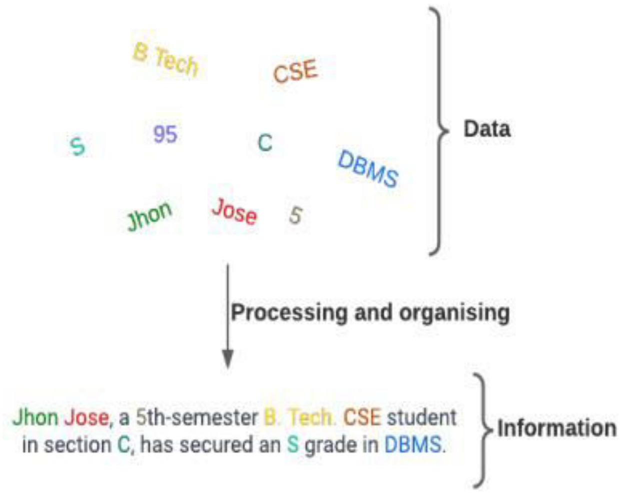
DBMS CHECKLIST/NOTES

DBMS UNIT-1

- data: raw and unorganized

- information: structured and organized


B Tech
CSE
S
95
C
DBMS
Jhon
Jose
5
Data
Processing and organising
Jhon Jose, a 5th-semester B. Tech. CSE student in section C, has secured an S grade in DBMS.
Information

- database: collection of related data, logically coherent

- database system: collection of interrelated data and a set of programs that allow users to access and modify these data

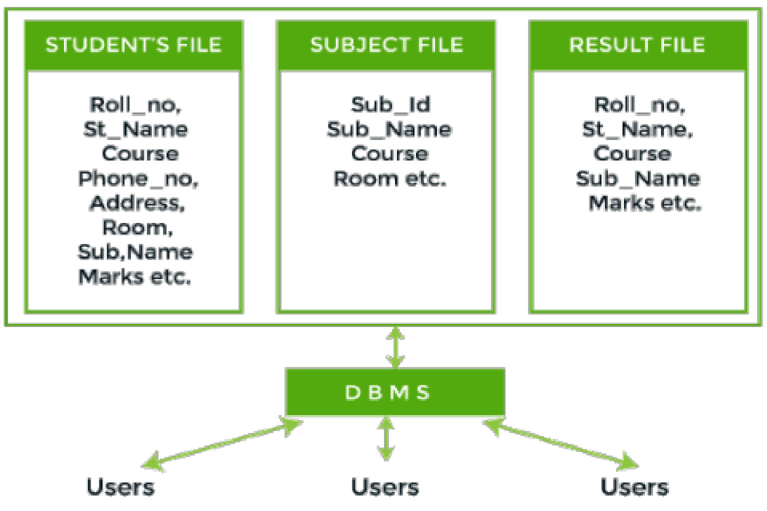
- dbms: define, construct, manipulate and share database

- why db

- applications

- how are db used: OLTP and data analysis

- File-processing Systems vs DBMS


STUDENT'S FILE
Roll_no, St_Name Course Phone_no, Address, Room, Sub,Name Marks etc.
SUBJECT FILE
Sub_Id Sub_Name Course Room etc.
RESULT FILE
Roll_no, St_Name, Course Sub_Name Marks etc.
D B M S
Users
Users
Users

DBMS APPROACH

- purpose:

  1. Data Redundancy & Inconsistency: centralized location
  2. Difficulty in accessing the data: use powerful query language
  3. Data Isolation
  4. Integrity problems: automatic constraint updation (bank balance <0 not possible)
  5. Atomicity problems: automatic rollback in case of failure
  6. Concurrent Access Anomalies: transaction and locking
  7. Security problems: administrator to give permissions

- Characteristics of Database Systems: (SISS)

  - Self-describing nature of a database system: metadata
  - Insulation between programs and data, and data abstraction
  - Support multiple views of the data
  - Sharing of data and multiuser transaction processing

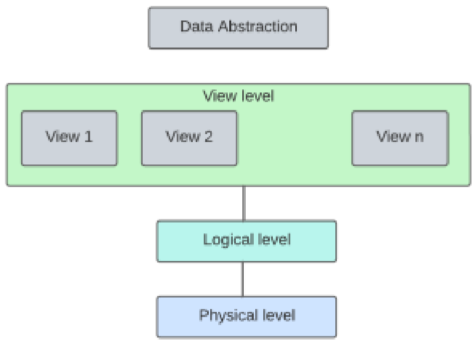
- Advantages of Database Systems

- Data abstraction: Hide the complexity of data structures from users

    - Physical Level

    - Logical Level

    - View Level



- PHYSICAL DATA INDEPENDENCE: able to modify the physical level without any alterations to the logical level.

- LOGICAL DATA INDEPENDENCE: able to modify the logical level without any alterations to the view level.

- Data Models:

    1. RELATIONAL MODEL-> tables

    2. Entity-Relationship Model-> database design

    3. Semi Structured Data Model-> JSON and XML

    4. Object-Based Data Model

- schema: overall design of the database

- instance: information stored at a particular moment

- Database Languages:

    1. DDL: Definition-> create, alter, drop, trauncate, rename, comment

        - Domain constraints

        - Referential integrity

        - Authorization: read, insert, update, delete

    2. DML: Manipulation-> select, insert, update, delete

        - Procedural DML: what and how?

        - Declarative DML: only what, not how. ex: SQL

        - query processor translates DML queries into sequence of actions at the physical level

    3. DCL: Control-> grant, revoke

    4. TCL: Transaction control-> commit, rollback

- DROP removes the entire table, DELETE removes specific rows, and TRUNCATE removes all rows but keeps the table structure.

- SQL does not support input from users, output to displays, or communication over the network.

- Database Access from Application Programs: ODBC/JDBC

- Database design:

    1. User requirement specification: understanding data needs, interaction w users and domain experts

    2. Conceptual design phase: choose data model(high-level), group attributes together-> ER model/Normalization

    3. Logical design phase: map high-level schema to implementation model

    4. Physical design phase: internal storage structures

- database engine(functional components):

    1. storage manager: provides interface between low-level data and application programs, interaction with OS file manager, efficient storing and retrieval of data

- Authorization and integrity manager
- Transaction manager
- File manager
- Buffer manager

2. query processor component: work at view level without understanding physical (parsing and translation, optimization, evaluation)

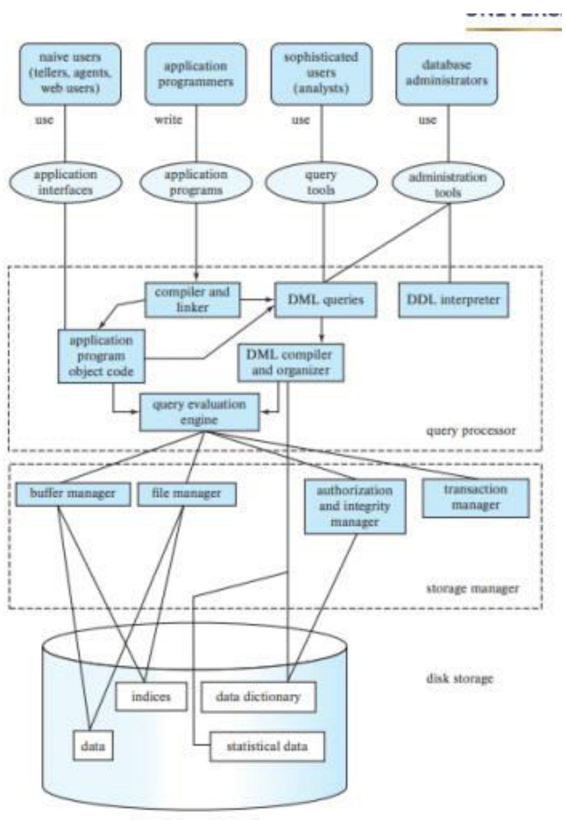
   - DDL interpreter
   - DML compiler-> choose lowest cost evaluation plan
   - Query evaluation engine

