DATABASE MANAGEMENT SYSTEMS L T P C
3 0 0 3

OBJECTIVES

- To learn the fundamentals of data models and to represent a database system using ER diagrams.
- To study SQL and relational database design.
- To understand the internal storage structures using different file and indexing techniques which
- will help in physical DB design.
- To understand the fundamental concepts of transaction processing- concurrency control techniques and recovery procedures.
- To have an introductory knowledge about the Storage and Query processing Techniques

UNIT I RELATIONAL DATABASES 10

Purpose of Database System – Views of data – Data Models – Database System Architecture – Introduction to relational databases – Relational Model – Keys – Relational Algebra – SQL fundamentals – Advanced SQL features – Embedded SQL– Dynamic SQL

UNIT II DATABASE DESIGN 8

Entity-Relationship model – E-R Diagrams – Enhanced-ER Model – ER-to-Relational Mapping – Functional Dependencies – Non-loss Decomposition – First, Second, Third Normal Forms,Dependency Preservation – Boyce/Codd Normal Form – Multi-valued Dependencies and Fourth Normal Form – Join Dependencies and Fifth Normal Form

UNIT III TRANSACTIONS 9

Transaction Concepts – ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – Locking Protocols – Two Phase Locking – Deadlock – Transaction Recovery - Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery.

UNIT IV IMPLEMENTATION TECHNIQUES 9

RAID – File Organization – Organization of Records in Files – Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing – Query Processing Overview – Algorithms for SELECT and JOIN operations – Query optimization using Heuristics and Cost Estimation.

UNIT V ADVANCED TOPICS 9

Distributed Databases: Architecture, Data Storage, Transaction Processing – Object-basedDatabases: Object Database Concepts, Object-Relational features, ODMG Object Model, ODL, OQL - XML Databases: XML Hierarchical Model, DTD, XML Schema, XQuery – Information Retrieval: IR Concepts, Retrieval Models, Queries in IR systems.

TOTAL: 45 PERIODS

OUTCOMES:

Upon completion of the course, the students will be able to:

- Classify the modern and futuristic database applications based on size and complexity
- Map ER model to Relational model to perform database design effectively
- Write queries using normalization criteria and optimize queries
- Compare and contrast various indexing strategies in different database systems
- Appraise how advanced databases differ from traditional databases.

TEXT BOOKS:

1. Abraham Silberschatz, Henry F. Korth, S. Sudharshan, “Database System Concepts”, Sixth Edition, Tata McGraw Hill, 2011.
2. Ramez Elmasri, Shamkant B. Navathe, “Fundamentals of Database Systems”, Sixth Edition, Pearson Education, 2011.

REFERENCES:

1. C.J.Date, A.Kannan, S.Swamynathan, “An Introduction to Database Systems”, Eighth Edition, Pearson Education, 2006.
2. Raghu Ramakrishnan, —Database Management Systems‖, Fourth Edition, McGraw-Hill College Publications, 2015.
3. G.K.Gupta,"Database Management Systems”, Tata McGraw Hill, 2011

UNIT - I

RELATIONAL DATABASES

1.1 Database Management Systems: (DBMS)

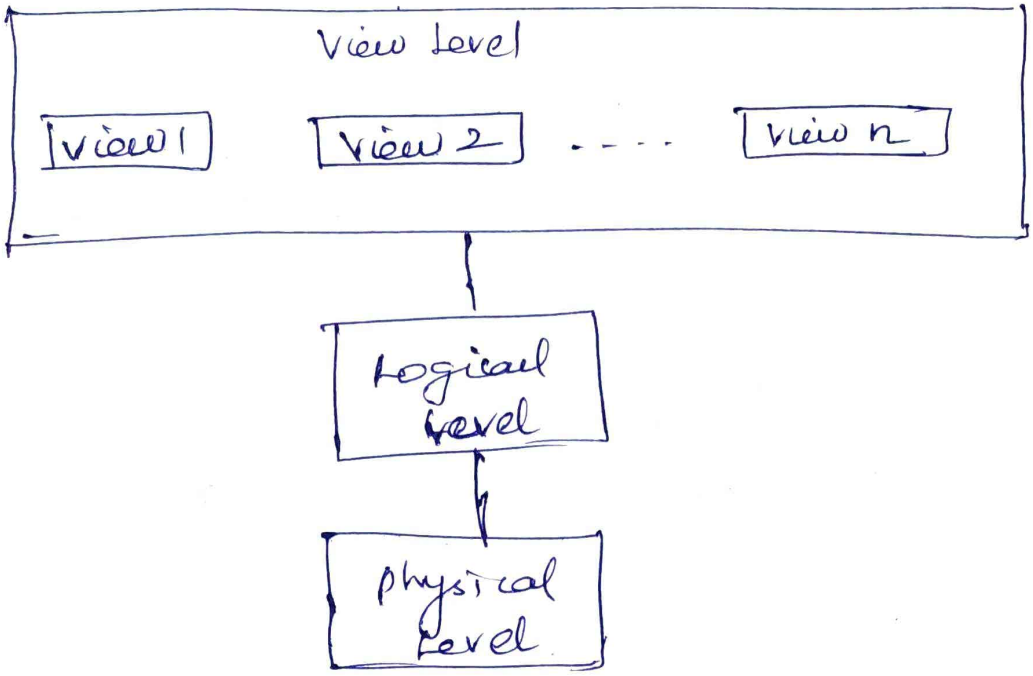
Database Management System refers to the technology of storing and retrieving users data with atmost efficiency along with safety and security features.

Disadvantages of File-processing system

(i) Data Redundancy and inconsistency
(ii) Difficulty in accessing data
(iii) Data isolation
(iv) Integrity problems
(v) Atomicity problems
(vi) Concurrent access anomalies
(vii) Security problems.

1.2 Views of Data:

* Physical Level - how data are actually stored
* Logical Level - what data are stored and what relationships exists.
* View Level - describes only part of the data



1.3 Data models:

Data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

Data model classification

(i) Relational model
(ii) Entity - Relational model
(iii) Object-Based Data model
(iv) Semi structured Data model.
(v) Network data model
(vi) Hierarchical data model

1.4 # Database System Architecture

Functional Components

* Database user and administrator
* Storage Manager
* Query Processor

1) Data base Users

4 user types

a) Naive users - interact with system by invoking application program

b) Application programmers - who write programs

c) Sophisticated users - uses query Language

d) Specialized users - who write Specialized database applications

DB administrator (DBA)

Person who control over the system.

Functions of DBA

* Schema definition
* Storage structure and access-method definition
* Schema and physical organization modification
* Granting for authorization for data access

* Routine Maintenance

2) Storage Manager

responsible for storing, retrieving and updating data in the database

Components

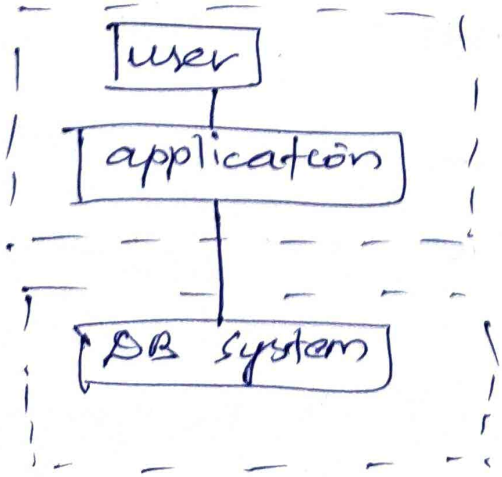
a) Authorization and integrity manager
b) Transaction manager
c) File manager
d) Buffer manager

3) Query Processor

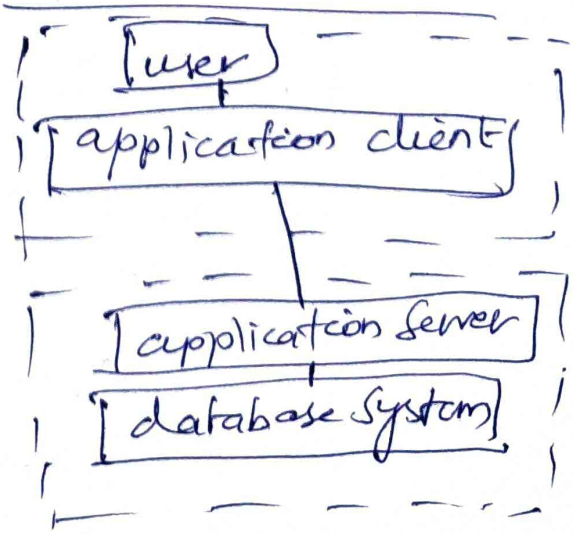
Components

a) DDL interpreter - Interprets DDL stmts
b) DML compiler - Translates DML stmts
c) Query Language engine - executes low level instruction

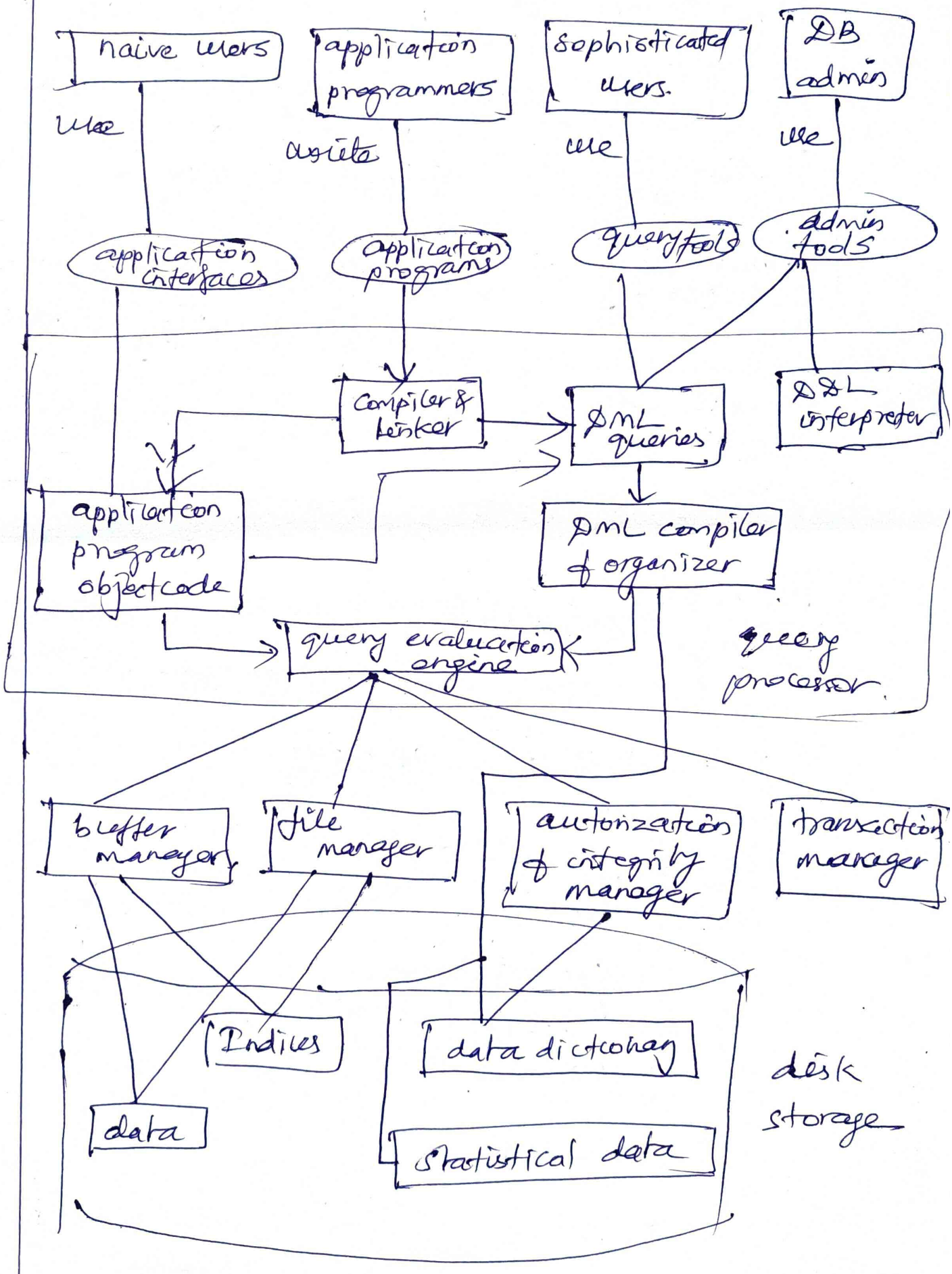
Two tier Architecture



3-tier Architecture



System Structure



1.5 Relational Model

Keys:

Superkey ⇒ Set of one or more attributes uniquely identify a tuple

Candidate key ⇒ minimal superkeys.

Primary key ⇒ Candidate key chosen to uniquely identify a tuple.

Foreign key ⇒ Attribute in r_1 is a primary key of another relation r_2.

1.6 Relational Algebra

* Procedural Query Language
* fundamental operations
  a) Select
  b) Project
  c) Union
  d) Set difference
  e) Cartesian product
  f) Rename

Select (σ):

Choose a subset of tuples from a relation that satisfies a selection condition

$$\sigma_{\langle \text{selection condn} \rangle}(R)$$

Project (Π):

Select certain columns

$$\Pi_{\langle \text{attribute list} \rangle}(R)$$

Union ($\cup$):

includes all either in R or in S or in both.

Duplicates are eliminated.

$$\Pi_{\text{names}}(R_1) \cup \Pi_{\text{names}}(R_2)$$

Set difference ($-$):

tuples in one relation but not in another.

$$\Pi_{\text{names}}(R_1) - \Pi_{\text{names}}(R_2)$$

Cartesian Product (X):

Combine information from any 2 different relations

$$\Pi_{names}(R_1 \times R_2)$$

Rename Operation (ρ):

rename the output relation

$$\rho_x(E)$$

Intersection operation ($\cap$):

includes all tuples that are in both R and S.

$$\Pi_{names}(R) \cap \Pi_{names}(S)$$

Natural Join ($\bowtie$):

Set of all combinations of tuples in R and S that are equal in their common attribute.

$$R \bowtie_{(condn)} S$$

1.7 # SQL fundamentals

Standard Language for Relational DB system.

Data types

* Char(n)
* varchar (n)
* int
* smallint
* numeric (p,d)
* real
* float (n)
* date
* time

DDL

create – used to create objects in database

create table <tablename> (coln datatype(size), col2 ....)

desc – describe the table

drop – delete the objects from data base

alter - alter the structure of objects

alter table <tablename>

add (new col1 datatype (size),

....)

truncate - delete the records but retain the structure

DML

insert - used to insert new records into database

insert into <tablename> values (a list of data values)

update - used to update the particular column value

update <tablename> set

<col = value> where <condition>

delete - allows to delete particular records.

delete from <tablename> where <condn>

Select – used to select particular records alone.

