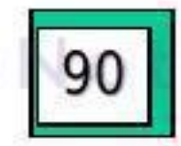
5.7.1 Creating B^+ tree

- Creating a B^+ tree is similar to B tree.
- When we split a leaf node, a copy of the middle key is stored in its left/right child. This is not done for a non-leaf node.

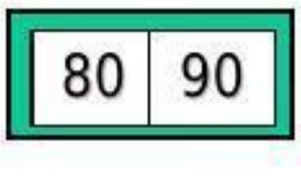
Q:- Create a B^+ tree of order 3 for the following set of key values

90, 80, 60, 5, 120, 14.

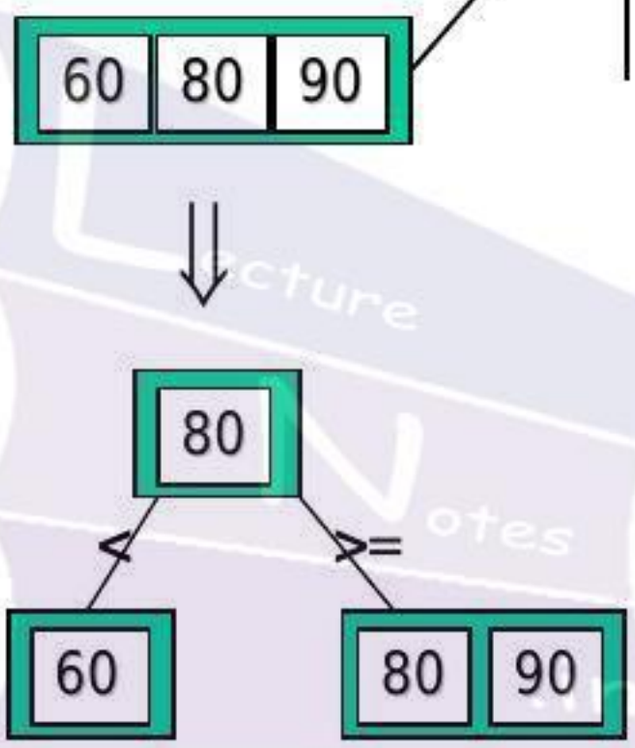
insert 90



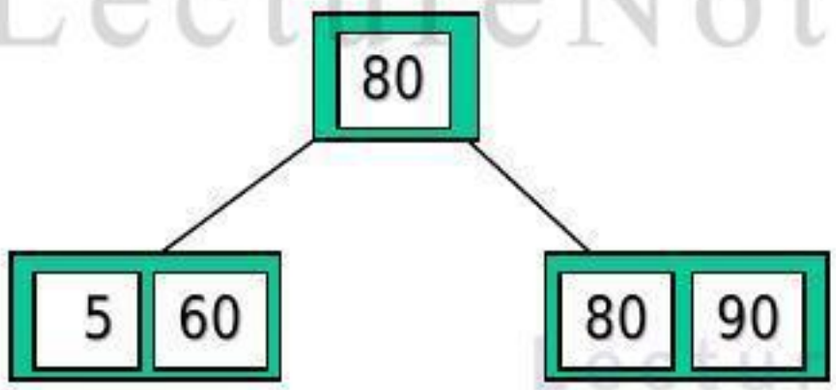
insert 80



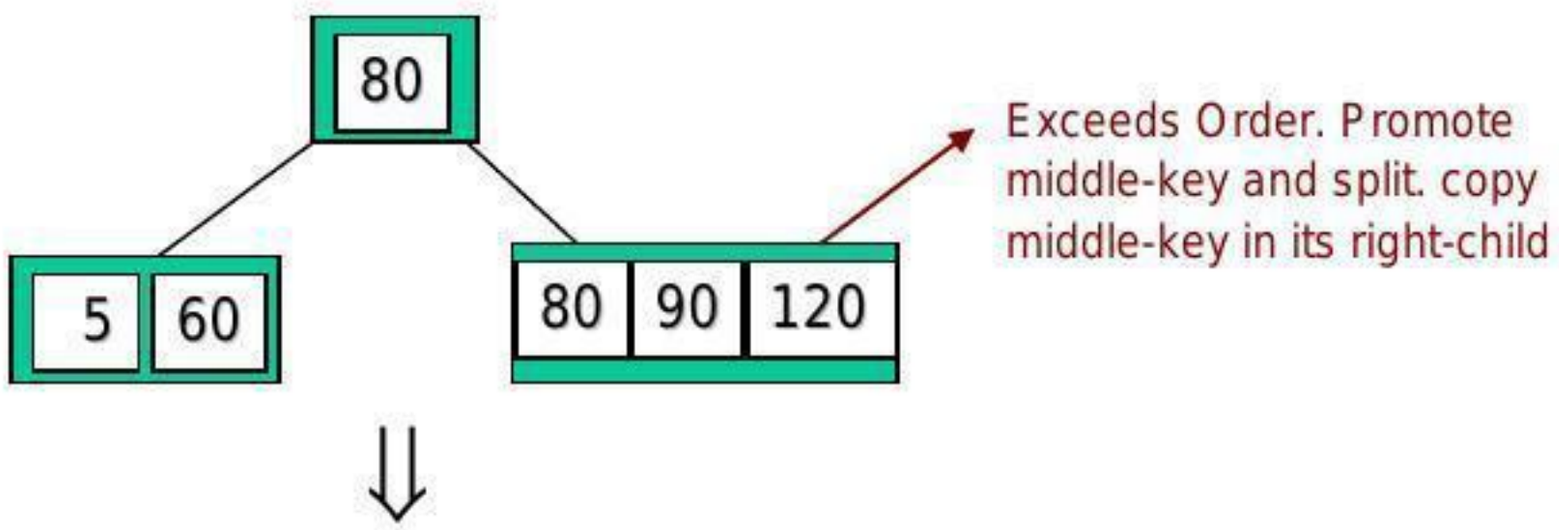
insert 60



insert 5



insert 120



3 Functional Dependency and Normalization

- 3.1 Inference Axioms
- 3.2 Closure of attribute
- 3.3 Equivalence set of FDs
- 3.4 Canonical cover
- 3.5 Decomposition : lossy and lossless decomposition
- 3.6 Algorithm for lossless join decomposition
- 3.7 Normalization
- 3.8 First normal form
- 3.9 Partial dependency, full dependency
- 3.10 Second Normal Form
- 3.11 Transitive dependency
- 3.12 Third Normal form
- 3.13 Boyce Codd Normal Form (BCNF)
- 3.14 Multivalue Dependency
- 3.15 Fourth Normal form
- 3.16 Fifth Normal form
- 3.17 Query processing and Optimization
  - 3.17.1 Translating Query to Relational Algebra
  - 3.17.2 Evaluation of Relational Algebra Expressions
- 3.18 Heuristic optimization

4 Transaction Processing

- 4.1 Properties of Transaction
- 4.2 States of transaction
- 4.3 Concurrent transaction
- 4.4 Schedule : Concurrent, Serial schedule
- 4.5 Serializability
  - 4.5.1 Conflict Serializability
  - 4.5.2 View Serializability
- 4.6 Concurrency Control
- 4.7 Lock Based Protocol
  - 4.7.1 Two Phase Locking(2PL) Protocol
  - 4.7.2 Timestamp Ordering Protocol
  - 4.7.3 Optimistic Concurrency Control
- 4.8 Recovery System
- 4.9 Log based recovery
- 4.10 Check - Pointing
- 4.11 Shadow Paging

5 Storage strategies

5.1 Magnetic disk (Hard disk)

5.2 RAID

5.3 Index

5.4 Types of index

- 5.4.1 Primary Index
- 5.4.2 Clustering Index
- 5.4.3 Secondary Index

5.5 Hashing

- 5.5.1 Hash Function
- 5.5.2 Collision avoidance technique

5.6 B-Tree

- 5.6.1 Creating a B-tree
- 5.6.2 Deletion from a B-tree

5.7 B^+ Tree

- 5.7.1 Creating B^+ tree
- 5.7.2 Deletion from a B^+ tree

Advance topics

5.8 Object oriented database (OODB)

5.9 Object Relational database (ORDB)

5.10 Parallel database

5.11 Distributed database

5.12 Parallel vs Distributed database

5.13 Data warehouse

5.14 Data mining

CHAPTER 1

Introduction to Database System

1.1 Basic concept & Definition

Database

A Database system is basically a computerized record keeping system.

" A Database is an integrated collection of logically related records or files "

Ex - A college database contain information about the following

- Entities such as student, faculty, course & room.
- Relationship between entities such as student enrollment in courses, faculty teaching courses.
- Information about an entity is stored in a table, which consists of rows & columns.

DBMS

- ✓ DBMS is the software for maintaining & utilizing a large collection of data.
- ✓ Database management system follows a specific database model such as Network, Relational, Object or Hierarchical Model.
- ✓ These models use the query language to access the database.
  [query means request or task]
- ✓ Commonly used query language is known as SQL (Structure Query Language)
- ✓ DBMS manages the performance, concurrency, integrity and recovery of database.

RDBMS

- ✓ A RDBMS is a DBMS which is based on the relational model. It was introduced by Edgar Frank Ted Codd.
- ✓ A RDBMS obeys (based on) 12 codd rules.
- ✓ A RDBMS is a DBMS in which data and the relationship among the data is stored in tables.
- ✓ Two tables are related by using primary key and foreign key.

1.2 Difference Between DBMS and RDBMS:

DBMS	RDBMS
1. Support relationship between data.	1. Support relationship between tables.
2. It may not follow all the codd rules.	2. It follows all the codd rules.
3. Example-FoxPro	3. Example-Oracle,DB2, Sql-Server
4. Single-User System	4. Multi-User System.

1.3 Data Dictionary

- ✓ Database system maintains information about every table and index. This information is stored in a collection of special tables called data dictionary.
- ✓ It is also called catalog table or metadata (data about data).
- ✓ Data dictionary contains information such as
  - (i) Integrity constraints defined on a table.
  - (ii) Names & data-types of all the columns of table.
  - (iii) User privilege information.

Consider a Student table as: Student (regdno, studentname, branch). Then its Catalog Table is shown below

Attribute Name	Relation name	Data Type	position
regdno	Student	integer	1
studentname	Student	varchar2	2
branch	student	varchar2	3

(Catalog Table)

1.4 Users of DBMS

Users of DBMS are classified into four categories

1. End user
2. Online user
3. Application programmer
4. DBA

End User

- ✓ End users are also called Naive users.
- ✓ End users do not know about the DBMS.
- ✓ End users have only read permission on database.
- ✓ Example: User of an ATM (Automated Teller machine).

Online user

- ✓ Online users access database by a user-id and password.
- ✓ Online users know about the DBMS.
- ✓ Online users have read and write permission on database.
- ✓ Example: bank officials, bank-manager of a bank.

Application programmer

- ✓ Application programmers develop the applications using programming languages like c, c++, java etc.
- ✓ Application programmers work under DBA. DBA authenticates Application programmers to access database.
- ✓ Application programmers are also called embedded users / front-end designers.

DBA

- ✓ DBA is responsible for design, implementation, maintenance & repairing of database.
- ✓ DBA is responsible for granting permission to the user for accessing the database.
- ✓ DBA is responsible to recover the database from failure due to human, natural or hardware causes.
- ✓ DBA is responsible for periodically back up the database either to magnetic tape or remote server to prevent loss of data.
- ✓ DBA is responsible for modification of Schema and physical organization to improve performance.

1.5 File system versus DBMS

Before DBMS is designed, we store data in files, but it has many drawbacks.

- ✓ We have to write a special program for a query.
- ✓ Data inconsistency due to concurrent access by several users.
- ✓ Data inconsistency when system crashes.
- ✓ Only the password system is used for security of data.

Advantages of DBMS / Disadvantages of File-system

Data Independence: In DBMS a particular record can be inserted or deleted without affecting other record. Hence, data is independent by itself.
But in file-system if we insert a record then the forwarding records need to be shifted (as in array), because it allows serial access mechanism.

Efficient data access: DBMS efficiently store and retrieve data from database. User can view the data according to his choice.
Example: Display students having cgpa>8.0. This query cannot be written in File-system.

Data Integrity: DBMS enforce/apply integrity constraint(condition) so that correct data is stored in database.
It is also called data validation (valid data is stored. Thus avoid invalid or useless data)
Example: Name field does not allow numeric value.

Concurrent Access: A DBMS manages concurrent access to the database.
Example: Many users accessing the yahoo server at same time.

Crash recovery: DBMS can recover the data from system crash (system failure).

Reduced Redundancy: DBMS use technique to reduce data redundancy (redundancy means repetition/duplicate).

Security: DBMS can authenticate users for the permission (privileges) to access database. But there is no such restriction in file-system.

1.6 Database Language

Database language is divided into following categories

1. DDL
2. DML
3. DCL
4. TCL

DDL (Data Definition Language)

DDL includes following statements/commands

create - To make a new database, table, view, index

drop - To delete an existing database, table, view, index

alter - To modify an existing database, table, view, index

DML (Data Manipulation Language)

DML is used to access & manipulate the data.

DML includes following statements

select - Display(Retrieval) of data

insert - Insertion of new data

update - Modification of data

delete - Deletion of data

DCL (Data Control Language)

DCL is used to control the access of data.

DCL includes following statements

grant - To allow specific user to perform specific task.

revoke - To cancel previously granted permission from user.

TCL (Transaction Control Language):-

TCL includes following statements

commit - permanently save the data to database.

rollback - undoing current transaction (and go to previous state/checkpoint) ctr+z in word

1.7 Data Models

Data model is a collection of tools to describe data and relationship among data.

(Like flow-chart describe a program, similarly data-model describe database)

Various data models are Relational model, Network model, Hierarchical model etc.

1.8 Relational Model

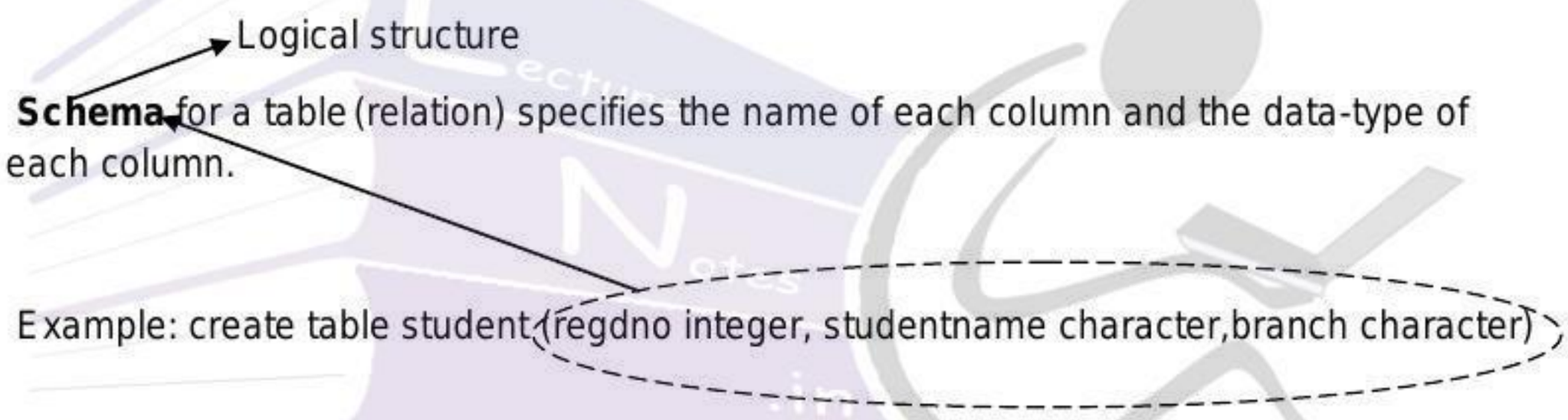
data and relationship among data is stored in tables(relations).

degree/arity of a relation = no of columns present in the relation.

Cardinality of a relation = no of records present in the relation.

Relational database is a collection of relations.

1.8.1 Schema , subschema and instances



Relational Database schema is the collection of schemas for the relations in database.

Subschema

A relation may have several schemas called as subschema that describes different views of database.

The concept of subschema is implemented by creating view.

View

- ✓ It is a logical table created from one (or more) table.
- ✓ change made in a view is reflected in the original table from which it is derived.
- ✓ It hides the complexity of database.
- ✓ View is used to hide personal & confidential matter

  Example: students are not permitted to see salary details of a faculty. so a view is created with all columns of faculty table except salary column.

Index

Index is a pointer to the records of a table. (just like index of a book)

syntax **:** create index index-name on table name(fieldname)

example**:** create index ind on student (regdno)

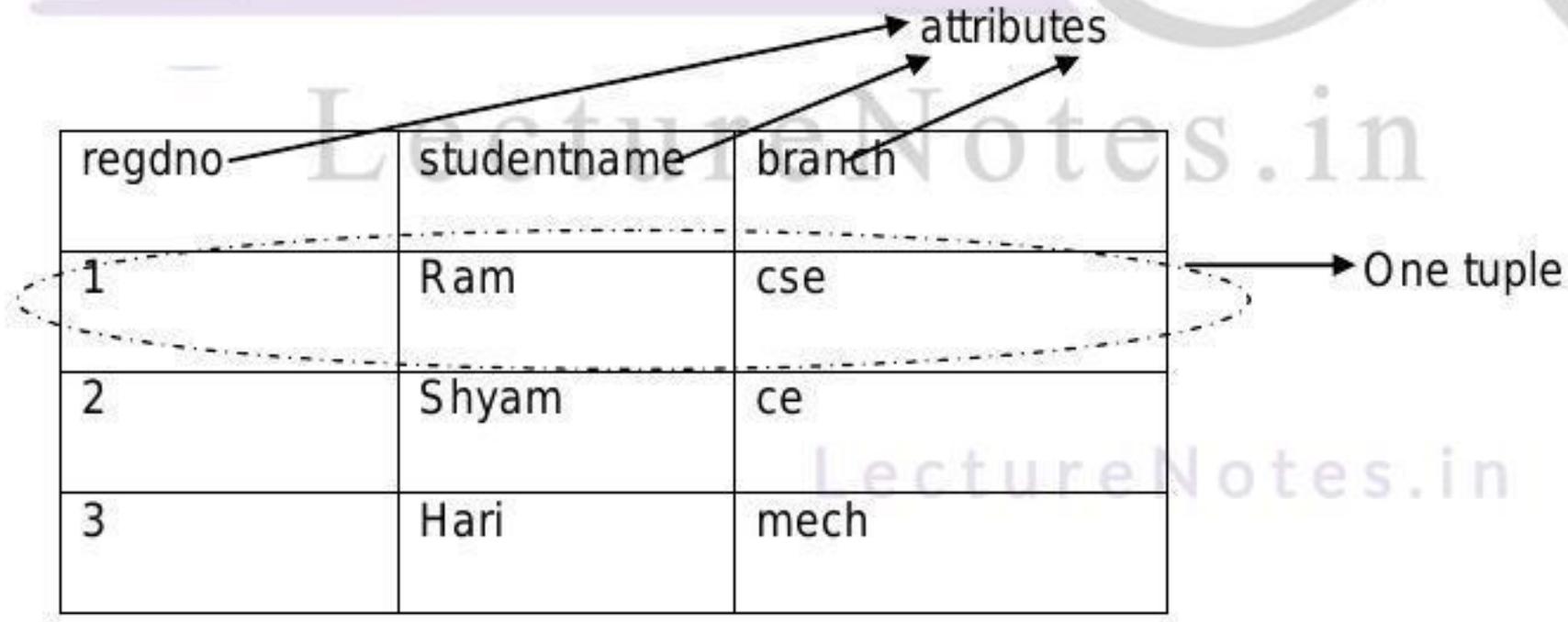
Database system creates an implicit index on a table. Index is used for quick search of a record from table.

Instance of a Relation

Instance of a relation represents data present at a particular instance (moment) of time.

A row or record of a relation is called a **tuple**.

columns are called attributes or fields.



(An instance of student relation)

1.8.2 Data Abstraction

Complexity of database is hidden through different levels of abstraction.

The data is described at 3 levels of abstraction.

1. Physical level
2. Logical level / Conceptual level
3. View level / External level

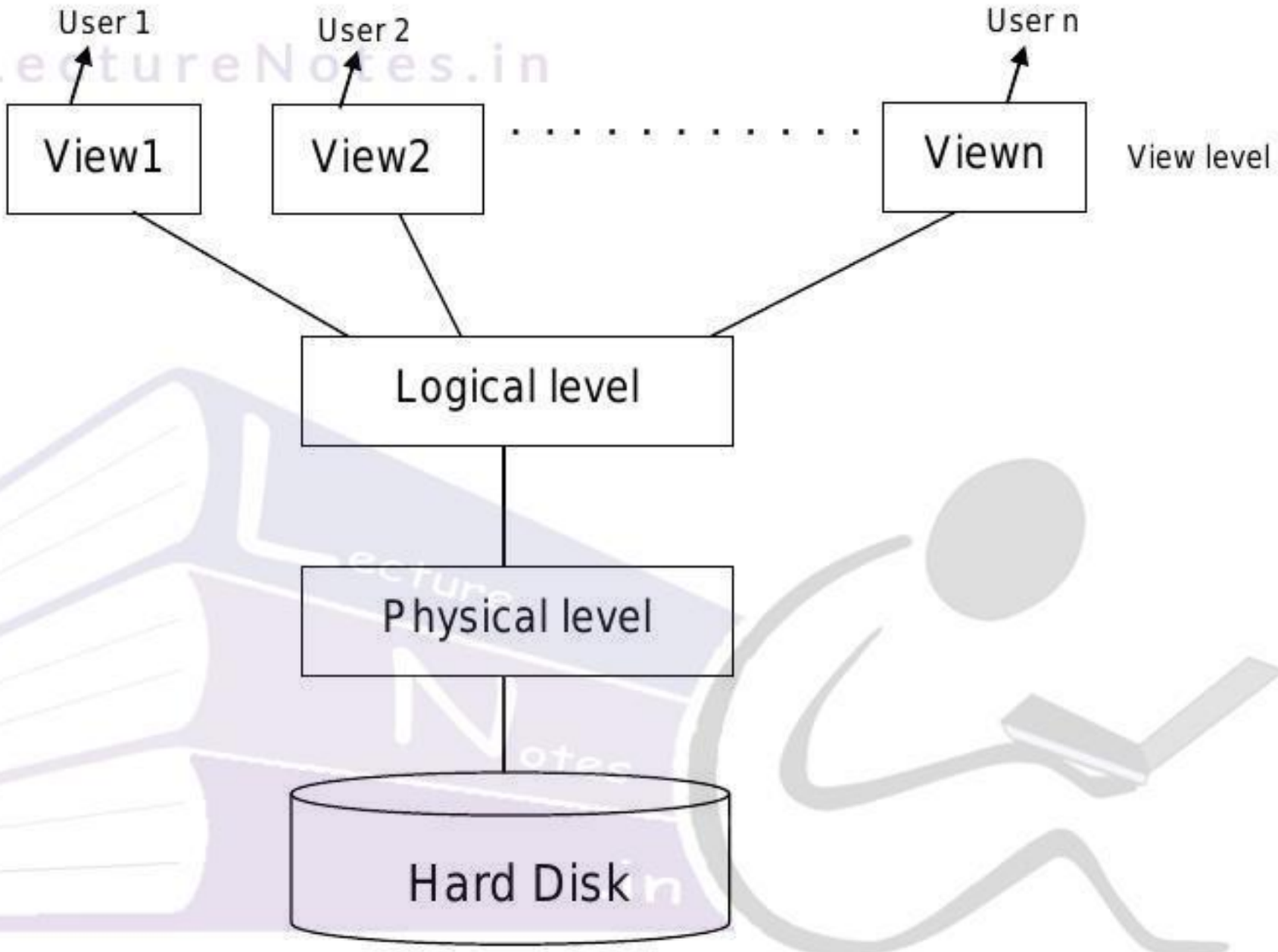


Figure: Three Levels of abstraction in DBMS

Physical level

It is the lowest level of abstraction.

It describes physical organization / structure of data which include

i) How data is stored in hard-disk.

ii) What data structure is used.

iii) About indexes.

These details are hidden to Application programmer working at logical level. But, DBA know these details.

Logical level

- ✓ It is the next higher level of abstraction.
- ✓ It describe logical schema which include
  - i) What data is stored in database.
  - ii) What kind of relationship exist between them.
- ✓ Application Programmers work at this level of abstraction.
- ✓ Example:- A college database may contain information about entities such as student, faculty & relationship such as student enrollment in courses.

View level:-

- ✓ It is the highest level of abstraction. It is also called user-level.
- ✓ There may be several view (external schema).
- ✓ Example:- users of Google database access information without knowing the logical & physical level.
- ✓ view also provides security by hiding important information
- ✓ Any given database has exactly one physical schema, one logical schema, but it may have several external schemas.

1.8.3 Data Independence

It is the capacity to change the schema at one level without changing at higher level.

Data independence is of two types

1. Logical Data independence:-

It is the capacity to change at logical level without changing at external level.

Example: change in datatype , constraint etc.

It is difficult to achieve Logical data independence. Because for a change in data type, we have to change the entire program/application.

2. Physical Data independence:-

Physical Data Independence is the capacity to change at physical level without changing at logical level. The external schema need not be changed as well.

Example: change in RAM size, harddisk capacity, file organization etc.

It is not difficult to achieve physical data independence. Because for a change in RAM size, we do not change the entire program (conceptual schema.)

1.9 Entity Relationship model(ER Model)

ER Model consists of entities and relationship among them.

Entity is a Real world object that is different from other objects.

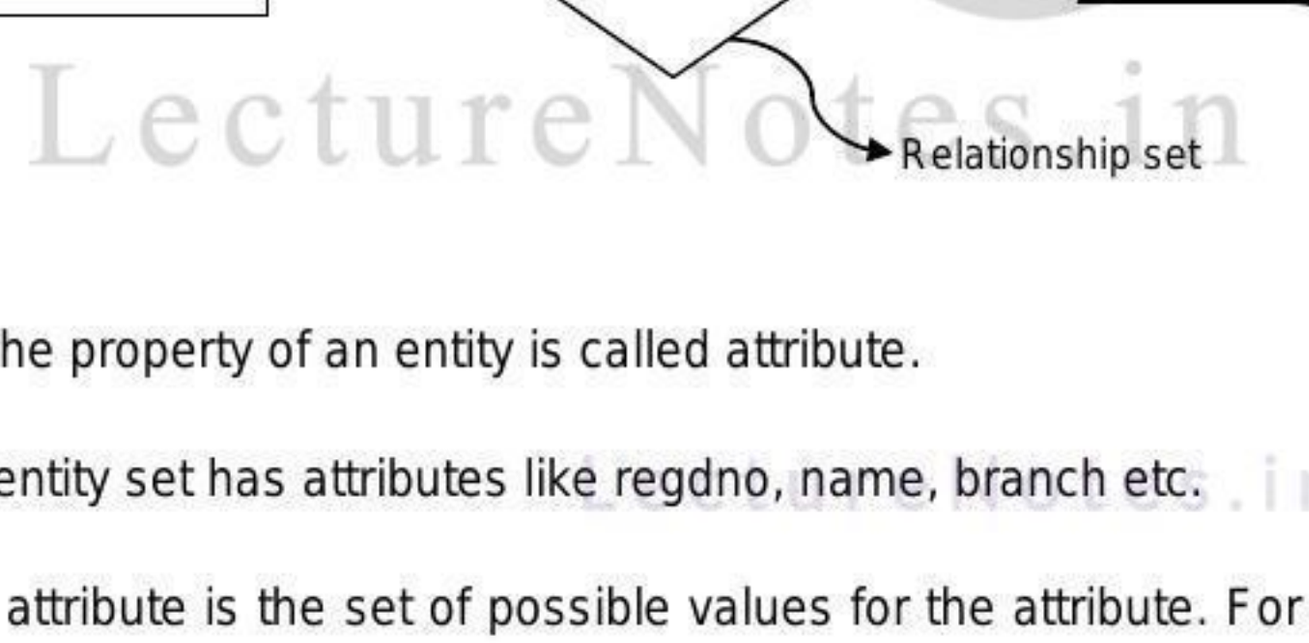
Something that exists and distinguished from other is an entity.

Example - An employee, A student, An Account.

Entity set is a collection of similar entities. Ex- employees, students, list of accounts.

A **Relationship** is an association among two or more entities.

Ex-



Relationship set is a set of similar relationship.

Example:

Attribute - The property of an entity is called attribute.

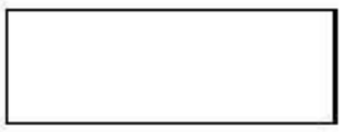
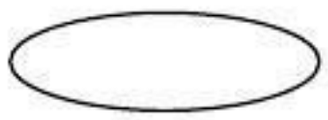
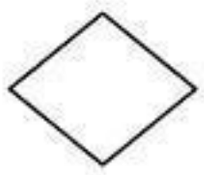
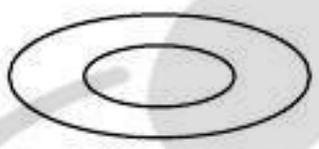
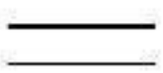
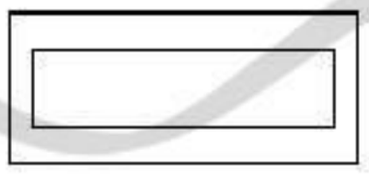
Ex- Student entity set has attributes like regdno, name, branch etc.