3. transaction management component: consistent state despite failures

   - recovery manager
   - concurrency control manager

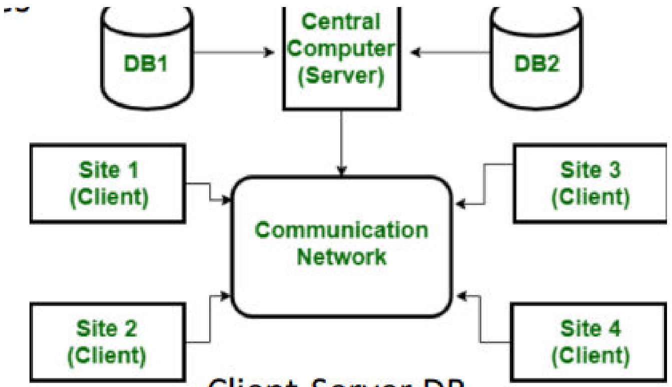
- Database Architecture:

1. centralized database: common shared memory, limited scalability



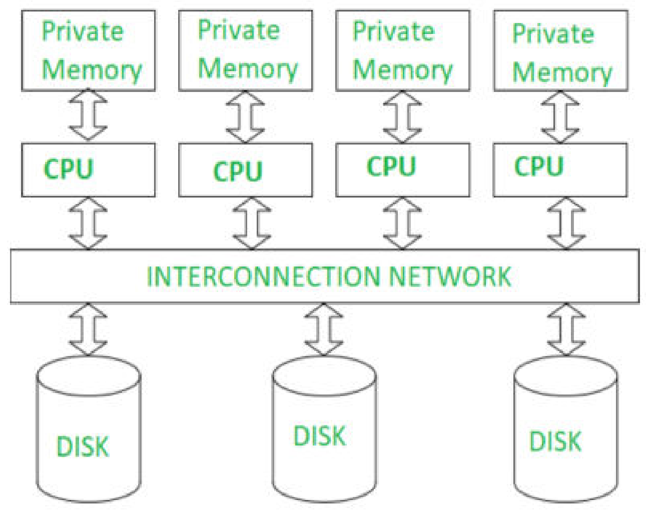
Centralized/Shared Memory DB

2. client-server: one server machine executes on behalf of multiple clients



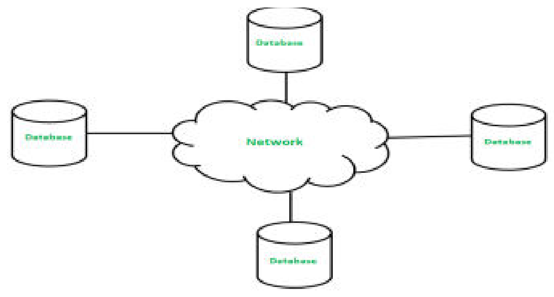
Client-Server DB

3. parallel databases: runs on a cluster of multiple machines, shared disk, better scalability



Parallel DB

4. distributed databases - data heterogeneity, geographically separated machines



- application architecture: Two Tier Architecture, Three Tier Architecture

1. Database users

    - naive users use application interfaces (inexperienced)

    - Application programmers use application programs (pro)

    - Sophisticated users use query tools (analyst)

2. Database administrator (DBA) use administration tools-> Schema definition, Storage structure and access-method definition, Schema and physical-organization modification, Granting authorization for data access, Routine maintenance

- RELATIONAL MODEL:

    - relation-> table, tuple-> row, attribute-> column

- order of arrangement doesn't matter: relation is a set of tuples
- domain: set of permitted values
- atomic values in domain: no multiple phoen numbers
- null values: missing/not known/ not applicable

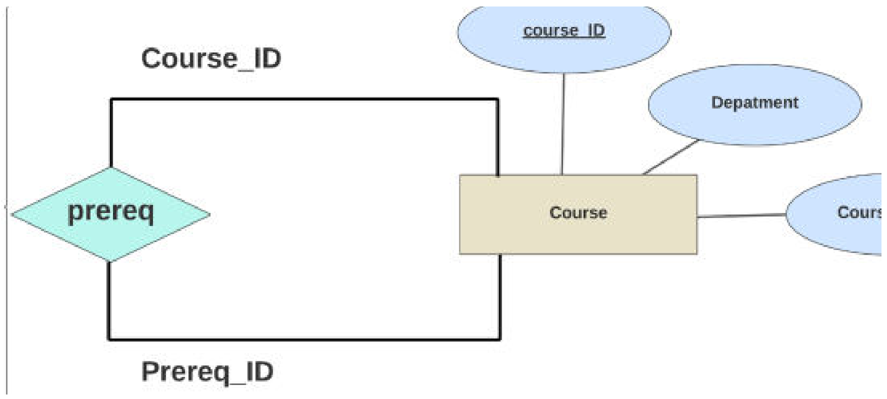
- relational schema: R = (A1, A2, ..., An)
- constraints:
    1. Implicit Constraints
    2. Explicit Constraints
        • Domain constraint - atomic/null value
        • Key constraints and Constraints on NULL values

        -no two tuples in a relation are allowed to have exactly the same value for all attributes.

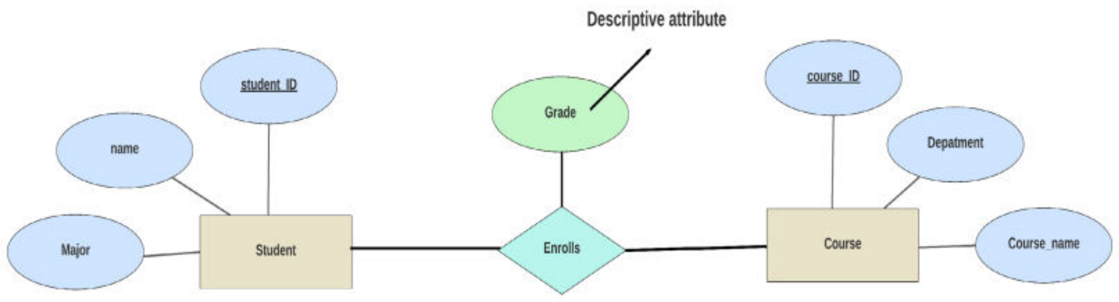
        - superkey, candidate key: minimal superkey, primary key: random CK

        • Integrity Constraints
            • Entity integrity constraints: PK cannot have null values
            • Referential integrity constraint: FK
    3. semantic constraints
- relational database state can be valid/ invalid.
- entity: distinctly identifiable items
- major pitfalls while designing database: redundancy and incompleteness
- ER Model: technique designed to assist in database design
- entity: distinct thing/object, can be concrete(people) or abstract(course)
- Non-Disjoint Entity Sets: overlapping
- Relationship: connections between entities.

- Recursive Relationship

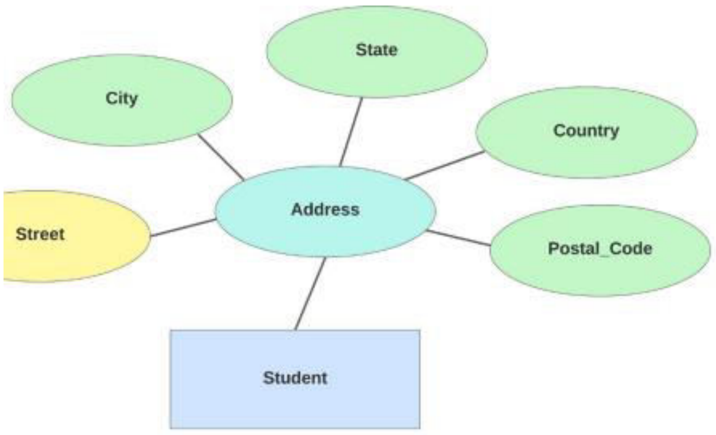


- Descriptive Attribute: of relationship

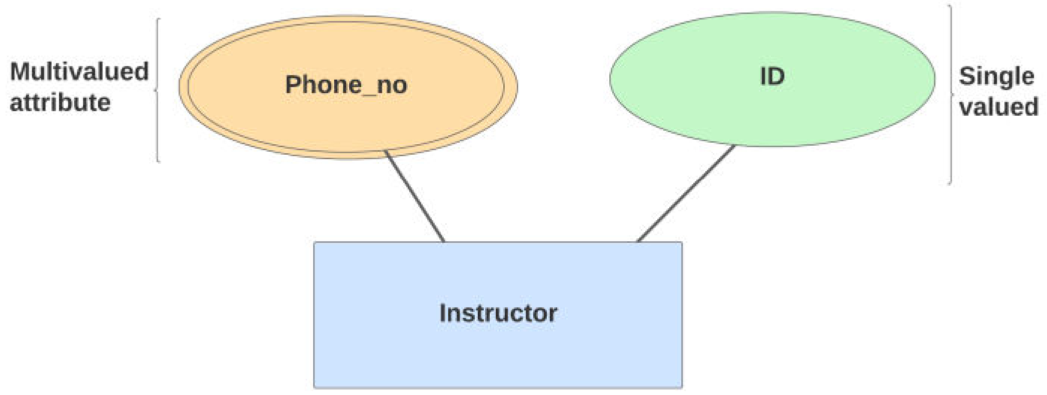


- The number of entity sets that participate in a relationship set is the degree of the relationship set.