Select <coln... coln) from <tablename>

DCL

used to control privilege in Database

Defines 2 commands.

* Grant
* Revoke

Grant <object privileges>
ON <object name>
to <user_name>
[with grant option]

revoke <object privileges>
on <object_name>
from <User_name>

TCL

used to manage transactions in database.

commit – permanently save transaction in DB.

Savepoint - point to which later we can rollback.

Savepoint savepoint_name

rollback - used to undo the work done

rollback [work] [to savepoint]

String operations

percent % : matches any substring

Underscore _ : matches any character

Order by

result of the query appear in sorted order.

Select * from tablename order by col-name;

Aggregate Functions

avg - average value
min - minimum value
max - maximum value
sum - sum of values
count - number of values

Group by clause

```
Select col_name , aggregate function
from tablename
group by col_name.
having aggregate function predicate
```

Procedure:

block that can take parameters and be invoked.

If the definition changes, only the procedure are affected.

```
Create or replace <procedure name>
(argument {in, out, inout}
datatype)
{is, as} variable declaration
constant declaration
begin
PL/SQL subprogram body
exception
exception PL/SQL block
end;
```

Functions:

Same as procedure except that it returns a value.

```
create or replace function <function_name>
    (argument in datatype, ...) return
                    datatype
{is, as}.
    variable declaration
    constant declaration
    begin
        PL/SQL subprogram body.
    exception
        exception. PL/SQL block
    end;
```

Triggers:

defines the action the database needs to perform whenever some database manipulation (Insert, Update and Delete) takes place.

Trigger has 3 parts.

1. Trigger Event
2. A trigger Constraint
3. Trigger Action

```
Create [or replace] Trigger Trigger_name
{before | after | instead of}
{Insert [or] | update [or] | Delete}
[of col_name]
ON table_name
[Referencing old As o new as n]
[for each row]
declare
    declaration statements
begin
    executable statements.
Exception
    Exception handling statements
end;
```

1.9. Embedded SQL

method of combining the power of a programming language and the database.

processed by special SQL precompiler.

```
EXEC SQL BEGIN DECLARE SECTION
    varchar dname [10], fname [10], ....
    char ssn [10], bdate [11], ...
    int dno, dnumber, SQLcode.
EXEC SQL END DECARE SECTION
```

1.10 Dynamic SQL:

allows the program to construct an SQL query as a character string at runtime, submit the query, and then retrieve the result into program variables a tuple at a time.

Static SQL	Dynamic SQL
how database will be accessed is predetermined	database will be accessed is determined at run time
more efficient	less efficient
compiled at compile time	compiled at run time
data distributed uniformly	data distributed non-uniformly
less flexible	more flexible.

UNIT-II

Database Design

2.1 Entity-Relationship Model:

* defines the conceptual view of a database
* 3 basic concepts
  (i) Entity set
  (ii) Attributes
  (iii) Relationship sets.

Entity Set:

* Entity - real world object
* Entity set - set of entities of same data type.

Attributes

* Attributes - properties of entities
* Domain - range of values that can be assigned to attribute

Types:

a) Simple b) Composite c) Derived
d) Single-valued e) multi-valued

Relationship sets.

* Relationship – association among entities

* Relationship set – set of relationships of similar type.

* Degree of relationship – number of participating entities in a relationship

Binary → 2

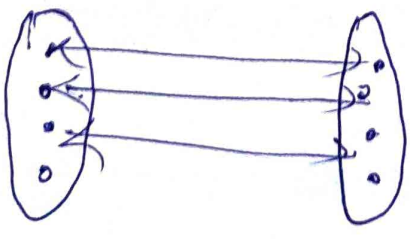
Ternary → 3

n-dry → n

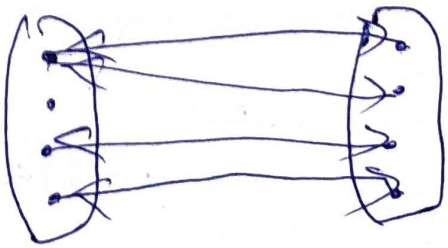
* Mapping cardinalities

* cardinality define the no. of entities in one entity set which can be associated with number of entities of other set

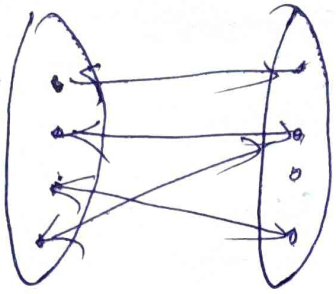
One-to-One



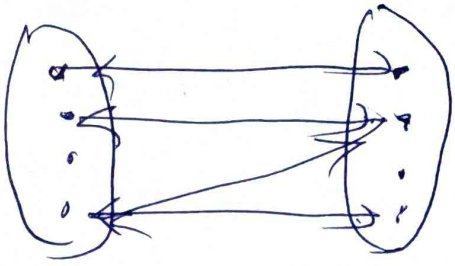
One-to-many



Many-to-one



many-to-many

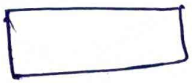
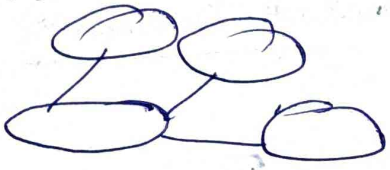


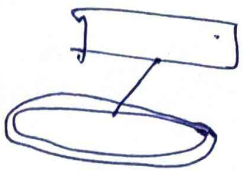
Participation constraints

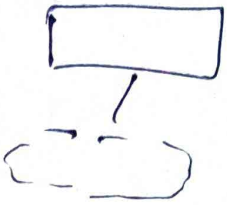
* total — every entity in E participates in at least one relationship
* partial — only some entities in E participated.

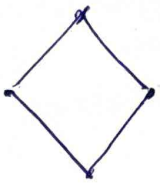
2.2. ER Diagrams

— defines the conceptual view of the database

- ER diagram expresses the overall logical structure of a database graphically
- Entity set - Rectangles 
  Attributes - ellipses 
  Composite - 

Multivalued – 

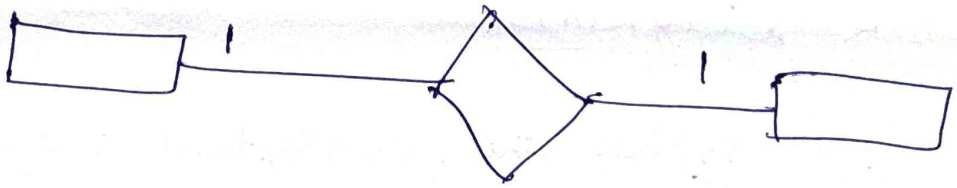
Derived – 

Relationships – 

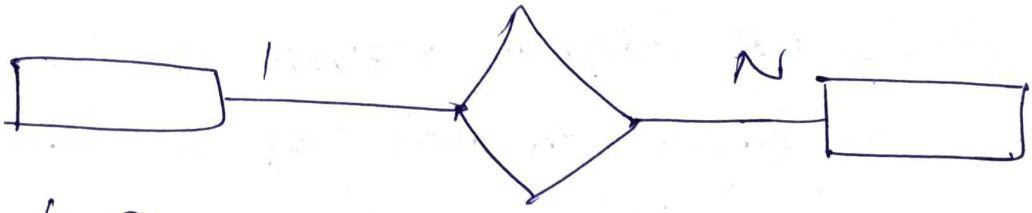
participation – 

Binary Relationship & Cardinalities

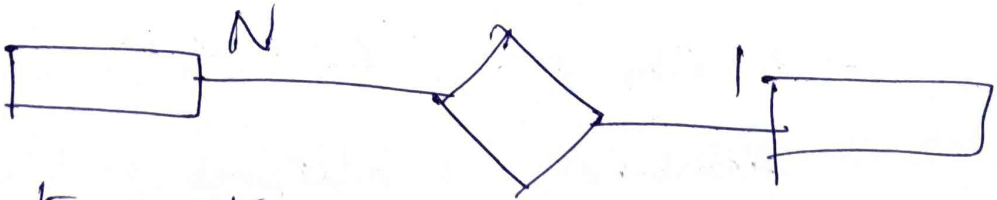
One to one



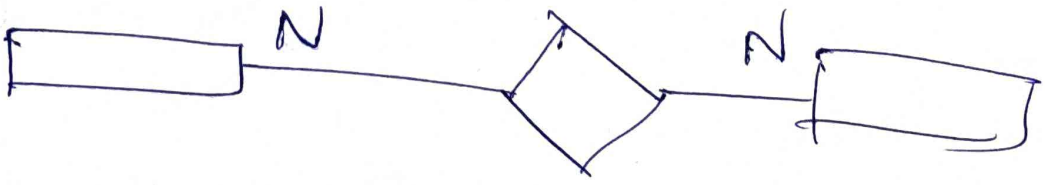
One-to-many



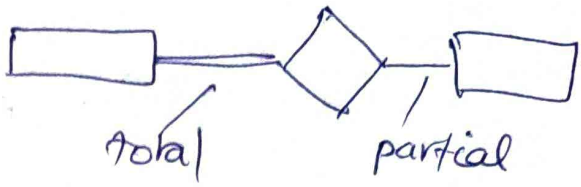
Many-to-one



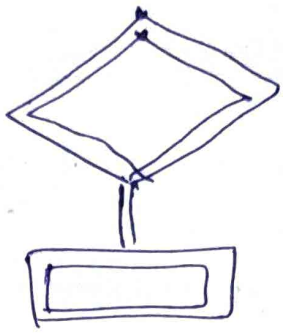
Many-to-many



Total participation and partial participation



Weak entity



2.3. E-R to Relational mapping

Step 1: Mapping of regular entity types

* For each regular entity type E, create a relation R.
* Include all simple attributes
* Choose one of the key attribute as primary key of R.

Step 2: Mapping of Weak entity types

* For each weak entity type W, create the relation R
* Include all simple attributes.
* Include a foreign key as the combination of primary key of E

and the partial key of W

Step 3: Mapping of 1:1 Relationship

* for each 1:1 type R, identify the relations S and T that correspond to the entity types for R

2 approaches

1) Foreign key Approach

2) the merged relationship Approach (When both are total)

Foreign key

1) Include PK of T as FK in S
   - Include all simple attributes
   - ~~Include~~ PK

2) Merged Relationship

- mapping of 1:1 relationship
- merge the 2 entity types and the relationship into a single relation

Step 4: Mapping of 1:N Relationship types.

* identify N side relation as S and other side as T
* Include PK of T as FK of S

Step 5: Mapping of M:N type

* Create a new relation S to represent R.
* Include PKs of both relations as FKs in S and their combination as the PK of S
* Include simple attributes of relationship type as attributes of S

Step 6: Mapping of multi-valued attributes

* for each multi-valued attribute A, create a relation R.
* Include attribute A in R
* Include K, the PK of relation which contains A as FK of R

* PK of R is the combination of A and K.

2.4. Enhanced ER (EER) model:

In addition to ER model, it includes

* Subclass and Superclass
* Specialization and Generalization
* attribute and relationship inheritance

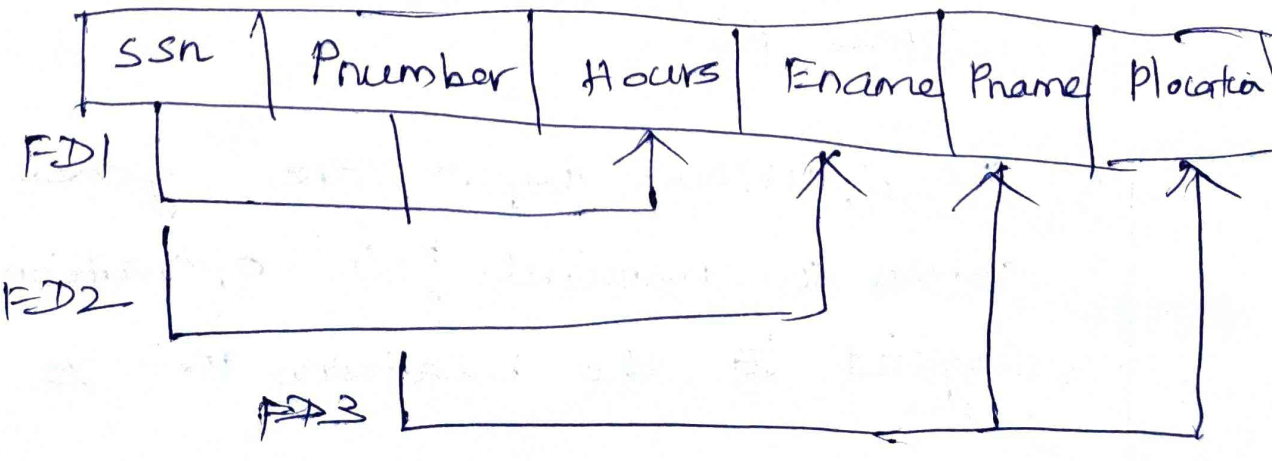
* Subgrouping are called Subclass
* Main class is called the super class
* An entity being a member of Subclass, it must also be a member of the super class.
* An entity can be included as a member of any number of subclasses.
* An important concept associated with Subclass is Inheritance

* Specialization is the process of defining a set of subclasses of an entity type.
* This entity type is called the superclass.
* Generalization process can be viewed as being functionally the inverse of the specialization process

2.5. Relational database design:

Functional Dependencies

* FD $X \rightarrow Y$, for any 2 tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$ they must also have $t_1[Y] = t_2[Y]$



Decomposition

A decomposition algorithm decomposes the universal relation schema R into a set of relation schema

$$D = \{R_1, R_2 \dots R_m\}$$

$$\bigcup_{i=1}^{m} R_i = R \quad \text{(attribute preservation)}$$

Properties of Decomposition

(1) Dependency Preservation

(2) Lossless (non additive) join property

(1) If each $X \to Y$ in F either appear directly in one of the relation schema R_i in D or could be inferred from the dependencies that appear in some R_i.

(2) no spurious tuples are generated when a natural join operation is applied to the relations in D

Normalization

Normalization is a systematic approach of decomposing tables to eliminate data redundancy

Types:

1. First Normal Form (1NF)
2. Second " " (2NF)
3. Third " " (3NF)
4. BCNF (Boyce-Codd Normal Form)
5. Fourth Normal Form (4NF)
6. Fifth " " (5NF)

1NF:

domain of an attribute must include only atomic values.

2NF:

If and only if it is in 1NF and every nonprime attribute A is fully functionally depend on the PK of R

3NF:

If and only if it is in 2NF and no nonprime attribute is transitively depend on the primary key.

BCNF:

BCNF eliminates all redundancy

for all FDs in F^+ of the form

$\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$ atleast one of the following holds.

