DBMS

Condensed Notes

May 4, 2025

Contents

Chapter 1

Intro

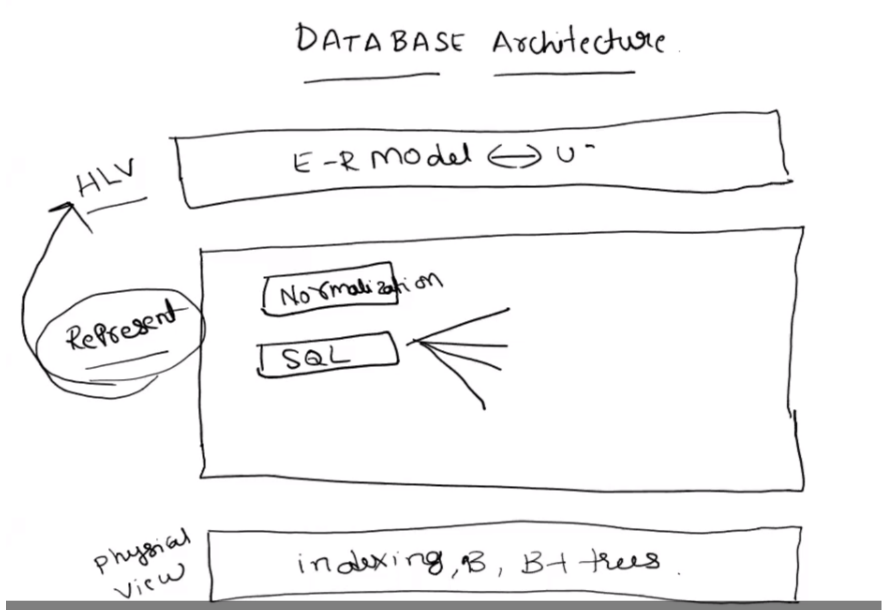
1.1 Definition

- **Data** : A fact that can be recorded
- **Record/Tuple** : Collection of interrelated Data
- **Database** : Collection of records
- **Datawarehouse** : a database containing large amount of data separate from client data for faster operation on client side
- **DBMS** : a Database with a software which manipulates the data in database

1.2 Relational data base management system

- **relation** : any subset of cartesian product
- **RDBMS** : a database which works on the principles of relation (i.e no repetations pf tuple etc.)

1.3 database architecture



Chapter 2

ER Model

- **ER model** : A datamodel view in which info is storerd in d/b is viewed as entities and relationships.
- **diagram convention** : rectangle represents entity diamond represents relation attribute represent by ovals
- **Entity** : A real world object with independent existence eg. student, professor etc.
- **Relatioships** : Asspciation between entities
- **Attributes** : the columns of entities (table)
- **relationship set** : the total no. of entities partcipating in a relationship aka degree of relation we have binary relation,ternary relation etc.
- **Types of relation** we have 1 to 1 relation (an attribute related to only one attribute of another table) m to n relation(student course example) etc. in 1 to n relation n will be primary key
- **Total participation** : when all the rows of table is participating in the relation represented by double line
- **partial participation** : one that is not total participation represented by single line

2.1 Cardinality

based on participation and relationship type there are two types of cardinality

- **Minimum cardinality** it is 1 for total participation and 0 for partial participation
- **Maximum cardinality** it is n if an entity (row of table) is maping to a maximum of n other entity (row in a table) by a relation R
- cardinality is a function of single entity and single relation

2.2 Entities

there are three types of entity

- **Strong** : if one is able to define a primery key from the entity
- **Weak** : if primary key can not be dertermined from a relation the relation with a weak entity is weak relation the key we define with primery key of another relation is partial key represented by concentric rectangle
- **Associative** : an Associative entity is a table which associates other table in many to many represented by a dimond in a rectangle.

2.3 Attributes

attributes are of following types

- **simple** : an attribute wich can not be further divided eg sl.no, year, etc.
- **Composite attribute** : an attribute which can be further divided eg address contains street no etc, name contains first name last name etc.
- **single valued** : an attribute which can only have singlr value
- **Multi valued** : an attribute where multiple values are allowed eg multiple phone numbers

- **Stored** : an attribute which can be used to calculate / derive some other attribute
- **Derived** : an ttribute which can be derived or calculated from other attribute

2.4 Conversion of ER models to relation

2.4.1 Conversion of entity into table

- simle attributes are to converted directly
- derived attributes are converted from their derived counterparts

2.4.2 Conversion of weak entity into table

- create a foreign key field in the weak entity table
- the foreign key and the partial key together forms the primary key
- we must use a on delete cascade so that if the data from strong entity is deleted the corresponding data in weak entity also gets deleted.