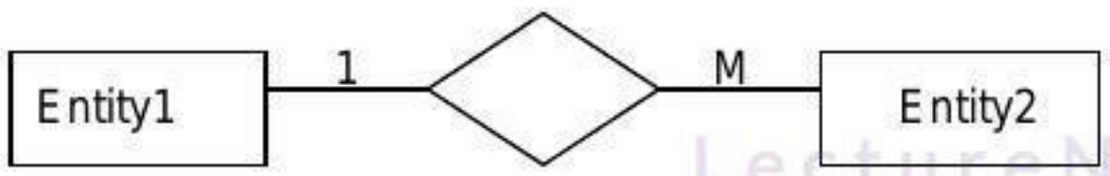
Domain of a attribute is the set of possible values for the attribute. For example, Domain of name attribute is all possible set of characters.

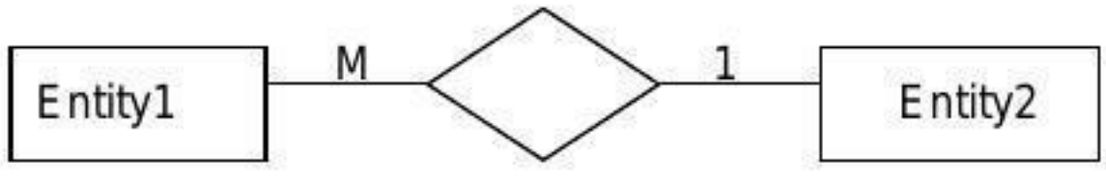
1.9.1 Entity-Relationship(ER) Diagram

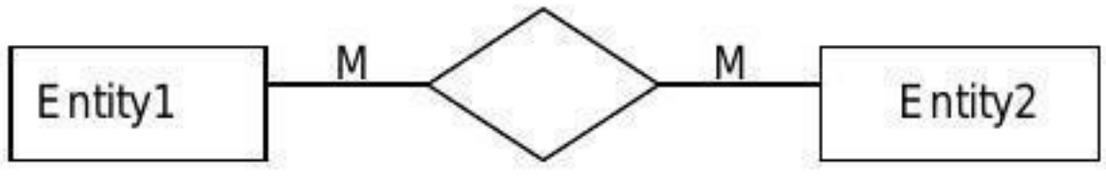
- ✓ It is a graphical representation of database system.
- ✓ It describes the data in terms of entity and their relationships.
- ✓ It is widely used to develop an initial database design.

ER Diagrams includes following symbols:-

1. Rectangles:- represents entity set. 
2. Ellipse:- represents attribute. 
3. Diamond:- represents relationship set. 
4. Line:- connect attribute to entity-set & entity-set to relationship-set.
5. Double ellipse:- represents multivalued attribute. 
6. Dashed Ellipse:- denote derived attribute. 
7. Double line:- indicate total participation of an entity-set in a relationship-set. 
8. Double Rectangle:- represents weak entity set. 
9. 

one-one relationship
10. 

one-many relationship
11. 

many-one relationship
12. 

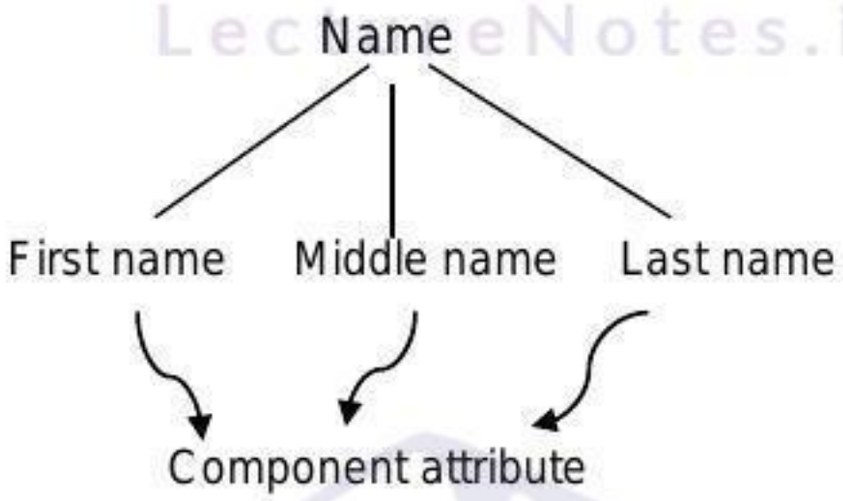
many-many relationship

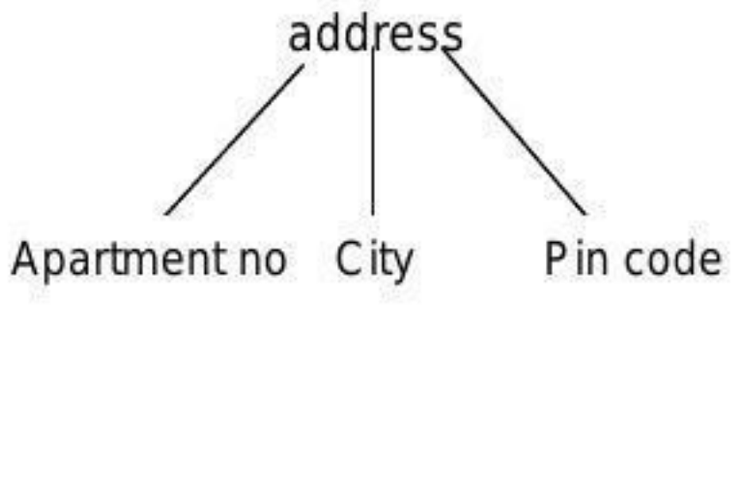
Different types of Attributes

Simple attribute:- The attributes which are not divided into subparts. Ex:- branch

Composite attribute:- The attributes which can be divided into subparts.

Ex- name is a composite attributes consisting of component attributes like first name, middle name, last name.





Multivalued attribute:- An attributes that have multiple values. Ex- phone-no.

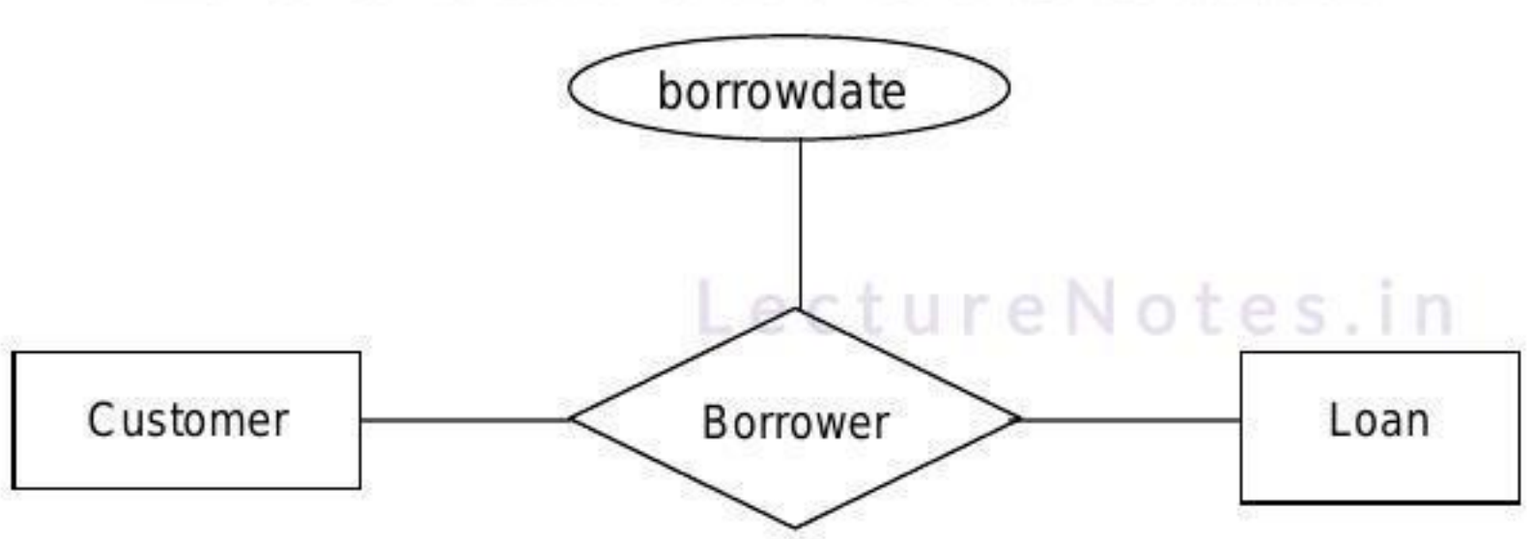
Derived attribute:- It is derived from other attribute.

Ex- age is a derived attribute. age is derived from date-of-birth attribute.

The value of derived attribute is not stored; It is calculated from base attribute whenever required.

Descriptive attribute:- It describes a relationship set.

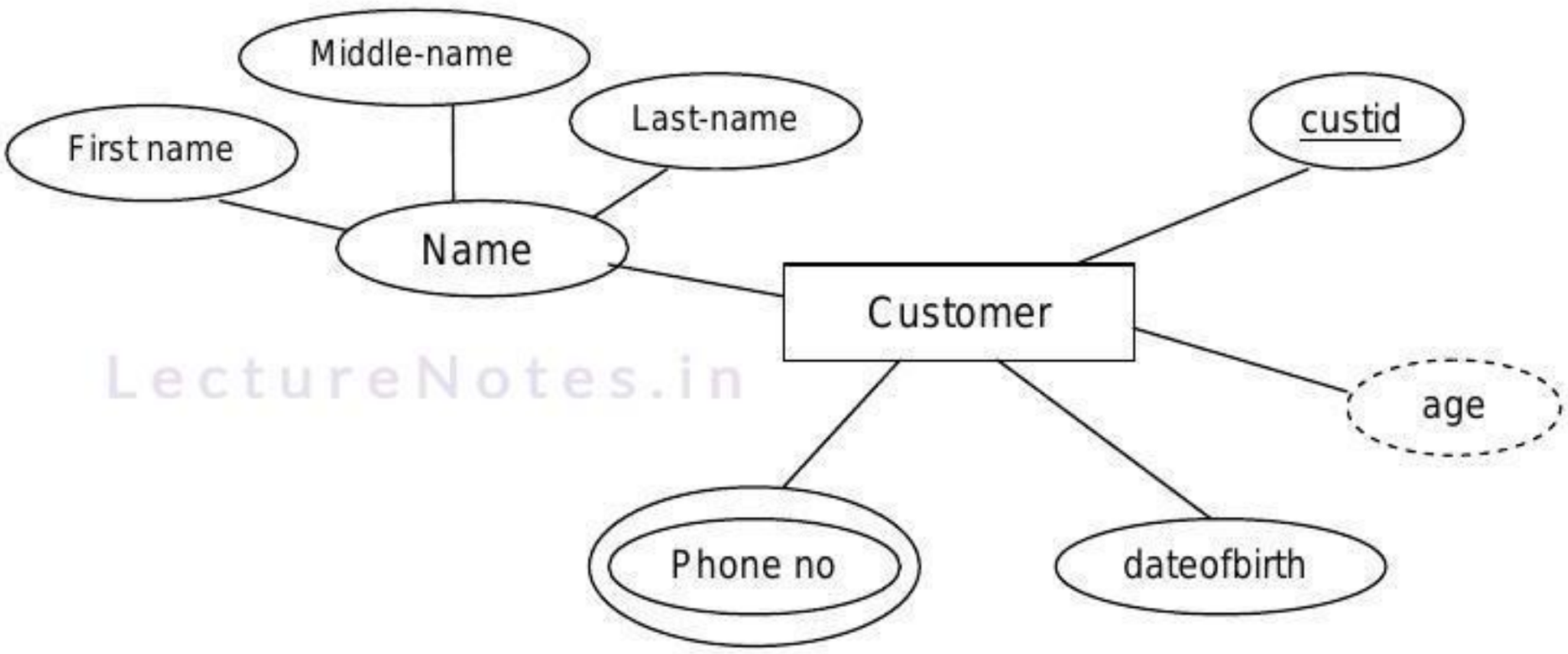
Ex- "borrowdate" is a descriptive attribute which describe the borrower relationship.



Key attribute:- It uniquely identify each entity of entity-set.

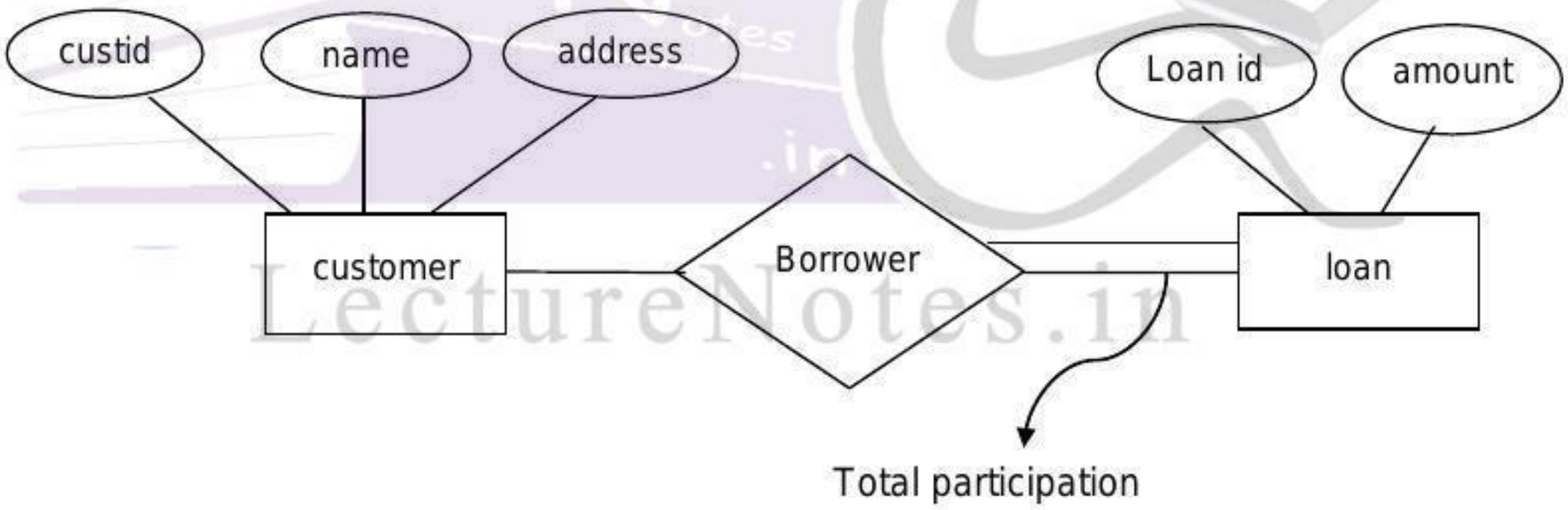
Ex - regdno, accountno, loanno .

1.9.2 ER Diagram of composite , component, multivalued, derived attribute



Total participation

If every entity in a entity set participate in a relationship set, then it is called total participation.



A double line from loan to borrower indicates total participation. because every loan must have a customer.

Partial participation: If some entity in an entity set participates in a relationship set, then it is called partial participation.

The relationship form customer to borrower is partial because some customer have borrowed loan not all customer.

Strong entity set: It is an entity set having a key attribute (primary key).

Weak entity set: It do not have a key attribute.

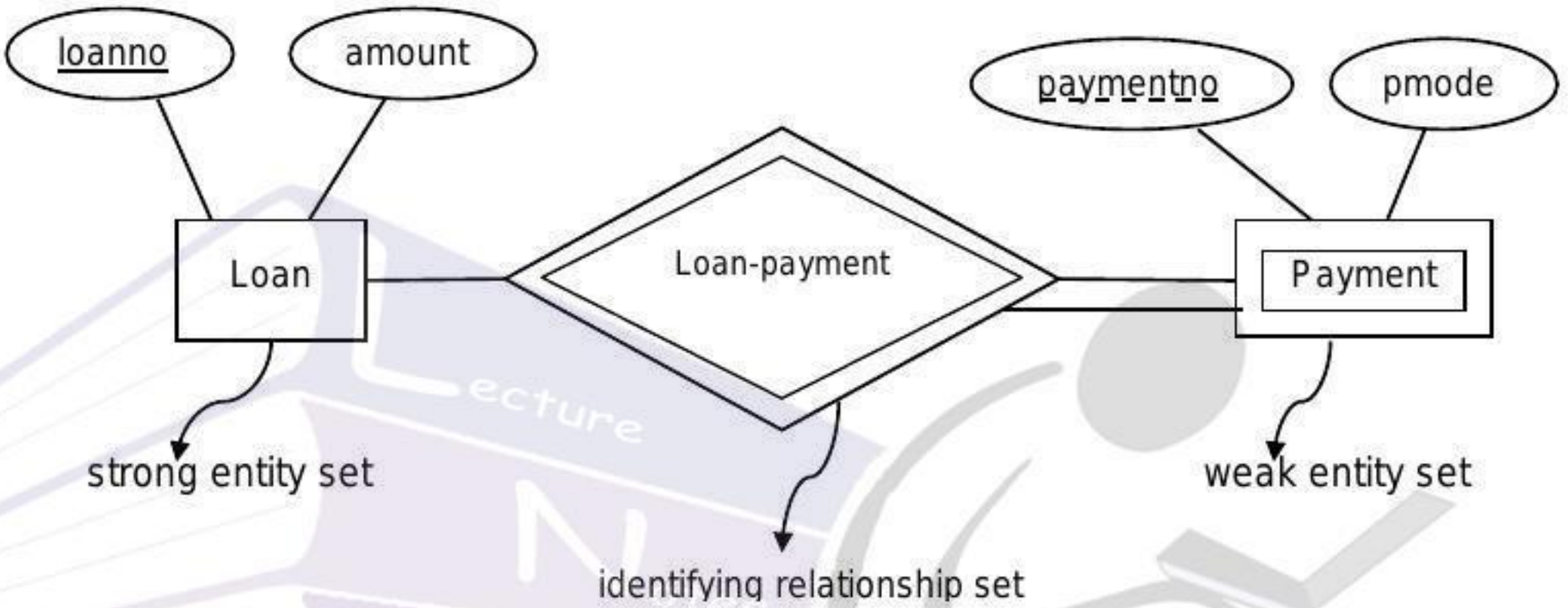
"An entity which depends on other entity to exist is called weak entity "

Weak entity set depends on identifying entity set (strong entity set).

Discriminator is an attribute that distinguish all entities of weak entity set.

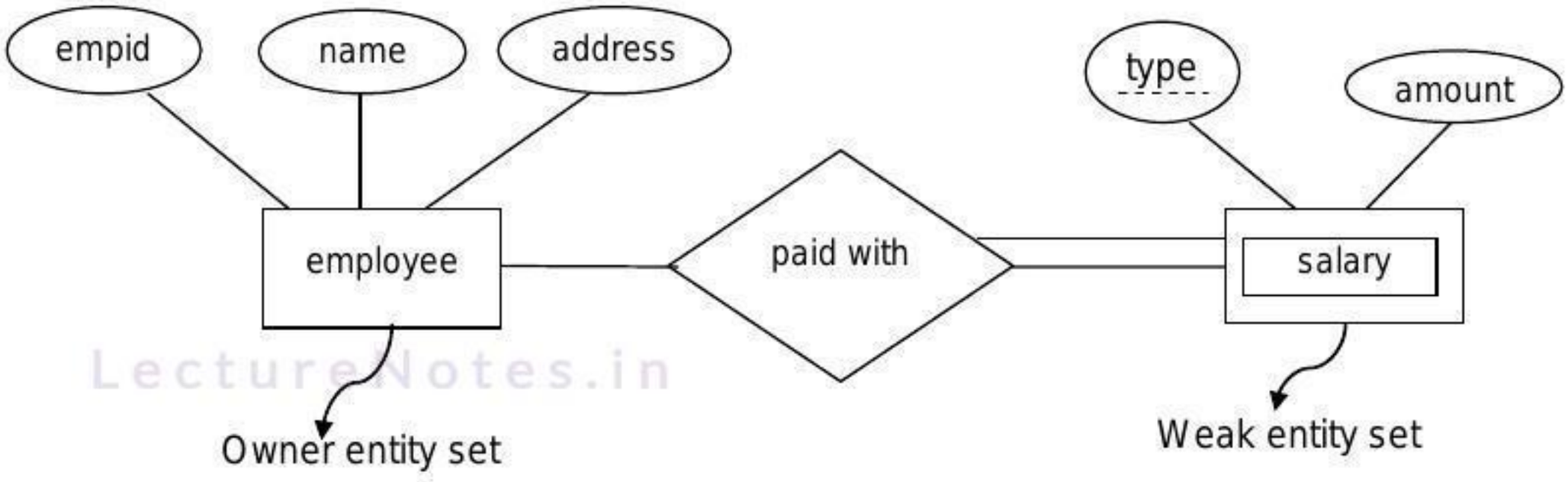
Primary key of weak entity = Primary key of strong entity + discriminator of weak entity

Example1: Primary key of payment is (loanno, paymentno). Identifying relationship set contains (loanno, paymentno)



- Strong entity set and weak entity set has one to many relationship.
- Weak entity set has total participation in the identifying relationship set.
- A weak entity set cannot be deleted as long as strong entity set exist.
- When a strong entity set is deleted the weak entity is automatically deleted.

Example2:-

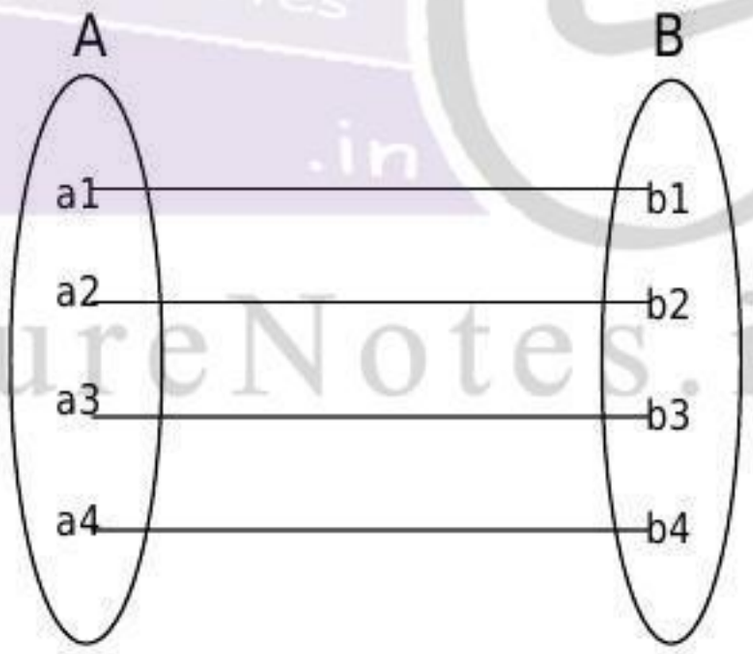


1.9.3 Mapping cardinality / Types of relationship

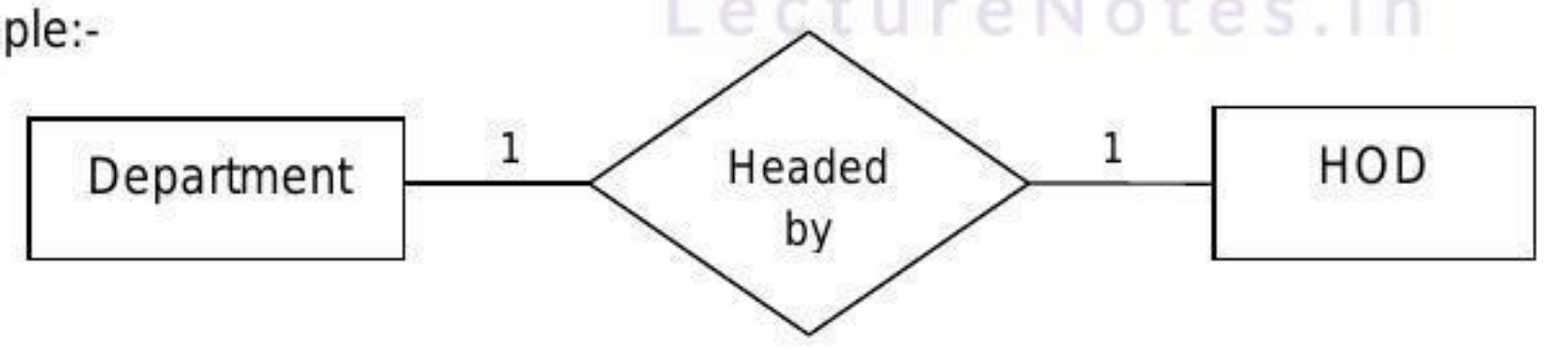
Mapping cardinality or cardinality ratio is the ratio no of entitles of one entity set with another entity via a relationship set. For a binary relationship between A and B, the mapping cardinality is one of the following

1) One to one

One entity of A is associated with one entity of B.



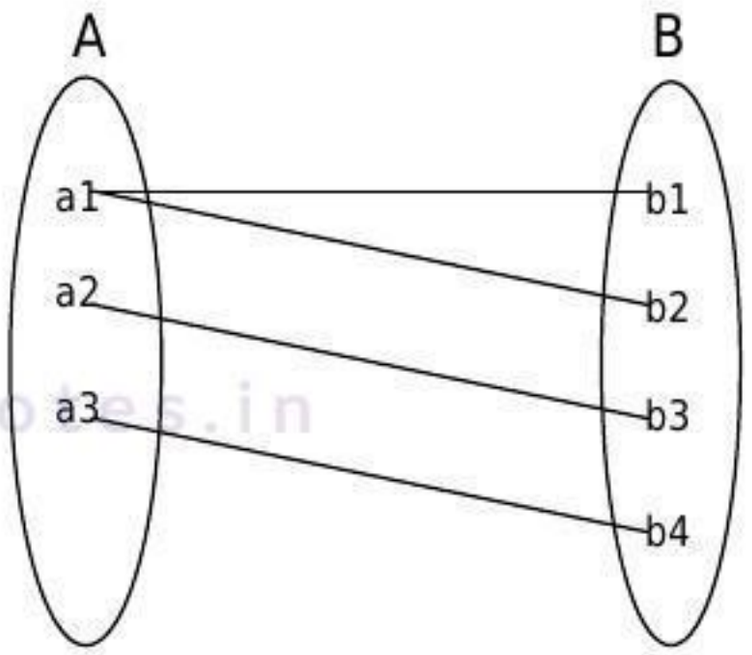
Example:-



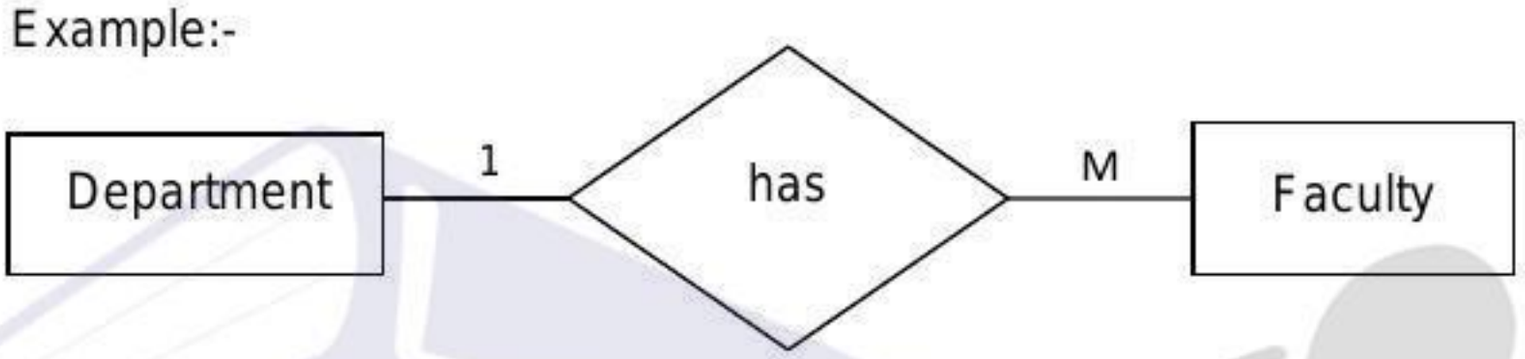
One department has one HOD.

2) One to many

One entity of A is associated with many entities of B.



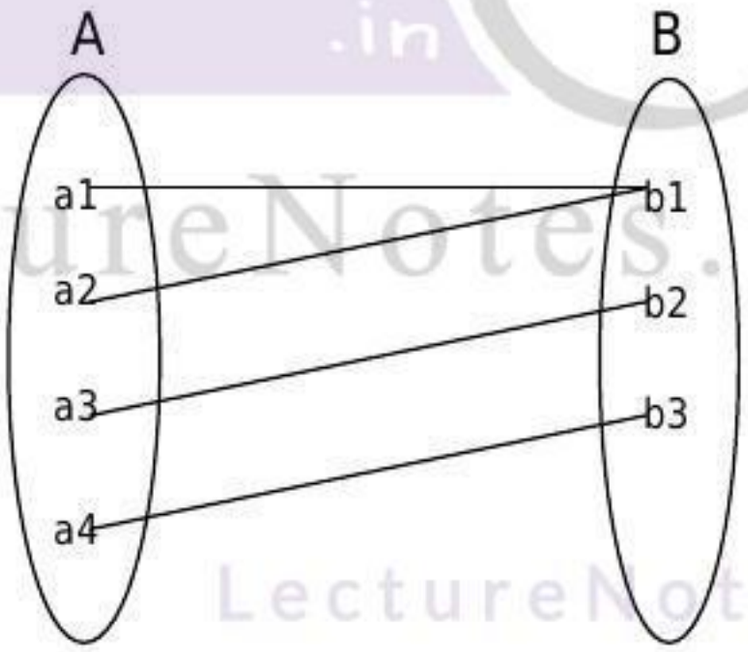
Example:-



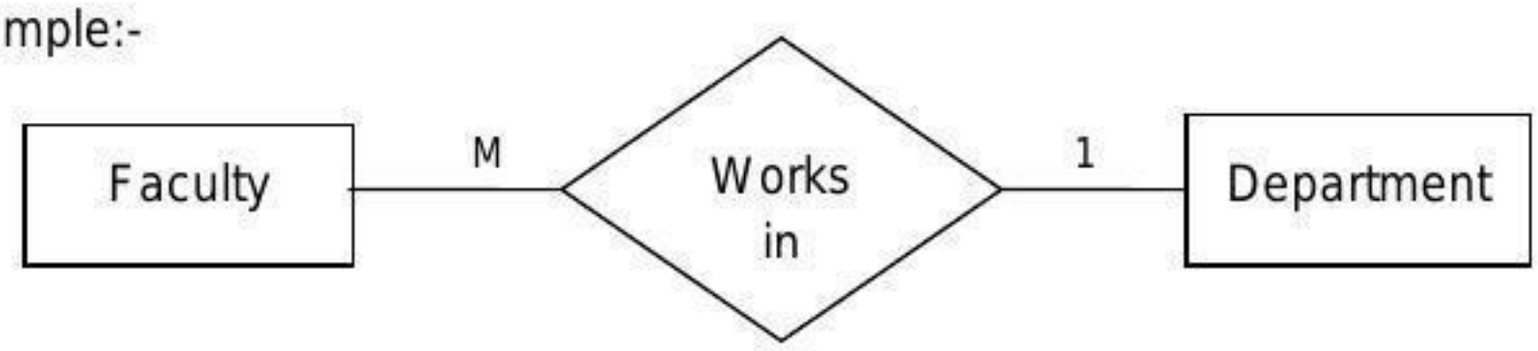
One department has many faculties.