* $\alpha \rightarrow \beta$ is a trivial FD ($\beta \subseteq \alpha$)
* α is a super key for schema R

General rule for decomposing into BCNF

R - schema not in BCNF

non trivial FD $\alpha \rightarrow \beta$. such that α is not a superkey for R.

We replace R in our design with

↳ $(\alpha \cup \beta)$

↳ $(R - (\beta - \alpha))$

closure of F (F^+)

Set of all FDs logically implied by F is the closure of F.

F^+ can be computed by applying Armstrong's Axioms.

* Reflexity Rule
* Augmentation Rule
* Transitivity Rule
* Union Rule
* Decomposition Rule.
* Pseduo Transitivity Rule

procedure for computing F^+.

$F^+ = F$

repeat
  for each FD f in F^+
    apply reflexivity and augmentation rule
    add result to F^+
  for each pair of FDs f_1 and f_2 in F^+
    if f_1 and f_2 can be combined using transitivity

add result to F^+

until F^+ does not change further.

Multi valued and Fourth Normal form

A table is in 4NF, if for every of its non-trivial multivalued FDs. X is a super key.

Join dependencies and Fifth Normal Form

5NF, if every non-trivial join dependency in it is implied by the candidate keys.

UNIT-III

Transactions

3.1. Transaction Concepts

- Collection of operations that form a single logical unit of work is called Transaction

3.2 ACID properties

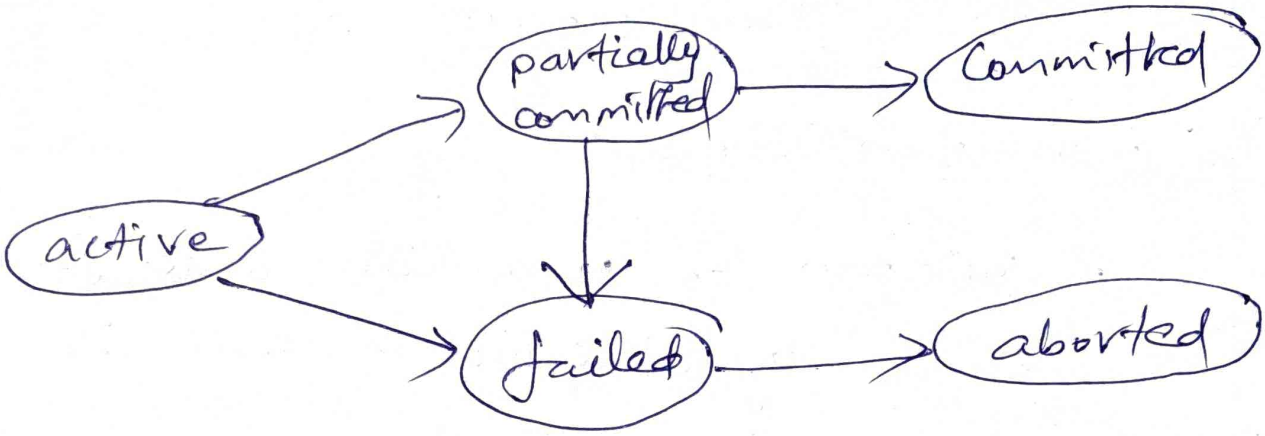
Atomicity - Either all operations are reflected properly in the database or none.

Consistency - Execution of a transaction in isolation

Isolation - Each transaction is unaware of other transactions executing concurrently.

Durability - After a transaction completes successfully, the changes it has made to the database persist even if there are system failures.

3.3. States of Transaction



3.4. Schedules:

Sequence of instructions that specify the chronological order in which instructions of concurrent Transactions are executed.

T_1	T_2
read (A)	
A := A - 50	
write (A)	
read (B)	
B := B + 50	
write (B)	
commit	read (A)

temp := A * 0.1
A := A - temp
write(A)
read(B)
B := B + temp
write(B)
commit.

A schedule is serializable if it is equivalent to a serial schedule

T_1	T_2
read(A)	
A := A - 50	
write(A)	
	read(A)
	temp := A * 0.1
	A := A - temp
	write(A)
read(B)	
B := B + 50	
write(B)	
commit	
	read(B)
	B := B + temp
	write(B)
	commit.

3.5. Serializability:

Types of serializability

1. conflict serializability
2. View serializability

1. Conflict Serializability

Instructions I_i and I_j of transactions T_i and T_j respectively, conflict iff there exists some item Q accessed by both I_i and I_j and atleast one of these instructions wrote Q.

1. I_i = read(Q), I_j = read(Q) no conflict
2. I_i = read(Q), I_j = write(Q), they conflicts
3. I_i = write(Q), I_j = read(Q), they conflict
4. I_i = write(Q), I_j = write(Q), they conflict.

View Serializability

Schedule S is view serializable, if it is view equivalent to a serial schedule.

S and S0 are view equivalent if the following 3 conditions are met.

1. For each data item Q, if transaction Ti reads the initial value of Q in schedule S, then transaction Ti must, in schedule S0, also read the initial value of Q.
2. For each data item Q, if transaction Ti executes read(Q) in schedule S, and that value was produced by Tj, then Ti must also read the value of Q that was produced by Tj
3. For each data item Q, the transaction that performs the final write(Q) in S then it must perform the final

write (Q) operation in S_0.

* Every conflict serializable schedule is also view serializable.
* But all view serializable schedules are not conflict serializable.
* Blind writes not allow view serializable to be conflict serializable.

Testing for serializability:

using a directed graph called precedence graph.

$G = (V, E)$.

Set of edges consists of all edges $T_i \rightarrow T_j$ for which one of three conditions holds.

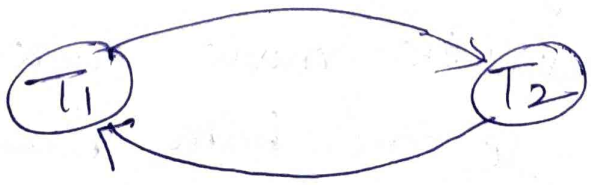
1) T_i executes write(Q) before T_j executes read (Q)
2) T_i executes read(Q) before T_j executes write (Q)

3) T_i executes write (Q) before T_j executes write (Q).

If the graph contains cycle, it is not conflict serializable.

T_1	T_2
read (A)	
A := A − 50	
	read (A)
	temp := A * 0.1
	A := A − temp
	write (A)
write (A)	
read (B)	
B := B + 50	
write (B)	
commit	
	read (B)
	B := B + temp
	write (B)
	commit

precedence graph



The precedence graph contains cycle So it is not conflict serializable.

3.6 Concurrency Control

* based on serializability property

1) Locking protocols

* when one transaction is accessing the database, lock may deny access to other transactions to prevent incorrect results.

Locks

2 modes of locks are

(1) Shared lock (S)

If a transaction T_i has obtained a shared-mode lock on item Q, then T_i can read, but cannot write Q.

(2) Exclusive lock (X)

If a transaction T_i has obtained an exclusive mode lock on item Q. then T_i can both read and write Q

Two-phase locking

* requires that each transaction issue lock and unlock requests in 2 phases.

(i) Growing phase

A transaction may obtain locks, but may not release any lock.

(ii) Shrinking phase

A transaction may release locks, but may not obtain any new locks.

Advantages:

* 2-phase locking ensures conflict serializability.
* Transactions can be ordered according to their lock points.

Disadvantages

* It does not ensure freedom from deadlock.
* Cascading rollbacks

Cascading rollbacks can be avoided by strict-2-phase locking

Types

1) Strict 2-phase locking

all exclusive-mode locks taken by a transaction should be held until transaction.

2) Rigorous 2-phase locking

all locks are to be hold until the transaction commits.

39. Deadlock

* refers to a particular situation where 2 or more processes are each waiting for another to release a resource, or more than 2 processes are waiting for resources in a circular chain

Deadlock prevention

uses time-stamps ordering protocols

Deadlock Prevention

Wait_Die Scheme:

* If $TS(T_i) < TS(T_j)$, i.e T_i which is requesting a conflicting lock, is older than T_j, T_i is allowed to wait until the data_item is available.

* If $TS(T_i) > TS(T_j)$ i.e, T_i is younger than T_j, T_i dies, T_i is restarted later with random delay but with same timestamp.

Wound_Wait Scheme:

* If $TS(T_i) < TS(T_j)$ i.e T_i, which is requesting a conflicting lock, is older than T_j, T_i forces T_j to be rolled back, i.e T_i wounds T_j, T_j is restarted later with random delay but with same timestamp.

* If $TS(T_i) > TS(T_j)$, ie T_i is younger than T_j, T_i is forced to wait until the resource is available.

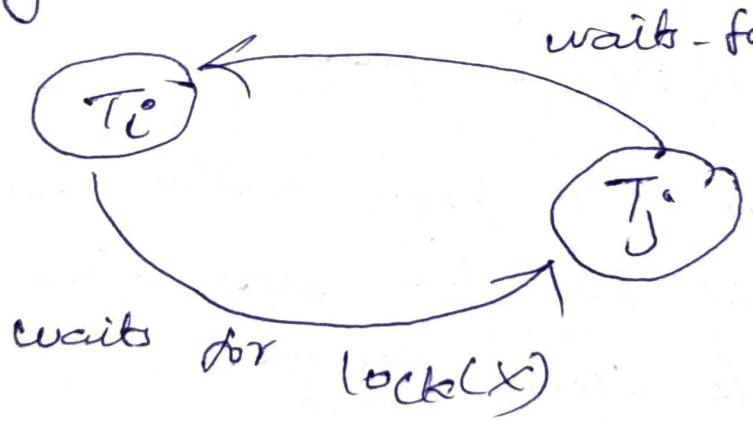
Deadlock Avoidance :

* Used to detect any deadlock situation in advance.
* Wait-for graph is used.

Wait-for graph

* For each transaction, a node is created.
* When T_i requests for a lock, on item say X., which is held by some other transaction T_j, a directed edge is created from T_i to T_j.
* If T_j releases item X, the edge between them is dropped.

* If T_j releases item x, the edge between them is dropped and T_i locks the data item.
* keeps checking if there is any cycle in the graph.
* If there is a cycle, then deadlock may occur.



* First options is not to allow any request for an item, which is already locked by some other transaction
* But starvation may occur.
* Second option is to roll back one of the transactions
* But it is not feasible.
* Choose the correct victim

3.10. Transaction Recovery:

* 2 common commit protocols suitable for distributed DBMSs.

* (i) 2-phase commit
  (ii) 3-phase commit

(i) 2-phase commit

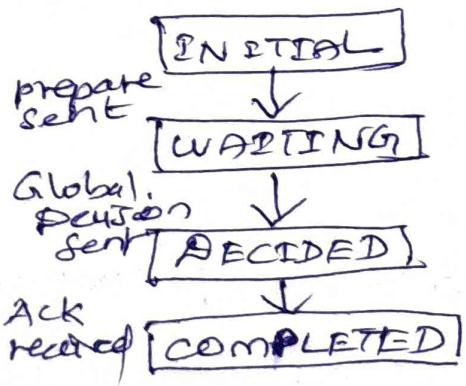
* operates in 2 phases
  a) voting phase
  b) decision phase.

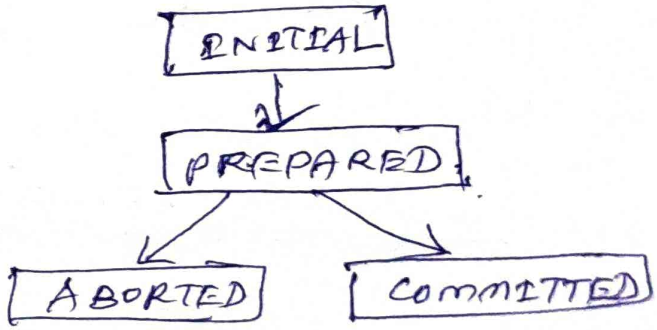
a) - Write a begin_commit record to the log file and force-write it to stable storage.
- Send a PREPARE message to all participants
- wait for participants to respond within a timeout period.

b) if a participant returns an ABORT vote

Write an abort record to the log file and force write it to stable storage

- send a GLOBAL_ABORT message to all participants.
- wait for participants to acknowledge within a timeout period.
- If a participant returns a READY_COMMIT, vote.
- write a commit record to the log file and force write it to stable storage
- send a GLOBAL_COMMIT message to all participants.
- wait for participants to acknowledge within a timeout period
- Once all acknowledgements have been received, write an end_transaction message to the log file.





Termination Protocols for 2PC

3.11. Save Points

* A savepoint is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction

* Savepoint Savepoint_

* Rollback command is used to undo a group of transactions

ROLLBACK TO SAVEPOINT_NAME

* RELEASE SAVEPOINT command is used to remove a SAVEPOINT already created.

RELEASE SAVEPOINT SAVEPOINT_NAME

* Once a savepoint is released, you can no longer use the ROLLBACK command to undo.

3.12. Isolation Levels.

* determine the type of phenomena that can occur during the execution

of concurrent transactions

* Developer sets this property only for the following databases:
  MSSQL, Informix and DB2.

* 3 Phenomena define SQL Isolation levels for a transaction

  1) Dirty Read
  2) Non_Repeatable Read.
  3) Phantoms.

Isolation level	Dirty Read	Non Repeatable Read	Phantoms
Read Uncommitted	X	X	X
Read committed	–	X	X
Repeatable Read	–	–	X
Serializable	–	–	–

Serializable – Highest Isolation Level.

3.13 SQL facilities for concurrency and Recovery

Crash Recovery

* DBMS is highly complex system
* durability and robustness depends on its complex architecture and its underlying hardware and system software.
* If it fails or crashes, it is expected that the system would follow some sort of algorithm or techniques to recover lost data.

Failure classification

a) Transaction failure
   - Logical errors
   - System errors

b) System Crash

c) Disk Failure

(10)

Recovery and Atomicity

- check the states of all the transactions
- ensure atomicity
- whether transaction can be completed or to be rolled back
- No transaction is allowed to leave the DBMS in an inconsistent state

Two methods for recovery

1) Maintaining logs of each transaction.
2) Maintaing shadow paging

Log-based Recovery

* Log is a sequence of records which maintains the records of actions performed by a transaction

* Logs are written prior to the actual modification and stored on a stable storage media,

* Works as follows

- When a transaction enters the system and starts execution, it writes a log about it.