2.4.3 Conversion of m:n relation into table

- create an Associative table which contains relation stored as primary key map from both table in relationship.

2.4.4 Conversion of 1:1 relation into table

- modify the table with total participation.
- add a foreign key corresponding toother table.

2.4.5 Conversion of one to many into table

- modify the many side of table.
- add the foreign key corresponding to other table.

2.4.6 Conversion of multi-valued attribute

- instead of duplicating rows in main table due to multi value.
- create another table with the multi value corresponding to primary key of other table.

2.5 Minimization Of ER Diagram

- 1:1 with partial participation at both end —— combime relation with any of the table —— 2 table
- 1:1 with total participation at one side —— combine all three table —— one table
- 1:M with total participation at one side —— combine relation with m side of table —- 2 tables
- M:M —- 3 tables

Chapter 3

Schema Refinement & Normal Forms

3.1 Introduction

a general db is a collection of interreletaed table we can a assign **user nam/schema** for a group of table. these user name can also be used to modify access to various table

3.2 Functional dependencies

we say an attribute is dependent on other set of attributes if we are able to identify the dependent attributes uniqly from a given set of determining attributes

given a table T(A,B,C)
if $A \to B, C$ (A determins B and C)
then given a value in A we can uniqly identify B and c
Rules for functional dependency

- the left side of arrow should be unique other wise if there are duplicates in left hand side the right hand side should also match.
- it should uniqly identify right hand side

Types of FD's

- given two sets x and y a fd $x \to y$ is triavial FD if $x \supseteq y$

- given two sets x and y a fd $x \rightarrow y$ is non-triavial FD if $x \cap y = \phi$
- if a FD can be decomposed into both trivial and non trivial FD it semi trivial FD

Armstrong Axioms

- Reflexivity if $x \supseteq y$ than $x \rightarrow y$
- Transivity if $A \rightarrow B$ and $B \rightarrow C$ than $A \rightarrow C$
- Augmentation if $x \rightarrow y$ than $xz \rightarrow yz$
- union if $x \rightarrow y$ and $x \rightarrow z$ than $x \rightarrow yz$
- spliting if $x \rightarrow yz$ than $x \rightarrow y$ and $x \rightarrow z$

Attribute set closure: the set of all atrributes that can be functionaly determined by an attribute s is called attribute closure of s

see l3/18 for examples on attribute closure

Super Key: an attribute or set of attributes that is able to determine all attributes of relation OR the attribute or attribute set in whose closure les all attributes of that relation

super key is not minimal if A is superkey of a realtion that AX is also a superkey

the minimal of all superkeys is called **Candidate key**

3.3 Applications of attribute closure

- finding candidate keys
- finding all super keys
- check if attribute set is candidate key/super key
- computing F D
- membership test

3.3.1 No. of SK that can be made with a CK

$2^{totalno.ofattributesinrelation - noofattributecomprisingthatCK}$

3.3.2 No. of SK that can be made with a no. CKs

$\sum 2^{totalno.ofattributesinrelation-noofattributecomprisingthatCK} - 2^{totalno.ofattributesinrelation-no.ofattributesinCKs}$

Primary Key:one of the CK is selected as PK
rules for sselecting PK

- An attribute of type integer can be selected as PK
- A CK having less no. of attributesof the relation

3.4 Foreign Key

definition: attribute or set of attributes which are referencing to PK/Alternate key of same or multiple tables

a key used in **referencing table** from **referenced table** is called as a fK to a PK.

the referenced and referencing table may be same
arrow head is toeards refrenced where PK is there

3.5 while deleting/updating the records from refrenced following things can be done to related records of refrencing relation

- on delete/update cascede
- on delete/update no action
- on delete/update set null

3.6 FD set closure/f+ closure

a set of all FDs that can be formed from given FDs go to lecture to see how.

3.7 Membership test

it implies if the given FDs is there in a FD closure

3.8 Normalization

Insertion anamoly: when inserting a new record un-necessory tupple values need to be inserted.

Deletion Ananmoly: if a value in a field is deleted then all the tupple having that value will be deleted.