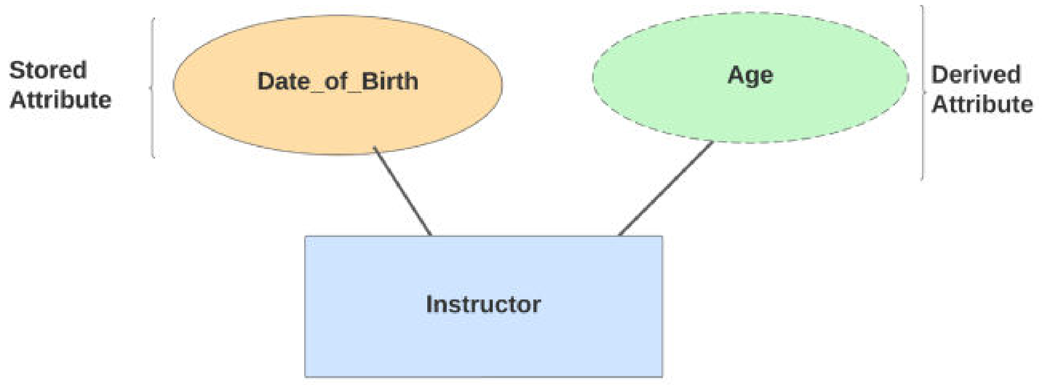
• Simple / Composite attribute: split into components



• Single-valued / Multivalued attributes: takes up more than a single value



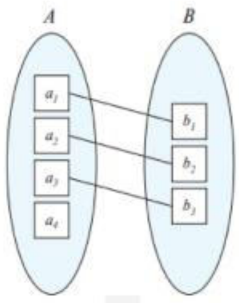
• Derived attributes:



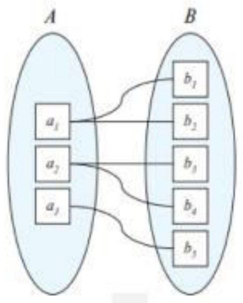
• Complex Attributes: nesting of composite and multi-valued attributes

- Cardinality Ratio: the number of entities to which another entity can be associated via a relationship

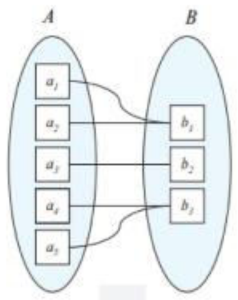
  • One-to-one
  • One-to-many
  • Many-to-one
  • Many-to-many



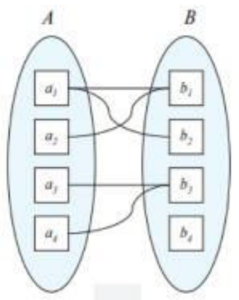
One-to-one



One-to-many



Many-to-one



Many-to-many

• A directed line (→) {one}

• An undirected line (—) {many}

• Total participation (double line)

• Partial participation

- l..h: l is the minimum cardinality, h is the maximum cardinality.

- choice of PK:

    • One-to-one: PK->either many or one

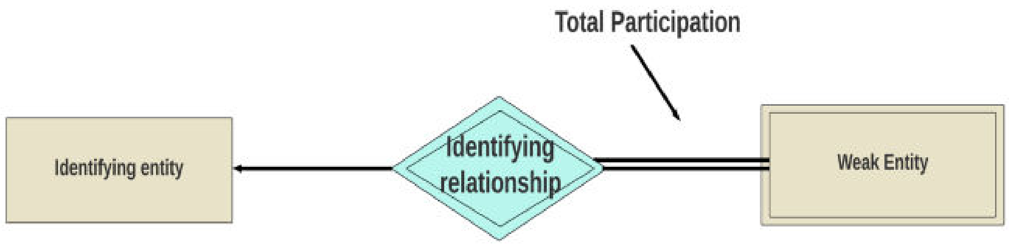
    • One-to-many/ Many-to-one: PK->use many side

    • Many-to-many: PK->union of both sets (no reduction)

- to avoid confusion, we permit only one arrow out of a nonbinary relationship set.

- weak entity set: whose existence is dependent on another entity set called identifying entity

    • One-to-many/ Many-to-one: many-> weak entity, one-> identifying entity



PK: (identifying) PK + (weak entity) discriminator

drawing ER diagram:

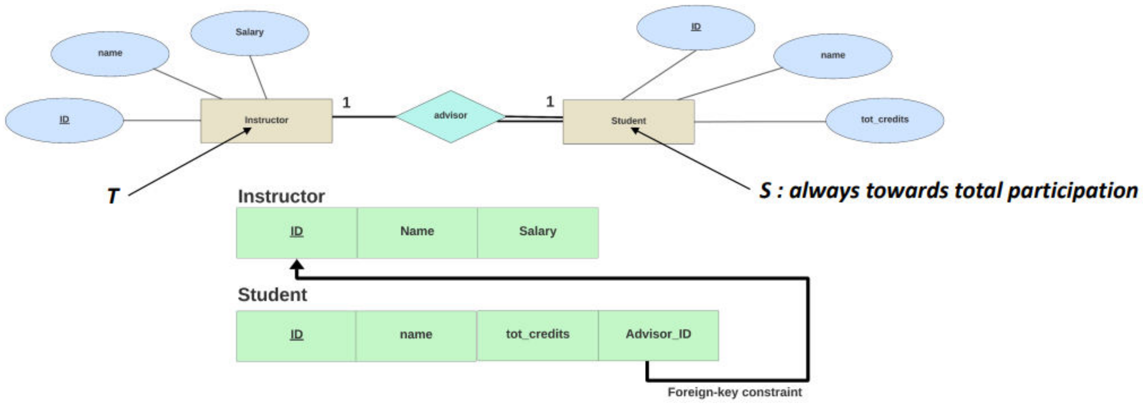
- Entities & attributes:
  - Classroom (**building**, room number, capacity).
  - Department (**dept name**, building, budget).
  - Course (**course id**, title, credits).
  - Instructor (**ID**, name, salary).
  - Section (sec_id, semester, year). → weak entity with all attributes as descriptive
  - Student (**ID**, name, tot cred).
  - Time slot (**time slot id**, {(day, start time, end time) }).

- Relationships:
  - Inst_dept: relating instructors with departments.
  - Stud_dept: relating students with departments.
  - Teaches: relating instructors with sections.
  - Takes: relating students with sections, with a descriptive attribute grade.
  - Course_dept: relating courses with departments.
  - Sec_course: relating sections with courses. → identifying relationship
  - Sec_class: relating sections with classrooms.
  - Sec_time_slot: relating sections with time slots.
  - Advisor: relating students with instructors.
  - Prereq: relating courses with prerequisite courses.