  $\langle T_n, \text{start} \rangle$

- When the transaction modifies an item X, it writes

  $\langle T_n, X, V_1, V_2 \rangle$

- reads T_n has changed the value of X, from V_1 to V_2

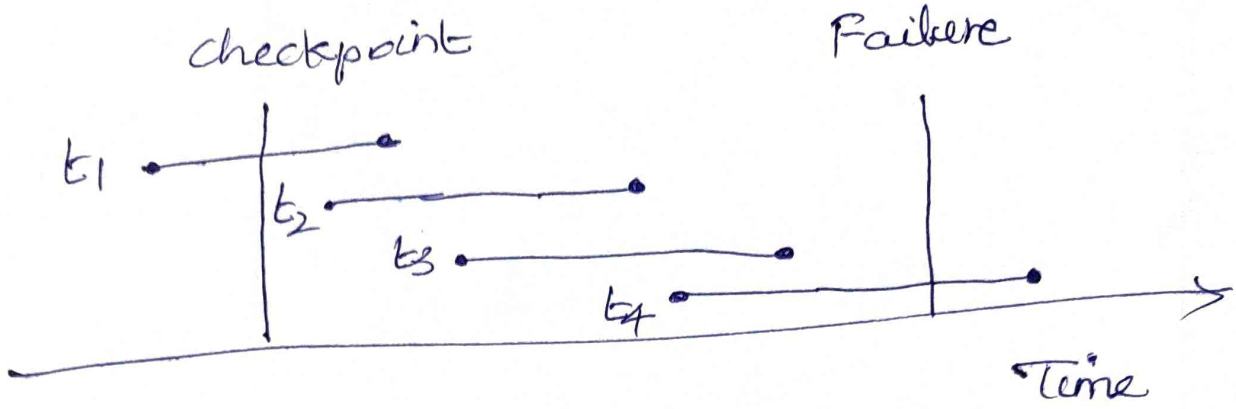
- When the transaction finishes it logs

  $\langle T_n, \text{commit} \rangle$

* DB can be modified using 2 approaches.

(1) Deferred DB modification

(2) Immediate DB modification



* Recovery system maintains 2 lists
  1) undo_list 2) redo_list

* when <Tn, start> and <Tn, commit> or just <Tn, Committ> it puts it in redo_list

* when it sees <Tn, start> but no commit or abort log found, it puts it in undo_list

Here Redo list <t1, t2, t3>
undo list <t4>

UNIT - IV

Implementation Techniques

4.1 RAID (Redundant Arrays of Independent Disks)

* A variety of disk organization techniques are collectively called RAID.

Improvement via Reliability via Redundancy

* to introduce redundancy is to duplicate every disk.
* This is called mirroring
* If one disk fails, data can be read from the other.

Improvement via performance via parallelism

* stripping data across multiple disks
* bit level striping – splitting the bits of each byte across multiple disks.

* Block-Level Striping — stripes blocks across multiple disks.

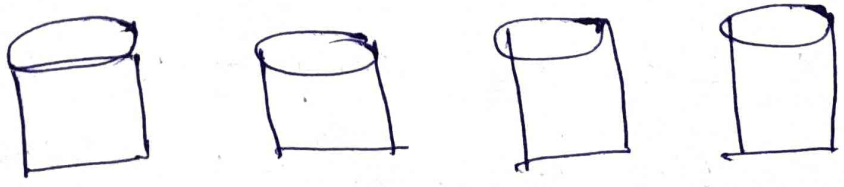
* Mirroring — high reliability, but it is expensive

Striping — high data-transfer rates, but does not improve reliability.

RAID Levels

RAID 0 : Block striping, non-redundant

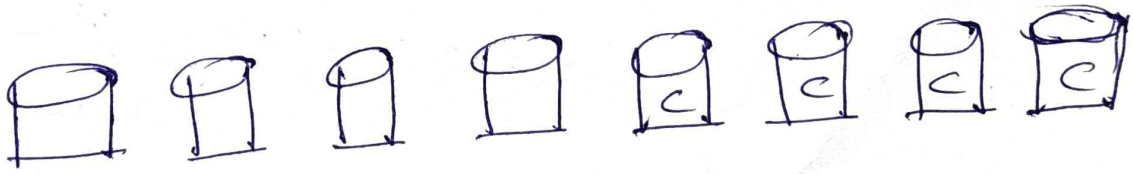
* no redundant data
* Data striping at block level.
* Used in high-performance applications where data lost is not critical



Raid 0 — Non Redundant stripping

RAID 1: Mirrored disks with block Striping

* maintains 2 identical copies
* If each disk has M blocks, logical blocks 0 to M-1 on disk 0, M to 2M-1 on disk 1 and so on, and each disk is mirrored.
* popular for applications such as storing log files in a database system.



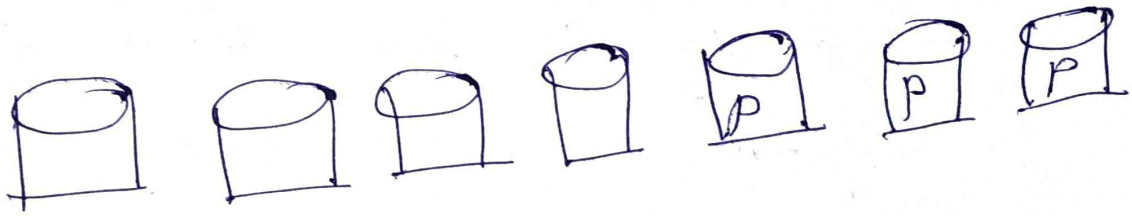
Raid 1: Mirrored Disks.

RAID 2: ECC with bit striping

* Striping unit is a single bit and hamming codes are used as the redundancy scheme.
* parity bits for error detection and correction

* Each byte in the memory system may have a parity bit associated with it that records whether the numbers of bits in the byte that are set to 1 is even (parity = 0) or odd (parity = 1)

* All one-bit errors can be detected.



Raid 2: Memory-style error correcting codes

RAID 3: Bit-Interleaved Parity

* a simple parity bits is computed for a set of individual bits in the same position on all of the data disks.

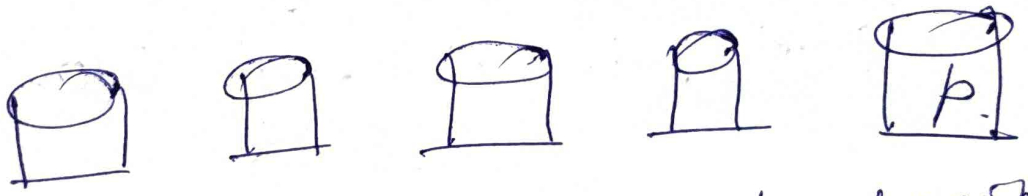
* When writing data, corresponding parity bits must also be computed and written to a parity bit disk.
* To recover data in a damaged disk, compute XOR of bits from other disks.
* Faster data transfer with a single disk, but fewer I/Os per second, since every disk has to participate in every I/O.

RAID 4: Block-Interleaved Parity

* uses block-level striping
* keeps a parity block on a separate disk for corresponding blocks from a number of other disks.
* If one disk fails, the parity block can be used with corresponding blocks from the other disks

to restore the block of the failed disk.

* Before writing a block, parity data must be computed.
* Provides higher I/O rates for independent reads than level 3.
* More efficient for writing large amount of data sequentially



Raid 4: block-interleaved parity

RAID 5: Block interleaved distributed parity.

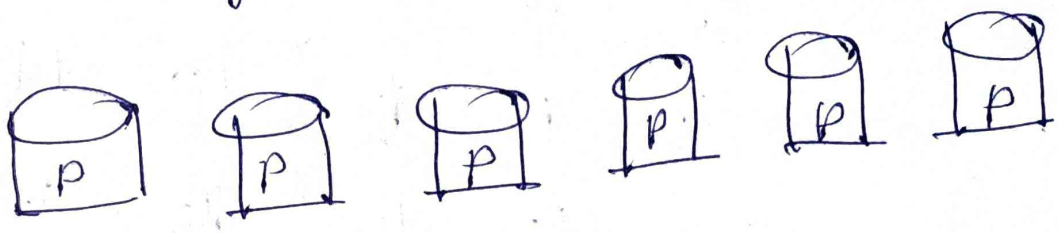
* Partitions data and parity among all N+1 disks, rather than storing data in N disks and parity in 1 disk.
* Higher I/O rates than level 4
* Block writes occur in parallel

P_0	0	1	2	3
4	P_1	5	6	7
8	9	P_2	10	11
12	13	14	P_3	15
16	17	18	19	P_4

RAID 6: Double Parity RAID scheme (or) P+Q Redundancy scheme

* similar to Level 5, but stores extra redundant information to guard against multiple disk failures.
* error-correcting codes are used instead of using parity.
* 2-bits of redundant data are stored for every 4 bits of data - unlike 1 parity bit in level 5 and the system can tolerate 2 disk failures

* Better reliability than 5. at a higher cost



Raid 6 : P + Q redundancy

A.2. File organization :

* The database is stored as a collection of files
* Each file is a sequence of record
* A record is a sequence of fields
* These records are mapped onto disk blocks.
* Each file is also logically partitioned into fixed-length storage units called blocks, which are the units of both storage allocation and data transfer.
* Most databases use block sizes of 4 to 8 KB by default

* A block may contain several records
* Two types of records
  (1) Fixed Length Records
  (2) Variable length Records.

4.3. Organisation of Records in files

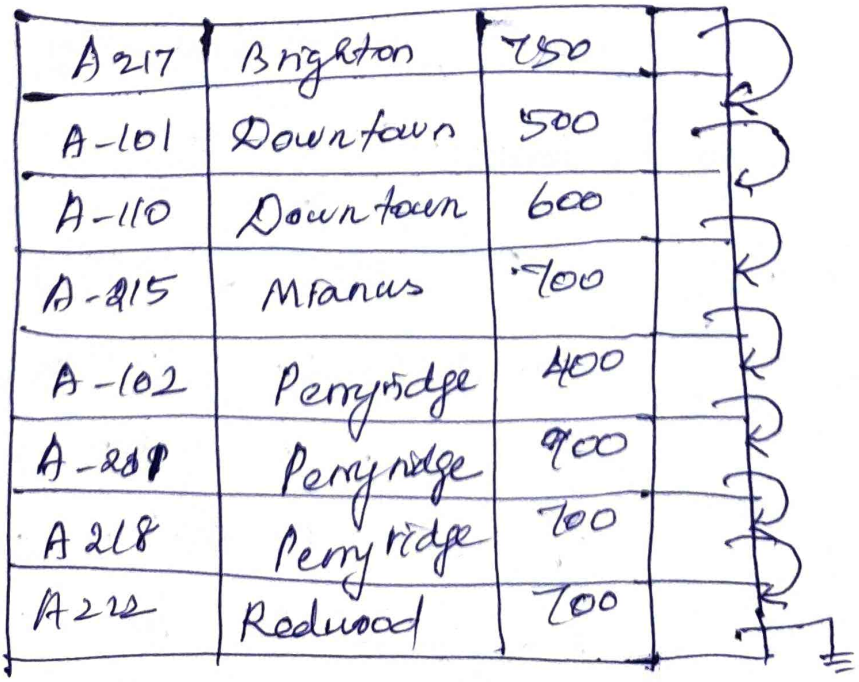
Several ways of organizing records in files are

* Heap – record can be placed anywhere in the file where there is space.
* Sequential – Store records in sequential order, based on the value of the search key of each record.
* Hashing – a hash function is computed on some attribute of each record; the result specifies in which block of the file the record should be placed.

* In a clustering file organization records of several different relations can be stored in the same file.

Sequential File Organization

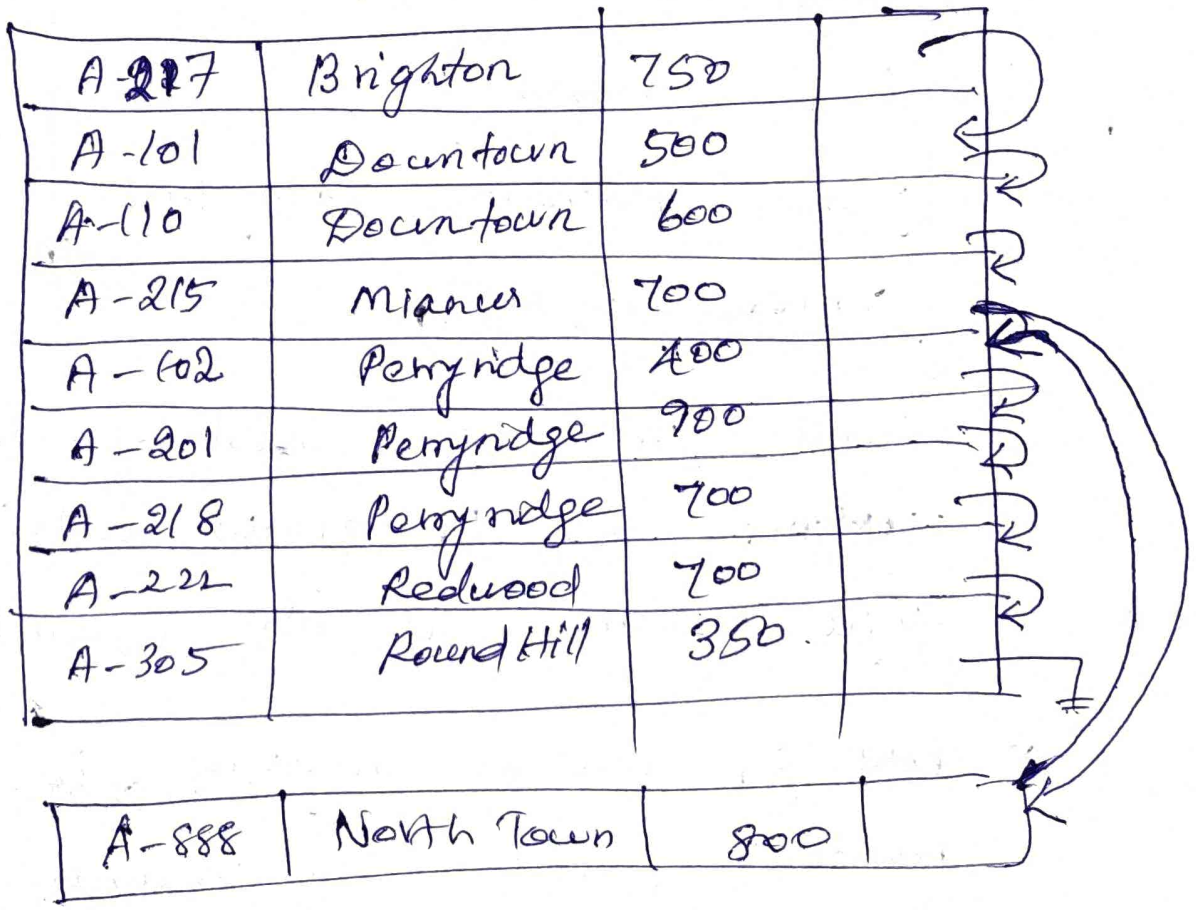
* efficient processing of records in sorted order based on some search key.
* Search key is any attribute or set of attributes.
* To permit fast retrieval of records in search key order, we chain together records by pointers.