Updation Anamoly: if a particular information needs changing we have to change all the repeated values due to redundency

to avoid such ananmolies due redundency we divide bigger table into smaller table by normalizing

3.9 Disadvantages of Normalization

due to division of bigger tables into smaller tables while getting information there are a lot of tables to join thereby leading to costlier ops.

3.10 Prperties of Normalization

- Lossless join decompsition
- dependency preservation

3.11 Lossless join decomposition

join operation is select on cross product

A table is decomposed in two or more and there is a common column among them joining is basically doing a cross product and then selecting rows whose values are common in that common column

lossless decomposition a decomposition where on "joining" we get the table back exactly **lossy decomposition** a decomposition where on "joining" we get possibly more rows then present initially in the table.

3.12 Rules for finding if relation is lossless

let R be decomposed into relational schema with FD decomposed into R1 and R2

R1 and R2 will be lossless if

- $R1 \cup R2 \equiv R$
- $R1 \cap R2 \neq \phi$
- $R1 \cap R2 \rightarrow R1$ $R1 \cap R2 \rightarrow R2$ or

3.13 Dependency Preservation

let R be relation schema with fd set decomposed into R1,R2,R3.....Rn
with fd sets fd1,fd2,fd3.....fdn respectively
then if $fd1 \cup fd2 \cup fd3..... = fd$
the decomposition is dependency preserved

3.14 Normalization and its properties

The redundebcy level at which the table will be decomposed will be explained by normal forms

Levels of Normalization

- 1 NF
- 2 NF
- 3 NF
- BCNF
- 4 NF

3.15 1 NF

R is in 1 NF only if R does not contain attributes having multiple values in a single Row.

3.16 2 NF

R is in 2 NF if
it is in 1 NF
it does not have **partial dependency**: if proper subset of CK determines a non prime attributes

3.17 3 NF

R is in 3 NF
it is in 2 NF
for every non trivial FD $x \to y$
x should be superkey or y should be prime attribute

3.18 BCNF

R is in BCNF
it is in 3 NF
for every non trivial FD $x \to y$
x should be superkey
A relation in 3 NF but not in BCNF always contains overlapping CKs

3.19 Cannonical Cover or Minimal Cover

for a given set of FDs in a relation can be derived minimal set of FDs without loosing the dependencies of the orignal set is called minimal cover.

rules for minimal cover

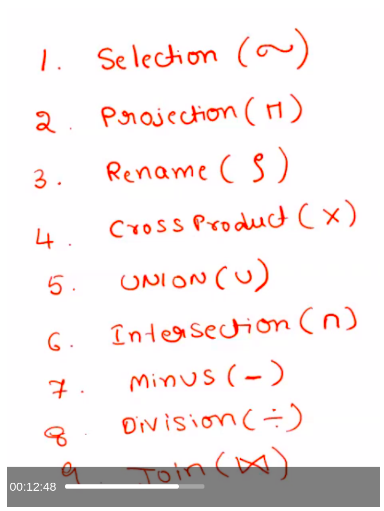
- make all fds have single attribute on right hand side use splitting rule
- eliminate left hand side to single attribute by taking attribute closure
- check for redundent dependencies by taking attribute closure

see (l 15/18)

Chapter 4

Relational ALgebra

4.1 Relational Algebra



4.2 Selection

used for selecting subset of tupples

4.3 projecton

select subsets of attributes projection removes the duplicates from the formed tuple

4.4 Rename

to create an alias for a relation

4.5 Cross Product

cross product of tuples of two tables

4.6 Joins

types of join

- - Inner Joins
  - Outer Join
  - - ∗ right outer join
      ∗ left outer join
      ∗ full outer
- - Natural
  - Natural join on condition

4.6.1 Natural join

take a cross product of two tables and select the tupples having same value on common columns.

duplicate columns are merged

4.6.2 Natural join with condition aka inner join

take a cross product of two tables and select the tupples satisfying the conditions

4.6.3 Left outter join

include all unmatched tuples on left side of relation

take all rows of LHS table if the condition stisfys then join the RHS tupple to LHS other wise join null filled RHS with LHS

4.6.4 Right outter join

similler to left outter join

4.6.5 Full outter join

union of left outer join and right outer join

4.7 Union/Intersection/minus

it gives the set of tupples from two relations with the correponding opperation applied

conditions for applying union opperation

- Both relations should have same degree(bo, of columnns)
- the domain of corresponding attribute should be same

4.8 Division operator

will be used in the scenarios to find some tuopple which satisfies the condition of type **All** eg: student who subscribed for all the courses ($student \div courses$) etc.