3) Many to one

Many entity of A is associated with one entity of B.



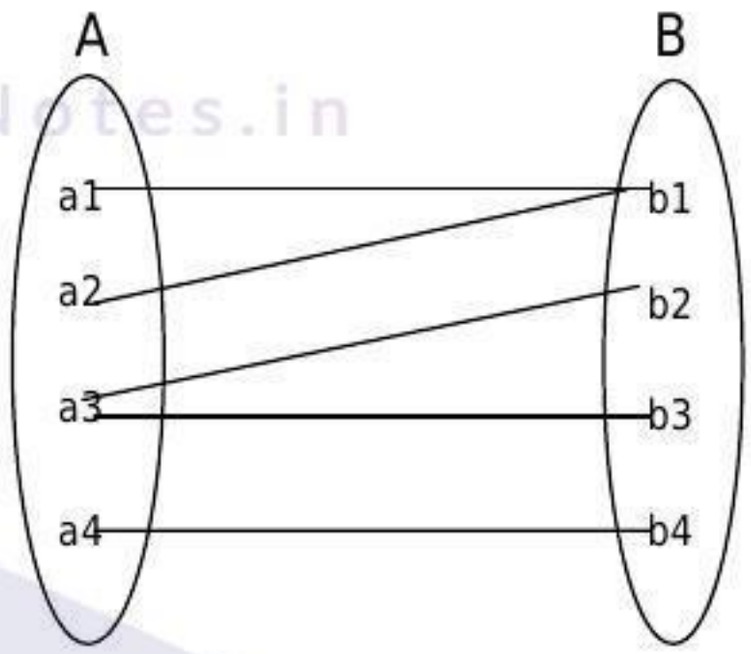
Example:-



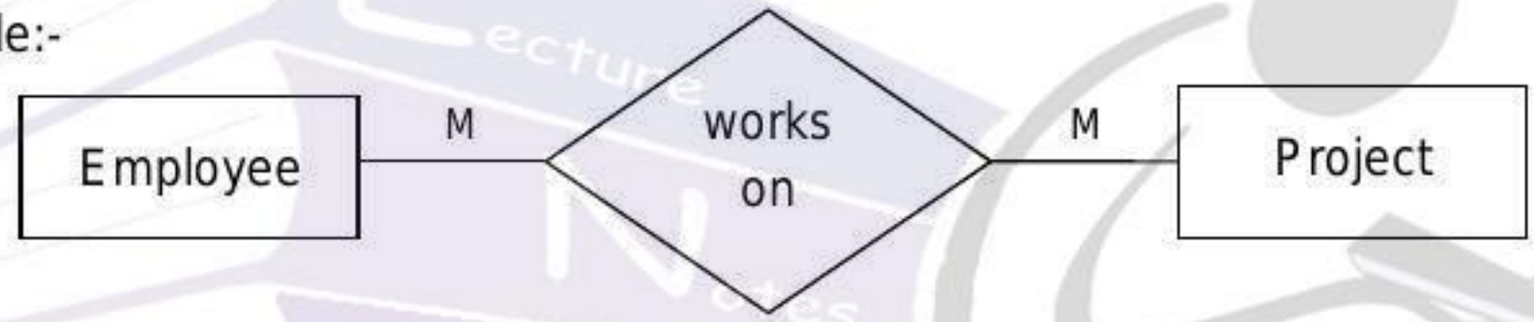
Many faculty works in one department.

4) Many to many

- ✓ Many entity of A is associated with many entities of B.
- ✓ An entity in A is associated with many entities of B & An entity in B is associated with many entities of A.
- ✓ many to many = many to one + one to many

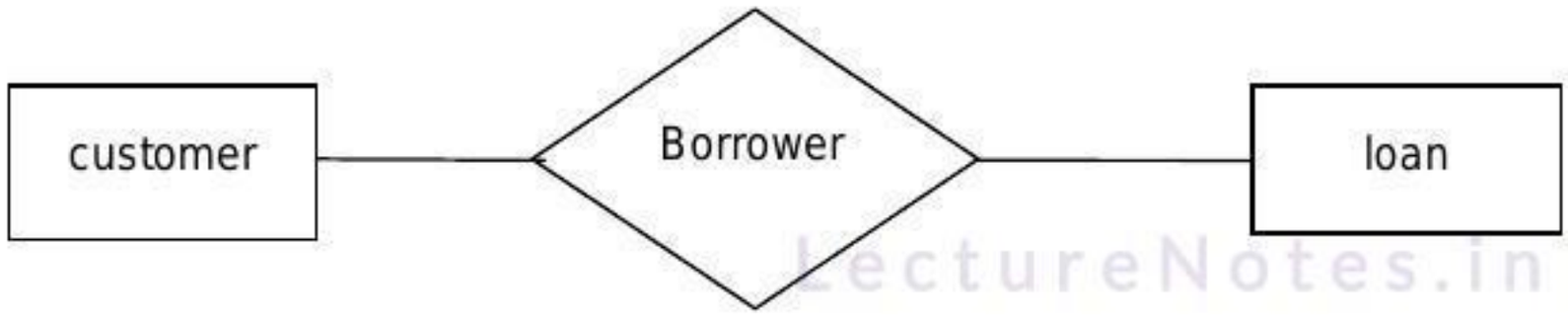


Example:-



Many employee works on many projects.

Example:- Consider the borrower relationship shown below



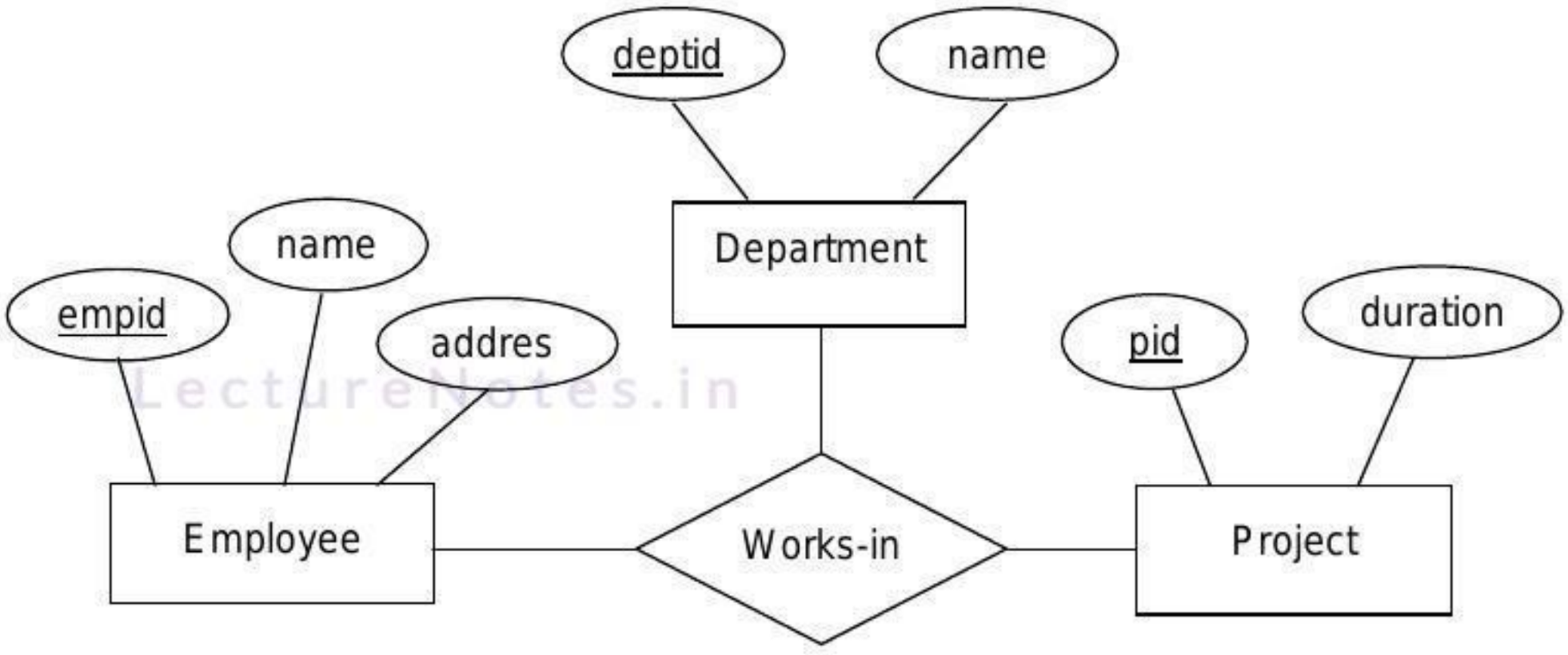
If one customer takes exactly one loan then relationship is one-one.

If one customer takes many loan then relationship is one-many.

If many customers take one loan then relationship is many-one.

If many customers take many loans then relationship is many-many.

Ternary Relationship:- Three entities participate in relationship set.

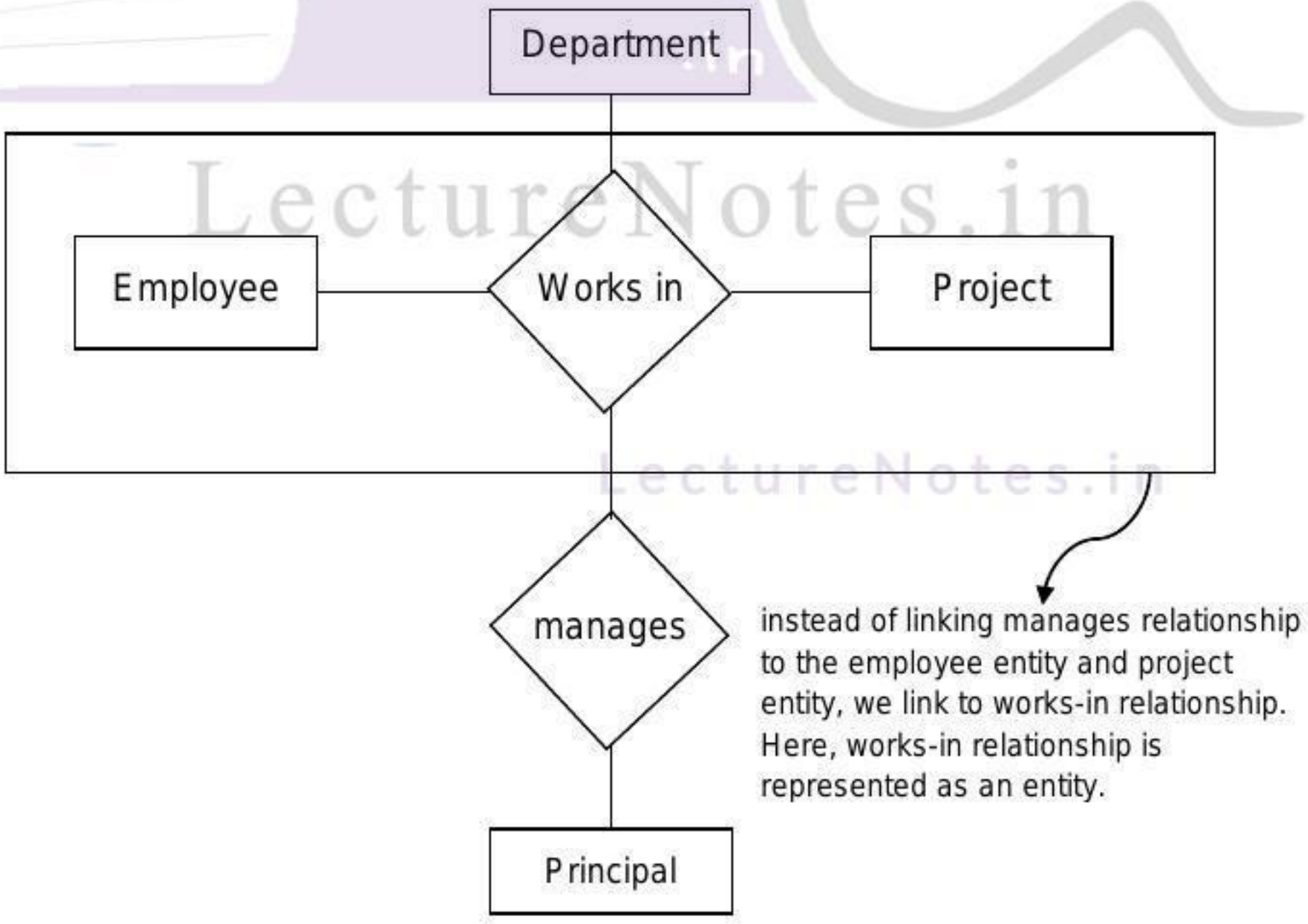


Note:- if n entities participate in relationship-set then it is called n-ary relationship.

1.9.4 Extended ER Features

Aggregation

It is an (abstraction) technique where a relationship is treated as an entity.



Specialization

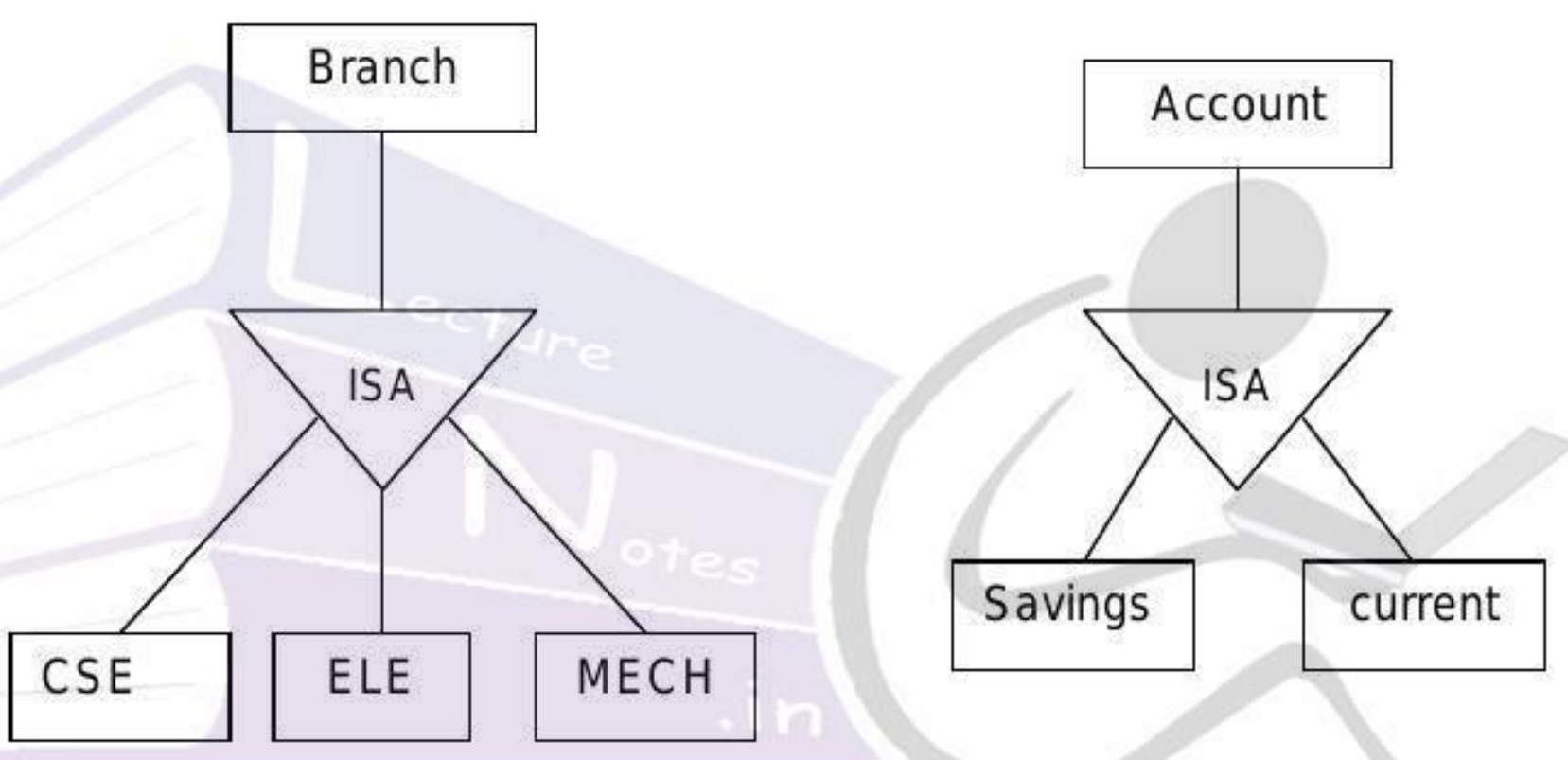
- ✓ The process of classifying/dividing an entity set by some special feature is called specialization.
- ✓ It is a top-down design approach.

Generalization

- ✓ It is the reverse process of specialization.
- ✓ It is a bottom-up design approach.

Specialization, generalization are represented by a triangle labeled with ISA.

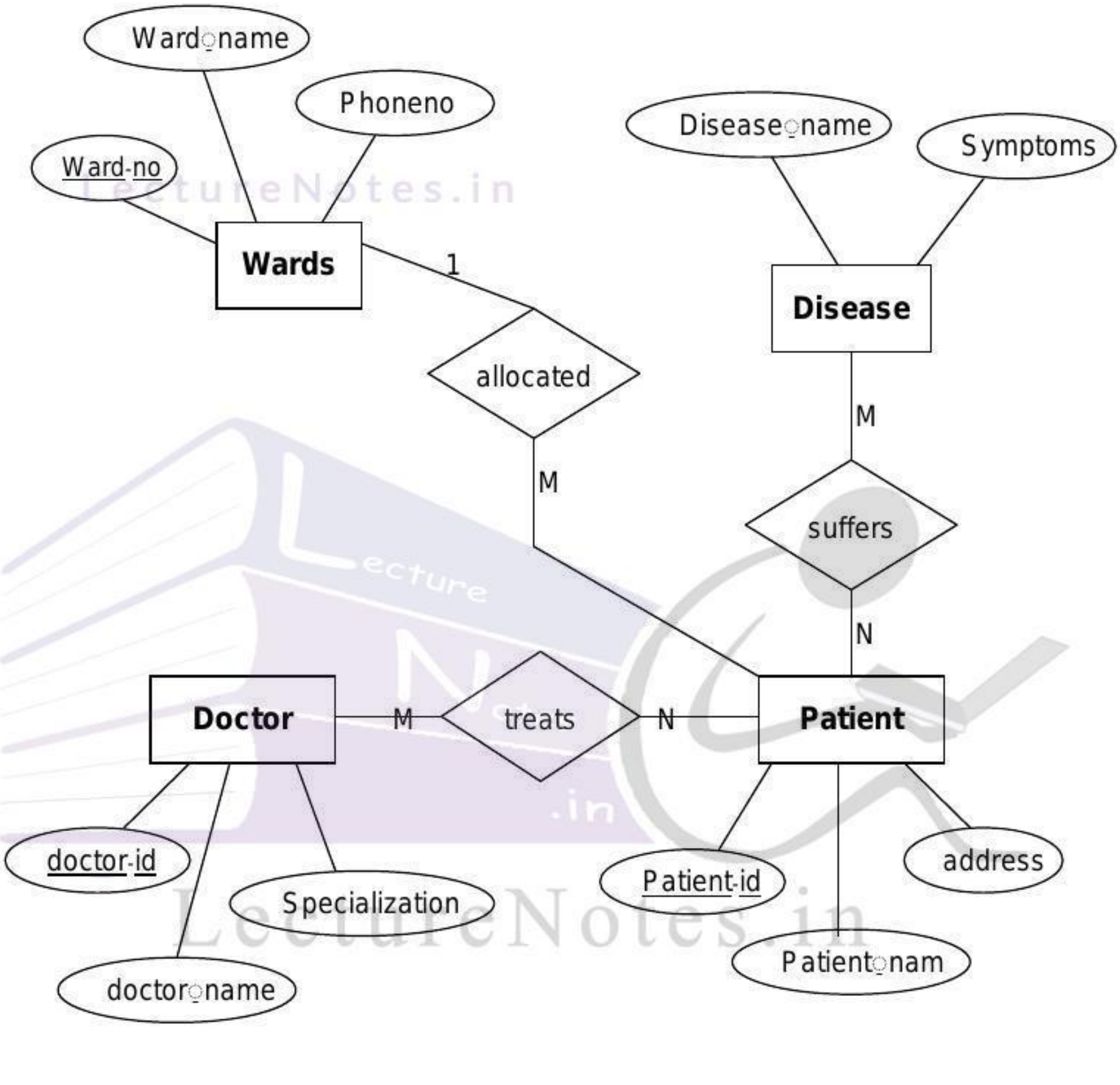
Examples:-



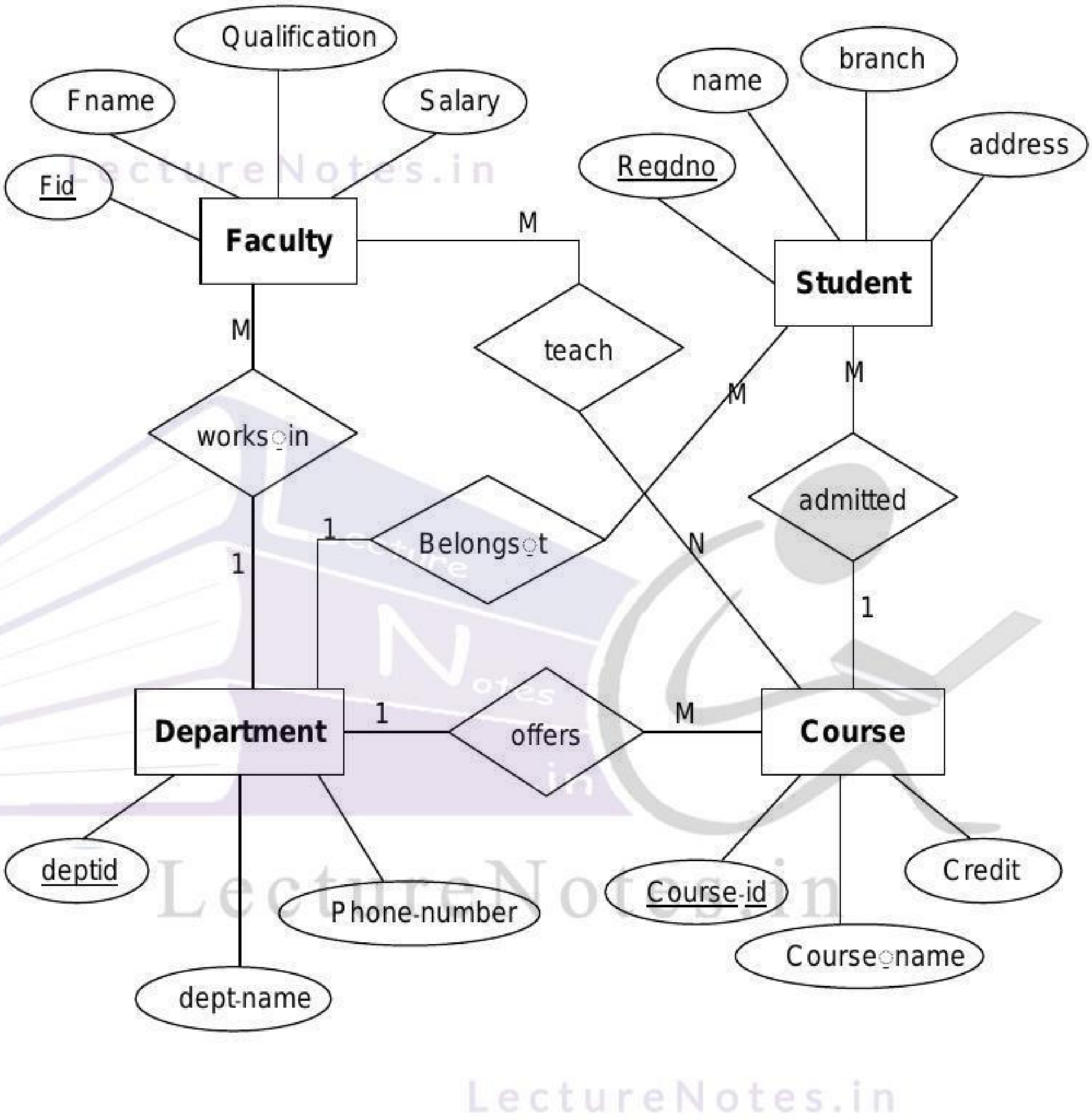
If we design above ER diagrams from top to down then it is specialization, But If we design from bottom to up then it is generalization.