A 217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A 218	Perryridge	700	
A 222	Redwood	700	

* For insertion, the following rules are applied.
* Locate the position where the record is to be inserted.
* If there is free space insert there otherwise insert the record in an overflow block
* In either case, pointer chain must be updated.
* Need to reorganize the file from time to time to restore sequential order.

After insertion



Clustering File Organization

* Simple file structure stores each relation in a separate file

* Can instead store several relations in one file using a clustering file organization

Eg: clustering organization of customer and depositor

Hayes	Main	Brooklyn
Hayes	A-102	
Hayes	A-220	
Hayes	A-503	
Turner	Putnam	Stamford
Turner	A-305	

* Good for queries involving depositor, customer, and for queries involving one single customer and his accounts.

* bad for queries involving only customer results in variable size records

4.4 Indexing and Hashing

Indexing

* Indexing provides the most powerful and flexible tools for organizing files
* Indexing mechanisms used to speed up access to desired data.
  Eg. author catalog in library
* Indexing mechanisms uses search key
* A search key is any attribute or set of attributes used to look up records in a file
* Index files are typically much smaller than the original file.
* The disadvantages of indexing are
  - → indexing take up space and time to keep them organized
  - → The amount of I/O increases with the size of index.
* There are 2 basic kinds of indices
  (1) Ordered indices - Search keys are stored in sorted order
  (2) Hash indices - Search keys are distributed uniformly across the buckets.

The bucket to which the value is assigned is determined by a function called hash function.

* Each technique must be evaluated on the basis of these factors.

Access Types:

* Access types can include finding records with a specified attribute value and finding records whose attribute values fall in a specified range.

Access Time:

* The time it takes to find a particular data item, or set of items using the technique.

1) Insertion Time

* The time it takes to insert a new data item
* This value includes the time it takes to find correct place to insert the new data as well as the time it takes to update the index structure

Space overhead:

* The additional space occupied by an index structure

Hashing

* A hash function is computed on some attribute of each record; the result specifies in which block of the file the record should be placed.

4.5. Ordered Indices:

Ordered indices:

* An ordered index stores the values of the search keys in sorted order and associates with each search key the records that contain it.

→ Primary Index

- is a sequentially ordered file, the index whose search key specifies the sequential order of the file
- Also called clustering index.
- The search key of a primary index is usually but not

necessarily the primary key

→ Secondary Index

* an index whose search key specifies an order different from the sequential order of the file.
* Also called non-clustering index.

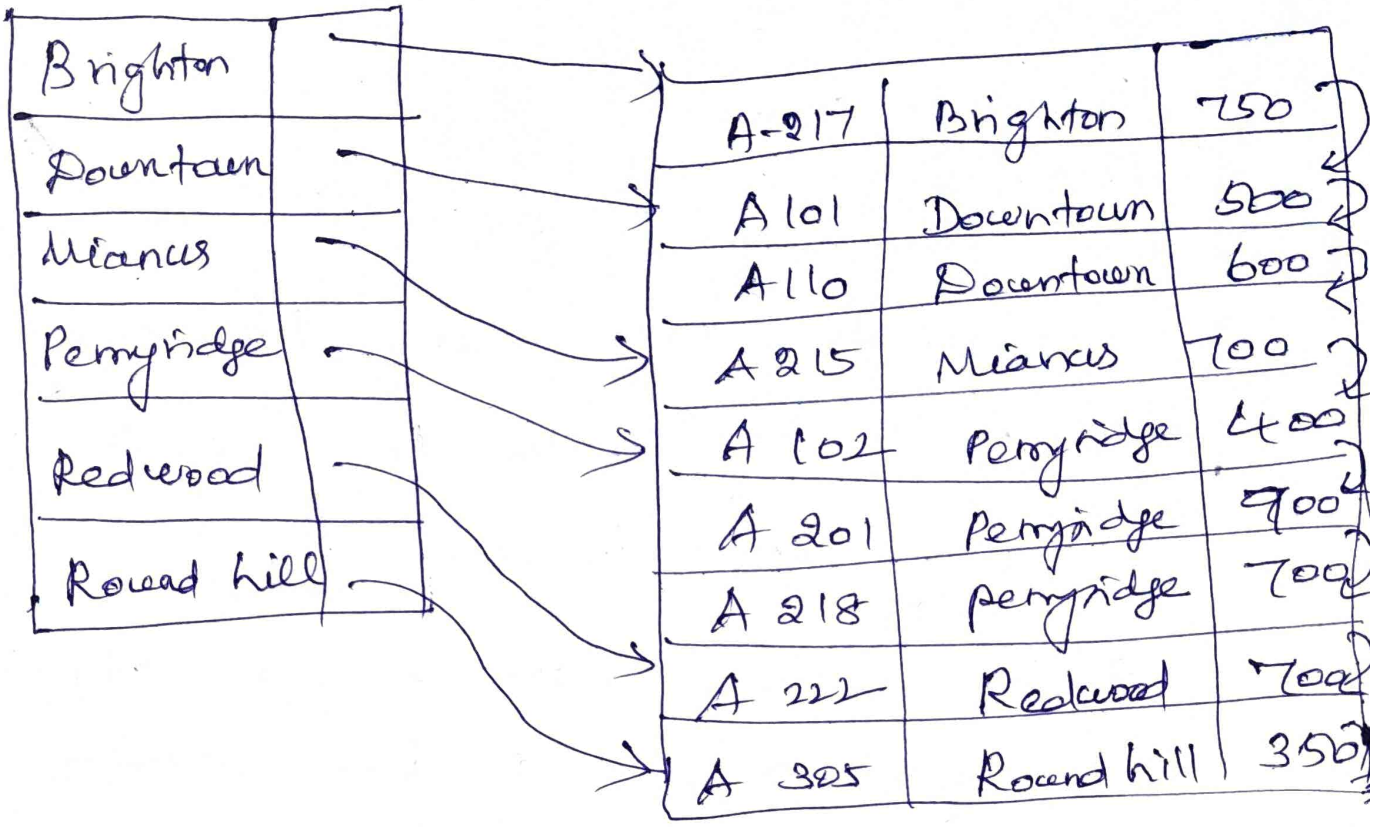
Primary Index:

* Files with a primary index on a search key, are called index-sequential file.
* 2 types of ordered indices
  a) Dense index
  b) Sparse index.

a) Dense Index

* An index record appears for every search key value in the file
* In a dense primary index, the index record contains the search key value and a pointer to the first data record with that search key value

* The rest of the records with the same search key value would be stored sequentially after the first record.



b) Sparse Index

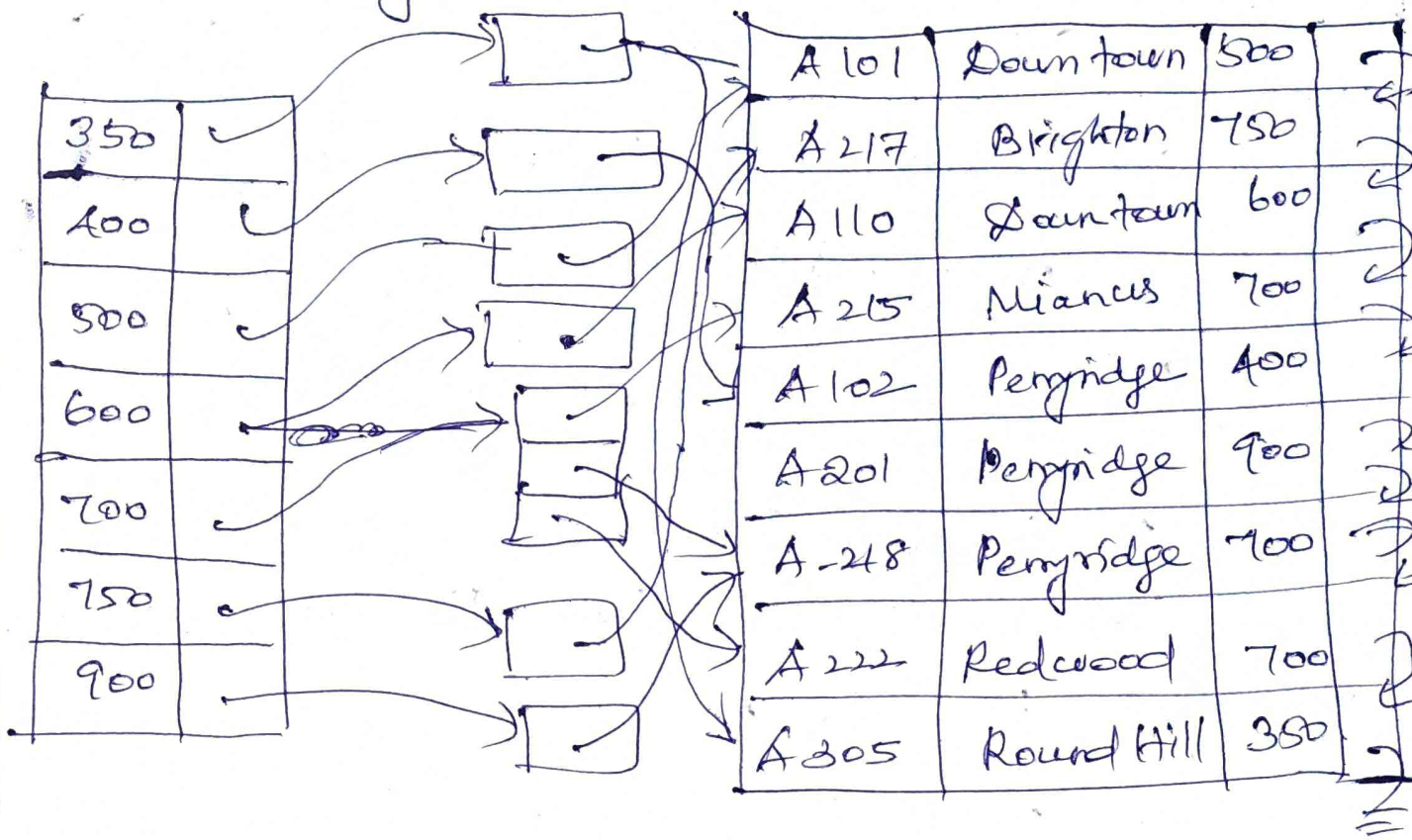
* An index records appears for only some search-key values.
* To locate a record with search-key value we
  - → Find index record with largest search-key value i.e) less than or equal to the search key value for which we are looking

* Less space and less maintenance overhead for insertions and deletions
* Generally slower than dense index for locating records.

Secondary Index:

* Secondary indices must be dense, with an index entry for every search key value and a pointer to every record in the file
* A secondary index on a candidate key looks like a dense primary index, except that records pointed to by successive values on the index are not stored sequentially
* The pointers in a secondary index do not point directly to the file
* Instead, each points to a bucket that contains pointers to the file
* The figure shows the structure of a secondary index that uses an extra level

of indirection on the accounts file on search key balance.



Advantages

* Secondary indices improve the performance of queries that use keys other than the search key of the primary index.

Disadvantages

* Secondary indices imposes a significant overhead on modification of the database

4.6. B+ Tree Index Files

* B+ Tree indices are an alternative to indexed-sequential files

Disadvantages of index-sequential files

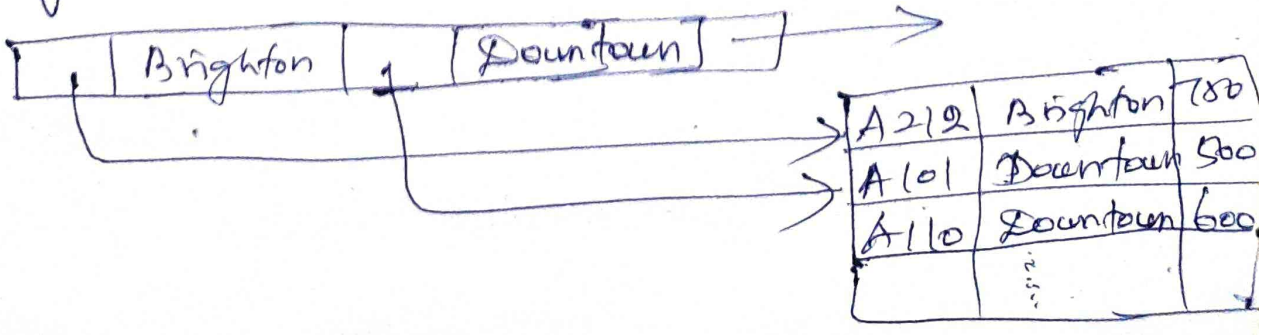
* Performance degrades as file grows, since many overflow blocks get created
* Periodic reorganization of entire file is required.

Structure of B+ Tree

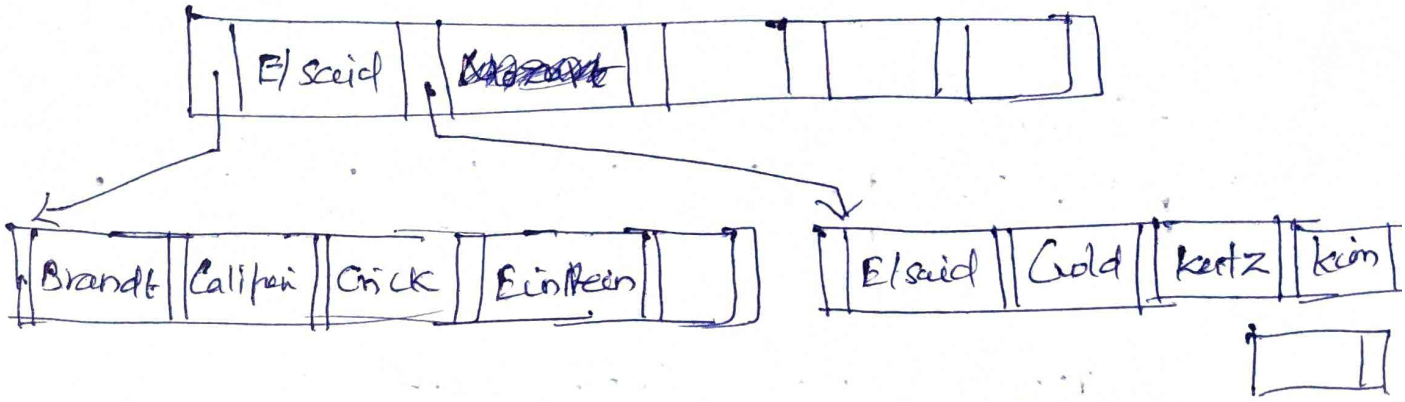
* Balanced Tree
* every path from the root of the tree to a leaf is of the same length.
* each non-leaf node of the tree has between $\lceil n/2 \rceil$ and n children.
* 

* Search key values within a node are kept in sorted order

Leaf node

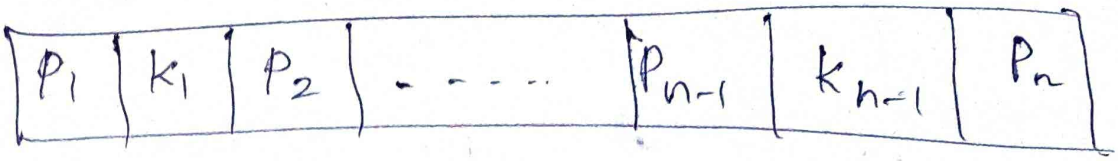


* A leaf node has between $\lceil \frac{(n-1)}{2} \rceil$ and $n-1$ values.