Chapter 5

TRC DRC

5.1 TRC - tupple relational calculus

Tupple: a variable that can store any as its value

trc Query: a query based on a peridicate which returns a sub set of tupples

Chapter 6

SQL

6.1 SQL operators

- Based on sarch condition
- Comparision operators
- range queries
- set membership functions
- pattern matching
- null condition
- aggregate functions
- Groub by, having, order by functions
- join operators
- any/all/exist operator

6.2 Aggregate functions

rules

- they cant be use in Comparision use sub quary instead
- these functions works on a single row

6.3 Groub by

characterstics

- applied on row having duplicate values
- groups the data from selecrt tables and produces a single summary row of each groups
- all colums names should appear in group by unless it is used in aggregate functions
- where clause must appear before groub by
- used in filtering from table durectly while having is used to filter from groub by

6.4 having clause

- aliasing cannot be used

6.5 where vs having

- aggeregate function cannot be used in where but vcan be used in having

6.6 order by clause

6.7 like clause

- % indicates any sequence h% = hyaaa,haaa etc.

6.8 defrent types of sub queries

6.8.1 nested

- inner query does not relate to outer query
- inner query executed first - bootom up approach

6.8.2 corelleted

- inner and sub queries are related
- top down approach - for every entry in outer query comparision with inner query takes place

6.8.3 exist clause

tests if inner subquery returns null or non null values returns true for non null values.

checks for the existance if records

6.8.4 join

select on cross product

Chapter 7

Transactions and concurrency control

7.1 Transaction

A transaction is a logical unit of work in dbms it comprises of a series of DMl operations to be performed atomically.

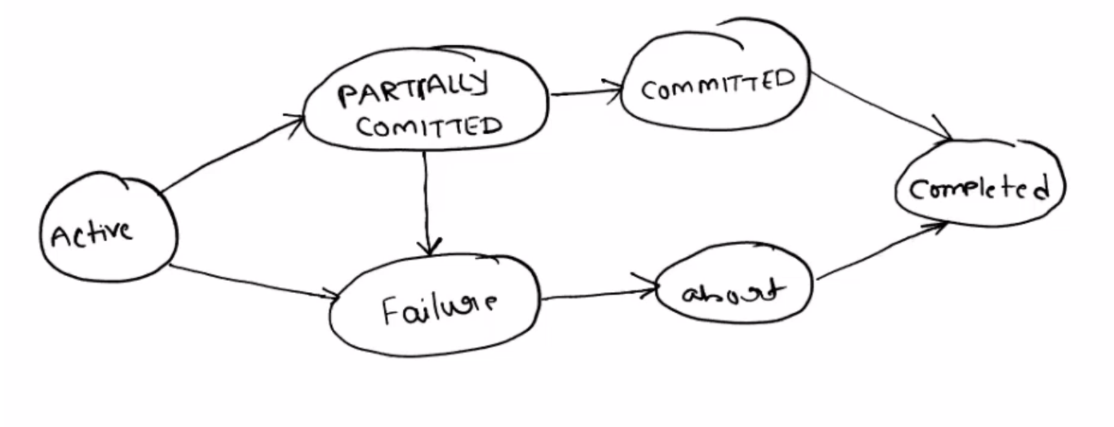
Transaction should follow 4 proporties

- Atomicity : the complete operation must be performed or none at all
- Consistency : the transaction should leave the db in Consistent state
- Isolation : updates made by transaction should be stored in physical storage
- Durability : each transaction must be carried out in Isolation and not in concurrent manner

7.2 States of transaction

7.3 schedules

A schedule is a list of actions (reading, writing, aborting, or committing) from a set of transactions, and the order in which two actions of a transaction T appear in a schedule must be the same as the order in which they appear in T.