- Constraints:
  - Each instructor must have exactly one associated department.
  - Every course must be in some department
  - Every student must be majoring in some department
  - Every course and every student can be related to only one department, not several
  - each student has at most one advisor

- Mapping of Binary 1:1 Relation Types:

  - Foreign Key (2 relations) approach:



- Merged relation (1 relation) option
- Cross-reference or relationship relation (3 relations) option

- Mapping of Binary 1:N Relation Types

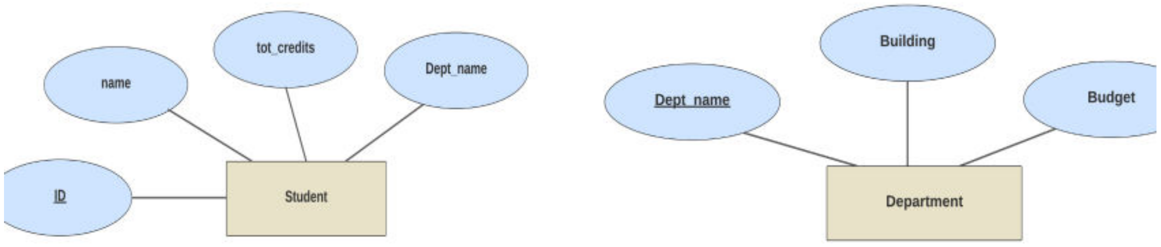
- Mapping of Binary M:N Relation Types

- converting er to relational schema:

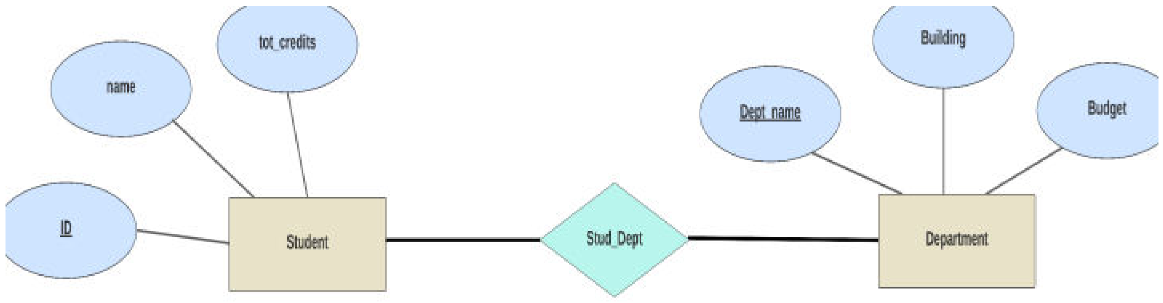
    1. strong entities
    2. weak entities
    3. map 1:1
    4. map 1:n
    5. map n:m
    6. map n-ary

- common mistakes in er diagram

    1. Incorrect Use of Primary Key

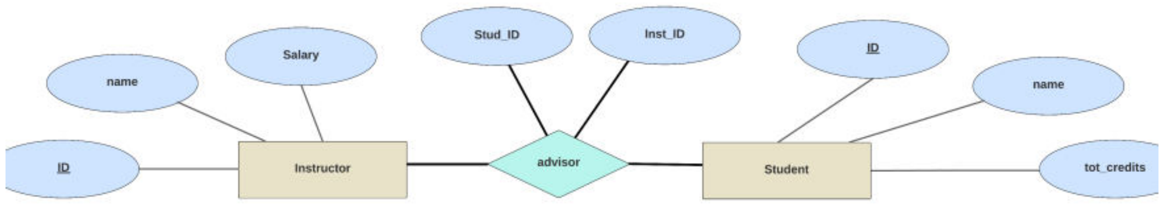


Error : student has Dept_name which is a PK of department

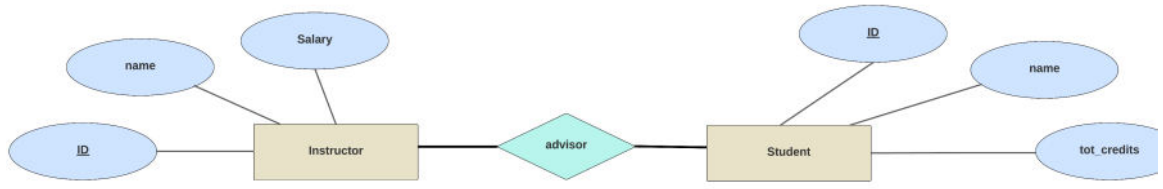


Solution : Using relationship stud_dept

2. Unnecessary Inclusion of Primary Key Attributes

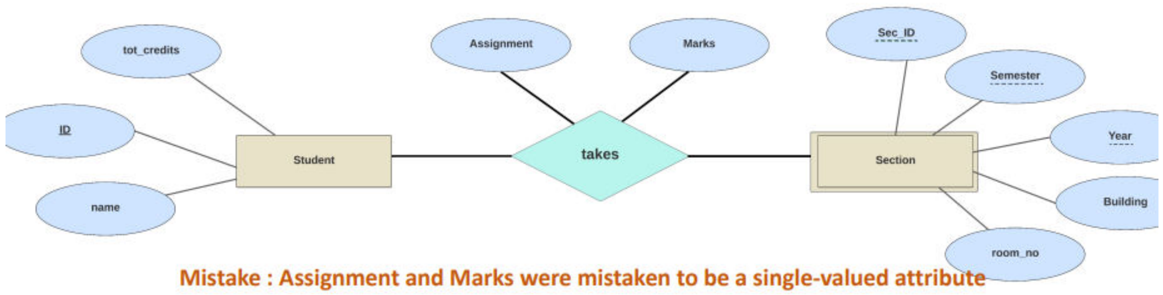


Mistake : PK of both entities as relational attributes

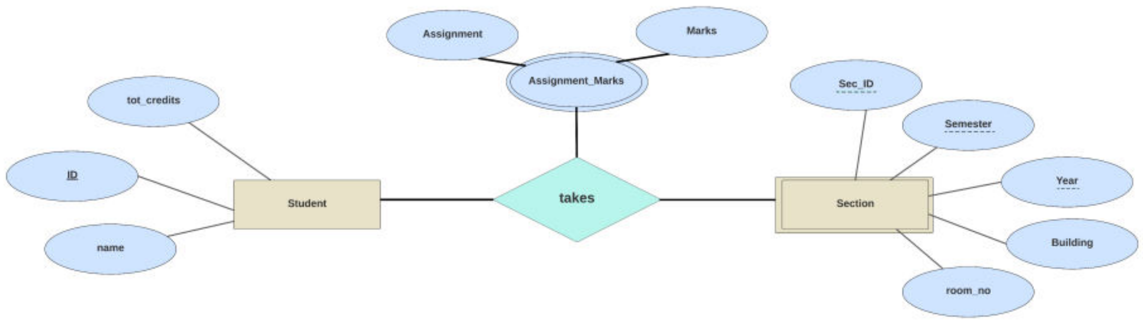


Corrected version of the advisor relationship

3. Misuse of Single-Valued Attributes

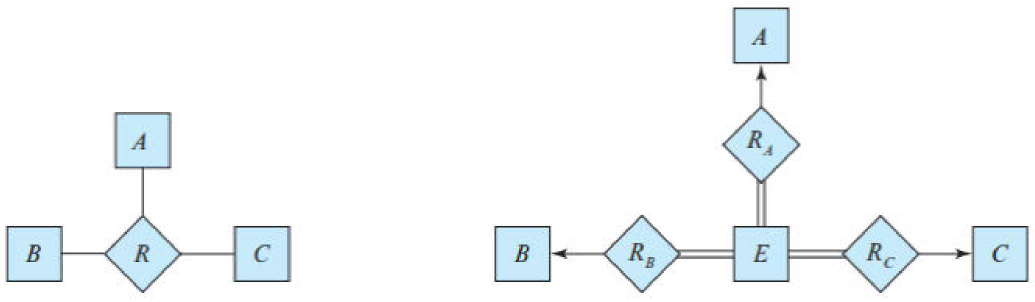


Mistake : Assignment and Marks were mistaken to be a single-valued attribute



4. Managing Large E-R Diagrams

- replace n-ary by binary when desired



(a)
Ternary relation

(b)
3 binary relations

DBMS UNIT-2

query language: user requests information from the database

- Imperative query language: user explicitly specifies a sequence of operations to be performed on the database.
- functional query language: computations are expressed as the evaluation of functions.
- declarative query language: users describe the desired information without specifying a sequence of steps or functions for obtaining that information.