1.9.5 ER Diagram of Different system

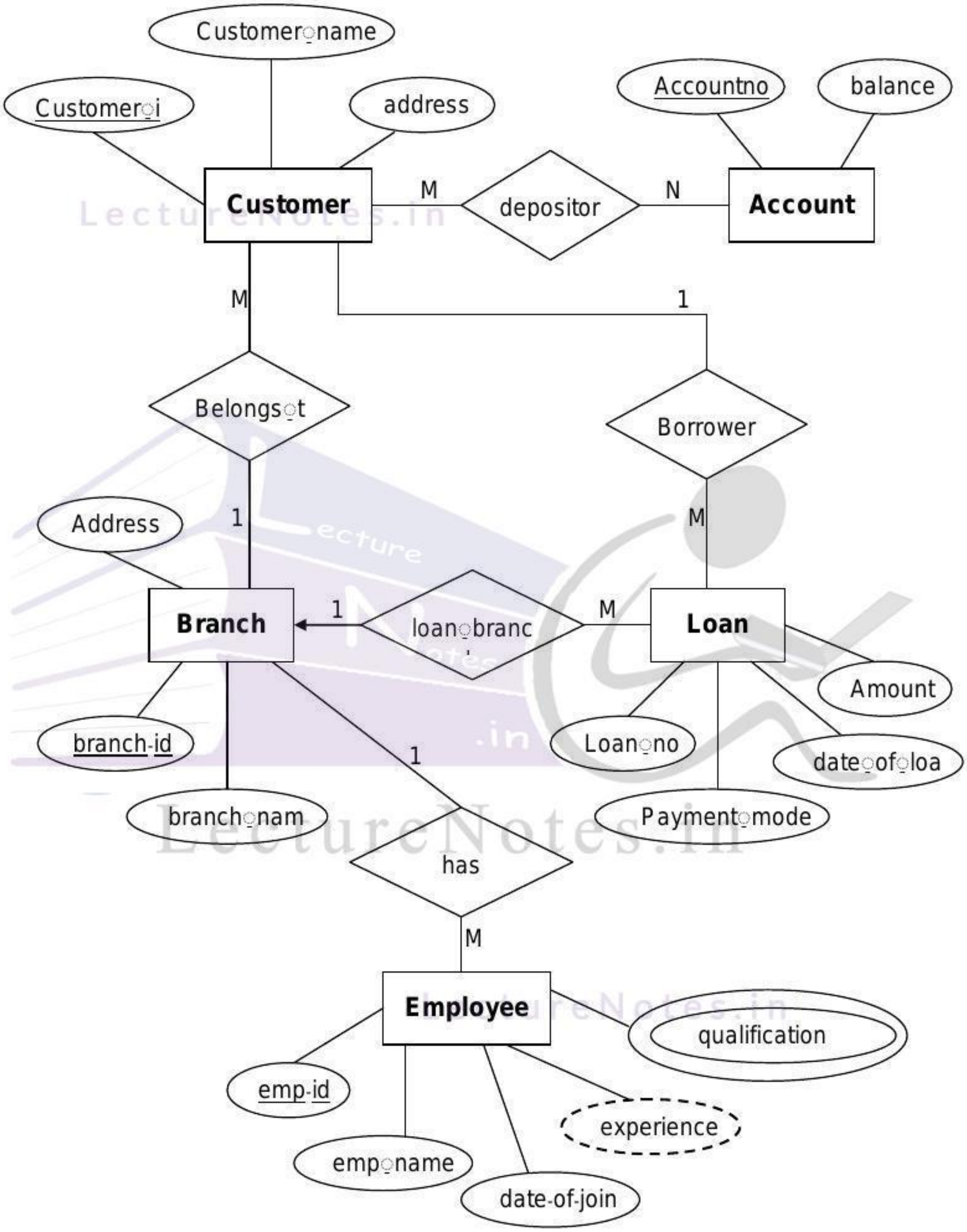
- **ER Diagram of Hospital Management System**



- **ER Diagram of College Automation System**



- **ER Diagram of Banking System**

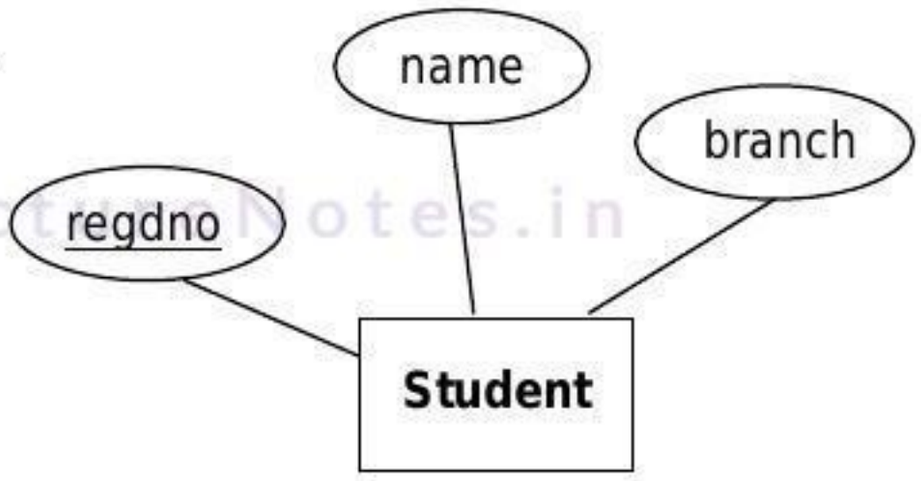


1.10 Mapping/converting ER model to Relation

Method for mapping

- name of relation= Entity set of ER diagram
- name of column =Attribute of ER diagram

Example:-



In the above example, name of relation =student

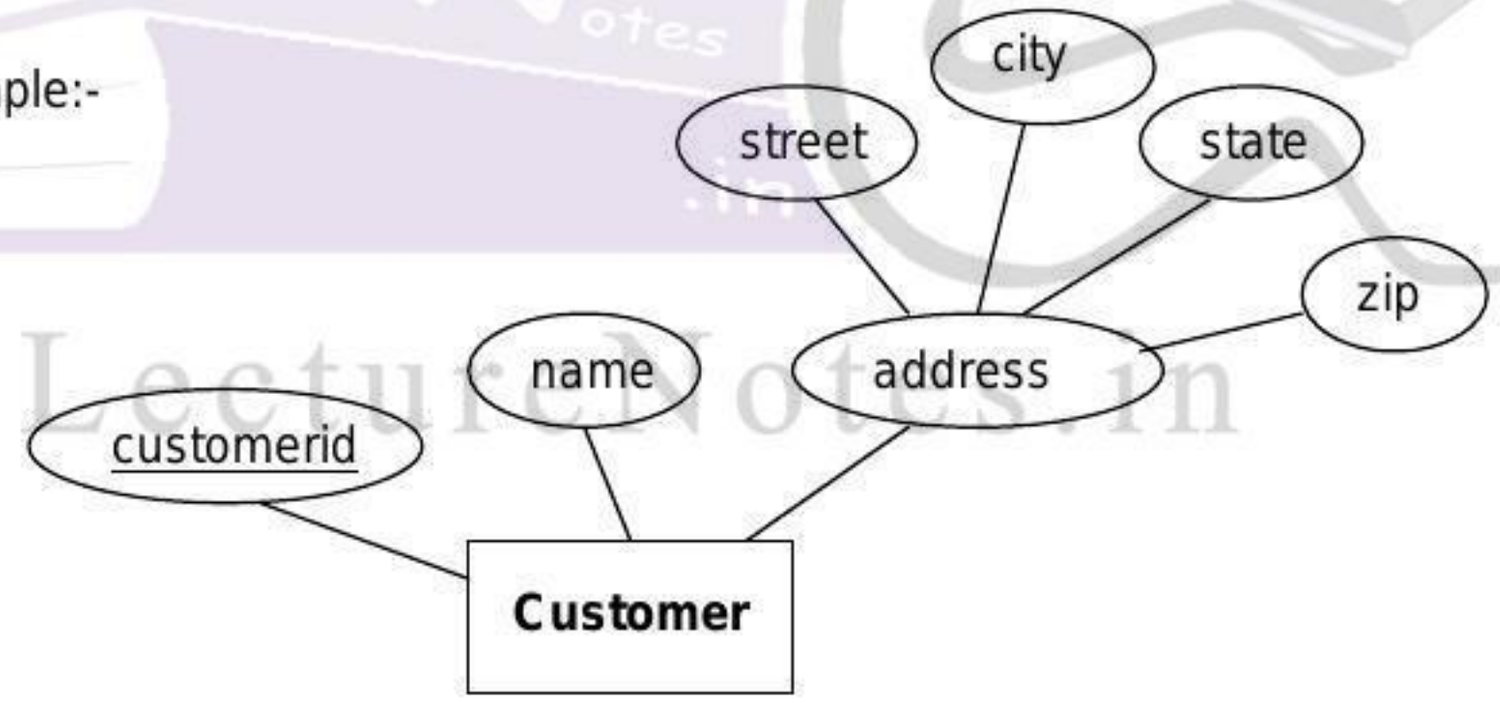
name of columns =regdno, name, branch

The equivalent relation for the above ER diagram is shown below

student	regdno	name	branch

Mapping ER diagram with a composite attribute:-

Example:-



Only the component attributes of the composite attribute are included in the relation.

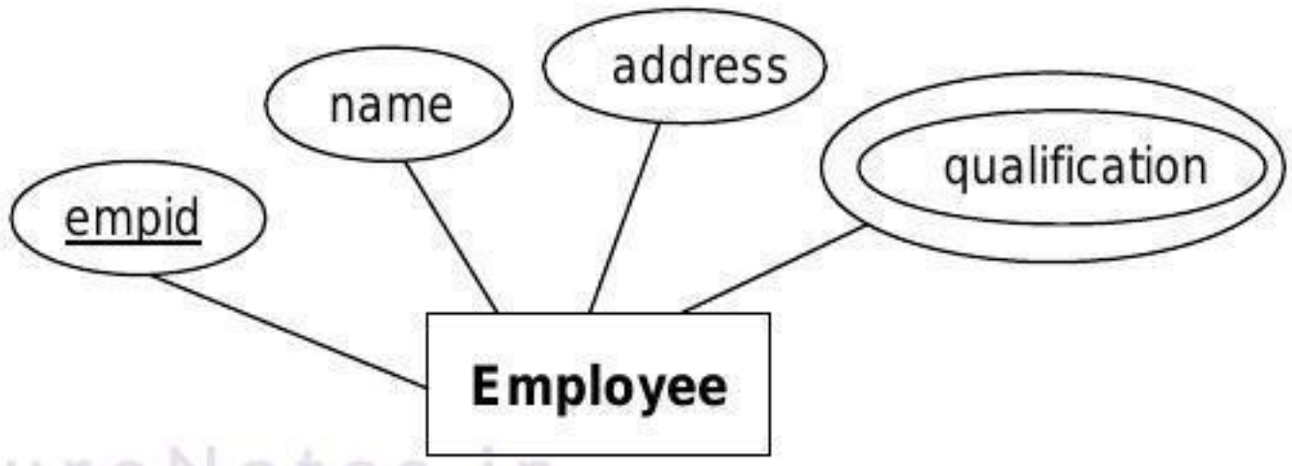
Here composite attribute is address and component attributes are street, city, state, zip

The equivalent relation for the above ER diagram is shown below

customer	customerid	name	street	city	state	zip

Mapping ER diagram with multivalued attribute:-

Example:-



Two relations are created. First contain all attribute except multivalued attribute and second contain primary key & multivalued attribute.

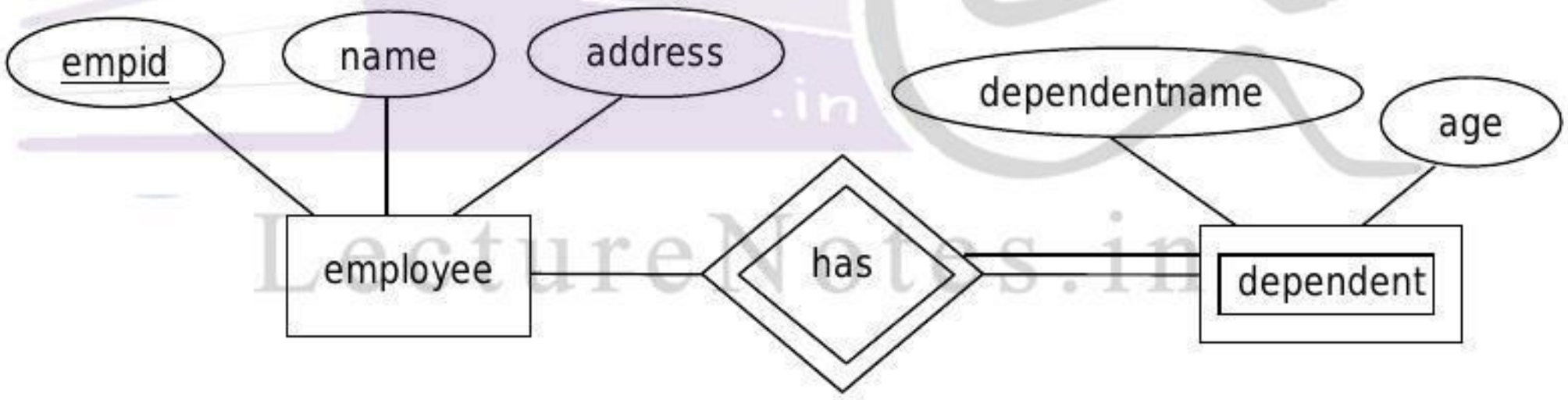
The equivalent relations for the above ER diagram is shown below

employee	empid	name	address

employee-qualification	empid	qualification

Mapping ER diagram with weak entity:-

Example:-



Here two tables are employee and dependent.

Primary key of strong entity is the foreign key of weak entity. The primary key empid of employee is a foreign key of dependent.

The equivalent relations for the above ER diagram is shown below

employee	empid	name	address

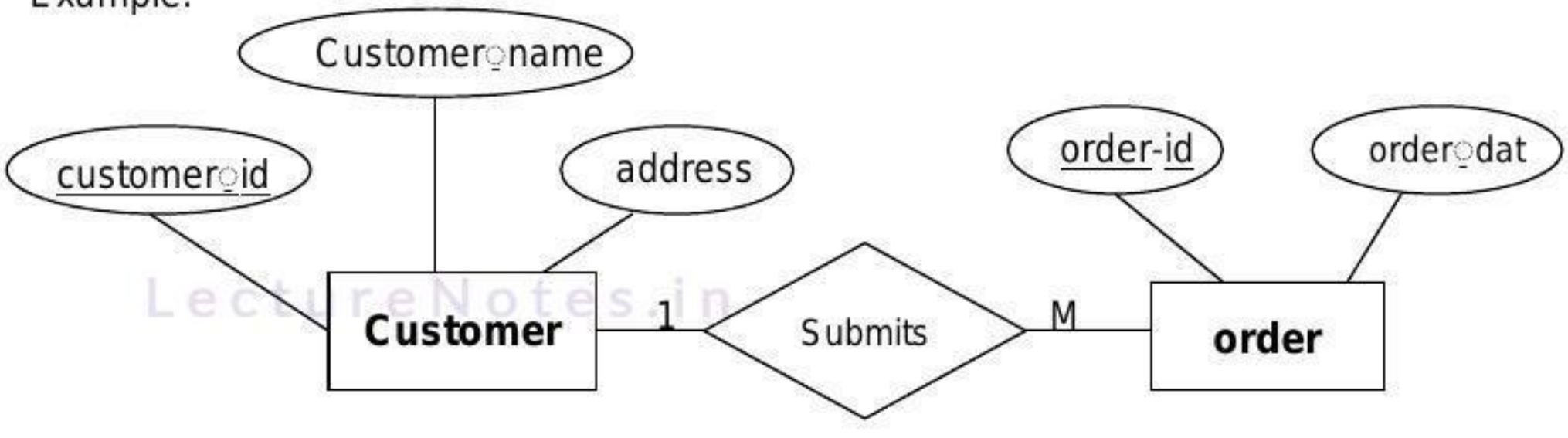
↑

dependent	empid	dependentname	age

Mapping ER diagram with binary relationship:-

Mapping one to many relationship:

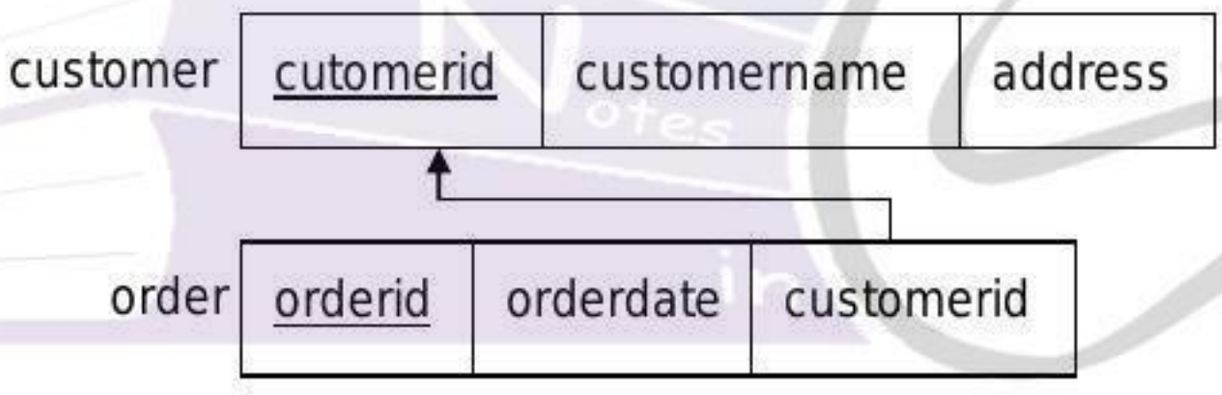
Example:-



create two tables for two entities. Primary key of one side relation is a foreign key for many side relation.

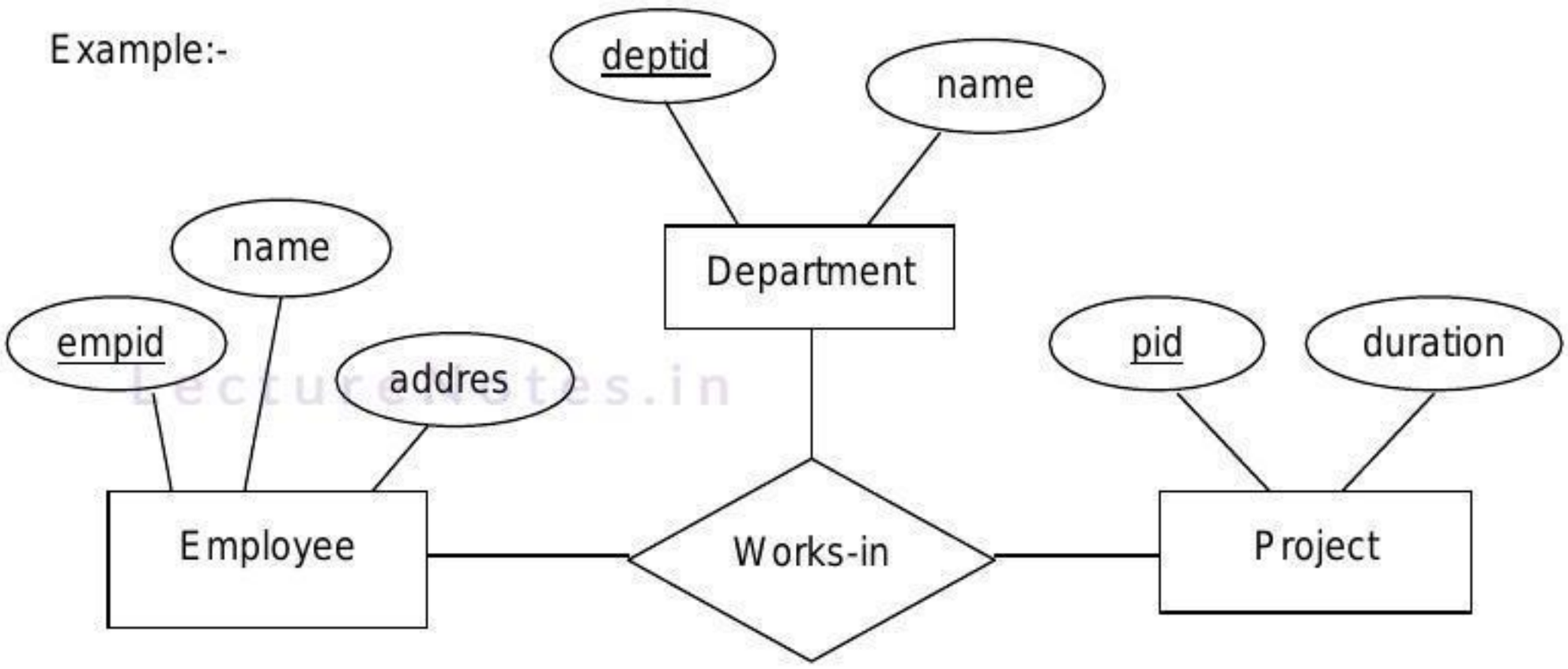
one side relation is customer & many side relation is order. The primary key customerid of customer is a foreign key of order.

The equivalent relations for the above ER diagram is shown below



one-one, many-one, many-many can be done in similar fashion as above.

Mapping ER diagram with ternary relationship:-



3 foreign keys of worksin table are empid,deptid,pid. They refer to primary key of employee,department,project respectively. These attributes are the component of primary key of worksin table i.e. primary key of worksin table is (empid,deptid,pid)
The equivalent relations for the above ER diagram is shown below

employee	empid	name	address

department	deptid	name

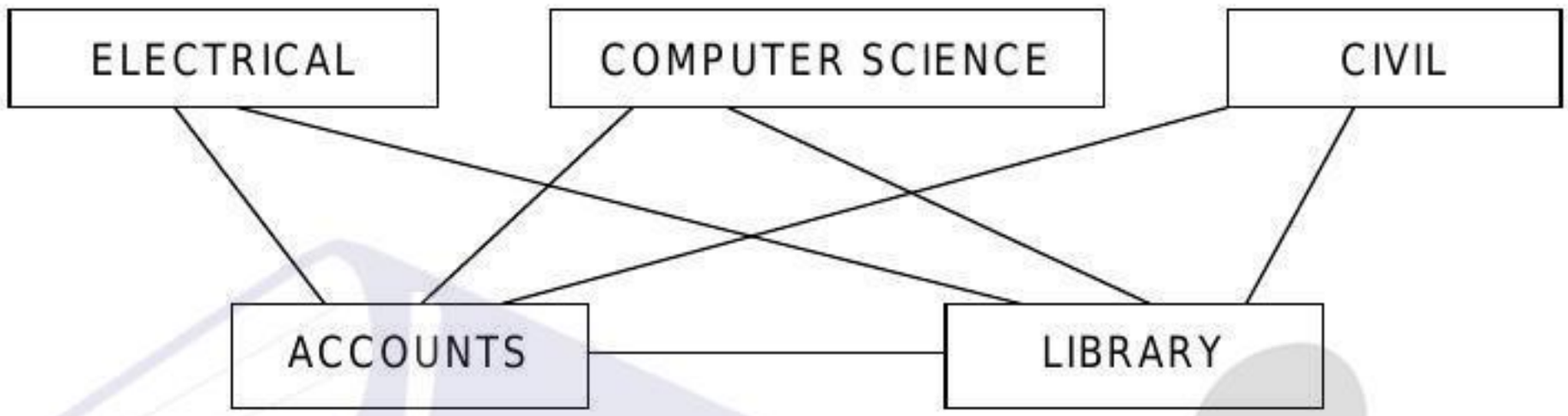
project	pid	duration

worksin	empid	deptid	pid	dateofjoin

1.11 Network Data Model

- ✓ Network Data Model consists of nodes and branches.
- ✓ A node represents a relation. A branch represent a relationship.
- ✓ Any node can be linked with any other node.
- ✓ A child may have multiple parents within the network structure.

Example:-



1.12 Object-Oriented Data Model

- ✓ An object-oriented database system is a database system object-oriented features.
- ✓ Object is similar to an entity in ER model.

  Object = data + methods. (object combine both data and methods as a single unit)

Benefits of Object-Oriented Modeling:

- Ability to deal with real life and complex problems
- Increases consistency in analysis, design, and programming
- Reusability of analysis, design, and programming results
- Improves communication between users, analysts, designers, and programmers
- System robustness

Note:- depending upon requirement appropriate model is chosen to design database.

storage manager includes

i) Buffer manager.
ii) File manager.
iii) Authorization & integration manager
iv) Transaction manager.

i) Buffer manager is responsible for transferring data from main-memory to hard-disk & hard-disk to main-memory.

ii) File manager manages the file on hard-disk storage. It also manages the allocation of space on disk.

iii) Authorization & integrity manager checks authority of users to access data and integrity constraints.

iv) Transaction manager manages all concurrent transaction and ensures that the database remains in a consistent state despite of system failure.

Query processor:- Query processor is responsible for execution of queries.

It includes following

1. **DDL interpreter** translates DDL instructions to machine language.
2. **DML compiler** translates DML instructions into machine language.

   The DML compiler also performs query optimization to generate an efficient query execution plan.
3. **Query evaluation engine** executes the machine language low-level instructions generated by the DDL interpreter & DML compiler to get final output for queries.

1.14 Types of database systems

Individual database

- ✓ Also called microcomputer database.
- ✓ Data and DBMS are under the control of user and stored on a local hard drive.

Company or Shared database

- ✓ stored on a mainframe system and managed by database administrator
- ✓ These databases are used for inventory, production, and sales

Distributed database

- ✓ Database is stored in several computers present at different locations. The computers communicate with one another through via a communications network
- ✓ The computers do not share main memory nor cpu clock.

Advantages **:**

- Reliability and Availability **:** If one site fails in a distributed system, the remaining sites are available. Thus the failure of a site does not imply the shutdown of entire system.
- Faster query processing **:** If query involves data at several sites, it is possible to split the query into sub queries that can be executed in parallel.

Disadvantages **:**

- Difficult to implement
- Difficult to debug/error checking (difficult to ensure correctness of algorithms
- More cost for exchange of messages and additional computation

Web database

- ✓ databases is available via the Internet
- ✓ Search engines use databases to provide information. Example : google
- ✓ Web forms often use CGI (Common Gateway Interface) scripts to process data

1.15 Integrity Constraints

Constraints mean rules/conditions. Integrity means completeness of the database. We achieve integrity by applying constraints on the data. We constrain the data using Not Null, unique, check, primary key or foreign key.

1. Domain Integrity constraint

A domain is defined as the set of values permitted for an attribute. For example, a domain of integer is all possible whole numbers, a domain of day is Monday, Tuesday **...**

Invalid data can't be stored in database. So, integrity of database is preserved.

Example:- age integer CHECK (age >0)

2. Entity Integrity constraint

Null value is not allowed in primary key.

This is because the primary key is used to identify individual rows of a table. If Null value is stored in primary keys, then we cannot identify those rows.

[Null value unknown value. Null value is different from zero or space]

3. Key Integrity constraints (or key constraints)

A key is a attribute(s) whose values uniquely identify each record of a table.

Consider a student table shown below

regdno	name	branch	section
01	Amit	cse	A
02	Sunil	entc	B
03	Prakash	mech	A
04	Sunil	civil	B

Candidate key: It is a attribute(s) whose values uniquely identify each record of a table.

A table can have more than one candidate keys.

Candidate key satisfy following conditions

i) Different records cannot be identical /same.

ii) Subset of candidate key cannot be a key (minimal super key)

For the above student table candidate keys are regdno and branch, because they uniquely identify each record.

Here {regdno,studentname} is not a candidate key since regdno alone is a candidate key.

Primary key: From many candidate keys, database designer select one candidate key for his database called as primary key.

Ex- Let regdno is taken as primary key for above student table.

primary key has following properties

i) uniqueness (two different record cannot be identical)

ii) NOT NULL(null value is not allowed)

primary key is underlined. Example:- regdno

A single column primary key is called simple key & multi-column primary key is called composite primary key.

Ex- regdno is a simple key, {order-no, product-no}is a composite primary key.

Alternate key: candidate keys which are not selected for primary key are called alternate keys.

(i.e. except primary key the remaining candidate keys are alternate keys.)

For the above example we have taken regdno as primary key so the rest candidate key i.e. {branch}is a alternate key.

Super key: It is a superset of candidate key.

For the above student table candidate keys.are {regdno}, {branch}

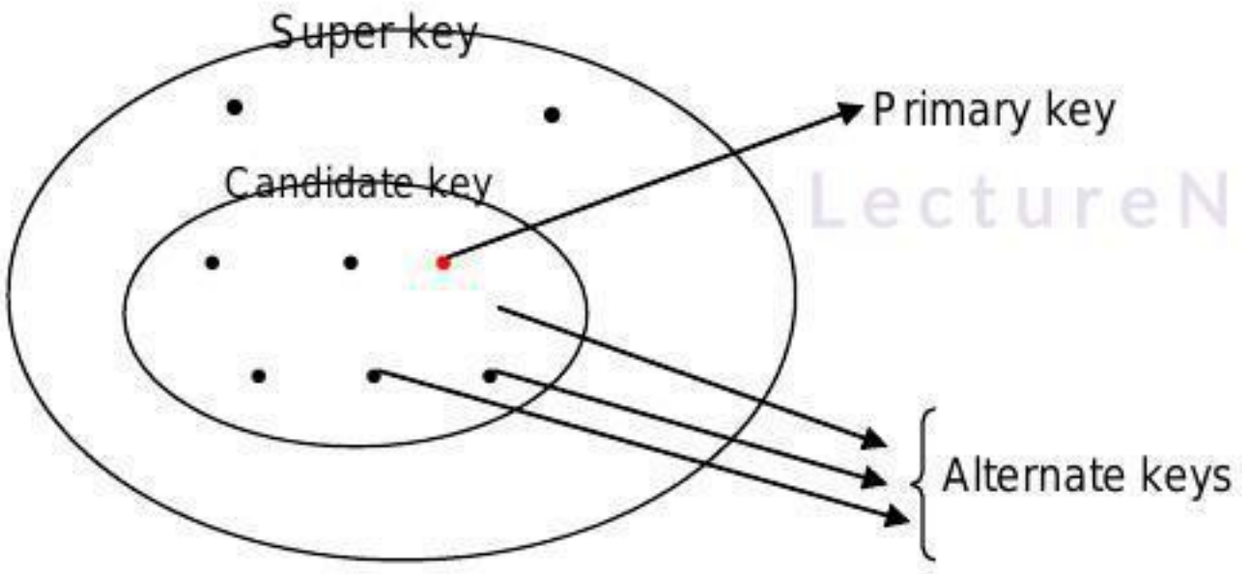
superkeys are **:** {regdno, studentname }

{regdno, address }

{branch, section }

{regdno} etc.

Superkeys are formed by adding extra attributes to candidate key.



4. Referential Integrity Constraints

Referential constraints are applied by primary-key & foreign-key relationship.

Foreign key creates a parent-child relationship between two tables.

The table having PRIMARY KEY is called PARENT table and the table having FOREIGN KEY is called CHILD table.

Example: Consider two tables, student and mark .

primary key — student (Parent table/Referenced table)

regdno	name	branch	section
01	Amit	cse	A
02	Sunil	entc	B
03	Prakash	mech	A
04	Sunil	civil	B

foreign key — mark (Child table /Referencing table)

regdno	semester	mark
01	1st	90
02	1st	70
03	1st	80
04	1st	80
01	2nd	72
02	2nd	85
03	2nd	75
04	2nd	95

FOREIGN KEY of child table refers PRIMARY KEY of parent table.

FOREIGN KEY of mark table is regdno. PRIMARY KEY of student table is regdno.

regdno of mark table refers regdno of student table.

We do not maintain all the attributes of student in mark table (because it will increase redundancy). When mark table(child table) requires any information from student table (Parent table), then it get so by referring its foreign key.

Referential constraints has following rules

Rule1: You cannot delete a record from student (Parent table) if that record exist in mark (Child table).

Rule2: You cannot insert a record into mark (Child table) if that record do not exist in student (Parent table).

[Rule3: You cannot change a primary key value in parent table if that record exists in child table. But, you can insert a Null value in foreign key, specifying that the record is not related]

[However, foreign key constraints can be used to delete and update the data of parent table. They are cascade delete and cascade update.

A. on delete cascade

Example: create table mark(regdno number(10) references student on delete cascade,. . . .);

If we delete the regdno-01 from the student table, the regdno-01 present in 1st and 5th row of mark table is automatically deleted. This constraint overrules the rule1 of referential constraints.

B. on update cascade

Example: create table mark(regdno number(10) references student on update cascade,. . . .);

If we change the regdno-01 to regdno-100 in the student table, this will be automatically changed in mark table. This constraint overrules the rule3 of referential constraints.]

Note :

- Attribute(s) present in primary and foreign key may be same or different.
- A table can have more than one foreign key.

CHAPTER 2

Query language

Query language is divided into two types

1. Procedural language:- Information is retrieved from database by specifying the sequence of operation to be preformed.

Example:- Relational algebra

Sql is based on Relational algebra.

2. Non-procedural language:- Information is retrieved from database without specifying the sequence of operation to be preformed. Users only specify what is the information to be retrieved.

Example:- Relational calculus

QBE (Query By Example) is based on Relational calculus.

2.1 Relational Algebra

Relational algebra consists of set of operations that take one or two relation as an input and produces a new relation as output.

2.1.1 Relational Algebra operations

The operations are

1. Select operation
2. Project operation
3. Rename operation
4. Union operation
5. Intersection operation
6. Difference operation
7. Cartesian product operation
8. Join operation
9. Division operation

Unary operation (operate on one table)

Binary operation (operate on two table)

Select operation:- It displays the records that satisfy a condition.