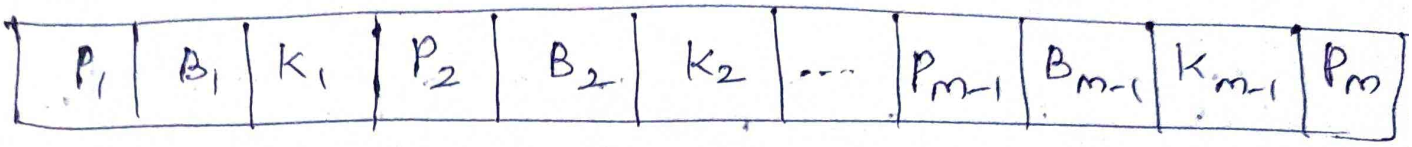
4.7. B-Tree Index Files

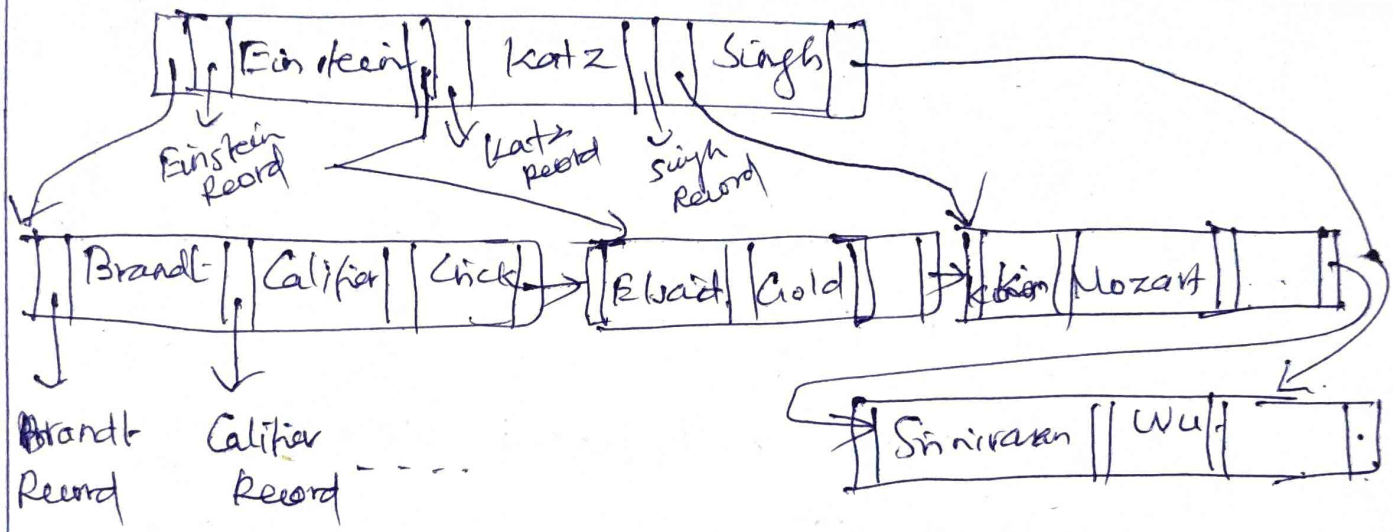
* B-Tree allows search-key values to appear only once.
* eliminates redundant storage of search keys.
* Search keys in nonleaf nodes appear nowhere else in the B-Tree;
* An additional pointer field for each search key in a nonleaf node must be included.
* Generalized B-tree leaf node - Non-Leaf node - pointers B_i are the bucket or file record pointers.



non leaf node

leaf node





Advantages

* May use less tree nodes than a corresponding B+ Tree
* Sometimes possible to find search-key value before reaching leaf node.

Disadvantages

* Only small fraction of all search-key values are found early

* Insertion and Deletion more complicated than in B^+ Trees
* Implementation is harder than B^+ Tree.

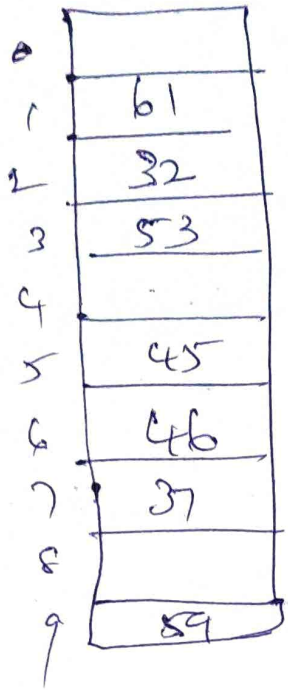
A.8 Static Hashing

* A bucket is a unit of storage containing one or more records
* In a hash file organization, we obtain the bucket of a record directly from its search-key value using a hash function
* Hash function h is a function from the set of all search-key values K to the set of all bucket addresses B.
* Hash function is used to locate records for access, insertion as well as deletion
* Records with different search-key values may be mapped to the same bucket; thus entire bucket has to be searched sequentially to locate a record.

Example of Hash File Organization

10 buckets.

37, 45, 32, 53, 61, 59, 46.



* Although the probability of bucket overflow can be reduced, it cannot be eliminated, it is handled by using overflow buckets
* Overflow chaining - the overflow buckets of a given bucket are chained together in a linked list.
* The above scheme is called closed hashing
* An alternative, called open hashing, which does not use overflow buckets, is

not suitable for database applications.

Hash Indices

* Hashing can be used for index-structure creation
* A hash index organizes the search keys, with their associated record pointers, into a hash file structure.
* Hash index is constructed as follows: apply hash function on a search key to identify a bucket and store the key and its associated pointers in the bucket.

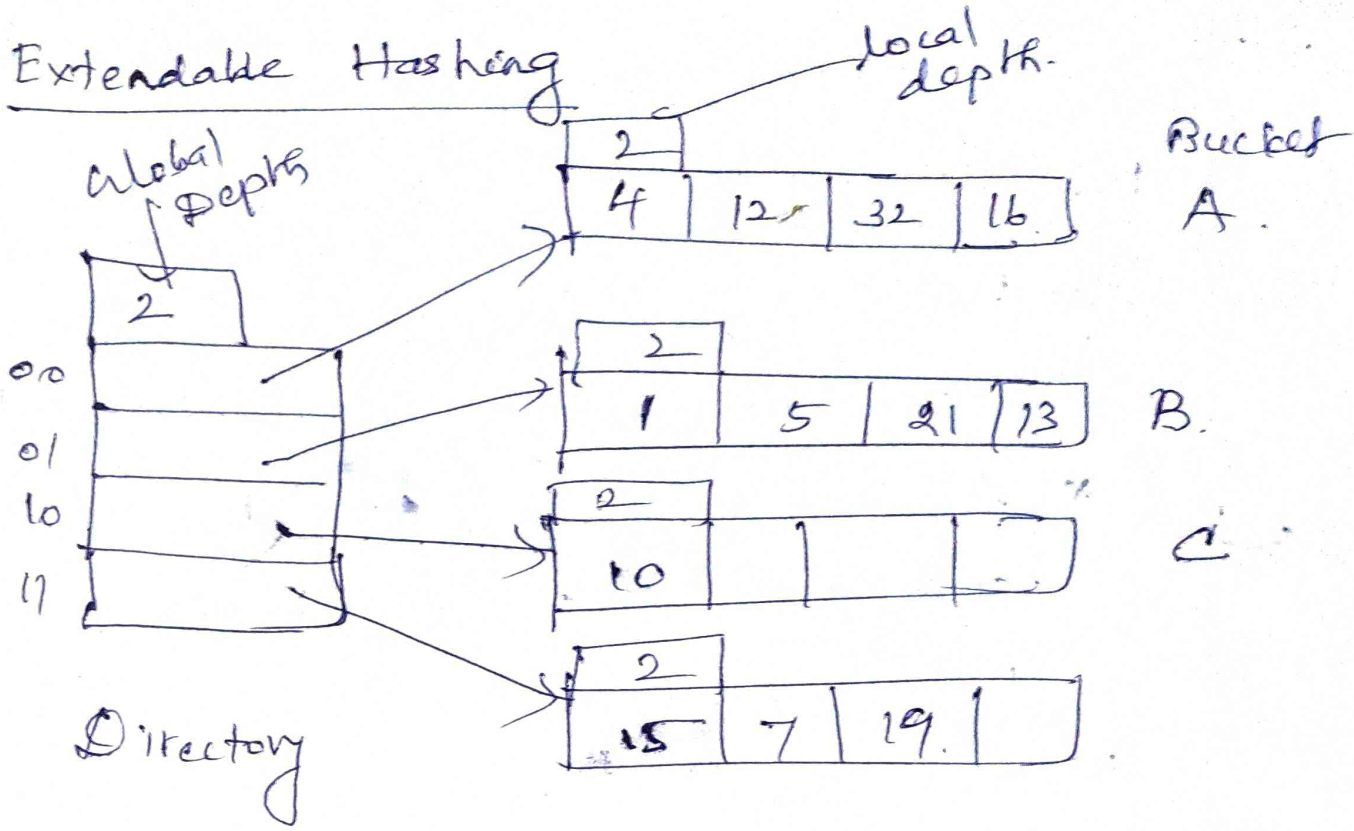
Disadvantages

* Periodic re-organization of the file is very expensive.

4.9. Dynamic Hashing

* Good for database that grows and shrinks in size
* Allows the hash function to be modified dynamically.

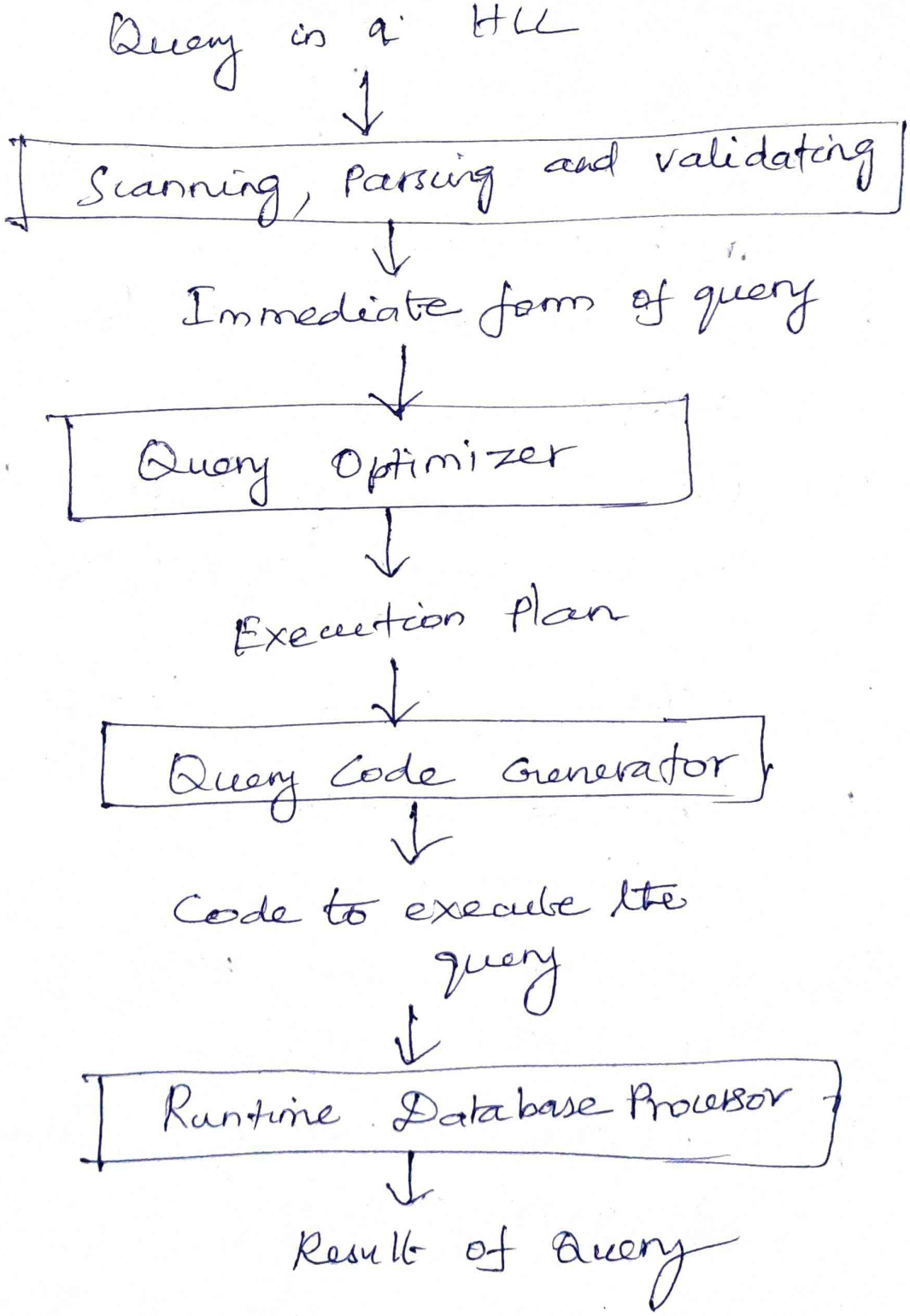
Extendable Hashing



4.10 Query Processing

- Scanner → identifies the query tokens
  parser → checks the query syntax.
- validated by checking that all attribute and relation names are valid and semantically meaningful names.
- internal representation of query is created by Query tree
- also possible with query graph.
- DBMS must devise an execution strategy or query plan to retrieve the results of query from the DB

* Query have many possible execution strategies, and the process of choosing a suitable one for processing a query is known as Query optimization



Algorithms for SELECT and JOIN operations

Search Methods for simple selection

S_1 : Linear Search

S_2 : Binary Search

S_{3a} : Using a primary Index.

S_{3b} : Using a hash Key

S_4 : Using a primary index to retrieve multiple records.

S_5 : Using a clustering index to retrieve multiple records.

S_6 : Using a secondary (B^+ tree) index on an equality comparison.

Search Methods for Complex Selection

S_7 : Conjuctive selection using an individual index.