Unary Operators:

Select: σ

Project: $\prod$

Rename: ρ

Binary Operator:

Union: U

Intersection :∩

Set difference: –

Cartesian product: X

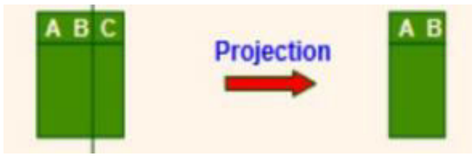
Join: ⋈

The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

SELECT -filter rows that satisfy a condition

PROJECT-unary operation that returns its argument relation, with certain attributes left out.

creates a vertical partitioning



PROJECT is *not* commutative

- $\pi_{<list1>} (\pi_{<list2>} (R)) = \pi_{<list1>} (R)$ as long as <list2> contains the attributes in <list1>

no duplicate tuples

The join operation allows us to combine a select operation and a Cartesian-Product operation into a single operation union-or

duplicate tuples eliminated in union

intersection- and

NOTE: Type Compatibility of operands is required for the binary set operation UNION U, (also for INTERSECTION ∩, and SET DIFFERENCE –)

There are five types of aggregate functions

1. AVERAGE

2. MINIMUM

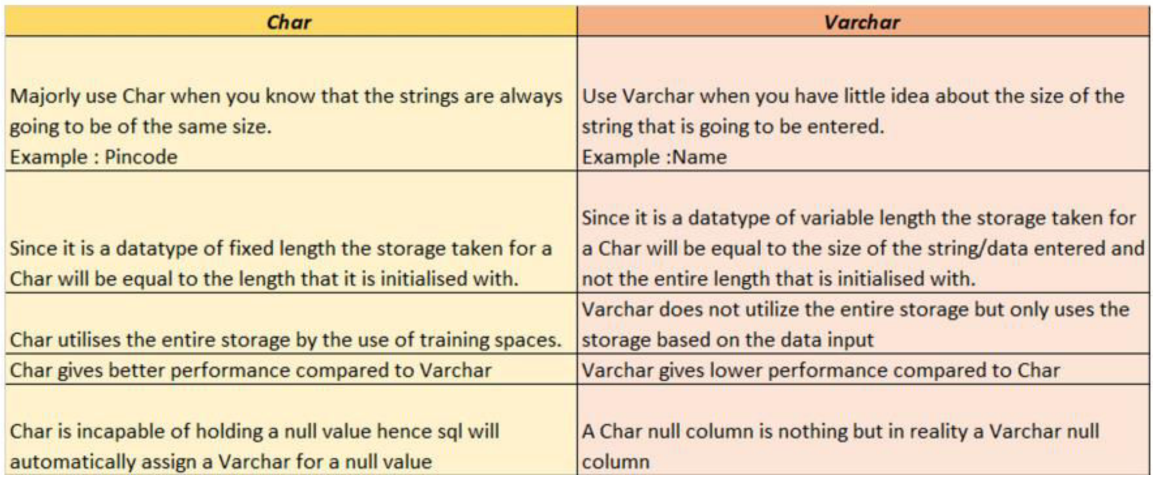
3. MAXIMUM

4. SUM

5. COUNT

• Grouping attribute placed to left of symbol

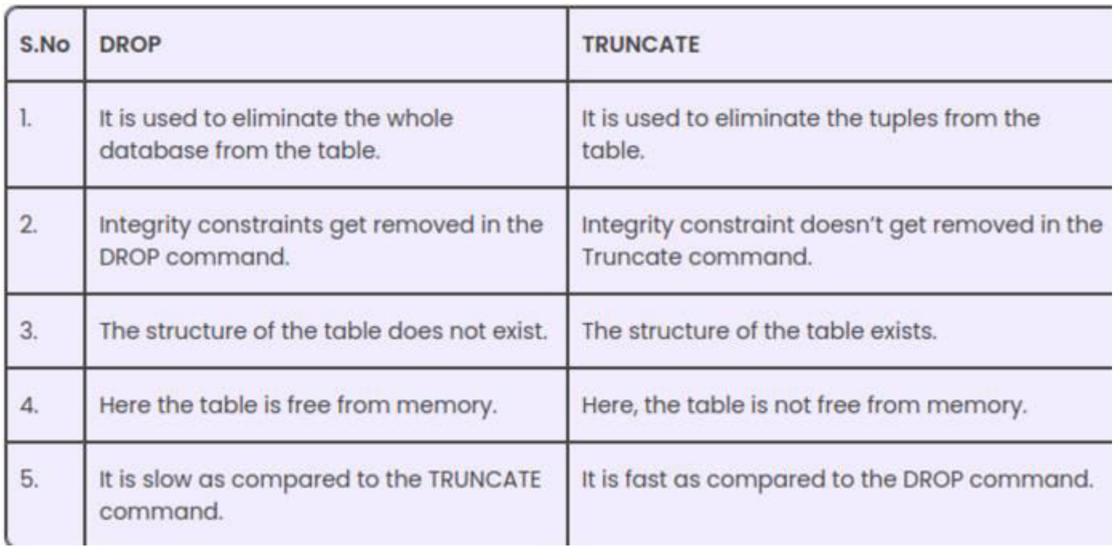
• Aggregate functions to right of symbol



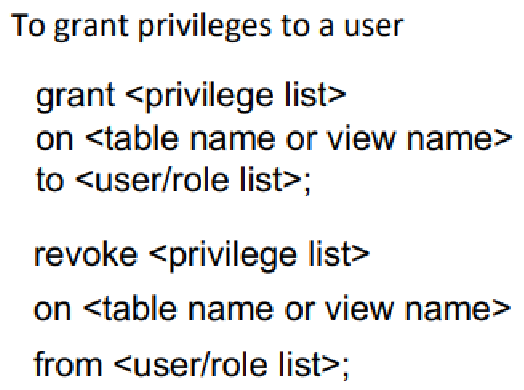
Char	*Varchar*
Majorly use Char when you know that the strings are always going to be of the same size. Example : Pincode	Use Varchar when you have little idea about the size of the string that is going to be entered. Example :Name
Since it is a datatype of fixed length the storage taken for a Char will be equal to the length that it is initialised with.	Since it is a datatype of variable length the storage taken for a Char will be equal to the size of the string/data entered and not the entire length that is initialised with.
Char utilises the entire storage by the use of training spaces.	Varchar does not utilize the entire storage but only uses the storage based on the data input
Char gives better performance compared to Varchar	Varchar gives lower performance compared to Char
Char is incapable of holding a null value hence sql will automatically assign a Varchar for a null value	A Char null column is nothing but in reality a Varchar null column

The table with the foreign key is called the child table, and the table with primary key is called the referenced or parent table.

SQL: practical rendering of the relational data model with syntax



S.No	DROP	TRUNCATE
1.	It is used to eliminate the whole database from the table.	It is used to eliminate the tuples from the table.
2.	Integrity constraints get removed in the DROP command.	Integrity constraint doesn't get removed in the Truncate command.
3.	The structure of the table does not exist.	The structure of the table exists.
4.	Here the table is free from memory.	Here, the table is not free from memory.
5.	It is slow as compared to the TRUNCATE command.	It is fast as compared to the DROP command.



To grant privileges to a user

```
grant <privilege list>
on <table name or view name>
to <user/role list>;

revoke <privilege list>
on <table name or view name>
from <user/role list>;
```

The letters “lob” in these data types stand for “Large OBject."

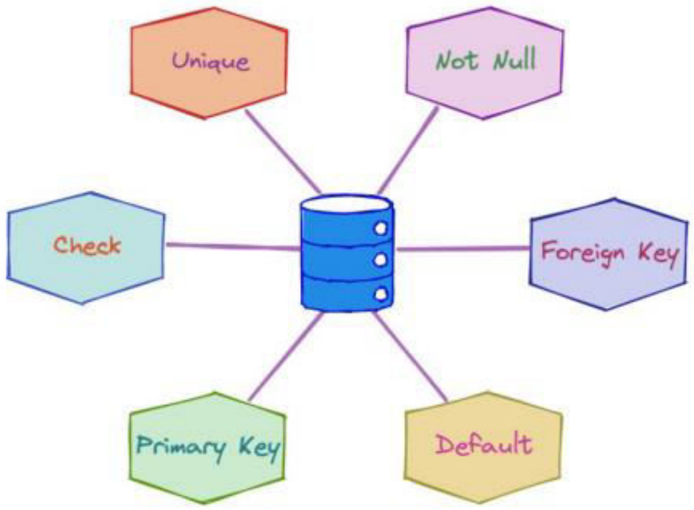
• BLOBs (Binary LOBs) used to store unstructured binary (also called “raw”) data, such as video clips.

● Using this you can stores files like videos, images, gifs, and audio files.