It is denoted by σ (sigma)

It is a horizontal subset of original relation

syntax:- $\sigma_{condition}$(tablename)

Consider the student table shown below

student

regdno	branch	section
1	cse	A
2	entc	B
3	mech	B
4	civil	A

Q:- display all the records of student table.

Ans:- σ (student)

Q:- display all the records of cse branch in student table.

Ans:- $\sigma_{branch = cse}$ (student)

o/p:-

regdno	branch	section
1	cse	A

Q:- display all the records in student table whose regdno>2.

Ans:- $\sigma_{regdno>2}$ (student)

o/p:-

regdno	branch	section
3	mech	B
4	civil	A

Q:- Display the record of civil branch section A students.

Ans:- $\sigma_{branch = civil \wedge section = A}$ (student)

Q:- Display the records of section B students of cse and civil branch.

Ans:- $\sigma_{section = B \wedge branch = cse \vee branch = civil}$ (student)

Project operation:- It displays the specific column(s) of a table.

It is denoted by $\prod$ (pie)

It is a vertical subset of original relation. It eliminates duplicate tuples.

Q:- display regdno column of student table.

Ans:- $\prod_{regdno}(student)$

o/p:-

regdno
1
2
3
4

Q:- display branch, section column of student table.

Ans:- $\prod_{branch,\ section}(student)$

o/p:-

branch	section
cse	A
entc	B
mech	B
civil	A

Q:-display regdno, section of entc students.

Ans:- $\prod_{regdno,section}(\sigma_{branch=entc}(student))$

o/p:-

regdno	section
2	B

Note:- condition can be written in select operation but not in project operation.

Rename operation:- It is used to assign a new name to a relation.

It is denoted by ρ (rho)

Syntax:- $\rho_{newname}$ (tablename or expression)

Example1:- $\rho_{newstudent}(student)$:The student table is renamed with newstudent.

Example2:- $\rho_{newname,\ newbranch}(\prod_{name,branch}(student))$:The name, branch column of student table are renamed with newname and newbranch respectively.

Binary operations are applied on two compatible relations. Two relations R1, R2 are said to be compatible if they are of same degree and the domain of corresponding attributes are same.

union operation:- union operation combine values in R1,R2 removing duplicate ones.

R1

regdno	branch	section
1	cse	A
2	entc	B
3	mech	B
4	civil	A
5	cse	B

R2

regdno	branch	section
1	civil	A
2	cse	A
3	entc	B

Q:- Display all the regdno of R1 & R2

Ans:- $\prod_{regdno}(R1) \cup \prod_{regdno}(R2)$

o/p:-

regdno
1
2
3
4
5

intersect operation:- displays the common values in R1 & R2.

It is denoted by $\cap$

Q:- Display regdno which are common to R1 & R2

$\prod_{regdno}(R1) \cap \prod_{regdno}(R2)$

o/p:-

regdno
1
2
3

Difference operation:- display the values present in R1 but not in R2.

It is denoted by –

Q:- Display regdno which are present in R1 but not in R2.

$\prod_{regdno}(R1) - \prod_{regdno}(R2)$

o/p:-

regdno
4
5

Cartesian product operation:- combines R1 and R2 without any condition.

It is denoted by X

Degree of R1XR2 = degree of R1+ degree of R2 [degree = total no of columns]

R1

regdno	branch	section
1	-	-
2	-	-
3	-	-
4	-	-
5	-	-

R2

name	regdno
s	2
d	4

⇒ R1XR2 =

regdno	branch	section	name	regdno
1	-	-	s	2
1	-	-	d	4
2	-	-	s	2
2	-	-	d	4
3	-	-	s	2
3	-	-	d	4
4	-	-	s	2
4	-	-	d	4
5	-	-	s	2
5	-	-	d	4

join operation:- combines R1 and R2 with respect to a condition. It is denoted by $\bowtie$

Theta join:- If we join R1 & R2 other than the equal to condition then it is called theta join /non-equi join

Example: $R1 \bowtie R2$ with condition R1.regdno>R2.regdno

regdno	branch	section	name	regdno
3	-	-	s	2
4	-	-	s	2
5	-	-	s	2
5	-	-	d	4

In the join operation, we select those rows from cartesian-product where R1.regdno > R2.regdno.

$\Rightarrow$ join operation=select operation + Cartesian product operation

Natural join:- If we join R1 & R2 on equality condition then it is called Natural join or equi join. Generally, join is referred as natural join. Natural join of R1 & R2 is shown below (i.e. we select those tuples from Cartesian product where R1.regdno=R2.regdno)

$\Rightarrow R1 \bowtie R2 =$

regdno	branch	section	name
2	-	-	s
4	-	-	d

Outer join:- It is an extension of natural join to deal with missing values of relation.

Consider R1 & R2 shown below

R1

regdno	branch	section
1	-	-
2	-	-
3	-	-
4	-	-
5	-	-

R2

name	regdno
s	2
d	4
e	7

Outer join is of 3 types

i) Left outer join:- denoted by $R1 ⟕ R2$

regdno	branch	section	name	regdno
2	-	-	s	2
4	-	-	d	4
1	-	-	NULL	NULL
3	-	-	NULL	NULL
5	-	-	NULL	NULL

here all the tuples of R1(left table) appears in output. The mismatching values of R2 are filled with NULL.

⇒ Left outer join = natural join + mismatch/extra tuple of R1

ii) Right outer join:- denoted by $R1 ⟖ R2$

here all the tuples of R2(right table) appears in output. The mismatching values of R1 are filled with NULL.

regdno	branch	section	collegename	regdno
2	-	-	s	2
4	-	-	d	4
NULL	NULL	NULL	e	7

⇒ Right outer join = natural join + mismatch/extra tuple of R2

iii) Full outer join:- denoted by $R1 ⟗ R2$

Full outer join = left outer join U right outer join

regdno	branch	section	collegename	regdno
2	-	-	s	2
4	-	-	d	4
1	-	-	NULL	NULL
3	-	-	NULL	NULL
5	-	-	NULL	NULL
NULL	NULL	NULL	e	7

division operation: The division operator is used for queries which involve the 'all'.

R1 ÷ R2 = tuples of R1 associated with all tuples of R2

A

sno	*pno*
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

B2

pno
p2
p4

B3

pno
p1
p2
p4

A÷B1

sno
s1
s2
s3
s4

A÷B2

sno
s1
s4

A÷B3

sno
s1

Generalized projection:- It perform mathematical operation on the attribute of a relation.

Example:- $\Pi_{empid,salary+1000}$ (emp)

emp

empid	salary
e1	8000

emp

empid	salary+1000
e1	9000

Aggregate functions: aggregate functions are max(), min(), sum(), average(),count()

It is denoted by $\mathcal{G}$ (Caligraphic G)

Examples: $\mathcal{G}_{max(salary)}$ (emp) – display maximum value in salary column of emp table

$\mathcal{G}_{sum(salary)}$ (emp) – display total-value in salary column of emp table

$\mathcal{G}_{count(city)}$ (emp) – display no of cities

$\mathcal{G}_{countdistinct(name)}$ (emp) – display no of cities which are distinct

Grouping :- $_{dept}\mathcal{G}_{avg(salary)}$ (emp) – display the average salary of each department.

Exercises:

Consider the following set of tables

emp(empid,ename,salary)

dept(deptid,dname)

project(projectid,pname)

worksin(empid,deptid)

assign(empid,projectid)

employee

empid	empname	salary
e1	ravi	40000
e2	sanjay	35000
e3	smruti	30000
e4	alok	30000
e5	pritam	30000

dept

deptid	dname
d1	cse
d2	ele
d3	civil

project

projectid	pname
p1	database
p2	networking

worksin

empid	deptid
e1	d1
e2	d3
e3	d2
e4	d2
e5	d1

assign

empid	projectid
e1	p1
e2	p2
e3	p1
e4	p2
e5	P2

Q:- display the details employee of who works in electrical department.

Ans:-

Step1 :- $T1=\prod_{deptid}(\sigma_{dname=ele}(dept))$

o/p :

T1

deptid
d2

i.e. T1= deptid of electrical

Step2 :-$T2 = \prod_{empid}(T1 \bowtie worksin)$

i.e T1.deptid = worksin.deptid

o/p :

T2

empid
e3
e4

i.e. T2 = all empid of electrical

Step3 :-T3 = (T2 ⋈ emp)

i.e T2.empid =emp.empid

o/p :

T3

empid	empname	salary
e3	smruti	30000
e4	alok	30000

i.e. T3=details of all employees of ele

Q:- display the names of employee of who works on database project

Ans:-

Step1 :- $T1 = \prod_{projectid}(\sigma_{pname=database}(project))$

T1

projectid
p1

Step2 :- $T2 = \prod_{empid}(T1 \bowtie assign)$

T2

empid
e1
e3

Step3 :- $T3 = \prod_{ename}(T2 \bowtie emp)$

o/p :

T3

empname
ravi
smruti

Q:- display the names of employee who works on those project on which Sanjay works.

Ans:-

Step1 :- Find empid of sanjay from employee

$$T1 = \prod_{empid} (\sigma_{ename = Sanjay}(emp))$$ (i.e. T1 contains e2)

Step2 :- join T1 with assign and display projectid.

$$T2 = \prod_{projectid} (T1 \bowtie assign)$$ (i.e. T2 contains projectid of Sanjay)

Step3 :- join T2 with assign and display empid.

$$T3 = \prod_{empid} (T2 \bowtie assign)$$

(i.e. T3 = all empid whose projectid same as Sanjay's projectid)

Step4 :- join T3 with employee and display empname.

$$T4 = \prod_{empname} (T3 \bowtie emp)$$

(i.e. T4 = employee-name working on same project as Sanjay)

Q:- display the name of highest paid employee of organization.

Ans:- $T1 = \mathcal{G}_{max(salary)}(emp)$ (i.e. salary column of T1 contain max^{m} salary)

$$T2 = \prod_{ename}(T1 \bowtie emp)$$ (T2 =all employee-name who are getting max^{m} salary)

Q:- display the name of second-highest paid employee of organization.

Ans:- $T1 = \prod_{salary}(emp)$ (i.e. T1 contain all salary values)

$T2 = \mathcal{G}_{max(salary)}(emp)$ (i.e. T2 contain max^{m} salary value)

$T3 = (T1 - T2)$ (i.e. T3 contain all salary values except max^{m} salary value)

$T4 = \prod_{empid} (\mathcal{G}_{max(salary)}(T3))$ (i.e. T4 contain empid having 2nd max^{m} salary value)

$T5 = \prod_{ename} (T4 \bowtie emp)$

(i.e. T5 contain all employee-name who are getting 2nd max^{m} salary)

Q:- display the empid who do not work on any project.

Ans:- $T1 = \prod_{empid}(emp)$ (i.e. T1 contain empid of all employees)

$T2 = \prod_{empid}(assign)$ (i.e. T2 contain empid of who are working)

$T3 = (T1 - T2)$ (i.e. T3 contain empid who are not working)

$T4 = \prod_{empname}(T3 \bowtie emp)$ (i.e. T3 contain empname who are not working)

Consider the following set of tables

boat(boatid, bname, color) **traveller**(tid, tname) **reservation**(boatid, tid, day)

Q:- display the name of travellers who have reserved the red color boat.

Ans:- $T1 = \prod_{boatid}(\sigma_{color=red}(boats))$ Suppose 1,3,5 are the boats having red color

$T2 = \prod_{tid}(T1 \bowtie reservation)$ T2 = tid attached with 1 , 3 or 5

$T3 = \prod_{tname}(T2 \bowtie traveller)$ T3=name of travellers matching with T2

Q:- display the name of travellers who have reserved the boats on friday.

Ans:- $T1 = \prod_{tid}(\sigma_{day=friday}(reservation))$

$T2 = \prod_{tname}(T1 \bowtie traveller)$

Q:- display the details of travellers who have reserved **all** the boats on monday.

Ans:- $T1 = \prod_{boatid}(\sigma_{day=monday}(reservation))$

$T2 = \prod_{tid}(reservation \div T1)$

$T3 = (T2 \bowtie traveller)$

Q:- display the details of boat that was reserved by Alok.

Ans:- $T1 = \prod_{tid}(\sigma_{tname=Alok}(traveller))$

$T2 = \prod_{boatid}(T1 \bowtie reservation)$

$T3 = (T2 \bowtie boat)$

2.2 Relational Calculus

It is a non-procedural query language in which information is retrieved from the database without specifying sequence of operation to be performed.

Relational calculus is of two types

1. Tuple calculus
2. Domain calculus

2.2.1 Tuple Calculus

The query is of the form { t | p(t) } , where t is a tuple variable and p is condition/predicate.

p(t) can be written using following

(i) $t \in R$ (t belongs to relation R)

(ii) t [A] operator value (A is attribute of R & operator can be $<,>,\leq,\geq,=,\neq$)

(iii) t1[A] operator t2[B] (t1, t2 are tuple variable of their relation)

Consider a table student(name,regdno,branch)

Q:- display the details of the students of cse branch

Ans- $\{t \mid t \in student \wedge t[branch] = 'cse'\}$

Q:- display the details of students whose regnno>4

Ans- $\{t \mid t \in student \wedge t[regdno] > 4\}$

Quantifier is of two types

there exist

i) Existential quantifier($\exists$):- if F is a formula/expression then $\exists t(F)$ evaluates to true for some tuple t

for all

ii) Universal quantifier($\forall$):- if F is a formula then $\forall t(F)$ evaluates to true for all tuple t

A tuple variable t is said to be bound variable if it is quantified (i.e. $\exists$ or $\forall$ symbol is present in expression/formula)

A tuple variable t is said to be free variable if it is not quantified (i.e. $\exists$ or $\forall$ is not present in expression)

Note: An expression is said to be finite expression if it yields finite number of tuples otherwise it is an unsafe expression Ex:- {t | NOT (t∈student)} is an unsafe expression

Consider the following tables

student (regdno, name, branch)

books (bookid, bname, author, price)

issue (regdno, bookid)

Q:- display the details of book written by Navathe

Ans- $\{ t \mid t \in books \wedge t[author] = 'Navathe' \}$

Q:- display the name of book whose price is greater than 500

Ans- $\{ t[bname] \mid t \in books \wedge t[price] > 500 \}$

Q:- display the details of book issued by regdno 6

Ans- $\{ t \mid t \in books \wedge t1 \in issue \wedge t1[regdno] = 6 \wedge t[bookid] = t1[bookid] \}$

Q:- display the names of book issued by Amar

Ans- $\{ t[bname] \mid \exists\ t1 \in student \wedge t1[name] = Amar \wedge \exists\ t2 \in issue \wedge t2[regdno] = t1[regdno] \wedge t \in books \wedge t[bookid] = t2[bookid] \}$

2.2.2 Domain Calculus:- The query is of the form $\{<x1,x2.....xn> \mid p(x1,x2.....xn)\}$,where x1,x2.....xn are domain variable(i.e. column of table) and p is condition/predicate which can be written using following

i) $<x1,x2.....xn> \in R$

ii) xi operator value (xi is domain variable of R & operator can be $<, >, \leq, \geq, =, \neq$)

iii) xi operator yi (xi,yi are domain variable of their relation)

Q:- display the details of book written by Navathe

Ans- $\{ <b1,b2,b3,b4> \mid <b1,b2,b3,b4> \in books \wedge b3 = 'Navathe' \}$

Q:- display the name of book whose price is greater than 500

Ans- $\{ b2 \mid <b1,b2,b3,b4> \in books \wedge b4 > 500 \}$

Q:- display the details of book issued by regdno 6

Ans- $\{ <b1,b2,b3,b4> \mid \exists\ <i1,i2> \in issue \wedge i1 = 6 \wedge <b1,b2,b3,b4> \in books \wedge i2 = b1 \}$

Q:- display the names of book issued by Amar

Ans- $\{ b2 \mid \exists <s1,s2,s3,s4> \in student \wedge s2 = Amar \wedge \exists <i1,i2> \in issue \wedge i1 = s1 \wedge <b1,b2,b3,b4> \in books \wedge i2 = b1 \}$

Exercises

1. Let R(a, b, c) and S(d, e, f) are two relations. Convert the following relational algebra to tuple calculus.

a. $\prod_a(R)$

b. $\sigma_{b=17}(R)$

c. $R \times S$

d. $\prod_{a,f}(\sigma_{c=d}(R \times S))$

Answer:

a. $\{t \mid \exists q \in R\ (q[a] = t[a])\ \}$

b. $\{t \mid t \in R\ \wedge t[b] = 17\}$

c. $\{t \mid \exists p \in R\ \exists q \in S\ (t[a] = p[a] \wedge t[b]=p[b] \wedge t[c]=p[c] \wedge t[d] =q[d] \wedge t[e]=q[e] \wedge t[f] = q[f])\ \}$

d. $\{t \mid \exists p \in R\ \exists q \in S\ (t[a] = p[a] \wedge t[f] = q[f] \wedge p[c] = q[d])\ \}$

2. Let $R_1 = (a, b, c)$ and $R_2 = (a, b, c)$ are two relations. Give an expression in domain relational calculus that is equivalent to each of the following:

a. $\prod_a(R_1)$

b. $\sigma_{b=17}(R_1)$

c. $R_1 \cup R_2$

d. $R_1 \cap R_2$

e. $R_1 - R_2$

Answer:

a. $\{<t> \mid \exists p, q\ (<t, p, q> \in R_1)\ \}$

b. $\{<a, b, c> \mid <a, b, c> \in R_1 \wedge b = 17\ \}$

c. $\{<a, b, c> \mid <a, b, c> \in R_1 \vee <a, b, c> \in R_2\}$

d. $\{<a, b, c> \mid <a, b, c> \in R_1 \wedge <a, b, c> \in R_2\}$

e. $\{<a, b, c> \mid <a, b, c> \in R_1 \wedge <a, b, c> \neg\in R_2\}$

3. Let R(a, b) and S(a, c) are two relations. Write relational algebra expressions equivalent to the following domain-relational calculus expressions:

a. $\{<a> \mid \exists\ b(<a, b> \in R\ \wedge b = 17)\ \}$

b. $\{<a, b, c> \mid <a, b > \in R\ \wedge\ <a, c> \in S\}$

c. $\{<a> \mid \exists c(<a, c> \in S\ \wedge \exists b_1, b_2(<a, b_1> \in R\ \wedge <c, b_2> \in R\ \wedge b_1 > b_2))\ \}$

Answer:

a. $\prod_a(\sigma_{b=17}(R))$

b. $R \bowtie S$

c. $\prod_{R[a]}((R \bowtie S) \bowtie_{c = R2[a] \wedge R[b] > R2[b]} (\rho_{R2}(R)))$

2.3 QBE (Query By Example)

QBE is a non-procedural language based on domain calculus. It has two dimensional syntax for expressing a query. i.e. by skeleton table shown below

book	bookid	bname	author	price

The queries are expressed by example rows (which consist of domain variable)

The domain variable is preceded by an underscore.

P. is used to print the value of corresponding column.

Q:- display the details of written by Navathe

Ans-

book	bookid	bname	author	price
P.			Navathe	

Q:- display the name of book whose price is greater than 500

Ans-

book	bookid	bname	author	price
		P.		>500

Q:- display the details of book issued by regdno 6

Ans-

issue	bookid	regdno
	_b1	6

book	bookid	bname	author	price
P.	_b1			

Q:- display the names of book issued to Amar

Ans-

student	regdno	name	branch
	_s1	Amar	

issue	bookid	regdno
	_b1	_s1

book	bookid	bname	author	price
	_b1	P.		

Aggregate functions are MAX. , MIN. , SUM. , AVG. , CNT.

Q:- display total no of books

Ans-

book	bookid	bname	author	price
	P. CNT.			

Q:- display total price of all books

Ans-

book	bookid	bname	author	price
				P. SUM.

Grouping (G.)

Q:- display number of employees in each department

Ans

emp	empid	ename	salary	city	dname
	P.CNT.				P.G.

Q:- Display Name Of highest paid employee of organization.

emp	empid	ename	salary	city	dname
		P.	MAX. _e1 _e1		

2.4 Phases of Database Development Life Cycle (DDLC)

1. Requirement Analysis

The most important step in implementing a database system is to find out what is needed i.e. what type of a database is required for the business organization, daily volume of the data, how much data needs to be stored in the master files etc. In order to collect all these information, a database analyst spend a lot of time within the business organization talking to people, end users and get acquainted with the day-to-day process.

2. Database Design

In this phase the database designers will make a decision on the database model that perfectly suits for the organization's requirement. The database designers will study the documents prepared by the analysts in requirement analysis stage and then start developing a system model that fulfills the needs.

3. Evaluation and Selection

In this phase, we evaluate the diverse database management systems and choose the one that perfectly suit the requirements of organization. In order to identify best performing database, end users should be involved.

4. Logical Database Design

Once the evaluation and selection phase is completed successfully, the next step is logical database design. This design is translated into internal model which includes mapping of all objects i.e. design of tables, indexes, views, transactions, access privileges etc.

5. Physical Database Design

This phase selects and characterizes the data storage and data access of the database. The data storage depends on the type of devices supported by the hardware, the data access methods.

Physical design is very vital because bad design can result in poor performance.

6. Implementation

Database implementation needs the formation of special storage related constructs. These constructs consist of storage group, table spaces, data files, tables etc.

7. Data Loading

Once the database has been created, the data must be loaded into the database. The data required to be converted, if the loaded data is in a different format.

8. Testing and Performance Tuning

In this phase, database is tested and tuned for performance. Database administrators and application programmers work together during this phase, because testing and performance tuning happens in parallel.

9. Operation

In operation phase, the database is accessed by the end users and application programs. This stage includes adding of new data, modifying existing data and deletion of obsolete data. This phase provides useful information and helps management to make a business decision.

10. Maintenance

It is one of the ongoing phases in DDLC. Factors such as new business needs, new information requirements formulate ongoing changes and improvements to the existing design. The major tasks include: database backup and recovery, access management , hardware maintenance etc.

2.5 Relational database design

Following four factors decides the quality of database design.

(a) Redundancy:- Some information is stored repeatedly. (so, wastage of memory occurs)The main goal of relational data base design is to store information without unnecessary redundancy.

Example:-

Empid	Name	Designation	Salary
001	ram	Manager	60000
002	hari	officer	40000
003	gopal	officer	40000
004	bishnu	Manager	60000

Here, Designation attribute determines Salary attribute i.e. for a particular Designation, there is only one permissible Salary value. So for a particular Designation value its associated hourly wages is repeated.

Problems / anomalies caused by redundancy:

(i) Insertion anomalies / (problem of inserting new data): we can't insert data unless some other data is stored. Ex:- we cannot insert a tuple for an employee unless we know the Salary for his designation.

(ii) Deletion anomalies / (problem of updating existing data): we can't delete certain related data. Ex:- if we delete all tuples having Designation as Manager, we lose the association between Designation and Salary.

(iii) Update anomalies / (problem of updating existing data): we can't update certain related data. Ex:- if Salary of Manager is updated in one record then we must update in each of its repetition/copy else inconsistency is created.

(b) NULL value:- NULL value take unnecessary memory. So, database should reduce NULL value in the attribute.

(c) Semantics of the attribute:- It refers to the interpretation of the value in a attribute. Valid / Meaningful data must be stored.

(d) Spurious tuple:- It refers to extra tuple(s) generated during join operation of two table. Database should control this to avoid inconsistency.

2.6 SQL

1. Command Name :- create

Description:- It is used to create table, view, index.

Syntax:- Create table tablename (column1 datatype (size), column2 datatype (size) …..(columnN datatype (size));

Example:- create table student (name char (30),regdno number(10),branch char (20));

Output:- Table created.

2. Command Name:- desc

Syntax **:** desc table name

Example **:** desc student

Description:- It describes structure /schema of a table.

Output :-

Name	Null	Type
NAME REGDNO BRANCH		CHAR (30) NUMBER (10) CHAR (20)

3. Command Name:- insert

Description:-It is used to insert data into table /view /index.

Syntax:- insert into tablename values (value1,value2,….value n);

Example:-

insert into student values ('bikash', 1001229055,'CSE');

Output:- 1 row created.

insert into student values ('smruti', 1001229068,'CSE');

Output:- 1 row created.

insert into student values ('smruti', 1001229068,'CSE');

1 row created.

insert into student values ('amaresh', 1001229077,'CSE');

Output:- 1 row created.

insert into student values ('mahesh', 1001229092,'CSE');

Output:- 1 row created.

4. Command Name:- select

Syntax:- select column1, column2,……,column1n from tablename.

Description:- It is used to display records/column values of a table.

Example:- select name,regdno,branch from student OR select * from student

Output:-

NAME	REGDNO	BRANCH
bikash	1001229055	CSE
smruti	1001229068	CSE
smruti	1001229068	CSE
amaresh	1001229077	CSE
mahesh	1001229092	CSE

Example:- select distinct name from student

Description: It display distinct name in sorted order.

Output:-

NAME	REGDNO	BRANCH
amaresh	1001229077	CSE
bikash	1001229055	CSE
mahesh	1001229092	CSE
smruti	1001229068	CSE

In the output, Name is displayed in sorted order removing duplicate values.

5. Command Name:- commit

Description: it is used to save the data permanently to database.

Example:- commit;

Output:- commited.

EXAMPLE

Step-1

Create table Book (bookname char(30), authorname char(30), noofcopies number(5));

Output:-Table Created.

Step-2

desc Book

Output:-

NAME	NULL?	TYPE
BOOK NAME		CHAR (30)
AUTHORNAME		CHAR (30)
NO OF COPIES		NUMBER (5)

Step-3

insert into book values ('DAA', 'CORMEN', 80);

insert into book values ('database engineering', 'Korth', 60);

insert into book values ('system programming', 'J.J Donovan', 50);

Output:- 3 rows created.

Step-4

select distinct * form Book

Output:-

BOOKNAME	AUTHORNAME	NOOFCOPIES
DAA	CORMEN	80
database engineering	Korth	60
system programming	J.J Donovan	50

Step-5

select lower(bookname) from book;

Description: - Display the from bookname field in lower case letter.

Output:-

LOWER (BOOKNAME)
daa
database engineering
system programming

Syntax:- update table tablename set column1=newvalue, column2=newvalue ……. where condition;

Example1:- update student set name='prakash' where name='bikash'

Output:- 1 row updated..

Example2:- update student set branch='civil',name='sambit' where name='mahesh';

Output:- 1 row updated

Example3:- update student set semester='4th' where regdno>0

Output:- 5 row updated.

8. order by clause:-

Description:- Display records in ascending or descending order w.r.t. a columnname

Syntax:- select columnname(s) from tablename order by columnname [asc/desc]

Example:- select * from student order by name desc;

Output:-

NAME	REGDNO	BRANCH
smruti	1001229068	CSE
smruti	1001229068	CSE
mahesh	1001229092	CSE
bikash	1001229055	CSE
amaresh	1001229077	CSE

9. Command Name:- delete

Description:- delete column(s) from a table

Syntax:- delete from tablename where condition;

Example1:- delete from student where name='smruti';

Output:-

NAME	REGDNO	BRANCH
mahesh	1001229092	CSE
bikash	1001229055	CSE
amaresh	1001229077	CSE

Example2:- delete from student; **OR** delete * from student; //deletes entire table

10. command Name :- drop

Description:- delete a table

Syntax:- drop table tablename;

Example1:- drop table student;

11. like clause:-

Syntax:- select columnname(s) from tablename where columnname like (pattern);

Example1:- select * from student where name like 'a%'; //start with letter a

Output:-

NAME	REGDNO	BRANCH
amaresh	1001229077	CSE

Example2:- select * from student where name like '%h'; //end with letter h

Output:-

NAME	REGDNO	BRANCH
mahesh	1001229092	CSE
bikash	1001229055	CSE
amaresh	1001229077	CSE

Example3:- select * from student where name like '%ka%'; //contains pattern 'ka'

Output:-

NAME	REGDNO	BRANCH
bikash	1001229055	CSE

Example4:- select * from student where name not like 'smruti';

Output:-

NAME	REGDNO	BRANCH
mahesh	1001229092	CSE
bikash	1001229055	CSE
amaresh	1001229077	CSE

12. in clause:-

Syntax:- select columnname(s) from tablename where columnname in(value1,value2,...);

Example:- select * from student where regdno in (1001229077, 1001229092);

Output:-

NAME	REGDNO	BRANCH
amaresh	1001229077	CSE
mahesh	1001229092	CSE

Note:- like clause is used for string but, in clause is used for integer.

13. between clause:-

Syntax:- select columnname(s) from tablename where columnname between value1 AND value2

Example:- select * from student where regdno between 1001229070 AND 1001229095;

Output:-

NAME	REGDNO	BRANCH
amaresh	1001229077	CSE
mahesh	1001229092	CSE

14. Command Name :- view

Description:- It is used to create a logical table.

Syntax:- create view viewname as select columnname(s) from tablename [where condition];

Example:- create view myview as select regdno from student;

Output:- view created.

select * from myview;

REGDNO
1001229068
1001229068
1001229092
1001229055
1001229077

16. as clause:-

Description:- It is used to give a different name(alias) to a column during display.

Syntax:- select columnname as newname from tablename; **OR** select columnname from tablename as newname;

Example:- select name as "4TH Sem Regdno of CSE" from student;

Output:-

4TH Sem Regdno of CSE
1001229068
1001229068
1001229092
1001229055
1001229077

15. Command Name:- rename

Description:- It is used to rename a table/view.

Syntax:- rename oldtablename to newtablename ;

Example:- rename student to student2;

Output:-Table renamed.

Aggregate functions:-

Step-1

create table emp (eno number(10),ename varchar2(30),dept varchar2(30),salary number(10));

Output:-Table created.

