that wants to update the database first creates a complete copy of the database. All updates are done on the new database copy, leaving the original copy, the shadow copy, untouched. If at any point the transaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.

The current copy of the database is identified by a pointer, called db-pointer, which is stored on disk. If the transaction partially commits (that is, executes its final statement) it is committed as follows:

First, the operating system is asked to make sure that all pages of the new copy of the database have been written out to disk. After the operating system has written all the pages to disk, the database system updates the pointer db pointer to point to the new copy of the database; the

new copy then becomes the current copy of the database. The old copy of the database is then deleted.

The transaction is said to have been committed at the point where the updated db-pointer is written to disk.

**Shadow copy**ing can be used for small databases, but copying a large database would be extremely expensive. A variant of shadow copying, called shadow-paging, reduces copying as follows:

The scheme uses a page table containing pointers to all pages; the page table itself and all updated

pages are copied to a new location. Any page which is not updated by a transaction is not copied, but instead the new page table just stores a pointer to the original page. When a transaction commits, it atomically updates the pointer to the page table, which acts as db-pointer, to point to the new copy. Shadow paging unfortunately does not work well with concurrent transactions and is not widely used in databases.

## *Let's check the take away from this lecture*

1) **From the following which one is beneficial for large database?**
   a) Shadowing c**) Shadow paging**
   b) Copy d) Shadow copy

Exercise:

Q.1 How proper designing of metal article can minimize the corrosion?

Questions/problems for practice:

Q. 2 Explain the following methods of corrosion prevention. I) Using pure metals ii) Using metal

Learning from the lecture: Learners will able to explain the different methods of checkpoints to be included to recover from failure.

**Conclusion**

The study of transaction gives idea of different states of transaction. It gives the idea of characteristics that database should possess. It gives idea of as execution can be serial or concurrent, it should be such that, it must maintain ACID properties of database. Concurrent executions can lead to deadlock which must be prevented using given mechanisms.

**Short Answer Questions:**

1.What is Locking Protocol?

Ans) The protocol to manage the multiple transactions is called Locking Protocol.

2.List Database properties.

Ans) Atomicity, Concurrency, Isolation, Durability

3.List different transaction states.

Ans) Active, Partially Committed, Failed, Aborted, Committed

4.What is throughput?

Ans) Number of Transactions executed per unit of time

5.What is deadlock?

Ans) A situation in which two or more processes are prevented from continuing while each waits for resources to be freed by the continuation of the other

**Long Answer Questions:**

1. What do you mean by transaction? What are different states of transaction?

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 14

2.Explain ACID properties of transaction with example.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 14

3. Illustrate serializability in detail.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 14

4.What is locking? Explain lock based protocols.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 15

5.Discuss failure classification in detail.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 16

6.What is checkpoint? Discuss in detail.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 16

7.Explain the term shadow paging.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 16

8.Describe the Conflict and View Serializability.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 14

9.What is View Serializability? Explain in detail.

Ans) Database System Concept – by Korth, Sudershan, Schilberchatz , 6th Edition, Chapter 14

**Set of Questions for FA/CE/IA/ESE**

Q. 1) What is checkpoint? Discuss in detail.

Q. 2) Explain ACID properties of transaction with example.

Q. 3) Discuss Discuss failure classification in detail.

Q. 4) Explain Serial execution in detail.

Q. 5) Explain concurrent execution in detail.

Q. 6) Explain states of transactions.

References:

1) Database System Concepts by Korth, Sudarshan, Siberschatz

**Practice for Module-06**

Q1) Explain concept of Transaction

Q2) Explain Concurrent Executions

Q3) Explain Serializability

Q4) Explain implementation of Isolation

Q5) Explain Acid Properties of Transaction

Q6) Explain Transactions State diagram

Q7) Explain Conflict Serializability

Q8) Explain View Serialazibility

Q9) Explain transaction definition in SQL

**Self-assessment**

Q.1) Define transaction. With the help of suitable diagram explain different states of transaction

Q. 2) Explain serial and concurrent Execution in detail.

Q. 3) Explain deadlock handling mechanism.

**Self-evaluation**

| | | |
|---|---|---|
| Name of Student | | |
| Class | | |
| Roll No. | | |
| Subject | | |
| Module No. | | |
| S.No | | Tick<br>Your choice |
| 21. | Do you understand what transaction is? | o Yes<br>o No |
| 22. | Do you know concurrency control mechanism? | o Yes<br>o No |
| 23. | Do you know the various factors of recovery? | o Yes<br>o No |
| 24. | Do you understand serializability? | o Yes |

| | | |
|---|---|---|
| | | ○ No |
| 25. | Do you understand module ? | ○ Yes, Completely.<br>○ Partialy.<br>○ No, Not at all. |