• CLOBs (Character LOBs) used to store large blocks of character data from the database character set

● Using this you can store files like text files, PDF documents, word documents etc

constraints:



keyword all to specify explicitly that duplicates are not removed

keyword distinct to specify no duplicates in result

Types of Set Operation

1.Union: eliminates duplicate rows from its result set

2.Union All: returns the set without removing duplication and sorting the data (c1+c2)

3.Intersect

4.Intersect All: min(c1,c2)

5.Except

6.Except All: max(c1-c2,0)

Built-in aggregate functions:

• COUNT - returns the number of tuples or values specified in a query

• SUM – returns the sum of a set (or multiset) of numeric values

• MAX – returns the maximum value from a set(or multiset) of numeric values

• MIN - returns the minimum value from a set(or multiset) of numeric values

• AVG – returns the average of a set(or multiset) of numeric values

SQL does not allow the usage of DISTINCT along with COUNT(*)

Aggregate functions in nested queries: SOME and ALL

-NESTED QUERIES

Some Important Keywords Related Nested Query

Ø In

Ø Not In

Ø All

Ø Any

Ø Some

Ø Exist

Ø Not Exist

- Find all customers who have both a loan and an account.'

select distinct customer_name from borrower where customer_name in (select customer_name from depositer)

- Find all customers who have both an account and a loan at the 'Perryridge' branch.

SELECT DISTINCT C.customer_name

FROM Customers C

JOIN Accounts A ON C.customer_id = A.customer_id

JOIN Loans L ON C.customer_id = L.customer_id

WHERE A.branch_name = 'Perryridge' AND L.branch_name ='Perryridge';

- Find all customer who do have a loan at the bank, but do not have an account at the bank

select distinct customer_name from borrower where customer_name not in (select customer_name from depositer)

- Select names of customer who have a loan at the bank and where names are neither "Smith" nor "Jones".

- Find the names of all the branches that have assets greater than those of at least one branch located in “Brooklyn”

select branch_name from branch where assets> some(select assets from branch where branch_city="Brooklyn")

MySQL doesn't support FULL OUTER JOIN directly.

DBMS UNIT-3

■ Informal Design Guidelines for Relational Databases

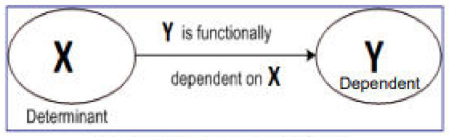
❖ Making sure that the semantics of the attributes is clear in the schema

❖ Reducing the redundant information in tuples

❖ Reducing the NULL values in tuples

❖ Disallowing the possibility of generating spurious tuples: satisfy the lossless join condition.

■ Functional Dependencies (FDs)



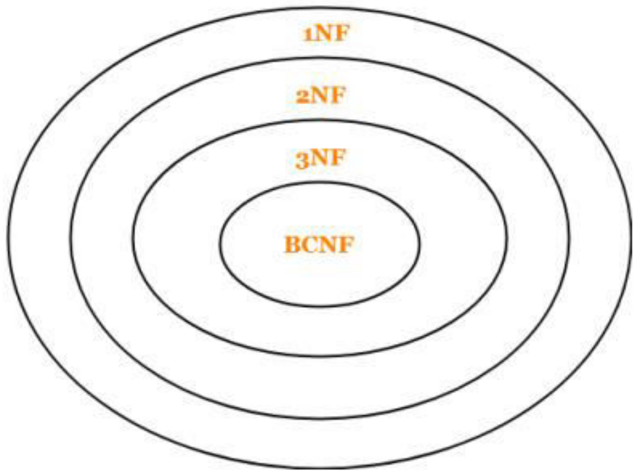
Functional dependency between X and Y

■ Normal Forms (1NF, 2NF, 3NF)Based on Primary Keys

■ General Normal Form Definitions of 2NF and 3NF (For Multiple Keys)

■ BCNF (Boyce-Codd Normal Form)

■ Fourth and Fifth Normal Forms



2NF- NO (LHS: PART OF CK AND RHS: NON PRIME ATTRIBUTE)

3NF-LHS:CK/SK OR RHS:PRIME ATTRIBUTE

BCNF: LHS:CK/SK

NON ADDITIVIE JOIN: ((R1 ∩ R2) → (R1- R2)) is in F+ OR

((R1 ∩ R2) → (R2 - R1)) is in F+.

BCNF decomposition is always lossless but not always dependency preserving

transactionsCollections of operations that form a single logical unit of work

Two main issues to deal with:

• Failures of various kinds, such as hardware failures and system crashes

• Concurrent execution of multiple transactions

ACID Properties

❑Transaction Atomicity and Durability

shadow-database scheme,

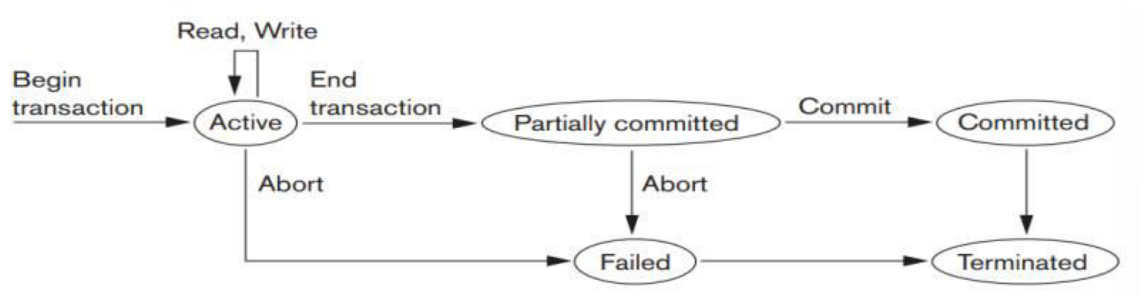
• A transaction that wants to update the database first creates a complete copy of the database.

• 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 simply deletes the new

❏ Transaction State: active, partially committed, failed, aborted, committed

state diagram:



❏ System Logkeeps track of transaction operations

❏Transaction Isolation: run serially

❏ Concurrent Executions

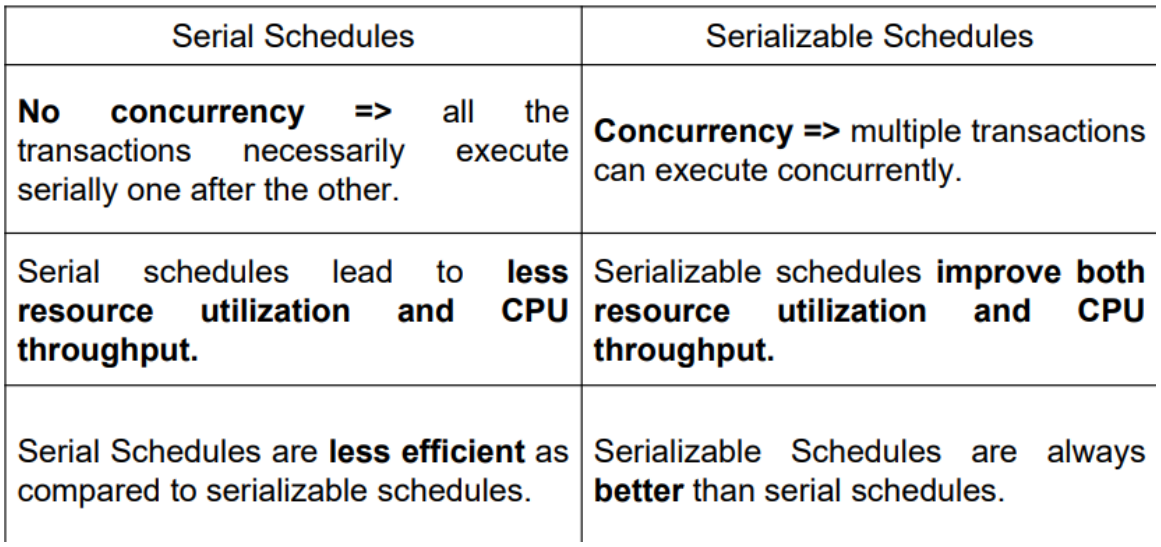
Improved throughput and resource utilization

Reduced waiting time

❏ Schedules: a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed

❏Serializability: If a given non-serial schedule of 'n' transactions is equivalent to some serialschedule of 'n' transactions, then it is called as a serializable schedule

Serial schedules are always Consistent



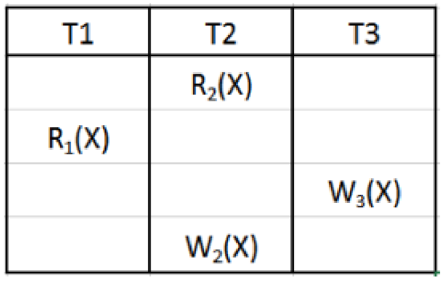
Serial Schedules	Serializable Schedules
No concurrency => all the transactions necessarily execute serially one after the other.	**Concurrency =>** multiple transactions can execute concurrently.
Serial schedules lead to **less resource utilization and CPU throughput.**	Serializable schedules **improve both resource utilization and CPU throughput.**
Serial Schedules are **less efficient** as compared to serializable schedules.	Serializable Schedules are always **better** than serial schedules.