Step-2

insert into emp values (01,'Ram','CSE',1000);

insert into emp values (02,'shyam','ENTC',1000);

insert into emp values (03,'Gopal','CSE',2000);

insert into emp values (04,'Madhu','ENTC',2000);

insert into emp values (05,'hari','ENTC',3000);

insert into emp values (06,'bikash','MECH',1000);

insert into emp values (07,'Ramesh','MECH',2000);

Output:-7 rows created.

Step-3

select * from emp;

Output:-

ENO	ENAME	DEPT	SALARY
01	Ram	CSE	1000
02	shyam	ENTC	1000
03	Gopal	CSE	2000
04	Madhu	ENTC	2000
05	hari	ENTC	3000
06	bikash	MECH	1000
07	Ramesh	MECH	2000

Step-4

Example1:- select min(salary) from emp; //it displays minimum salary

Output:-

MIN(SALARY)
1000

Example2:- select max(salary) from emp; //it displays maximum salary

Output:-

MAX(SALARY)
3000

Example3:- select avg(salary) from emp; //it displays average salary

Output:-

AVG(SALARY)
1714.28571

Example4:- select sum(salary) from emp; //it displays total salary

Output:-

SUM(SALARY)
12000

group by clause :- //attribute(s) having same value is grouped to a single unit

Step-5

Syntax:-

select columnname(s),aggregatefunction(columnname) from tablename [where condition] group by columnname(s)

same

Example1:- select dept,min(salary) from emp group by dept;

//it displays minimum salary of each department

Output:-

DEPT	MIN(SALARY)
CSE	1000
ENTC	1000
MECH	1000

Example2:- select dept,max (salary) from emp group by dept;

//It displays maximum salary of each department

Output:-

DEPT	MAX (SALARY)
CSE	2000
ENTC	3000
MECH	2000

Example3:- select avg(salary) from emp group by dept;

//it displays average salary of each department

Output:-

AVG (SALARY)
1500
2000
1500

Example4:- select dept,sum(salary) from emp group by dept;

//It displays total salary of each department

Output:-

DEPT	SUM(SALARY)
CSE	3000
ENTC	6000
MECH	3000

group by - having :-

Example1:- select dept,sum(salary) from emp group by dept having sum(salary)>4000;

same/equal

Output:-

DEPT	SUM(SALARY)
ENTC	6000

In group by

- We can display the column on which group by is applied. i.e. dept
- We can't display other columns. i.e. eno,ename,salary. Because the query execution starts from group by clause.after grouping according to department the resulting table has 3 rows i.e. CSE,ENTC,MECH. [if you display ename.the query is unable to find ename from the resulting table].
  But a column(if any)which fit according to group can be displayed e.g.deptno.
- We can display any aggregate function.
- The condition in having clause can contain grouping-column and aggregate function only. i.e. dept, sum()

Join operation:-

Description:- used to join 2 tables

Step-1:- create table student (name char (30), regdno number(10));

Output:- Table created.

Step2:- insert into student values ('Ram',1);

insert into student values ('shyam',2);

insert into student values ('Gopal',3);

Output:- 3 rows created.

Step-3:- select * from student;

Output:-

NAME	REGDNO
Ram	1
shyam	2
Gopal	3

Step-4:- create table smark(regdno number(10), mark number(10));

Output:- Table created.

Step-5:- insert into smark values (1,90);

insert into smark values (2,80);

insert into smark values (5,70);

Step-6:- select * from mark;

Output:-

REGDNO	MARK
1	90
2	80
5	70

natural join:-

Syntax:- select columnname(s) from tablename1 join tablename2 on tablename1.columnname=tablename2.columname;

Step-7:- select * from student join smark on student.regdno =smark.regdno;

Output:-

Name	Regdno	Regdno	Mark
Ram	1	1	90
Shyam	2	2	80

left join:-

Step-8:- select * from student left join smark on student.regdno=smark.regdno;

Output:-

Name	Regdno	Regdno	Mark
Ram	1	1	90
Shyam	2	2	80
Gopal	3		

right join:-

Step-9:- select * from student right join smark on student.regdno=smark.regdno;

Output:-

Name	Regdno	Regdno	Mark
Ram	1	1	90
Shyam	2	2	80
		5	70

full join:-

Step-10:- select * from student full join smark on student.regdno =smark.regdno;

Output:-

Name	Regdno	Regdno	Mark
Ram	1	1	90
Shyam	2	2	80
Gopal	3		
		5	70

Union, Intersect ,Minus operation:-

Create two tables R1 having regdno & branch and R2 having Regdno& branch and use SQL commands UNION,INTERESECT and MINUS among these two tables

Step1:- create table R1 (regdno number (10), branch varchar2 (10));
Output:-Table created.

Step2:- insert into R1 values (1,'CSE');
insert into R1 values (2,'CSE');
insert into R1 values (3,'CSE');
insert into R1 values (4,'CSE');
insert into R1 values (5,'CSE');

Output:- 5 rows created.

Step3:- create table R2 (regdno number (10), branch varchar2 (10));
Output:-Table created.

Step4:- insert into R2 values (1,'CSE');
insert into R2 values (2,'CSE');
insert into R2 values (3,'CSE');

Output:-3 rows created.

Step-5:- select * from R1;

Output:-

REGDNO	BRANCH
1	CSE
2	CSE
3	CSE
4	CSE
5	CSE

Step-6:- select * from R2;

Output:-

REGDNO	BRANCH
1	CSE
2	CSE
3	CSE

Syntax:- select columnname(s) from tablename1 operatorname select columnname(s) from tablename2;

union

Step5:-select regdno from R1 union select regdno from R2;

Output:-

REGDNO
1
2
3
4
5

intersect

Step6:- select regdno from R1 interest select regdno from R2;

Output:-

REGDNO
1
2
3

minus

Step7:- select regdno from R1 minus select regdno from R2;

Output:-

REGDNO
4
5

Primary key and Foreign key constraints:-

Step1:- create table student2 (regdno number(10) primary key, name char (30));

create table mark (regdno number(10) references student2,mark number(3));

Output:- 2 table created.

Foreign key

Step2:- insert into students2 values (1,'Ram');

insert into students2 values (2,'Shyam');

Output:- 2 rows created.

Step3:- insert into mark values (1,90);

insert into mark values (2,80);

Output:- 2 rows created.

Step4:- select * from student2;

select * from mark;

Output: -

Master table (Student2)

REGDNO	NAME
1	Ram
2	Shyam

Child table (Mark)

REGDNO	MARK
1	90
2	80

Step5:- insert into mark values (5,90);

Output:-ERROR at line 1; // we can't insert a record into child table (mark) if it does not exist in master table (student2)

ORA-02291:Integrity constraint (CSE08.SYS-C003061) violated-parent key not found.

Example:- delete from student2 where regno =1;

Output: - ERROR at line 1; //we can't delete a record from master table(student2) as long as it exist in child table(mark)

ORA-02292: Integrity constraint

(CSE08.SYS-C003061) violated-child rewrd found.

EXAMPLE:-

Step1:- create table student4 (name varchar2(30), regdno number(10), branch varchar2 (20));

Output: - Table Created

Step2:- Create table book4 (bookid number(10), bookname varchar2 (30), author varchar2 (30),price number (20));

Output: - Table Created

Step3:- create table issue4 (regdno number (10), bookid number (10));

Output: - Table Created

Step4:- insert into student4 values ('mahesh',3,'CSE');

insert into student4 values ('Ramesh',4,'CSE');

insert into student4 values ('Paresh',5,'CSE');

Output: - 3 row created.

Step5:- insert into book4 values (1,'DE','korth',700);

insert into book4 values (2,'DAA','Cormen',600);

insert into book4 values (3,'SP','Donovan',400);

insert into book4 values (4,'DEC',' Umera',500);

insert into book4 values (5,'DM',' liu',400);

insert into book4 values (6,'OB',' davis',300);

Output: - 6 rows created.

Step6:- insert into issues4 values (3,1);

insert into issues4 values (4,5);

insert into issues4 values (5,3);

insert into issues4 values (3,4);

insert into issues4 values (4,6);

insert into issues4 values (3,5);

insert into issues4 values (3,6);

Output: - 7 rows created.

Step7:- select * from student4;

Output: -

NAME	REGDNO	BRANCH
Mahesh	3	CSE
Ramesh	4	CSE
Paresh	5	CSE

Step8:- select * from Book4;

Output: -

BOOKID	BOOKNAME	AUTHOR	PRICE
1	DE	Korth	700
2	DAA	Cormen	600
3	SP	Donovan	400
4	DEC	Umera	500
5	DM	liu	400
6	OB	davis	300

Step9:- select * from issue4;

Output:-

REGDNO	BOOKID
3	1
4	5
5	3
3	4
4	6
3	5
3	6

QUESTION:- Display the details of book whose author is korth.

Ans:- select * from book4 where author='korth';

Output:-

BOOKID	BOOKNAME	AUTHOR	PRICE
1	DE	Korth	700

QUESTION:- Display the name of book whose price is greater than 400

Ans:- select bookname from book4 where price>400;

Output:-

BOOKNAME
DE
DAA
DEC

QUESTION:- Display the details of book issued by regdno 4

Ans:- Select * from book4,issue4 where issue4.regdno=4 and book.bookid=issue4.bookid;

Output:-

BOOK ID	BOOKNAME	AUTHOR	PRICE	REGDNO	BOOK ID
5	DM	liu	400	4	5
6	OB	davis	300	4	6

QUESTION:- Display the name of book issued by Mahesh

Ans:- Select book4.bookname from student4,book4,issue4 where student4.name='Mahesh' and issue4.regdno=student4.regdno =student4.regdno and issue4.bookid =book4.bookid;

Output:-

BOOK NAME
DE
DEC
DM
OB

To copy a table (present in other user, say cse26):

create table student as select * from cse26. student;

Consider the following set of tables

Student

Rollno	Name	Subject	Marks
1	Ram	C	20
2	shyam	C	30
3	Gopal	C	25
4	Madhu	OOPS	20
5	hari	OOPS	20
6	bikash	OOPS	35

Teacher

Tid	Tname	Subject	Rollno
t1	Abhay	C	1
t2	Badri	C	2
t1	Abhay	C	3
t2	Badri	OOPS	4
t2	Badri	OOPS	5
t2	Badri	OOPS	6

Q:- Find the name of those students who have secured maximum mark in the class ?

select name from student where marks = max(marks) //**ERROR**

select name from student where marks =(select max(marks) from student)

// inner query is executed first then outer query is executed

Q:- Find the name of those students who have secured more than average marks in the class ?

select name from student where marks >(select avg(marks) from student)

Q:- Find the name of those students who have registered with teacherid =t1.

select name from student,teacher where tid=t1 and student.rollno=teacher.rollno;

Q:- Find the name of those teachers who have more than 2 registered students.

select tname from teacher group by tname having count(tname)>2

Q:- Find the name of those students who have secured the highest mark in each subject ?

select name from student where mark in (select max(mark) from student group by subject)

consider the following tables

Employee

empcode	empname	salary	deptcode
1	Ram	28000	d1
2	shyam	23000	d2
3	Gopal	27000	d3
4	Madhu	24000	d4

Department

deptcode	deptname
d1	cse
d2	ele
d3	etc
d4	mech

Q:- Find the name and empode of all employees who work for the 'CSE' Department.

Ans:- select empname,empcode from employee,department where deptname = 'cse' and employee.deptcode = department.deptcode

Q:- Find the name of all employees whose salary is above Rs. 25000/- along with their department name.

Ans:- select empname,deptname from employee,department where salary > 25000 and employee.deptcod =department.deptcode

Chapter 3

Functional Dependency (FD) and Normalization

Let x, y are two attribute(s) of R. x is functionally dependent on y iff value of x uniquely determines value of y. (i.e. for one value of x there is only one value of y)

x→y (read as x functionally dependent on y OR x determines y)

Example1:-

R	A	B	C	D
	a1	b1	c1	d1
	a1	b1	c1	d2
	a2	b2	c2	d3
	a2	b2	c2	d3

Here F: A→B
B→C

Set of FD

but A↛D i.e. A does not determine D ,because a1 determine d1 in one row and a1 determine d2 in another row.

Example2:-

book

bookid	bname	author	price
b1	database	Navathe	400
b2	database	korth	300
b3	datastructure	patel	200
b4	oops	patel	200

bookid→ bname

bname↛ author

author→ price

Note: There is no algorithm to identify functional dependency. We have to use our commonsense and judgment to identify functional dependency.

3.1 Inference Axioms / Rules

There are six inference axioms

1. Reflexivity:- $X \rightarrow X$ // An attribute(s) determines itself
2. Augmentation:- if $X \rightarrow Y$ then $XZ \rightarrow YZ$
3. Transitivity:- if $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$

} Armstrong's Axioms

4. Addivity/union:- if $X \rightarrow Y$ & $X \rightarrow Z$ then $X \rightarrow YZ$
5. Projectivity/decomposition:- if $X \rightarrow YZ$ then $X \rightarrow Y$ & $X \rightarrow Z$
6. Pseudo-Transitivity:- if $X \rightarrow Y$, $YZ \rightarrow W$ then $XZ \rightarrow W$

Q:- R(A,B,C,D,E,F)

F: $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $E \rightarrow F$, $CF \rightarrow B$. Prove that $F \models CD \rightarrow B$ (⊨ : logically implies)

Ans:- $D \rightarrow E$, $E \rightarrow F \models D \rightarrow F$ (Transitivity)

$D \rightarrow F$, $CF \rightarrow B \models CD \rightarrow B$ (Pseudo-Transitivity)

OR

$D \rightarrow E$, $E \rightarrow F \models D \rightarrow F$ (Transitivity)

$D \rightarrow F \models CD \rightarrow CF$ (Augmentation)

$CD \rightarrow CF$, $CF \rightarrow B \models CD \rightarrow B$ (Transitivity)

3.2 Closure of attribute

Closure of a attribute x is the set of all attribute that are functionally dependent on X w.r.t. F . It is denoted by X^+ (X^+ means what X can determine)

Algorithm to compute X^+ :-

1. $X^+ = X$
2. Repeat until X^+ does not change
   For each FD $Y \rightarrow Z$ in F
   if $Y \subseteq X^+$ then $X^+ = X^+ \cup Z$

Q:- R(A,B,C,D,E,F)

F: $E \rightarrow A$, $E \rightarrow D$, $A \rightarrow C$, $A \rightarrow D$, $AE \rightarrow F$, $AG \rightarrow K$. Find the closure of E or E^+.

Ans:- $E^+ = E$

$= EA$ (for $E \rightarrow A$ add A)
$= EAD$ (for $E \rightarrow D$ add D)
$= EADC$ (for $A \rightarrow C$ add C)
$= EADC$ (for $A \rightarrow D$ D already added)
$= EADCF$ (for $AE \rightarrow F$ add F)
$= EADCF$ (for $AG \rightarrow K$ don't add k $AG \not\subseteq D^+$)

Q:- R(A,B,C,D,E,F)

F: B→C , BC→AD , D→E , CF→B. Find the closure of B (i.e. B^+)

Ans:- B^+={B,C,A,D,E }

Q:- R(A,B,C,D,E)

F: AB→C , D→E , BC→AD

Prove that F ⊨ AB→D

Ans:- $(AB)^+$={ABCDE }

since D ⊆ AB^+ Hence AB→D

Q:- R(A,B,C,D,E)

F: A→B , B→C , C→D , A→E . Prove that F ⊨ AC→E

Ans:- AC^+={ACBDE }

since E ⊆ AC^+ Hence F ⊨ AC→E

Closure is used to (i) find the candidate keys of R (ii) compute F^+

(i) Candidate Keys of R:- X is a candidate key of R if X→{R }

Q:- R(A,B,C,D,E,F)

F: A→BC , B→D , C→DE , BC→F. Find the candidate keys of R.

Ans:- A^+ = {A,B,C, D,E,F }={R} ⇒A is a candidate key

B^+ ={B,D } ⇒B is not a candidate key

C^+ = {C, D,E } ⇒C is not a candidate key

BC^+ = {B,C, D, E, F } ⇒BC is not a candidate key

Q:- R(A,B,C,D,E)

F: AB→C , D→A , AE→B , CD→E, BE→D . Find all the candidate keys of R.

Ans:- AB^+ = {A,B,C } ⇒ AB is not a candidate key

D^+ = {A,D } ⇒ D is not a candidate key

AE^+ = {A, E, B, D, C }={R} ⇒ AE is a candidate key

CD^+ = {C, D, A, E, B }={R} ⇒ CD is a candidate key

BE^+ = {B, E,D, A, C }={R} ⇒ BE is a candidate key

BD^+ = {B, D, A, C, E }={R} ⇒ BD is a candidate key

(ii) Closure of F (F^+):- F^+ is the set of all FDs that can be inferred/derived from F. Using Armstrong Axioms repeatedly on F, we can compute all the FDs.

Q:- R(A,B,C,D,E)

F: $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $A \rightarrow E$. Find the closure of F (i.e. F^+).

Ans:- $A^+ = \{A,B,C,D,E\}$

$B^+ = \{B,C,D\}$

$C^+ = \{C,D\}$

$\Rightarrow F^+ = \{A \rightarrow A , A \rightarrow B , A \rightarrow C , A \rightarrow D , A \rightarrow E , B \rightarrow B , B \rightarrow C , B \rightarrow D , C \rightarrow C , C \rightarrow D\}$

Q:- R(A,B,C,D,E,F)

F: $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CF \rightarrow B$. Find the closure of F (i.e. F^+).

Ans:- $AB^+ = \{A,B,C,D,E\}$

$BC^+ = \{B,C,A,D,E\}$

$D^+ = \{D,E\}$

$CF^+ = \{C,F,B,A,D,E\}$

$\Rightarrow F^+ = \{AB \rightarrow A , AB \rightarrow B , AB \rightarrow C , AB \rightarrow D , AB \rightarrow E , BC \rightarrow B , BC \rightarrow C , BC \rightarrow A ,$
$BC \rightarrow D , BC \rightarrow E , D \rightarrow D, D \rightarrow E, CF \rightarrow C , CF \rightarrow F , CF \rightarrow B , CF \rightarrow A ,$
$CF \rightarrow D , CF \rightarrow E\}$

Cover:- F and G are two sets of FDs. F cover G if all the functional dependency of G is logically inferred(derived) from F.

Q:- Given two set of FDs

F: $\{A \rightarrow BC , B \rightarrow D\}$

G: $\{A \rightarrow D\}$

Check whether F cover G ?

Ans:- consider $A \rightarrow D$ from G.

Now, A^+ under F $= \{ABCD\}$

since $D \subseteq A^+ \Rightarrow A \rightarrow D$ can be written in F

Hence F cover G.

3.3 Equivalence set of FDs:- F and G are equivalent (F≡G) iff F cover G and G cover F i.e. $F^+ = G^+$

Q:- Given two set of FDs. Check whether both are equivalent or not.

F: {A→B , AB→C, D→AC, D→E }

G: {A→BC , D→AE }

Ans:-

check whether F cover G or not

(i) consider A→BC

A^+ under F = {ABC } since $BC \subseteq A^+ \Rightarrow$ A→BC can be written in F

(ii) consider D→AE

D^+ under F ={DACEB } since $AE \subseteq D^+ \Rightarrow$ D→AE can be written in F

since F determines every FD of G . Hence F cover G .

check whether G cover F or not

(i) consider A→B

A^+ under G = {ABC } since $B \subseteq A^+ \Rightarrow$ A→B can be written in G

(ii) consider AB→C

AB^+ under G = {ABC } since $C \subseteq AB^+ \Rightarrow$ AB→C can be written in G

iii) consider D→AC

D^+ under G = {DAEBC } since $AC \subseteq D^+ \Rightarrow$ D→AC can be written in G

(iv) consider D→E

D^+ under G ={DAEBC } since $E \subseteq D^+ \Rightarrow$ D→E can be written in G

since G determines every FD of F . Hence G cover F .

$\therefore$ F cover G and G cover F $\Rightarrow$ F≡G

[**Redundant FD:** $X \rightarrow Y$ is a redundant FD if it can be derived from remaining FDs in F.

i.e. $F - (X \rightarrow Y) \models X \rightarrow Y$

$X \rightarrow Y$ is a Non-redundant FD if it can't be derived from remaining FDs in F.

Q:- F: { $A \rightarrow B$
$D \rightarrow B$
$A \rightarrow D$
}

Ans:-

(i) A^+ under $F - (A \rightarrow B) = \{ADB\}$ $\Rightarrow A \rightarrow B$ is redundant ($\because B \subseteq A^+$)

(i) D^+ under $F - (D \rightarrow B) = \{D\}$ $\Rightarrow D \rightarrow B$ is non-redundant ($\because B \not\subseteq D^+$)

(i) A^+ under $F - (A \rightarrow D) = \{AB\}$ $\Rightarrow A \rightarrow B$ is non-redundant ($\because D \not\subseteq A^+$)

Non-redundant cover: F is a non-redundant cover if no proper subset of F can cover F.

Q:- F: { $A \rightarrow BC$
$CD \rightarrow E$
$E \rightarrow C$
$D \rightarrow AEH$
$ABH \rightarrow BD$
$DH \rightarrow BC$ }. Find the non redundant cover of F.

Ans:- (i) A^+ under $F - (A \rightarrow BC) = \{A\}$ $\Rightarrow A \rightarrow BC$ is non-redundant ($\because BC \not\subseteq A^+$)

(ii) CD^+ under $F - (CD \rightarrow E) = \{CDAEHB\}$ $\Rightarrow CD \rightarrow E$ is redundant ($\because BC \subseteq CD^+$)

(iii) E^+ under $F - (E \rightarrow C) = \{E\}$ $\Rightarrow E \rightarrow C$ is non-redundant ($\because C \not\subseteq E^+$)

(iv) D^+ under $F - (D \rightarrow AEH) = \{D\}$ $\Rightarrow D \rightarrow AEH$ is non-redundant ($\because AEH \not\subseteq D^+$)

(v) ABH^+ under $F - (ABH \rightarrow BD) = \{ABHC\}$ $\Rightarrow ABH \rightarrow BD$ is non-redundant ($\because BD \not\subseteq ABH^+$)

(vi) DH^+ under $F - (DH \rightarrow BC) = \{DHAEBC\}$ $\Rightarrow DH \rightarrow BC$ is redundant ($\because BC \subseteq DH^+$)

Hence the non-redundant cover is $\{A \rightarrow BC, E \rightarrow C, D \rightarrow AEH, ABH \rightarrow BD\}$]

3.4 Cannonical Cover

Canonical Cover is called minimal Cover (i.e. minimum set of FDs)

A Set of FD F_C is called Canonical Cover of F if Each FD in F_C is a

i) Simple FD .
ii) Left reduced FD .
iii) Non-redundant FD .

Simple FD:- $X \rightarrow Y$ is a simple FD if Y is a single attribute.

Extraneous attribute: Let $XA \rightarrow Y$ then A is a extraneous attribute if $X \rightarrow Y$ (i.e. $XA^+ = X^+$)

Left reduced FD:- $X \rightarrow Y$ is a left reduced FD if there is no extraneous attribute in X

Non-redundant FD:- $X \rightarrow Y$ is a Non-redundant FD if it can't be derived from $F - (X \rightarrow Y)$.

Trivial Functional dependency:- A FD $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$

Ex:- $CD \rightarrow C$, $CD \rightarrow D$, $CD \rightarrow CD$. Trivial FD is a redundant FD .

Exercises:

Q:- F: { $A \rightarrow C$
$AC \rightarrow D$
$E \rightarrow ADF$ }

Find the Canonical cover of F.

Ans:-

step1:- convert all the FDs to Simple FD (single attribute in RHS)

$\Rightarrow$ F: { $A \rightarrow C$, $AC \rightarrow D$, $E \rightarrow A$, $E \rightarrow D$, $E \rightarrow F$ }

step2:- left reduced FD .

//consider FDs with more than one attribute in LHS

(i) Consider $AC \rightarrow D$

$AC^+ = \{ACD\}$

$A^+ = \{ACD\} \Rightarrow$ C is extraneous ($\because AC^+ = A^+$)

$C^+ = \{C\} \quad \Rightarrow$ A is not extraneous ($\because AC^+ \neq C^+$)

$\Rightarrow$ $AC \rightarrow D$ becomes $A \rightarrow D$ ($\because$ C is extraneous)

$\Rightarrow$ F: { $A \rightarrow C$, $A \rightarrow D$, $E \rightarrow A$, $E \rightarrow D$, $E \rightarrow F$ }

Note: there are two attributes. if one attribute is already extraneous then other can't be extraneous. so in the above case we shouldn't check whether A is extraneous since C is already extraneous.

step3:- non-redundant FD

//consider every FD of F and delete the redundant FD from F .

A^+ under $F-(A\to C)=\{AD\}$ $\Rightarrow A\to C$ is non-redundant ($\because C\subseteq A^+$)

A^+ under $F-(A\to D)=\{AC\}$ $\Rightarrow A\to D$ is non-redundant ($\because D\subseteq A^+$)

E^+ under $F-(E\to A)=\{EDF\}$ $\Rightarrow E\to A$ is non-redundant ($\because A\subseteq E^+$)

E^+ under $F-(E\to D)=\{EAFCD\}$ $\Rightarrow E\to D$ is redundant ($\because D\subseteq E^+$)

E^+ under $F-(E\to F)=\{EADC\}$ $\Rightarrow E\to F$ is non-redundant ($\because F\subseteq E^+$)

F_c = Canonical cover of $F=\{A\to C, A\to D, E\to A, E\to F\}$

Group the functional dependencies that have common LHS together

$\Rightarrow F_c=\{A\to CD, E\to AF\}$

Q:- F: { A→BC
CD→E
E→C
D→AEH
ABH→BD
DH→BC } . Find the Canonical cover of F.

Ans:-

step1:- convert all the FDs to Simple FD

$\Rightarrow$ F: { A→B , A→C , CD→E , E→C , D→A , D→E , D→H , ABH→B , ABH→D , DH→B , DH→C }

ABH→B is a trivial FD in F. So it is deleted from F .

$\Rightarrow$ F: { A→B , A→C , CD→E , E→C , D→A , D→E , D→H , ABH→D , DH→B , DH→C }

step2:- left reduced FD .

//consider FDs with more than one attribute in LHS

(i) Consider CD→E

$CD^+=\{CDEAHB\}$

$C^+=\{C\}$ $\Rightarrow$D is not extraneous ($\because CD^+\neq C^+$)

$D^+=\{DAEHBC\}$ $\Rightarrow$C is extraneous ($\because CD^+=D^+$)

$\Rightarrow$CD→E becomes D→E ($\because$ C is extraneous)

$\Rightarrow$ F: {A→B ,A→C , D→E , E→C , D→A, D→H, ABH→D, DH→B, DH→C }

(ii) Consider $ABH \rightarrow D$

$ABH^{+} = \{ABHDCE\}$

$BH^{+} = \{BH\} \Rightarrow A$ is not extraneous $(\because ABH^{+} \neq BH^{+})$

$AH^{+} = \{AHBCDE\} \Rightarrow B$ is extraneous $(\because ABH^{+} = AH^{+})$

$AB^{+} = \{ABC\} \Rightarrow H$ is not extraneous $(\because ABH^{+} \neq AB^{+})$

$\Rightarrow ABH \rightarrow D$ becomes $AH \rightarrow D$ ($\because B$ is extraneous)

$\Rightarrow F: \{A \rightarrow B, A \rightarrow C, D \rightarrow E, E \rightarrow C, D \rightarrow A, D \rightarrow H, AH \rightarrow D, DH \rightarrow B, DH \rightarrow C\}$

(iii) Consider $DH \rightarrow B$

$DH^{+} = \{DHBECA\}$

$D^{+} = \{DECAHB\} \Rightarrow H$ is extraneous $(\because DH^{+} = D^{+})$

$\Rightarrow DH \rightarrow B$ becomes $D \rightarrow B$ ($\because H$ is extraneous)

$\Rightarrow F: \{A \rightarrow B, A \rightarrow C, D \rightarrow E, E \rightarrow C, D \rightarrow A, D \rightarrow H, AH \rightarrow D, D \rightarrow B, DH \rightarrow C\}$

(iv) Consider $DH \rightarrow C$

$DH^{+} = \{DHCEAB\}$

$D^{+} = \{DECAHB\} \Rightarrow H$ is extraneous $(\because DH^{+} = D^{+})$

$\Rightarrow DH \rightarrow C$ becomes $D \rightarrow C$ ($\because H$ is extraneous)

$\Rightarrow F: \{A \rightarrow B, A \rightarrow C, D \rightarrow E, E \rightarrow C, D \rightarrow A, D \rightarrow H, AH \rightarrow D, D \rightarrow B, D \rightarrow C\}$

step3:- non-redundant FD

//consider every FD of F and delete the redundant FD from F .

A^{+} under $F - (A \rightarrow B) = \{AC\} \Rightarrow A \rightarrow B$ is non-redundant $(\because B \subseteq A^{+})$

A^{+} under $F - (A \rightarrow C) = \{AB\} \Rightarrow A \rightarrow C$ is non-redundant $(\because C \subseteq A^{+})$

D^{+} under $F - (D \rightarrow E) = \{AHDBC\} \Rightarrow D \rightarrow E$ is non-redundant $(\because E \subseteq D^{+})$

E^{+} under $F - (E \rightarrow C) = \{E\} \Rightarrow E \rightarrow C$ is non-redundant $(\because C \subseteq E^{+})$

D^{+} under $F - (D \rightarrow A) = \{DECHB\} \Rightarrow D \rightarrow A$ is non-redundant $(\because A \subseteq D^{+})$

D^+ under $F-(D\rightarrow H) = \{DECAB\}$ $\Rightarrow$ $D\rightarrow H$ is non-redundant ($\because H \subseteq D^+$)

AH^+ under $F-(AH\rightarrow D) = \{AHBC\}$ $\Rightarrow$ $AH\rightarrow D$ is non-redundant ($\because D \subseteq AH^+$)

D^+ under $F-(D\rightarrow B) = \{DECAHB\}$ $\Rightarrow$ $D\rightarrow B$ is redundant ($\because B \subseteq D^+$)

D^+ under $F-(D\rightarrow C) = \{DECAH\}$ $\Rightarrow$ $D\rightarrow C$ is redundant ($\because C \subseteq D^+$)

F_c = Canonical cover of $F = \{A\rightarrow B, A\rightarrow C, D\rightarrow E, E\rightarrow C, D\rightarrow A, D\rightarrow H, AH\rightarrow D\}$

Group the functional dependencies that have common LHS together

$\Rightarrow F_c = \{A\rightarrow BC, D\rightarrow EAH, E\rightarrow C, AH\rightarrow D\}$

3.5 Decomposition

- ✓ A Relation R is decomposed/divided into ($R_1, R_2, \ldots\ldots R_n$)
- ✓ Decomposition should be dependency preserving and lossless.