S_8 : Conjuctive selection using a composite index

S_9 : Conjuctive selection by intersection of record pointers.

Implementing JOIN operations

* 2-way JOIN

$$R \bowtie_{A=B} S$$

* Joins involving more than 2 files are called multi-way joins

Methods

* J1 - Nested Loop Join (or Nested Block Join)
* J_2 - Single Loop Join (using an access structure to retrieve the matching records)
* J_3 - Sort-merge Join
* J_4 - Partition-Hash Join

Example:

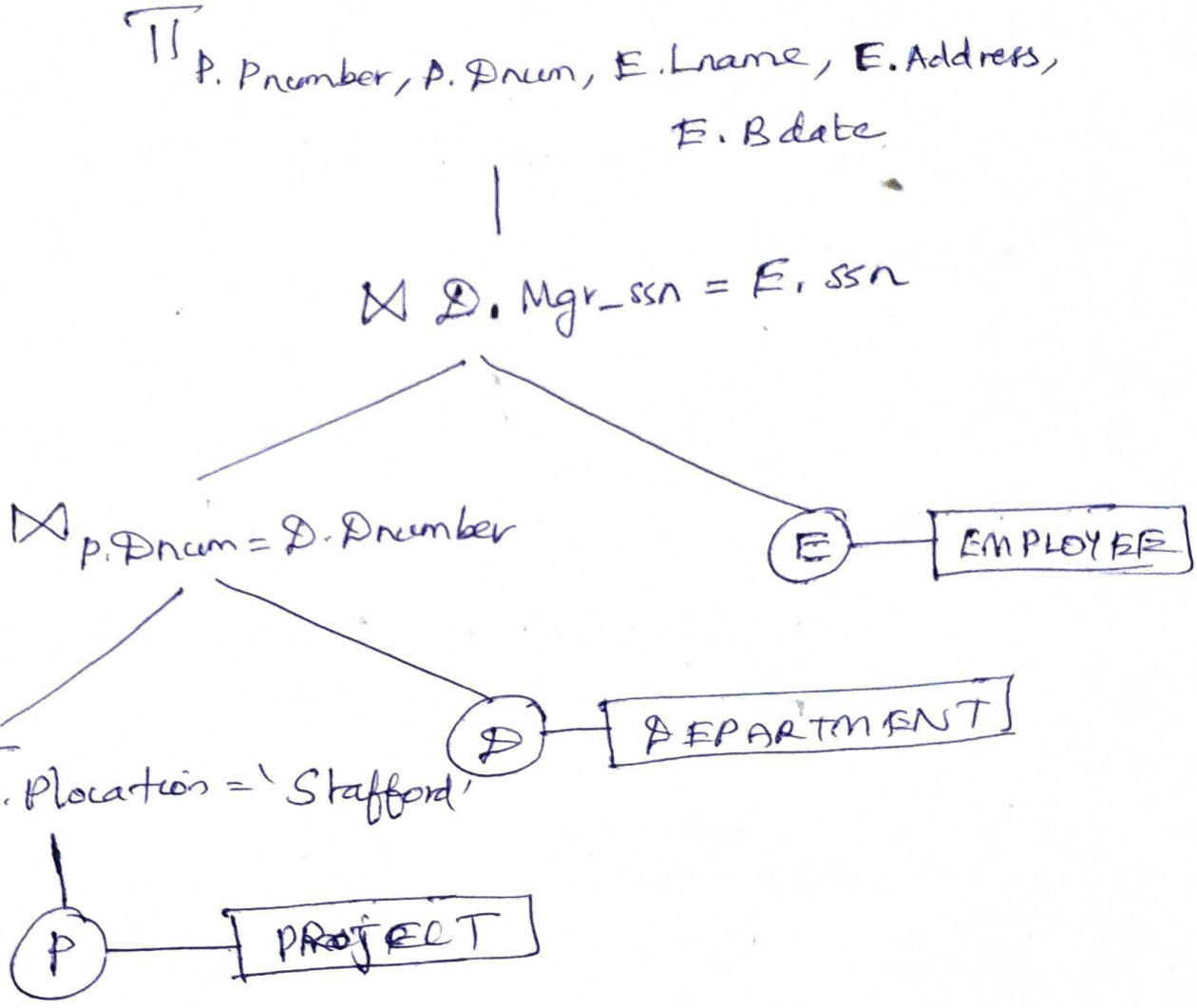
```
SELECT P.Pnumber, P.Dnum, E.Lname,
       E.Address, E.Bdate
FROM PROJECT AS P, DEPARTMENT AS D,
     EMPLOYEE AS E
```

WHERE P.Dnum = D.Dnumber AND

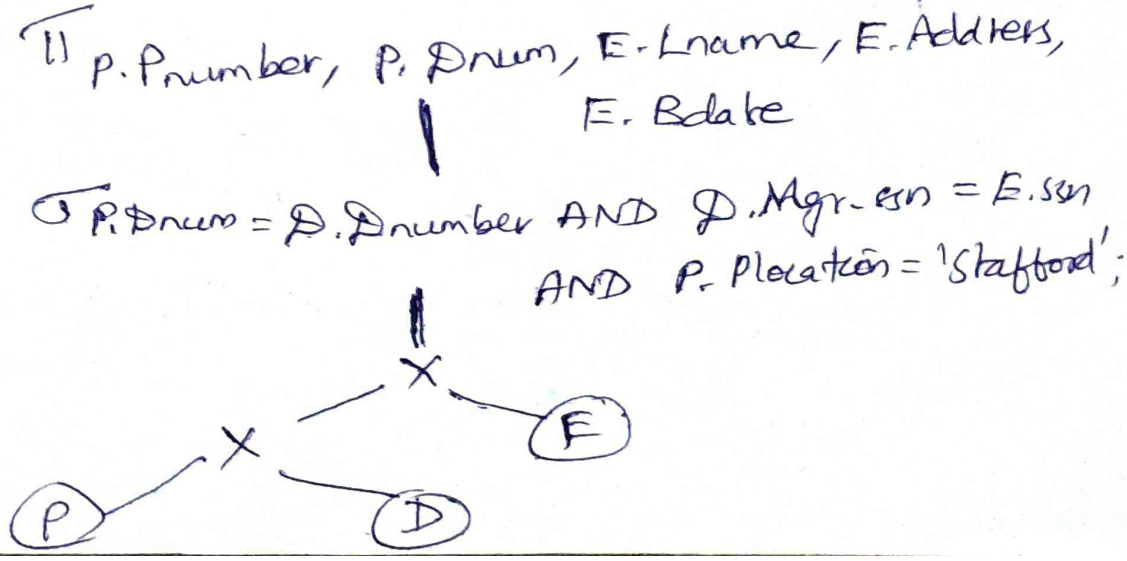
D.Mgr_ssn = E.ssn AND

P.Plocation = 'Stafford';

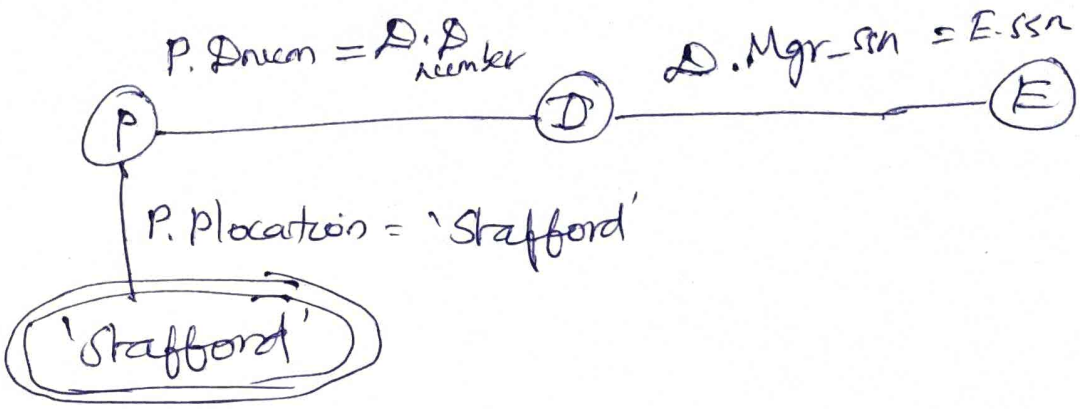
(a)



(b)



(c) Query Graph.



Cost Components for Query Execution

1. Access cost to secondary storage
2. Disk storage cost
3. Computation cost
4. Memory usage cost
5. Communication cost

UNIT- V

Advanced Topics

5.1. Distributed databases :

* Database is stored on several computers
* They do not share main memory or disk.

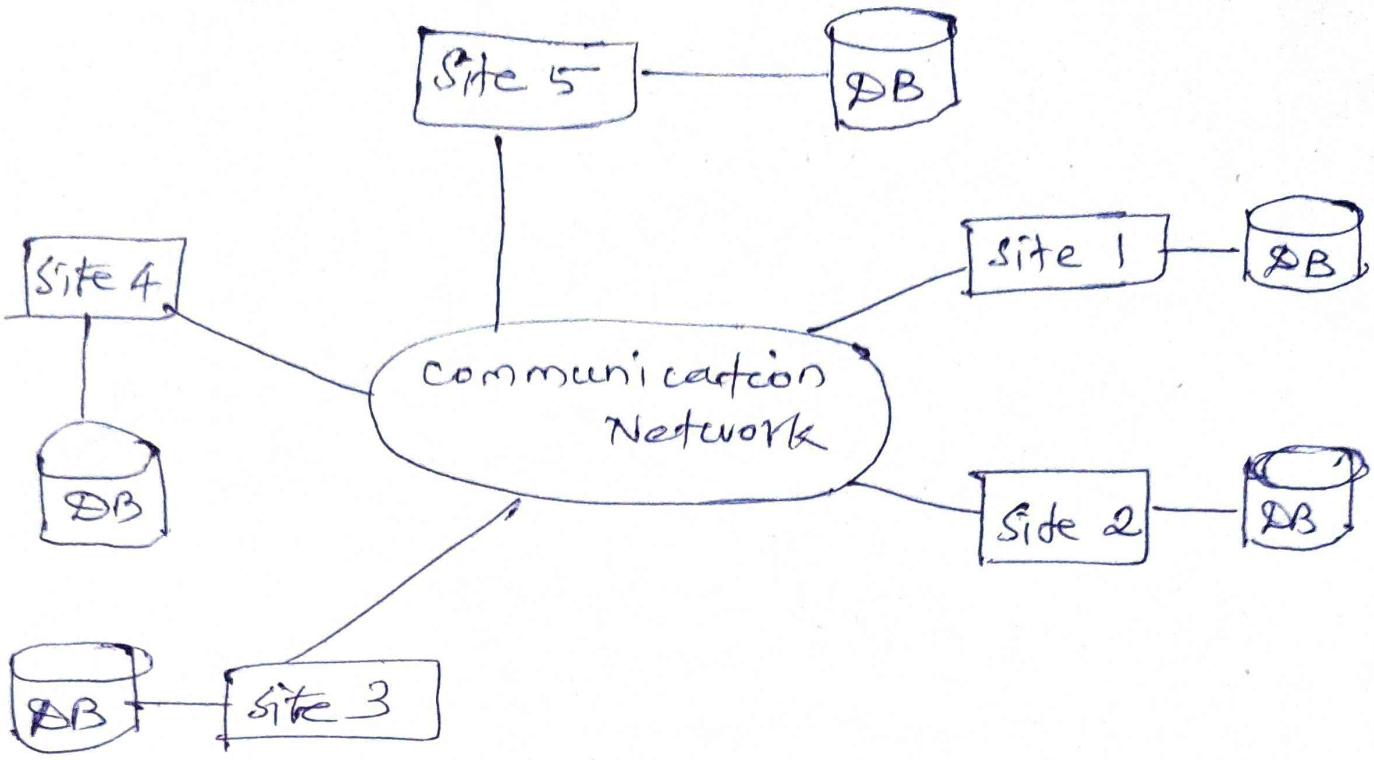
Types of Distributed Databases

(a) Homogeneous Databases
(b) Heterogeneous Databases

(a) All sites have identical schema and DBMS software
They are aware of one another, and agree to cooperate in processing user's requests.

(b) Different sites may use different schemas, and different DBMS software.
may not be aware of one another, and they may provide only limited facilities for cooperation in transaction processing.

5.1.1. Architecture



5.1.2. Data storage

* 2 approaches to store a relation in distributed DB

(1) Replication - system maintains several identical replicas (copies) of the relation and stores each replica at a different site

(2) Fragmentation - system partitions the relation into several fragments and stores each fragment at a different site.

Data Replication

* copy of relation is stored in two or more sites.
* In full replication a copy is stored in every site in the system
* There are number of advantages and disadvantages to replication
  (1) Availability
  (2) Increased parallelism
  (3) Minimizes movement of data
  (4) Increased overhead on update

Data fragmentation

* Relation is fragmented, it is divided into fragments $r_1, r_2 \ldots r_n$
* 2 types
  a) Horizontal – each tuple to one or more fragments
  b) Vertical – decomposing

Transparency

* users should not be required to know where the data are physically located nor how the data can be accessed

at the specific local site

* This characteristic, is called data transparency.

* 2 forms
  a) Fragmentation transparency
  b) Replication transparency
  c) Location Transparency

5.1.3 Transaction Processing

* Access to the various data items in a distributed system is usually accomplished through transactions, which must preserve the ACID properties

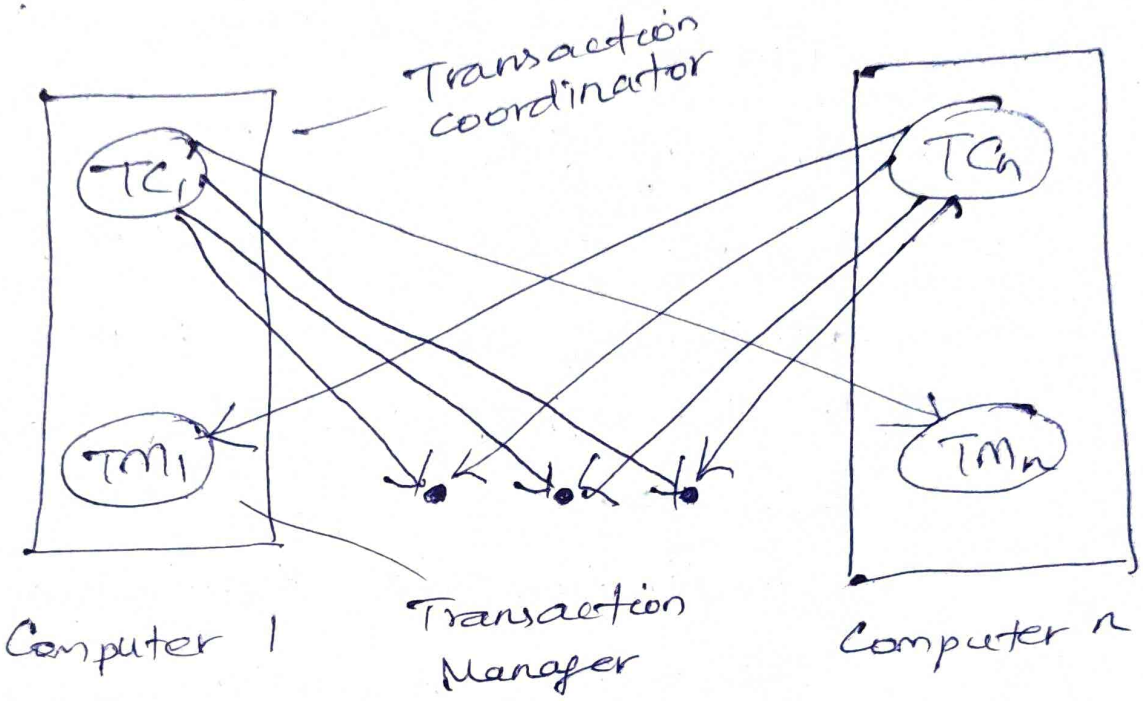
* 2 types of transactions
  (1) Local transaction
  (2) Global "

System structure

* each site has its own local transaction manager, whose function is to ensure the ACID properties

* Various transaction managers cooperate to execute global transactions.