❑ Conflicting Instructions

❑ Conflict Serializability: If a given non-serial schedule can be converted into a serialschedule by swapping its non-conflicting operations

loop/cycle in precedence graph-> not conflict serializable

View Serializability: Two schedules are said to be view equivalent if the order of initial read, final writeand update operations is the same in both the schedules.



T1	T2	T3
	$R_2(X)$	
$R_1(X)$		
		$W_3(X)$
	$W_2(X)$	

W3(X) IS BLIND WRITE

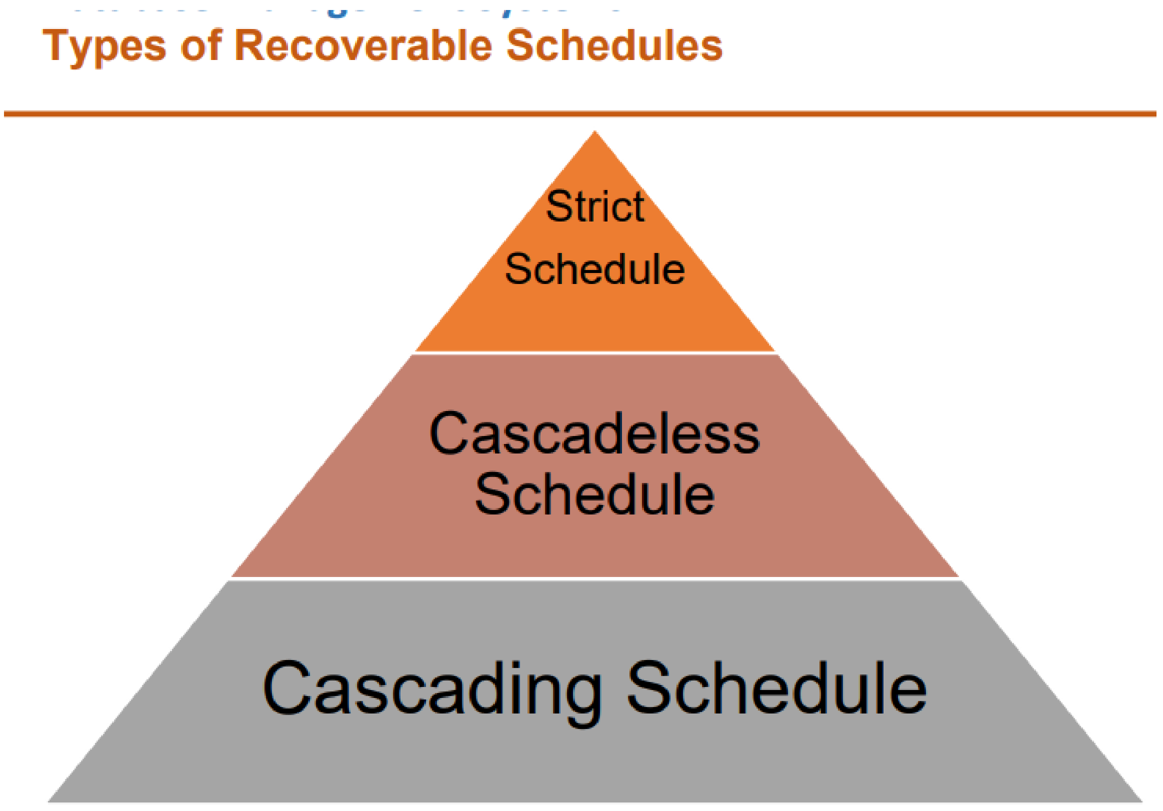
Recoverable Schedules: if a transaction Tj reads a data item previously written by a transaction Ti , then the commit operation of Ti appears before the commit operation of Tj

Cascading Rollbacks: a single transaction failure leads to a series of transaction rollbacks.

Cascadeless Schedule: cascading rollbacks cannot occur

Reading from an uncommitted transaction is called dirty read

Checking Whether a Schedule is Recoverable or Irrecoverable: check seq of dirty read == sequence from precedence graph(commit)



Types of Recoverable Schedules

Cascadeless schedule allows only committed readoperations.

Strict Schedule: a transaction is neither allowed to read nor write data item until the last transaction that has written is committed or aborted

❑ What is Concurrency Control: procedure of managing simultaneous transactions ensuring their atomicity, isolation, consistency, and serializability

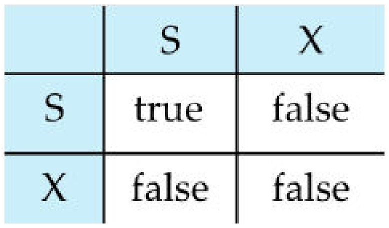
❑ Lock-Based Protocols

shared exclusive locking:

❑ Exclusive (X) mode : Data item can be both read as well as written. X-lock is requested using lock-X instruction.

❑ Shared (S) mode. Data item can only be read. S-lock is requested using lock-S instruction.

lock compatibility matrix:



	S	X
S	true	false
X	false	false

❑ The Two-Phase Locking Protocol: growing and shrinking phase

Once the transaction releases a lock, it enters the shrinking phase, and it can issue no more lock requests

to remove irrecoverability and cascading rollback problem:

1. Strict two-phase locking: a transaction must hold all its exclusive locks till it commits/aborts.

2. Rigorous two-phase locking: a transaction must hold all locks till commit/abort (shared and exclusive)

conservative 2 PL removes deadlock problem also

Transactions can be serialized in the order in which they commit.

❑ Lock Conversions: upgrade and downgrade

❑ Implementation of Locking

The lock manager maintains an in-memory data-structure called a lock table to record granted locks and pending requests

❑ Graph-Based Protocols: restricted to employ only exclusive locks

If di → dj then any transaction accessing both di and dj must access di before accessing dj.

❑ Tree-Based Protocols

❑ Deadlocks

❑ Deadlock Prevention

-Waitdie

If TS(Ti) < TS(Tj), then (Ti older than Tj) Ti is allowed to wait; otherwise (Ti younger than Tj) abort/rollback Ti (Ti dies) and restart it later with the different timestamp.It is a non-preemptive technique

- Woundwait.

If TS(Ti) < TS(Tj), then (Ti older than Tj) abort Tj (Ti wounds Tj) and restart it later with the same timestamp; otherwise (Ti younger than Tj) Ti is allowed to wait.It is a preemptive technique.

❑ Deadlock Detection and Deadlock Recovery: if cycle->deadlock

❑ Problems

1) Transaction Failure:

There are two types of errors that may cause a transaction to fail:

- Logical error
- System error

Fail-stop assumption: The assumption that hardware errors and bugs in the software bring the system to a halt, but do not corrupt the non-volatile storage contents

1. Volatile storage
2. Non-Volatile storage
3. Stable storage

➢ Recovery Atomicity

➢ Recovery Algorithm

We represent an update log record as <Ti, Xj, V1, V2>, indicating that transaction Ti has performed a write on data item Xj. Xj had value V1 before the write and has value V2 after the write

Two approaches using logs:

• Immediate database modification

• Deferred database modification

• The undo operation using a log record sets the data item specified in the log record to the old value contained in the log record.

• The redo operation using a log record sets the data item specified in the log record to the new value contained in the log record.