Dependency preserving decomposition:- Let R is decomposed into (R1,R2 ,……, Rn) with projected FD set (F1,F2……..,Fn). This decomposition is dependency preserving if $F^+ = (F_1 \cup F_2 \cup \ldots\ldots Fn)^+$

Q:- R {A,B,C,D,E}

F : {AB→C , C→D , AB→D }

R is decomposed to R_1(A,B,C) , R_2(D,E) . Prove decomposition is dependency preserving

Ans:- $F_1 = \{AB\rightarrow C\}$

$F_2 = \{C\rightarrow D\}$

$\Rightarrow (F_1 \cup F_2) = \{AB\rightarrow C, C\rightarrow D\}$

AB^+ under $(F_1 \cup F_2) = \{A,B,C,D\}$ $\Rightarrow AB\rightarrow D$ is under $(F_1 \cup F_2)$

$F^+ = (F_1 \cup F_2)^+$

$\Rightarrow$ decomposition is dependency preserving

Q:- R {A,B,C,D,E,F,G,H,I,J }

F : {AB→C , A→DE , B→F, F→GH , D→IJ }

R is decomposed to R_1(A,B,C,D) , R_2(D,E) , R_3(B,F) , R_4(F,G,H) , R_5(D,I,J).

Check decomposition is dependency preserving or not .

Ans:- $F_1 = \{AB \rightarrow C\}$ $F_2 = \{\}$ $F_3 = \{B \rightarrow F\}$

$F_4 = \{F \rightarrow GH\}$ $F_5 = \{D \rightarrow IJ\}$

$\Rightarrow (F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5) = \{AB \rightarrow C, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$

A^+ under $(F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5) = \{A\}$

$\Rightarrow A \rightarrow DE$ is not under $(F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5)$

$\Rightarrow F^+ \neq (F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5)^+$

$\Rightarrow$ decomposition is not dependency preserving

Lossless join decomposition:

if $R = (R_1 \bowtie R_2 \ldots\ldots \bowtie R_n)$ then

the decomposition of R into $(R_1, R_2, \ldots\ldots, R_n)$ is called lossless

Example :

R	A	B	C
	a1	b1	c1
	a2	b2	c2
	a3	b2	c3

If R is decomposed to R1 & R2 as shown below

R_1	A	B
	a1	b1
	a2	b2
	a3	b2

$\bowtie$

R_2	A	C
	a1	c1
	a2	c2
	a3	c3

=

$R_1 \bowtie R_2$	A	B	C
	a1	b1	c1
	a2	b2	c2
	a3	b2	c3

Here, $R = R_1 \bowtie R_2$. hence it is a lossless join

Note : The decomposition is lossless if the common attribute is a key.

If R is decomposed to R1 & R2 as shown below

R_1	A	B
	a1	b1
	a2	b2
	a3	b2

$\bowtie$

R_2	B	C
	b1	c1
	b2	c2
	b2	c3

=

$R_1 \bowtie R_2$	A	B	C	
	a1	b1	c1	
	a2	b2	c2	extra
	a2	b2	c3	Spurious tuples
	a3	b2	c2	
	a3	b2	c3	

Here, $R \neq R_1 \bowtie R_2$ hence it is a lossy join

if spurious tuples come in join operation, then join is lossy .

> Let R is decomposed to R_1 and R_2 .
>
> if $R_1 \cap R_2 \rightarrow R_1$ OR $R_1 \cap R_2 \rightarrow R_2$ then
>
> decomposition is lossless
>
> else
>
> decomposition is lossy

Exercises

Q:- R (A,B,C,D)

F : {A→B , A→C , C→D }

R is decomposed to R_1(A,B,C) , R_2(C,D) . Check decomposition is lossless or not .

Ans:- $R_1(A,B,C) \cap R_2(C,D) = C$

$C^+ = \{CD\} = R_2 \quad \Rightarrow C \rightarrow R_2$

$\because R_1 \cap R_2 \rightarrow R_1 \quad \Rightarrow$ decomposition is lossless

Q:- R (A,B,C,D,E)

F : {AB→C , AD→E , C→D } R is decomposed to R_1(A,B,C,D) , R_2(D,E) .

Check decomposition is lossy or lossless .

Ans:- $R_1 \cap R_2 = D$

$D^+ = \{D\}$

$\because R_1 \cap R_2 \not\rightarrow R_1$ OR R_2 (i.e. $R_1 \cap R_2$ neither determine R_1 nor R_2)

$\Rightarrow$ decomposition is lossy

3.6 Algorithm to check lossy or lossless decomposition

Step1: create a table with M rows and N columns.

M = number of decomposed relation

N = number of attribute of original relation

Step2: if a decomposed relation R_i has attribute A then

insert a symbol(say 'a') at position (R_i, A)

Step3: consider each FD x→y.

if column x has two or more symbols then

insert symbols in the same place(rows) of column y.

Step4: if any row is completely filled with symbols then

Decomposition is lossless.

Else

Decomposition is lossy.

Q :- consider R(A B C D E)

F : { A→ B , BC → E , ED → A }

R is decomposed into R1(AB) and R2 (ACDE).

Check the decomposition is lossy or lossless.

Ans :

Step1:

	A	B	C	D	E
R1					
R2					

Step2:

	A	B	C	D	E
R1	a	a			
R2	a		a	a	a

Step3: for A→ B , insert symbol a

	A	B	C	D	E
R1	a	a			
R2	a	a	a	a	a

R2 is completely filled ⇒decomposition is lossless

Q :- consider R(A B C D E)

F : { AB→ C , C → E , B → D , E → A }

R is decomposed into R1(BCD) , R2 (ACE).

Check the decomposition is lossy or lossless.

Ans :

Step1:

	A	B	C	D	E
R1		a	a	a	
R2	a		a		a

Step2: for AB→ C , no symbol is inserted

for C → E , insert symbol a

	A	B	C	D	E
R1		a	a	a	a
R2	a		a		a

for B → D , no symbol is inserted

for E → A , insert symbol a

	A	B	C	D	E
R1	a	a	a	a	a
R2	a		a		a

R1 is completely filled ⇒decomposition is lossless

3.7 Normalization

Normalization is the process of organizing data to minimize

(i) redundancy/duplication/repetition

(ii) insertion,deletion,updation anomalies

There are six Normal Forms

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

3.8 First Normal Form (1NF)

A relation is in 1NF when there is no multivalue attribute.

OR

A relation is in 1NF when all its attributes are simple.

Consider a relation student (rollno, name, branch, address, Phoneno)

rollno	name	branch	address	Phoneno
1	A	cse	cuttack	9437270668,8906790236,9938172361
2	B	etc	bhubaneswar	9861240673,7208271923

The above relation is not in 1NF because the multivalue attribute phoneno is present.

Now we represent the above table by creating a new row for each phoneno as shown below.

rollno	name	branch	address	phoneno
1	A	cse	cuttack	9437270668
1	A	cse	cuttack	8906790236
1	A	cse	cuttack	9938172361
2	B	etc	bhubaneswar	9861240673
2	B	etc	bhubaneswar	7208271923

The above table contains redundant data due to phoneno. Because for each phoneno we have to repeat all the other information of student. so phoneno attribute should be separated from above table.

We divide/decompose the above table R into two tables :

R1(key, multivalue attribute) , R2 (R–multivalue attribute)

$\Rightarrow$ R1(rollno, phoneno) , R2(rollno, name, branch, address)

R1

rollno	phoneno
1	9437270668
1	8906790236
1	9938172361
2	9861240673
2	7208271923

R2

rollno	name	branch	address
1	A	cse	cuttack
2	B	etc	bhubaneswar

R1, R2 are in 1NF

key of R1 = rollno

key of R2 = (rollno, phoneno)

3.9 Partial dependency

if non-key attribute(s) depend on a part of candidate key then it is partial dependency.

[This is applicable where primary key has multiple attributes]

if non-key attribute(s) depend on the entire candidate key then it is **full dependency**.

Example:- Let R(A,B,C) and AB = candidate key .

A→C is a Partial dependency (c depend on a part of candidate key)

AB→C is a Full dependency (c depend on the entire candidate key)

A FD X→Y is a partial dependency if

(i) X is a part of candidate key and

(ii) Y is nonkey attribute(s)

Ex:- R(ABCDE)

F : {AB→C , C→D , B→E }. Find Partial dependency .

Ans:- AB^+=ABCDE C^+=CD B^+=BE

⇒AB is the only candidate key .

⇒key attributes =A,B and nonkey attributes =C,D,E

key attributes are also called prime attributes

nonkey attributes are also called nonprime attributes

AB→C is not a PD // it is full dependency

C→D is not a PD // it is full dependency

B→E is a PD (∵B is a part of candidate key and E is nonkey attribute)

Note: X→Y is full dependency when X =key or Y =key attribute(s)

3.10 Second Normal Form (2NF)

A relation is in 2NF when it is in 1NF and there is no partial dependency.

OR

A relation is in 2NF when it is in 1NF and all non-key attributes fully depend on primary key(candidate key).

Consider a relation R (rollno, name, subjectcode, subjectname, duration)

rollno	name	subjectcode	subjectname	duration
1	Alok	S1	C	50 days
1	Alok	S2	C++	40 days
2	Sanjay	S1	C	50 days
2	Sanjay	S2	C++	40 days
3	Rohit	S3	Database	45 days

F: { rollno → name
subjectcode → (subjectname,duration) }

$(\underline{\text{rollno, subjectcode}})^+$ = {rollno, name, subjectcode, subjectname, duration}

⇒ (rollno, subjectcode) is the candidate key.

The above table is in first normal form because no multivalue attribute is present. But the table is not in 2NF because following two partial dependencies are present.

rollno→ name is a partial dependency [∵ name depend on a part of key]

subjectcode →{subjectname,duration} [∵ {subjectname,duration} depend on a part of key]

The table R suffers from all three anomalies

(a)Insertion anomaly:- We cannot insert a new course such as "java" to the table unless we have a student who has to take the subject.

(b)Updation anomaly:- If we change the subject from "c++" to "oops" we have to make changes in more than one place otherwise the table will be inconsistent.

(c)Deletion anomaly:- If Rohit is deleted from the table we also loose information that we had "database" subject.

Decomposition for 2NF:- To overcome these anomalies table R should be divided to smaller tables.

if partial dependency is $X \rightarrow Y$ then divide R into $R1(X^+)$ and $R2(R - Y^+)$

rollno→ name is a partial dependency [∵ name depend on a part of key]

so divide the table R into

$R1 = \text{rollno}^+ = (\text{rollno, name})$

$R2 = R - \text{name}^+ = (\text{rollno, subjectcode, subjectname, duration})$

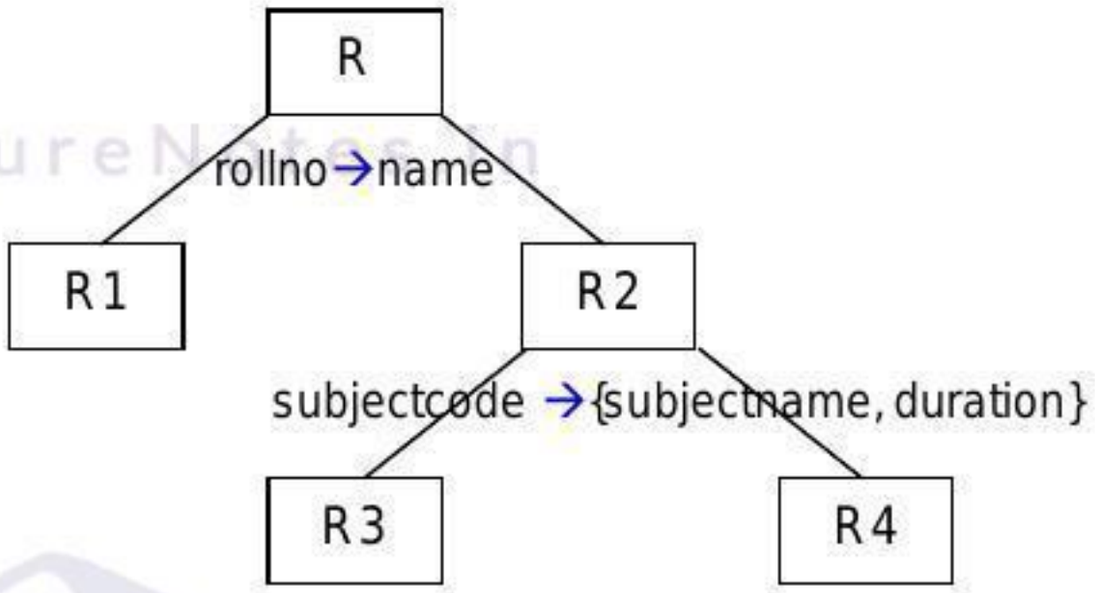
subjectcode →{subjectname,duration} is a partial dependency [∵ {subjectname,duration} depend on a part of key]

so divide R2 table into

$R3 = subjectcode^{+} = (subjectcode, subjectname, duration)$

$R4 = R2 - \{subjectname, duration\}^{+} = (rollno, subjectcode)$

[There should be a table which contain candidate key and any attribute(s) which fully depend on candidate key. if not so then make it.]



R1

Rollno	name
1	Alok
2	Sanjay
3	Rohit

R3

subjectcode	subjectname	duration
S1	C	50 days
S2	C++	40 days
S3	Database	45 days

R4

Rollno	subjectcode
1	S1
1	S2
2	S1
2	S2
3	S3

The above three tables are free from all anomalies. Let us clarify this in more detail.

No Insertion anomaly:- Now a new course "java" can be inserted to the subject table without any student information.

No Updation anomaly:- To change any subject only one change is needed in subject table.

No Deletion anomaly:- if we delete Rohit's record from both R1 and R4 table then it does not have any side effect. Because "database" is untouched in the subject table.

Note: if key is a single attribute then relation is always in 2NF .

Q:- R(ABCDE) F : {A→C , B→DE , D→C }

check which FD violate 2NF. Decompose R into 2NF.

Ans:- $A^{+} = AC$ ⇒A is not candidate key

$B^{+} = BDEC$ ⇒B is not candidate key

$D^{+} = DC$ ⇒D is not candidate key

$AB^+ = ACBDE \Rightarrow$ AB is candidate key

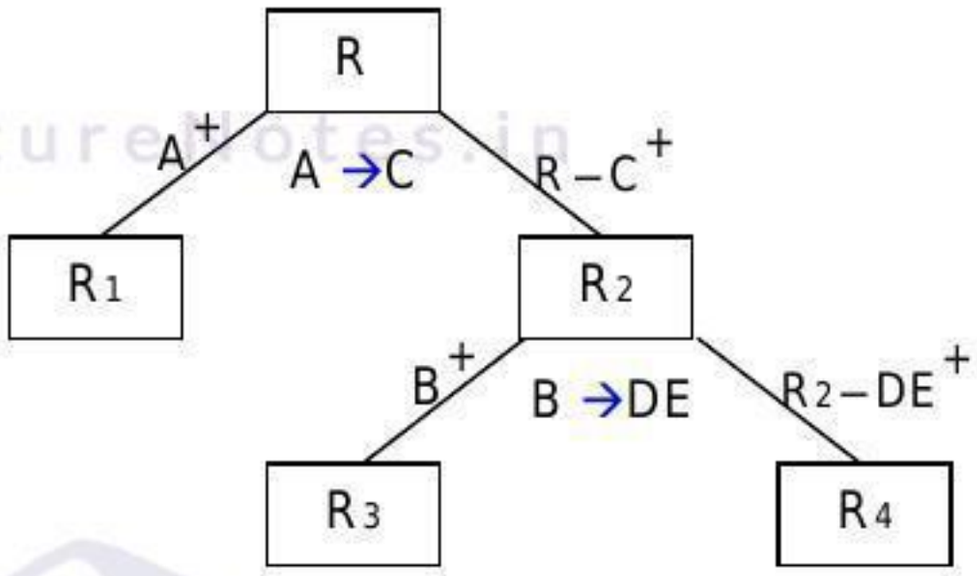
$\Rightarrow$ key attributes = A,B and nonkey attributes = C,D,E

$A \rightarrow C$ violate 2NF ($\because A \rightarrow C$ is PD)

$B \rightarrow DE$ violate 2NF ($\because B \rightarrow DE$ is PD)

$D \rightarrow C$ is in 2NF ($\because D \rightarrow C$ is not PD)

2NF decomposition is shown below



We consider FDs which violate 2NF

For $A \rightarrow C$ R(ABCDE) decomposed to R_1(AC) & R_2(ABDE)

For $B \rightarrow DE$ R_2(ABDE) decomposed to R_3(BDEC) & R_4(AB)

$\Rightarrow$ 2NF decomposition of Relation R is R_1(AC), R_3(BDEC), R_4(AB).

Q:- R (eid, ename, pid, pname, hours)

F : { eid→ename

pid→pname

eid,pid→hours }

decomposed R into 2NF .

Ans:- $eid^+ = \{eid, ename\}$

$pid^+ = \{pid, pname\}$

$(eid,pid)^+ = \{eid, pid, ename, pname, hours\}$ $\Rightarrow$ (eid,pid) is candidate key

$\Rightarrow$ key attributes = {eid , pid } and nonkey attributes = {ename, pname, hours }

2NF decomposition:- eid→ename violate 2NF ($\because$ it is PD)

pid→pname violate 2NF ($\because$ it is PD)

eid,pid→hours is in 2NF ($\because$ it is not PD)

We consider FDs which violate 2NF

For eid→ename R decomposed to R_1(eid,ename) & R_2(eid,pid,pname,hours)

For pid→pname R_2 decomposed to R_3(pid,pname) & R_4(eid,pid,hours)

⇒2NF decomposition of R is R_1(eid,ename) , R_3(pid,pname) , R_4(eid,pid,hours) .

Q:- R(ABCDEF) F : {A→BCDEF , BC→ADEF , B→F , D→E }

Decompose R into 2NF.

Ans:- A^+ = ABCDEF ⇒A is candidate key

BC^+ = BCADEF ⇒BC is candidate key

B^+ = BF ⇒B is not candidate key

D^+ = DE ⇒D is not candidate key

⇒key attributes = A, B, C and nonkey attributes = D, E, F

A→BCDEF is in 2NF (∵A→BCDEF is not PD)

BC→ADEF is in 2NF (∵BC→ADEF is not PD)

B→F violate 2NF (∵B→F is PD)

D→E is in 2NF (∵D→E is not PD)

2NF decomposition:- We consider FDs which violate 2NF

For B→F R decomposed to R_1(BF) & R_2(ABCDE)

⇒2NF decomposition of R is R_1(BF) , R_2(ABCDE) .

3.11 Transitive Dependency(TD)

A FD X→Y is a transitive dependency if

X is not candidate key/superkey and

Y is nonkey attribute(s)

Example1:- Let R(A,B,C)

key attributes = A & non-key attributes = B,C

B→C is a Transitive dependency

Example2:- Let R(A,B,C,D)

key attributes = A,B

non-key attributes = C,D

C→D is a Transitive dependency

AB→D is not a Transitive dependency

C→DB is not a Transitive dependency

Ex:- R(ABCDEF)

F : {AB→C , C→A , B→DE , ABD→F }. Find transitive dependency .

Ans:- AB^+=ABCDE ⇒AB is candidate key .

C^+=CA

B^+=BDE

ABD^+=ABDFCE ⇒ABD is not candidate key (∵AB is candidate key)

⇒key attributes={A,B} and nonkey attributes ={C,D,E }

AB→C is not a TD

C→A is not a TD

B→DE is a TD (∵B not a candidate key/superkey and DE nonkey attributes)

ABD→F is not a TD (∵ABD is a superkey)

3.12 Third Normal Form (3NF):

A relation is in 3NF when it is in 2NF and there is no transitive dependency.

OR

A relation is in 3NF when it is in 2NF and all non-key attributes directly depend on candidate key.

Consider a relation STUDENT (rollno, game, fee)

rollno	game	fee
1	cricket	100
2	cricket	100
3	cricket	100
4	football	200
5	football	200
6	football	200
7	badminton	300

F: { rollno → game
rollno → fee
game → fee }

$rollno^+$ = { rollno, game, fee }

⇒rollno is primary key

The above STUDENT Table is in first normal form because no multivalue attribute is present.

Also STUDENT is in second normal form because all non-primary attributes are fully functionally dependent on the primary key (rollno).

But the table is not in 3NF because there is Transitive dependency game → fee . fee has transitive/indirect dependency on rollno via game.

STUDENT table suffers from all 3 anomalies;

Insertion anomaly:- A new game can't be inserted into the table unless we get a student to play that game.

Deletion anomaly:- If rollno 7 is deleted from the table we also loose the information that we had a badminton game.

Updation anomaly:- to change the fee for cricket, we must make changes in more than one place.

Decomposition for 3NF:- To overcome these anomalies STUDENT table should be divided to smaller tables.

> if $X \rightarrow Y$ is a transitive dependency then divide R into $R1(X^+)$ & $R2(R-Y^+)$

game → fee is a transitive dependency [∵ neither game is a key nor fee is a key attribute]

$R1 = game^+ = (game, fee)$

$R2 = (STUDENT - fee^+) = (rollno, game)$

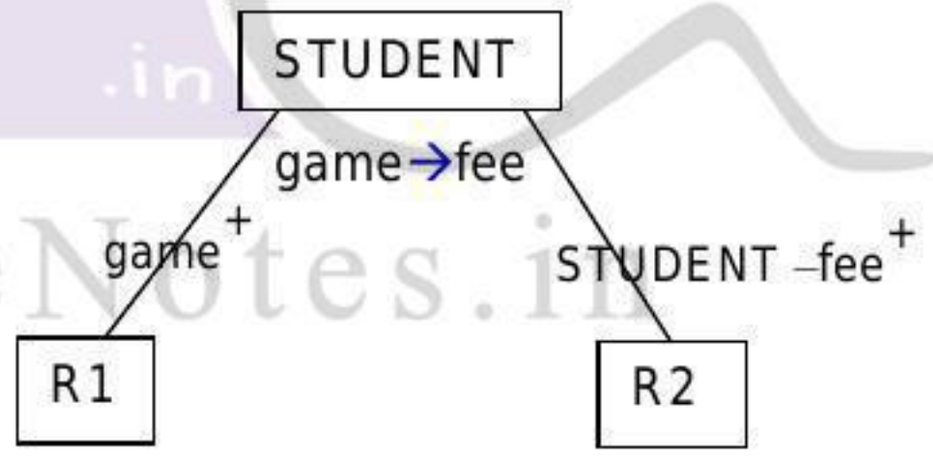
so divide STUDENT table into

(i) R1(game, fee)

(ii) R2(rollno, game)

R1

rollno	game
1	cricket
2	cricket
3	cricket
4	football
5	football
6	football
7	badminton



R2

game	fee
cricket	100
football	200
badminton	300

The above two tables are free from all anomalies.

Q:- R(ABCDE) F : {A→C , B→DE , D→C }

check which FD violate 3NF. Decompose R into 3NF.

Ans:- $A^+ = AC$ ⇒A is not candidate key

$B^+ = BDEC$ ⇒B is not candidate key

$D^+ = DC$ ⇒D is not candidate key

$AB^+ = ACBDE$ ⇒AB is candidate key

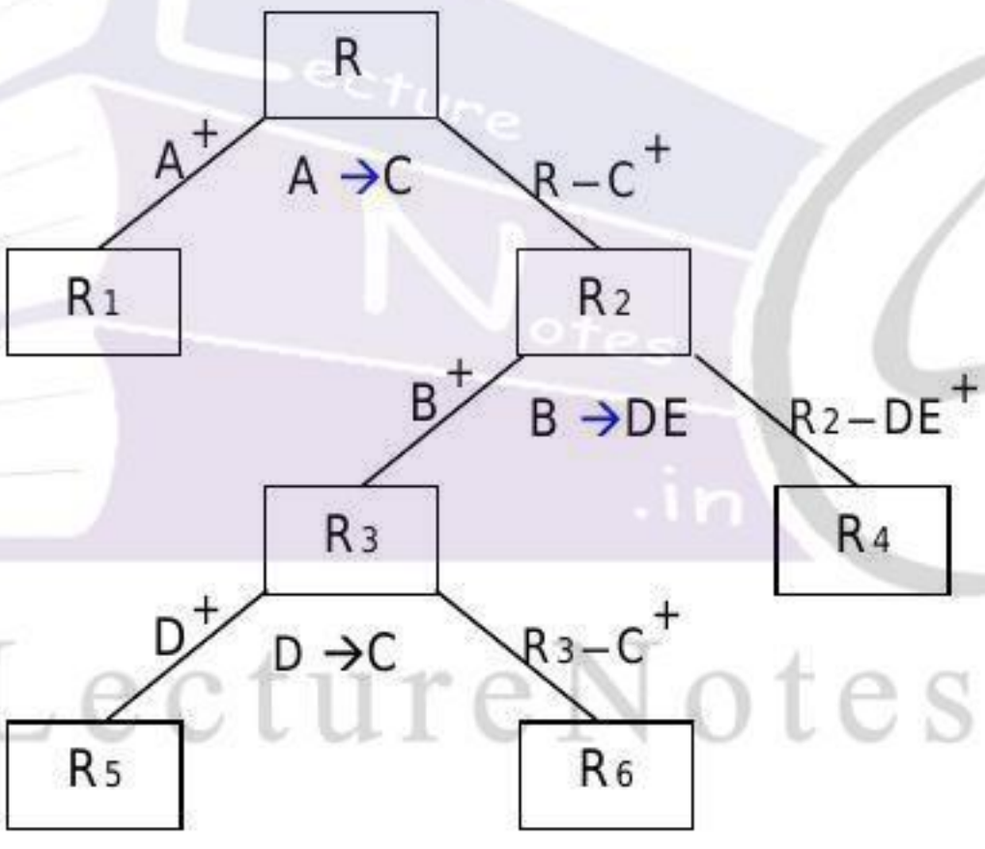
⇒ key attributes =A,B and nonkey attributes =C,D,E

A→C violate 3NF (∵A→C is TD)

B→DE violate 3NF (∵B→DE is TD)

D→C violate 3NF (∵D→C is TD)

3NF decomposition is shown below



We consider FDs which violate 3NF

For A→C R(ABCDE) decomposed to R_1(AC) & R_2(ABDE)

For B→DE R_2(ABDE) decomposed to R_3(BDEC) & R_4(AB)

For D→C R_3(BDEC) decomposed to R_5(DC) & R_6(BDE)

⇒3NF decomposition of Relation R is R_1(AC) , R_4(AB) , R_5(DC) , R_6(BDE).

Q:- R(ABCDEF) F : {A→BCDEF , BC→ADEF , B→F , D→E }

Find the highest normal form of R & decomposed R into 3NF .

Ans:- A^+ ={ABCDEF } ⇒A is candidate key

BC^+ ={BCADEF } ⇒BC is candidate key

B^+ = {BF } ⇒B is not candidate key

D^+ = {DE } ⇒D is not candidate key

⇒ key attributes ={A, B, C } and nonkey attributes ={D, E, F }

R is in 1NF (∵ there is a key & no multivalue attribute)

R is not in 2NF (∵ B→F is a PD)

⇒ highest normal form of R is 1NF.

3NF decomposition:- A→BCDEF is in 3NF (∵ it is not TD)

BC→ADEF is in 3NF (∵ it is not TD)

B→F violate 3NF (∵ it is TD)

D→E violate 3NF (∵ it is TD)

We consider FDs which violate 3NF

For B→F R decomposed to R_1(BF) & R_2(ABCDE)

For D→E R_2(ABCDE) decomposed to R_3(DE) & R_4(ABCD)

⇒3NF decomposition of Relation R is R_1(BF) , R_3(DE) , R_4(ABCD) .

3.13 Boyce Codd Normal Form (BCNF)

A relation is in BCNF when every determinant is a key (super key) .

Consider a relation R with attributes (Student, Subject, Teacher).

Student	Teacher	Subject
Avas	K. Das	database
Avas	G. Sahoo	C++
Manas	K. Das	database
Manas	R. Mishra	C++

F: { (Student, Teacher) → Subject
(Student, Subject) → Teacher
Teacher → Subject }

candidate keys are (Student, Teacher) & (Student, Subject)

The above relation is in 3NF. (∵ there is no TD)

A relation R is in BCNF if for every non-trivial FD X→Y , X must be a key .

The above relation is not in BCNF. Because In the FD (Teacher →Subject) , Teacher is not a key.

[A relation is not in BCNF when it contains two(or more) composite candidate keys, having at least one common attribute. There are two composite candidate keys (Student, Teacher) & (Student, Subject). Student is a common attribute in both the candidate keys. So, this relation must be normalized according to BCNF]

This relation suffers again from the anomalies as discussed below:
For example, if we delete the student Manas, we will loose the information that
R. Mishra teaches C++.
These difficulties are caused by the fact that teacher is determinant but not a candidate key.