* Transaction coordinator coordinates the execution of the various transactions initiated at that site.



* Each transaction manager is responsible for
  - Maintaining a log for recovery purposes.
  - participating in an appropriate concurrency-control scheme to coordinate the concurrent execution of the transactions executing at that site.

* For each such transaction, the transaction coordinator is responsible for.

  - Starting the execution
  - Breaking the transaction into a number of sub transactions and distributing these sub transactions to the appropriate site for execution.
  - Coordinating the termination of the transaction, which may result in the transaction being committed at all sites or aborted to all sites

System failure modes

* A distributed system may suffer from some types of failures.
* Basic failure types

  1) failure of a site
  2) Loss of messages
  3) Failure of a communication Link
  4) Network Partition

5.2. Object-based Databases

* An object corresponds to an entity in E-R model.
* The object-oriented paradigm is based on encapsulating code and data related to an object into single unit.
* Object-oriented data model is a logical data model (like the E-R model)

5.2.1 Object database Concepts

* An object has associated with it
  - A set of variables that contain the data for the object
  - A set of messages to which the object responds; each message may have zero, one or more parameters
  - A set of methods, each of which is a body of code to implement a message; a method returns a

value as the response to the message.

Messages and Methods

* Methods can be read-only or update methods
* Read only methods do not change the value of the object.
* Every attribute of an entity must be represented by a variable and 2 methods, one to read and the other to update the attribute.

Object Classes :

* Similar objects are grouped into a class; each such object is called an instance of it's class
* All objects in a class have the same
  - Variables, with the same types
  - message interface
  - methods
* They may differ in the values assigned to variables.

* Classes are the entity sets in the ER model.

Class Definition Example

```
class employee
{
    String name;
    String address;
    date   start_date;
    int    salary;

    int annual_salary();
    String get_name();
};
```

variables

Messages

* Methods to read and set the other variables are also needed with strict encapsulation
* Methods are defined separately

```
String get_address()
{
    return address;
}
```

```
int set_address (string new_address)
{
  address = new_address;
}
```

while the method employment_length()
would be defined as:

```
int employment_length()
{
  return today() - start date;
}
```

5.2.2 Object-Relational features

Inheritance
Object identity
Object identifiers
Object containment

5.2.3 ODMG Object model, ODL, OQL

* ODMG (Object Data Management Group) is a proposed standard that is known as the ODMG-93 or ODMG 1.0 standard

* The standard is made up of several parts, including the object model, the object definition language (ODL), the object query language (OQL) and the bindings to object-oriented programming language.

Object Model of ODMG:

* Data model upon which the object definition language (ODL) and OQL are based.

* provide a standard data model for object databases, just as SQL describes a standard data model for relational databases.

Objects and Literals.

* basic building blocks of the object model.

* object has * an object identifier
  * a state (value)

Literal has a value but no object id.

5.24. Object-Oriented Data Model (ODL)

* Support semantic constructs of the ODMG object model and is independent of any particular programming language.
* Mach use is to create object specifications ie) classes and interfaces
* ODL is not a full programming Language.

Interface

class

Relationships

1 : 1

1 : N

M : N

Inheritance

(is-a)
using

class
inheritance
extends

5.2.5 Object Query Language (OQL)

* proposed for ODMG object
* Simple queries

Ex: Select D.Dname
from D in DEPARTMENTS
where D.college = 'Engineering';

DEPARTMENTS — returns a reference to the collection of all persistent DEPARTMENT objects.

CS_DEPARTMENT; — returns a reference to that individual object of type DEPARTMENT

Other features of OQL

* Specifying braces and named queries
* View mechanism uses the concept of a named query.
* Query definition is persistent.

V_1 : define Has-majors (Dept_name) as

select S

from S in STUDENTS

where S.Majors_in.Dname = Dept_name

5.3 XML Databases :

* Extensible Markup Language
* Defined by WWW Consortium
* Derived from SGML (Standard Generalized Markup Language) but simpler to use than SGML

5.3.1. XML Hierarchical model :

* Documents have tags giving extra information about sections of the document.
* E.g. <title> XML </title>
* Users can add new tags, and separately specify how the tag should be handled for display

* The ability to specify new tags, and to create nested tag structures make XML a great way to exchange data, not just documents.

* Tags make data (relatively) self-documenting

* E.g.

```
<university>
  <department>
    <dept_name> Comp. Sci </dept_name>
    <building> Taylor </building>
    <budget> 100000 </budget>
  </department>

  <course>
    <course_id> CS_101 </course_id>
    <title> Intro. to Computer Science </title>
    <dept_name> Comp. Sci </dept_name>
    <credits> 4 </credits>
  </course>
</university>
```

5.4.2. XML Schema and DTD:

* XML documents are not required to have an associated schema
* However, schemas are very important for XML data exchange
* Otherwise, a site cannot automatically interpret data received from another site.
* Two mechanisms for specifying XML schema
  * Document Type Definition (DTD)
  * XML Schema

1. DTD

* Type of an XML document can be specified using a DTD.
* DTD constraints structure of XML data
  - what elements can occur
  - what attributes can an element have
  - what sub elements can occur inside each element, and how many times

* DTD doesnot constrain datatypes
* All values represented as strings in XML
* DTD syntax
  * <!ELEMENT element (subelements-specification) >
  * <!ATTLIST element (attributes) >

University DTD

<!DOCTYPE university [
<!ELEMENT university ((department|course|instructor|teaches)+) >
<!ELEMENT department (dept name, building, budget) >
<!ELEMENT course (course id, title, dept name, credits) >
<!ELEMENT instructor (IID, name, dept name, salary) >

<!ELEMENT teaches (IID, courseid)>
<!ELEMENT dept name (#PCDATA)>
<!ELEMENT building (#PCDATA)>
<!ELEMENT budget (#PCDATA)>
<!ELEMENT courseid (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT credits (#PCDATA)>
<!ELEMENT IID (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT salary (#PCDATA)>

]>

XML Schema

* XML Schema is a more sophisticated schema language which addresses the drawbacks of DTDs. Supports
  - Typing of values
  - Eg. integer, string etc.
  - Also, constraints on min/max values

* User-defined, complex types
* Many more features, including
  - uniqueness and foreign key constraints, inheritance
* XML Schema is itself specified in XML Syntax, unlike DTDs
* Standard representation
* XML Schema is significantly more complicated than DTDs.
* XML Schema is integrated with namespaces.
* XML Namespaces provide a method to avoid element name conflicts.
* In XML, elements names are defined by the developer.
* This often results in a conflict when trying to mix XML documents from different XML applications.

```
<table>
    <td> Apples </td>
    <td> Bananas </td>
</table>
```

```
<table>
  <name> African Coffee Table </name>
  <width> 80 </width>
  <length> 120 </length>
</table>
```

XML schema version of University DTD

```
<xs: schema xmlns: xs = " http://www.w3.org/
        2001/XMLSchema" >
  <xs: element name = "university" type = "universityType" />
  <xs: element name = "department" >
    <xs: complexType>
      <xs: sequence>
        <xs: element name = "dept name" type = "xs:string" />
        <xs: element name = "building" type = "xs:string" />
        <xs: element name = "budget" type = "xs:decimal" />
      </xs: sequence>
    </xs: complexType>
  </xs: element>
  ----
```

```
<xs: element name = "instructor" >
  <xs: complex Type>
    <xs: sequence>
      <xs: element name = " IID" type = "xs: string"/>
      <xs: element name = "name" type = "xs: string"/>
      <xs: element name = "deptname" type = "xs: string"/>
      <xs: element name = "salary" type = "xs: decimal"/>
    </xs: sequence>
  </xs: complexType>
</xs: element>

- - - -

<xs: complexType name = "University Type">
  <xs: sequence>
    <xs: element ref = "department" minOccurs = "0"
                 maxOccurs = "unbounded" />
    <xs: element ref = "course" minOccurs = "0"
                 maxOccurs = "unbounded" />
    <xs: element ref = "instructor" minOccurs = "0"
                 maxOccurs = "unbounded" />
    <xs: element ref = "teaches" minOccurs = "0"
                 maxOccurs = "unbounded" />
```

```
</xs:sequence>
</xs:complexType>
</xs:schema>
```

5.3.3. XQuery:

* Translation of information from one XML schema to another
* Querying on XML data
* Above two are closely related, and handled by the same tools
* Standard XML querying/translation languages
  - XPath: Simple language consisting of path expressions
  - XSLT: Simple language designed for translation from XML to HTML
  - XQuery: An XML query language with a rich set of features

* Xquery is a general purpose query language for XML data
* Currently being standardized by the W3C

* XQuery is derived from the Quilt query language, which itself borrows from SQL, XQL and XML-QL
* XQuery can be used to:
  - Extract information to use in a web service
  * Generate summary reports
  * Transform XML data to XHTML
  * Search web documents for relevant information
* XQuery uses functions to extract data from XML documents.
* The doc() function is used to open the xml file: doc("books.xml")
* XQuery uses path expressions to navigate through elements in an XML document
* The following path expression is used to select all the title elements in the "books.xml" file: doc("books.xml")

  /bookstore/book/title

* XQuery uses predicates to limit the extracted data from XML documents

* The following predicate is used to select all the book elements under the bookstore element that have a price element with a value that is less than 30.

* doc("books.xml")/bookstore/book[price < 30]

* Xquery uses a syntax. for SQL from
where SQL where
orderby SQL order by
return SQL select

```
for $x in doc("books.xml")/bookstore/book
where $x/price > 30
orderby $x/title
return $x/title
```

5.4. Information Retrieval

* process of retrieving documents from a collection in response to a query by a user.

5.4.1. IR concepts

* IR systems use a user's information need expressed as a free-form search request.

* An IR System can be characterized at different levels: by types of users, types of data, and the types of the information need, along with the size and scale of the information repository it addresses.

- Types of users

  * user may be an expert user who is searching for specific information that is clear in mind and forms relevant queries for the task, or a layperson user with a generic information need.

- Types of data

  * Search systems can be tailored to specific types of data.

* For example, the problem of retrieving information about a specific topic may be handled more efficiently by customized search systems that are built to collect and retrieve only information related to that specific topic.

- Types of Information Need

* In the context of Web search, users information needs may be defined as navigational, informational, or transactional

a) Navigational — refers to finding a particular piece of information that a user needs quickly

b) Informational — refers to find current information about a topic.

c) Transactional — refers to reach a site where further interactions happens.

5.42 Retrieval models

* There are 4 main statistical models

1) Boolean 2) Vector Space
3) Probabilistic 4) Semantic Model

Boolean Model

* Documents are represented as a set of terms.
* Queries are formulated as a combination of terms using the standard Boolean logic set-theoretic operators such as AND, OR and NOT.
* There is no notion of ranking
* All documents are equally important
* easy to associate meta data information and write queries that match the contents of the documents as well as other properties of documents, such as date of creation, author, and type of document

Vector Space Model

* provides framework in which term weighting, ranking and relevance feedback are possible.

* Documents are represented as features and weights of term features in an n-dimensional vector space of terms.

* Document term weight w_{ij} (for term i in document j) is represented based on some variation of the TF (Term Frequency) or TF-IDF (Inverse Document Frequency) scheme

* TF-IDF is a statistical weight measure that is used to evaluate the importance of a document word in a collection of documents.

* The following formula is used.

$$\text{Cosine}(d_j, q) = \frac{\langle d_j \times q \rangle}{\|d_j\| \times \|q\|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}$$

d_j - document vector

q - query vector

w_{ij} - weight of term i in document j

w_{iq} - weight of term i in query vector q

$|V|$ - no. of dimensions in the vector that is the total no. of important keywords (or features)

Probabilistic Model

* Based on Probability Ranking Principle
* Decide whether the documents belong to the relevant set or the nonrelevant set for a query
* Given a document representation D of a document, estimating the relevance R and nonrelevance NR of that document involves computation of conditional probability $P(R|D)$ and $P(NR|D)$
* These conditional probabilities can be calculated using Baye's Rule

$$P(R|D) = P(D|R) \times P(R) / P(D)$$

$$P(NR|D) = P(D|NR) \times P(NR) / P(D)$$

* relevant if $P(R|D) > P(NR|D)$

Semantic Model

* Includes different levels of analysis
  1) Morphological
  2) Syntactic
  3) Semantic analysis.

* Morphological analysis, roots and affixes are analysed to determine the parts of speech of the words.

* Syntactic analysis follows to parse and analyze complete phrases in documents

* Resolve word ambiguities

* Requires techniques from artificial intelligence and expert system.

5.4.3 Queries in IR Systems

Keyword Queries

* Simplest and most commonly used forms of IR queries

* Query keyword terms are implicitly connected by a logical AND operator.

* either and or or type of matching is applied
* Some system don't follow the order of words.
* All retrieval models provide support for keyword queries

Boolean Queries

* Use of AND, OR, NOT, () connectives
* Complex boolean queries can be built.
* No ranking is possible

Phrase Queries

* represented using an inverted keyword index for searching
* each phrase should be encoded in the inverted index or implemented differently.
* Consists of sequence of words forms the phrase

Wildcard Queries

* Supports regular expressions and pattern matching

* usually words with any trailing characters (data * could retrieve data, database, dataset, datapoint ...)

Natural Language Queries

* aim to understand the structure and meaning of queries written in natural Language text.
* employs techniques like shallow semantic parsing of text, or query reformulations based on natural language understanding.
* The system tries to formulate answers for such queries from retrieved results.