7.3.1 types of schedules

- serial schedule : the transaction occur in order without any interleaving
- concurreny schedule : the transaction occur in parellel interleaving yakes place

7.3.2 Serializability

whether the given concurrent schedule is equivalent to serial schedule

7.3.3 problems due to serial sechedule

- dirty read problem : The dirty read problem occurs when one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction.
- unrepeatable read problem :

7.3.4 checking for conflict serializablity

draw a presidence graph
draw an arrow only if

- transactions are different

- on same data item
- one of the opertaion is write

if theere is no cycle then it ic conflict serializable

equivalent serial sechedule can be found by a topographical sort on precidence graph

there may be serial schedule but not conflict serializable if it is not conflict serializable then check for view serializable if the schedule is view serializable the it is serializable

7.3.5 view serializability

A cschedule s is said to be view equivalent to other schedule s' if following conditions are met

- for each data item same transaction should read a data item initially in both the schedules
- for each data item same transaction must write a data item finally in both schedules
- producer-consumer(write then read) if exist should exist in both schedules

7.4 deciding on serializability of a schedule

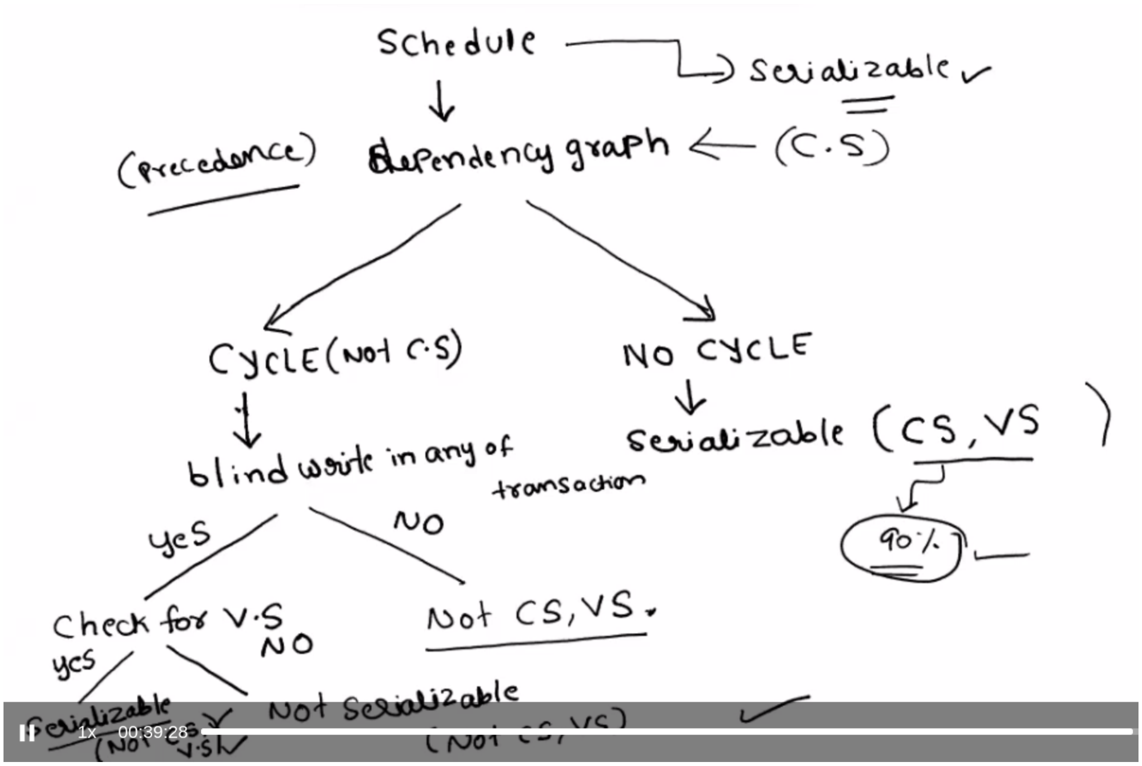
Recoverable schedules : a schedule is not recoverable if there is a producer-consumer problem and consumer is commited before producer

cascadeless schedules : consumer should read only after commited

strict schedules : before any read or write we should check if it is commited

7.5 Concurrency Prototcols

these protocol ensures serializability of schedules



7.5.1 types of protocols

- lock based
- timestamp
- granualirity

7.6 Lock Based Protocol

these are of two types

shared: when we only want to read a data. once we aquire a lock of a data other transaction can access it because it is used only for reads.