Transaction Ti needs to be undone if the log Contains the record

<Ti start>,But does not contain either the record <Ti commit> or <Ti abort>

Transaction Ti needs to be redone if the logContains the records <Ti start>And contains the record <Ti commit> or <Ti abort>

Threats to Database

❑ Loss of integrity

❑ Loss of availability

❑ Loss of confidentiality

Control Measures: AIFD-> access Control, Inference Control, Flow Control, Data Encryption

ACTIONS: ❑ Account creation. This action creates a new account and password for user or a group of users to enable access to the DBMS.

❑ Privilege granting. This action permits the DBA to grant certain privilegesto certain accounts.

❑ Privilege revocation. This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.

❑ Security level assignment. This action consists of assigning user accounts to the appropriate security clearance level.

1. Access Control, User Accounts, and Database Audits

2. Sensitive Data and Types of Disclosures: Inherently sensitive, From a sensitive source, From a sensitive source, A sensitive attribute or sensitive record, Sensitive in relation to previously disclosed data.

factors must be considered before deciding whether it is safe to reveal the data: ❑ Data availability

❑ Access acceptability

❑ Authenticity assurance

Relationship between Information Security and Information Privacy

Discretionary Access Control: the owner can determine the access and privileges and can restrict the resources basedon the identity of the users

mandatory access control: the system only determines the access and the resources will berestricted based on the clearance of the subjects

minimal cover: rhs into single, redundant attributes, extraneous (lhs)

DBMS UNIT-4

• Why python with MySQL: supports SQL cursors

• Installing Mysql connector and establishing a connection

    pip install mysql-connector-python

    4 parameters, host, user, password and an optional database

CODE:

```
import mysql.conector
mydb = mysql.conector.connect(
        host="localhost",
        user="root",
        password="root",
        database="little_lemon"
)
c = mydb.cursor()
query="select * from customers"
c.execute(query)
results=c.fetchall()
def add_data(cust_id,name):
        c.execute('insert into customer(cust_id,name)
        values (%s,%s)',(001,nishank))
        mydb.commit()
c.close()
mydb.close()
```

- Executing Sql queries
- Retrieving and handling results
- Building a Python-MySQL application
- Web Interfaces and Hypertext Documents
- XML for Data Exchange (html vs xml)
- Why XML(self-describing docs) over HTML: in text files, dynamic
- Dynamic Web Pages and its importance
- Structured, Semistructured and Unstructured data

• Semistructured Data Example: Directed Graph

• Unstructured Data

• Comparisons

• XML Hierarchical (Tree) Data Model-> internal nodes represent complex elements, whereas leaf nodes represent simple elements.

• Types of XML documents: data centric, document centric, hybrid

NoSQL databases: document-> mongodb, couchdb

key-value-> redis, dynamodb

wide-column-> cassandra, hbase

graph-> neo4j, ibm graph

Characteristics of NoSQL Systems:

• those related to distributed databases and distributed systems-> Horizontal scalability

Availability, Replication and Eventual Consistency

Replication Models: master-slave and master-master

Sharding: load balancing

High-Performance Data Access: hashing

• those related to data models and query languages:

Not Requiring a Schema

Less Powerful Query Languages

Versioning

CAP Theorem: consistency (among replicated copies), availability (of the system for read and write operations) and partition tolerance (in the face of the nodes in the system being partitioned by a network fault)

"It is not possible to guarantee all three of the desirable properties—consistency, availability, and partition tolerance—at the same time in a distributed system with data replication".

■ Document-based NOSQL systems

■ MongoDB data model: ad-hoc fashion query, BSON format, collection

■ RDBMS vs MongoDB

■ MongoDB CRUD Operations

db.createCollection("person")

db.person.insertOne({name:"Nishank", age:20})

db.person.insertMany([{likes:"Maggie"},{likes:"Oreo"}])

db.person.find(query,projection)

db.person.updateOne($set)

Format: db.createCollection(name, options)

db.<collection_name>.insert(<document(s)>)

The parameters of the insert operation can include either a single document or an array of documents.

db.<collection_name>.find(<condition>)

To display only necessary data from documents in the collection

Format: db.collection_name.find(selection_criteria, fields required)

use $set in update:

Format: db.collection_name.update(selection_criteria, update_value)

to delete:

db.<collection_name>.remove(<condition>)

- Insert a document with the following fields and values into this collection:

```
title: "G20 Summit",
body: "The summit addressed pressing geopolitical challenges.",
category: "News",
likes: 15
```

db.posts.insertOne({title: "G20 Summit",body: "The summit addressed pressing geopolitical challenges.",category: "News",likes: 15 }

- Retrieve the posts where category is "News"

db.posts.find({category:"News"})

- Retrieve information of only title and body of all posts

db.posts.find({},{title:1,body:1,_id:0})

- Set the number of likes on the post "Benefits of Blockchain" to 8

db.posts.updateOne({title: "Benefits of Blockchain"},{$set:{likes:8}}

■ MongoDB Distributed System Characteristics

Horizontal scaling: adding machines to share the data set and load.

Sharding is a method for distributing or partitioning data across multiple machines. ➢ Range partitioning, ➢ Hash partitioning

■ Replication in MongoDB

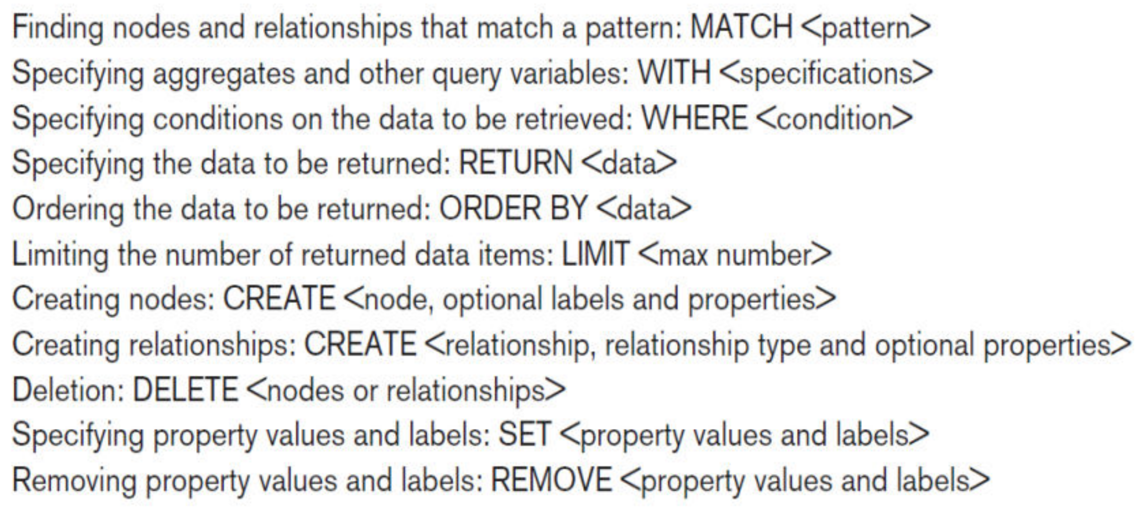
The total number of participants in a replica set must be at least three

■ Sharding in MongoDB: query router

Key-Value Database(DynamoDB)

Voldemort- consistent hashing - linkedin - get.put.delete

● Neo4j has a high-level query language, Cypher.



Finding nodes and relationships that match a pattern: MATCH <pattern>
Specifying aggregates and other query variables: WITH <specifications>
Specifying conditions on the data to be retrieved: WHERE <condition>
Specifying the data to be returned: RETURN <data>
Ordering the data to be returned: ORDER BY <data>
Limiting the number of returned data items: LIMIT <max number>
Creating nodes: CREATE <node, optional labels and properties>
Creating relationships: CREATE <relationship, relationship type and optional properties>
Deletion: DELETE <nodes or relationships>
Specifying property values and labels: SET <property values and labels>
Removing property values and labels: REMOVE <property values and labels>

- Query Processing

1. Parsing and translation

• Translate the query into its internal form. This is thentranslated into relational algebra. • Parser checks syntax and verifies relations.

1. Optimization

• Construct a query-evaluation plan that minimizes thecost of query evaluation.

1. Evaluation

• The query-execution engine takes a query-evaluationplan, executes that plan, and returns the answers to thequery.

A relationalalgebra operation annotated with instructions on how to evaluate it called an evaluation primitive.

order the records physically-> sorting