Exclusive: when we want to read as well as write a data. once we aquire a lock of a data no other transaction can access it

simple locks may give inconsistent results and deadlock

7.7 2 phase locking

in 2 phase locking we have a growing phase and a shrinking phase we lock all data in growing phase and unlock them back in shrinking phase

types of 2-phase locking

- strict 2pl :an exclusive lock must unlock only after commit
- rigorous 2pl:exclusive and shared lock both must unlock only after commit
- conservative 2pl: in addition to above conditions it must also not have any operation between aquiring locks ie. all locks must be aquired in thje begining only

7.8 Timestamp based concurrency controll

here concurrency is achieved by following certain rules based on timestamp

rules

if Transaction T_i is issuing a write on dataitem A

if(RTS(A)¿TS(ti)) rollback; if(WTS(A)¿TS(ti)) rollback; else execute WTS(A) = TS(Ti)

if Transaction T_i is issuing a read on dataitem A

if(WTS(A)¿TS(ti)) rollback; else execute RTS(A) = max(RTS(A),TS(Ti))

7.9 Graph based protocol

if we have hirarchical db then this protocol can be used

it states that if a data is to be locked lock its parent instead

rules

- the first lock may be on any data item
- subsequemtly the data item can be locked if the corresponding parent is locked by the transaction
- data items can be unlocked in any time in any order

Chapter 8

Indexing

8.1 Indexing

each record of a table is kept inside a block

8.2 Strategies for storing records in block

- spanned : the records are spanned across blocks so that space is not wasted
- un-spanned : the records are kept inside a single block so that searching is easier

8.3 use of indexing

indexing is used for making the search faster

8.4 Types of indexes

- primary index : used with ordered data having primary key
- clustered index : used with ordered and non-pk
- secondary index : used with unordered and non-pk

8.5 Primary index

A primary in dex is an ordered file where records are of fixed length with two fields (key and block pointer)

no of index here is no of blocks

8.6 Clustered index

no of keys = no of unique keys in data

8.7 Secondary index

no of keys = no of unique keys in data store the index in asending order

8.8 Indexing tecqniques

two types

- Dense : if index entry is created for every search key value
- Sparese : if an index emtry is created for some of the search value

8.9 Multi Level Indexing

it is implemented using B and B+ trees

8.10 m-way search tree

each node of the tree consist of m-1 elements and has m children

example a node of a 3 - way search tree will have 2 elements lets say 30 and 40 then all the nodes where elements are less than 30 will form a child btn. 30 and 40 will form another child and above 40 will form another.

m-way search trees are not height balanced

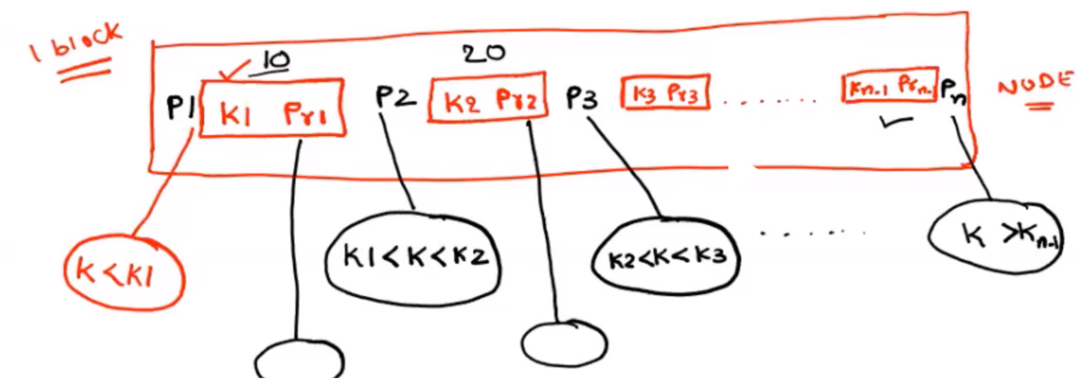
8.11 B trees

rules for b trees

- root should have minimum two children
- every node should have m/2 children by the time tree is constructed
- all leaf node should be in the same Level
- creation process should be bottom up

concepts

- the node represents block
- $n * p_n + (n-1)(k + p_r) \leq Blocksize$



8.11.1 insertion in b-tree

points

- max no of children in node is order -1
- min no of children in node is ceil(order/2) -1

inserting elements inside a node

- if the node has space left put the elements inside the node and sort it

- if the node over flows
  - find median left or right(depends?)
  - push the median to root or push it up
  - elements to right of median will be right child and left will be left child
  - insert the element in either left or right depending on whether it is greater then or less than root

8.12 B+ tree

it is similler to b tree except that here the block pointer is present in leaf nodes only and all leaf nodes are connected using pointer like linked list

$n * p_n + (n-1)k \leq Blocksize$