Decomposition for BCNF

Teacher → Subject violates BCNF [∵ teacher is not a candidate key]

if X→Y violates BCNF then divide R into R1(X,Y) and R2(R–Y)

so R is divided into two relation R1(Teacher, Subject) and R2(Student, Teacher).

R1

Teacher	Subject
K. Das	database
G. Sahoo	C++
R. Mishra	C++

R2

Student	Teacher
Avas	K. Das
Avas	G. Sahoo
Manas	K. Das
Manas	R. Mishra

All the anomalies which were present in R, now removed in these two relations.

Q:- R(ABCDEF) F : {A→BC , BC→ADEF , B→F , D→E }
decomposed R into BCNF .

Ans:- A^+ ={ABCDEF } ⇒A is candidate key
BC^+ ={BCADEF } ⇒BC is candidate key
B^+ ={BF }
D^+ ={DE }

BCNF decomposition:- A→BCDEF is in BCNF (∵ A is a key)
BC→ADEF is in 3NF (∵ BC is a key)
B→F violate BCNF (∵ B is not a key)
D→E violate BCNF (∵ D is not a key)

We consider FDs which violate BCNF

For B→F R decomposed to R_1(BF) & R_2(ABCDE)
For D→E R_2(ABCDE) decomposed to R_3(DE) & R_4(ABCD)
⇒BCNF decomposition of R is R_1(BF), R_3(DE), R_4(ABCD) .

Q:- R(ABC) F : {AB→C , C→B } .

Find highest normal form of R . decompose R into BCNF .

Ans:- $AB^+ = \{ABC\}$ ⇒AB is candidate key

R is in 2NF (∵there is no PD)

R is not in 3NF (∵C→B is a TD)

⇒highest normal form of R is 2NF

<u>BCNF decomposition</u>:-

C→B violate BCNF

⇒R(ABC) decomposed to R_1(BC) & R_2(AC)

⇒F_1 : {C→B } & F_2 : {}

⇒$F \neq F_1 \cup F_2$

⇒decomposition is not dependency preserving

Note:- BCNF decomposition not always satisfy dependency preserving property. After BCNF decomposition if dependency is not preserved then we have to decide whether we want to remain in BCNF or rollback to 3NF. This process of rollback is called denormalization.

Q:- A relation R has following dependencies

F: { rollno → name, game
subjectcode → subjectname
game → fee }

Decompose the above relation into 3NF.

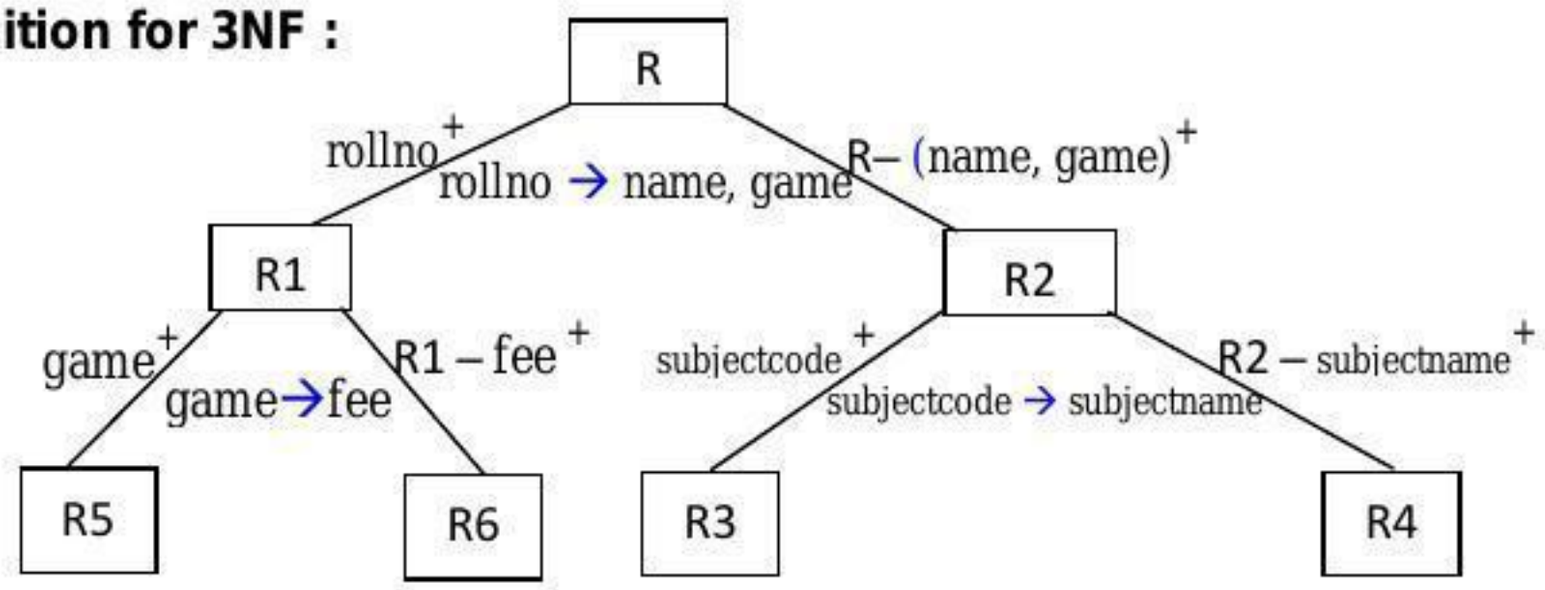
Ans:-

$rollno^+ = \{rollno, name, game, fee\}$

$subjectcode^+ = \{subjectcode, subjectname\}$

⇒ (rollno, subjectcode) is primary key.

Decomposition for 3NF :



R3 = {subjectcode, subjectname }
R4 = {rollno, subjectcode }
R5 = {game,fee }
R6 = {rollno, name, game }

Q:- Does 3NF decomposition always satisfy dependency preserving and lossless join property? Is there any algorithm for the same?

Ans:- It will not satisfy in all cases.

Following algorithm can be used for 3NF decomposition which always satisfy dependency preserving and lossless(nonadditive) join property

Algorithm:-

step1: Find minimal cover(F_C) for the given set of FD.

step2: For each FD $X \rightarrow Y$ in F_M create a relation ($X \cup Y$)

step3: If none of the relation contain key, then create a relation which contain key.

Q:- R(ABCDE) F : {A→C , B→DE , D→C }

Decompose R into 3NF.

Ans:- candidate key = AB

step1: minimal cover(F_C) = {A→C , B→DE , D→C }

step2: R_1(AC) , R_2(BDE) , R_3(DC)

step3: R_4(AB)

⇒ 3NF decomposition of Relation R is R_1(AC) , R_2(BDE) , R_3(DC) , R_4(AB)

Q:- F: {rollno → game
rollno → fee
game → fee }

Decompose R into 3NF.

Ans:- candidate key = rollno

step1: minimal cover(FC) = { rollno → game
game → fee }

step2: R1(rollno, game) , R2(game , fee)

Exercises

Question 1 Suppose you are given a relation R = (A, B, C, D, E) with the functional dependencies: {CE →D, D →B, C →A}.

a. Find all candidate keys.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

c. If the relation is not in BCNF, decompose it until it becomes BCNF. At each step, identify a new relation, decompose and re-compute the keys and the normal forms they satisfy.

Answer

a. The only key is {CE}

b. The relation is in 1NF

c. Decompose into R1=(A,C) and R2=(B,C,D,E). R1 is in BCNF, R2 is in 2NF.
Decompose R2 into R21=(C,D,E) and R22=(B,D). Both relations are in BCNF.

Question 2 Suppose you are given a relation R=(A,B,C,D,E) with the functional dependencies: {BC →ADE, D →B}.

a. Find all candidate keys.

b. Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).

Answer

a. The keys are {BC}and {CD}

b. The relation is in 3NF

Question 3 You are given the following set of functional dependencies for a relation R(A,B,C,D,E,F), F = { AB →C , DC →AE , E →F }

a. What are the keys of this relation?

b. Is this relation in BCNF? If not, explain why by showing one violation.

c. Is the decomposition (A,B,C,D) (B,C,D,E,F) a dependency preserving decomposition? If not, explain briefly.

Answer

a. {ABD}and {BCD}.

b. No, all functional dependencies are actually violating this. No dependency contains a superkey on its left side.

c. Yes, AB →C and DC →A are preserved in the first relation. DC →E and E →F are preserved in the second relation.

Question 4 You are given the below functional dependencies for relation R(A,B,C,D,E), F = {AB →C, AB →D, D →A, BC →D, BC →E}.

a. Is this relation is in BCNF? If not, show all dependencies that violate it.

b. Is this relation in 3NF? If not, show all dependencies that violate it.

c. Is the following dependency implied by the above set of dependencies? If so, show ABC →AE using the Armstrong's Axioms

Answer

Keys for the relation: {AB}, {BD}, {BC}.

a. Not in BCNF since D →A does have a superkey on the left hand side.

b. The relation is in 3NF . None of the FD violate 3NF .

c. BC →E (given)

ABC →AE by the augmentation rule.

Question 5 You are given the following set F of functional dependencies for a relation R(A,B,C,D,E,F) F = {ABC →D, ABD →E, CD →F, CDF →B, BF →D}

a. Find all keys of R based on these functional dependencies.

b. Is this relation in Boyce-Codd Normal Form? Is it 3NF? Explain your answers.

Answer

a. Keys: ABC & ACD

b. It is not in BCNF. Consider ABD →E and ABD is not a superkey.

It is not in 3NF. Consider ABD →E, and ABD is not a superkey and E is not prime attribute (part of a key).

Question 6 Given relation R(W, X, Y, Z) and set of functional dependencies G = {Z →W, Y →XZ, XW →Y}, where G is a minimal cover:

a) Decompose R into a set of relations in Third Normal Form.

b) Is your decomposition in (a) also in Boyce Codd Normal Form? Explain your answer.

Answer

a. Possible keys: {Y}, {XZ}, {WX}

R1=(Z, W), R2 = (Y, X, Z), R3 =(X, W, Y)

b. Yes. In each of the three relations, the left side of the funcational dependencies that apply are superkeys for the relation. Hence, all three relations satisfy the definition of BCNF.

Question 7 consider the following functional dependencies

Date_of_Birth →Age

Name →Roll_Number

Roll_Number →Name

The relation (Roll_Number, Name, Date_of_Birth, Age) is

(A) in 2NF

(B) in 3NF

(C) in BCNF

(D) in none of the above

Answer: (D)

Explanation: To check that a relation is in which normal form we should apply test from lower level. First apply test for 2NF

Candidate keys are: {Name , Date_of_birth} & {Roll_number , Date_of_birth}

Now check for partial dependencies

Date_of_Birth →Age – partial dependency

Name →Roll_Number – partial dependency

Roll_Number →Name – partial dependency

There exist partial dependency in relation ⇒relation is not in 2NF

⇒relation will not be in neither 3NF nor BCNF.

Question 8 The relation Student (name, courseno, rollno, grade) has the following functional dependencies:

name, courseno→grade

rollno, courseno →grade

name →rollno

rollno →name

The highest normal form of this relation is

(A) 2NF (B) 3NF (C) BCNF (D) 4NF

Answer: (B)

Explanation: candidate keys of relation are: {name, courseno} and {rollno, courseno}

First check for 2NF

name, courseno→grade – not partial dependency
rollno, courseno →grade – not partial dependency
name →rollno – not partial dependency
rollno →name – not partial dependency

in this relation no partial dependency exist so relation is in 2NF

Now check for **3NF**: for every X→Y either X is candidate key or Y is prime attribute.

name, courseno→grade – not violating 3NF (candidate key at left side)

rolino, courseno →grade – not violating 3NF (candidate key at left side)

name →rollno – not violating 3NF (prime attribute at right side)

rollno →name – not violating 3NF (prime attribute at right side)

No dependency is violating 3NF condition ,so relation is in 3NF

Now check for **BCNF**: for every X→Y , X should be super key.

name, courseno→grade – not violating BCNF(super key at left side)

rollno, courseno →grade – not violating BCNF(super key at left side)

name →rollno – violating BCNF

rollno →name – violating BCNF

Relation is not in BCNF.

⇒Highest normal form of relation is 3NF.

Question 9 Relation R is decomposed into BCNF. The redundancy (arising out of functional dependencies) in the resulting set of relations is

(A) Zero

(B) More than zero but less than that of an equivalent 3NF decomposition

(C) Proportional to the size of F^+

(D) Indeterminate

Answer: (A)

Explanation: if a relation is in BCNF then there is no redundancy left in relation , but if a relation is in 3NF then there will be redundancy with no anomalies.

Question 10 Which one of the following statements about normal forms is FALSE?

(A) BCNF is stricter than 3NF

(B) Lossless, dependency-preserving decomposition into 3NF is always possible

(C) Lossless, dependency-preserving decomposition into BCNF is always possible

(D) Any relation with two attributes is in BCNF

Answer: (C)

3.14 MultiValue Dependency(MVD)

X→Y relates one value of X to one value of Y

X→→Y (read as X multidetermines Y) relates one value of X to many values of Y.

A **Nontrivial MVD** occurs when X→→Y and X→→Z where Y and Z are not dependent. are independent to each other. Nontrivial MVD produce redundancy.

Trivial Multivalued Dependency:- A MVD X→→Y is trivial if {Y ⊆ X OR X ∪ Y = R }
Else it is a nontrivial MVD

Ex:- R

regdno	phoneno	qualification
1	P1	diploma
1	P1	B.Tech
1	P1	M.Tech
1	P2	diploma
1	P2	B.Tech
1	P2	M.Tech

Here, regdno→→phoneno
regdno→→qualification
} nontrivial MVD

Phoneno and qualification are independent to each other (i.e. no FD bet^{n} them)

A relation having non-trivial MVDs must have at least three attributes; two of them multivalued.

3.15 Fourth Normal Form (4NF)

A relation R is in 4NF if it is in BCNF and there is no non-trivial multivalue dependency.

In the above example R is in BCNF (∵ there is no FD exist)

But R is not in 4NF (∵ there is nontrivial MVD)

The above relation suffers from following anomalies

Insertion anomaly:- if we want to insert a new phoneno for regdno3 then we have to insert 3 rows, because for each phoneno we have store all three combination of qualification

deletion anomaly:- if we want to delete the qualification diploma , then we have to delete in more than one places.

updation anomaly:- if we want to update the qualification diploma to ITI , then we have to update in more than one places.

4NF decomposition

if R(XYZP) has X→→Y and X→→Z then R is decomposed to R_1(XY) and R_2(XZP)

⇒R(regdno,phoneno,qualification) is decomposed to R_1(regdno, phoneno) and R_2(regdno, qualification)

R_1

regdno	phoneno
1	P1
1	P2

R_2

regdno	qualification
1	diploma
1	B.Tech
1	M.Tech

All the anomalies discussed above is removed.

The above two relation are in 4NF.

Now, regdno→→phoneno is trivial MVD (∵ {regdno}U {phoneno}=R_1)

⇒ R_1 is in 4NF

regdno→→qualification is trivial MVD (∵ {regdno}U {qualification}=R_2)

⇒ R_2 is in 4NF

3.16 Fifth Normal Form (5NF)

It is also called Project Join Normal Form.

A relation R is in 5NF if it is in 4NF and there is no join dependency.

no join dependency means lossless-join decomposition.

A decomposition is a lossless-join decomposition if it preserves all the data in original relation and does not result additional/spurious tuples.

A relation R satisfies join dependency iff R is equal to the join of R_1, R_2, ..., R_n where R_i are subsets of the set of attributes of R.

R

dept	subject	sname
comp.Sc.	C	Alok
comp.Sc.	C	Amar
comp.Sc.	JAVA	Amar
IT	C	Sanjay

Here, dept→→subject

dept→→sname

subject and sname are dependent

The above relation is in 4NF. Anomalies can occur in relations in 4NF if the primary key has three or more fields. Here primary key is (dept, subject, sname).

sometimes decomposition of a relation into two smaller relations does not remove redundancy. In such cases it may be possible to decompose the relation in three or more relations using 5NF.

The above relation says that dept. offers many elective subjects which are taken by a variety of students. Students have the option to choose subjects. Therefore all three fields are needed to represent the information.

The above relation does not show non-trivial MVDs since the attributes subject and sname are dependent; they are related to each other (A FD subject→sname exists).

The relation cannot be decomposed in two relations (dept, subject) and (dept, sname) (∵spurious tuple generated when we join these two relations).

Therefore the relation can be decomposed into following three relations

R1(dept, subject), R2(dept, sname) and R3(subject, sname) and it can be shown that this decomposition is lossless. (∵no spurious tuple when we join these three relations)

R1

dept	subject
comp.Sc.	C
comp.Sc.	JAVA
IT	C

R2

dept	sname
comp.Sc.	Alok
comp.Sc.	Amar
IT	Sanjay

R3

subject	sname
C	Alok
C	Amar
JAVA	Amar
C	Sanjay

In above decomposition of table reduction of redundancy is not apparent, but it can be realized when table contain large amount of data.

R = (R1 ⋈ R2) ⋈ R3 is shown below // lossless decomposition

R1 ⋈ R2

dept	subject	sname
comp.Sc.	C	Alok
comp.Sc.	C	Amar
comp.Sc.	JAVA	Alok
comp.Sc.	JAVA	Amar
IT	C	Asit

⋈

R3

subject	sname
C	Alok
C	Amar
JAVA	Amar
C	Asit

=

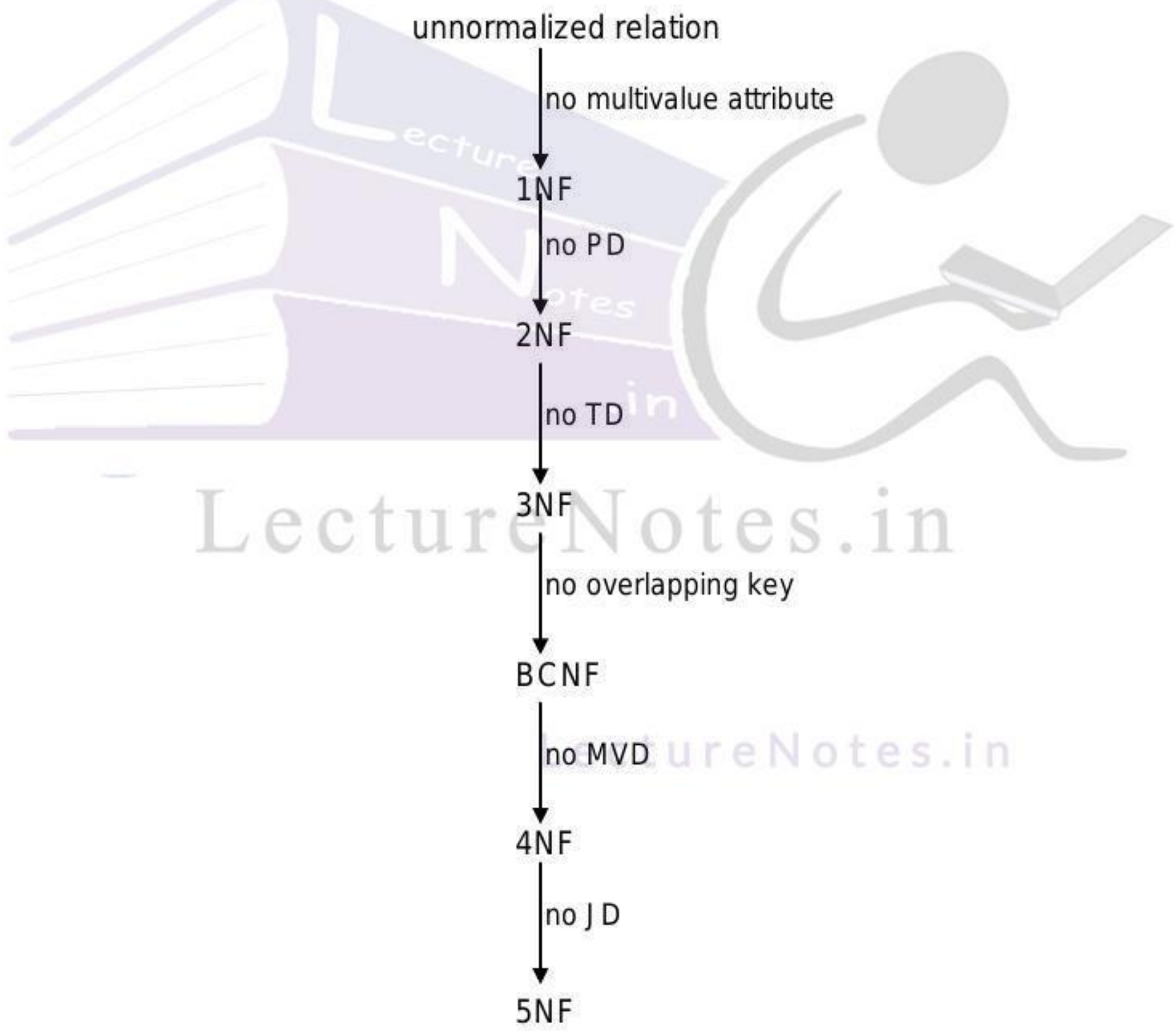
R

dept	subject	sname
comp.Sc.	C	Alok
comp.Sc.	C	Amar
comp.Sc.	JAVA	Amar
IT	C	Asit

R1and R2 are joined by common attribute 'dept' .(R1⋈R2) has spurious tuple

(R1⋈R2) and R3 are joined by common attribute (subject,sname)

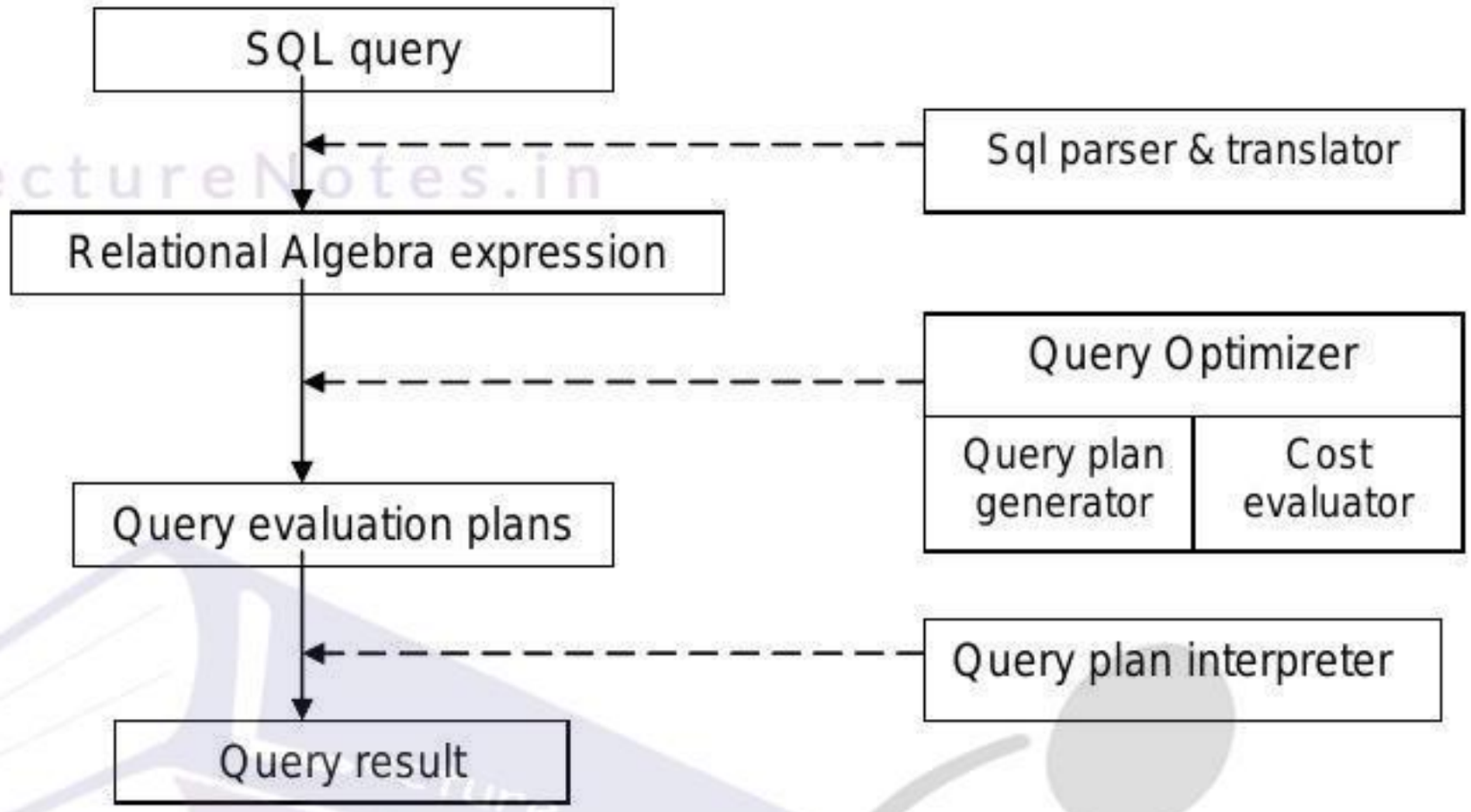
Final table R has no spurious tuple.



As normalization increases the level of redundancy decreases.

3.17 Query processing and optimization

Architecture



Parser checks syntax of sql & verify relation.

Translator translates sql to Relational Algebra expression (i.e. set of operations).

Query Optimization:- For any given query, there may be a number of different ways to execute it. The process of choosing a suitable one for processing a query is known as query optimization.

Two forms query optimization : Heuristic Optimization or Cost Based Optimization

- In **Heuristic Optimization**, the query execution is refined based on heuristic rules for reordering the individual operations.
- In **Cost Based Optimization**, the overall cost of executing the query is systematically reduced by estimating the costs of executing several different execution plans.

Query plan generator generate many evaluation plans (i.e. ordering of operations)

Ex:- select name from Customer, Account where Customer.name=Account.name and Account.balance>2000;

there are two evaluation plans

(i) $\prod_{customer.name}(\sigma_{customer.name=account.name \wedge account.balance>2000}(Customer \times Account)$

(ii) $\prod_{customer.name}(\sigma_{customer.name=account.name}(Customer \times \sigma_{account.balance>2000}(Account))$

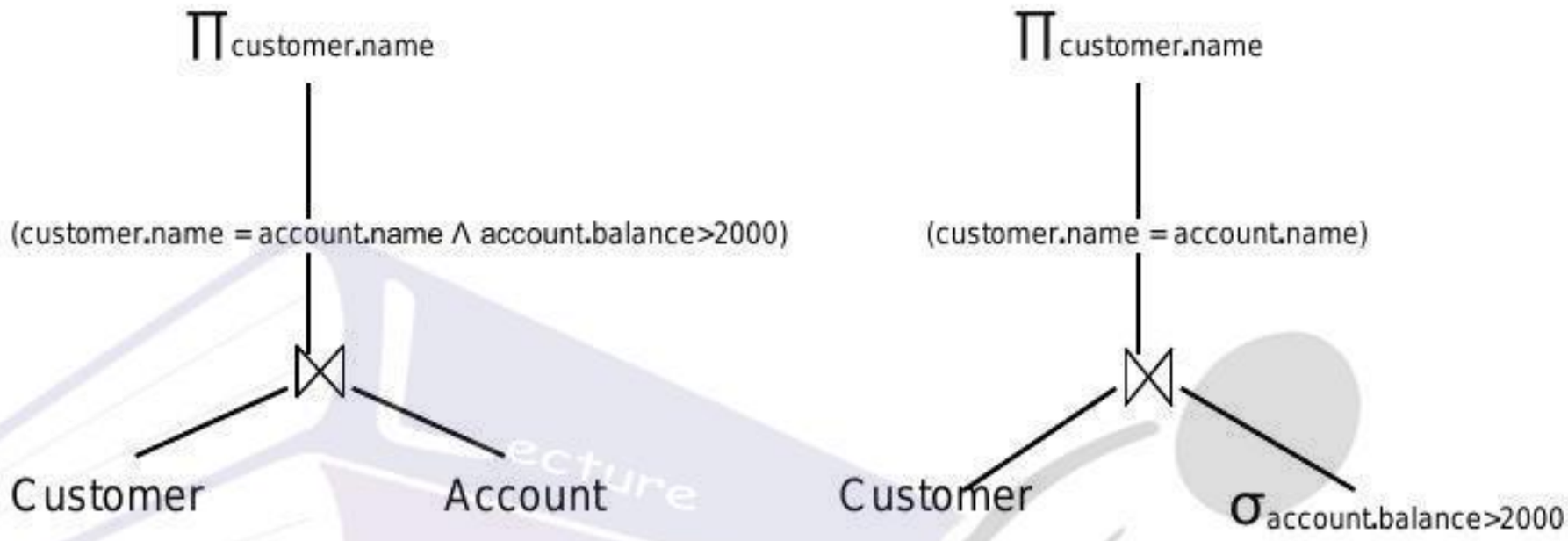
Cost evaluator evaluates the cost of different evaluation plans and choose the evaluation plan with lowest cost.

Disk access time, cpu time, no of operation, no of tuples in relation, size of tuple are considered for cost calculation.

Expression Tree

It is also called query tree. It demonstrates an evaluation plan.

Expression trees for the above evaluation plans are shown below



3.17.1 Translating Query to Relational Algebra

SQL queries are decomposed into query blocks. One query block contains a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clauses (if any). nested queries are split into separate query blocks.

Eg: select lname, fname from employee where salary > (select max(salary) from employee where deptname=CSE);

Split the above query into two blocks.

1. c = (select max (salary) from employee where deptname=CSE); // inner block
2. select lname, fname from employee where salary > c; // outer block

where c represents the result returned from the inner block.

The relational algebra for inner block is $\mathcal{G}_{max(salary)}(\sigma_{dname=CSE}(employee))$

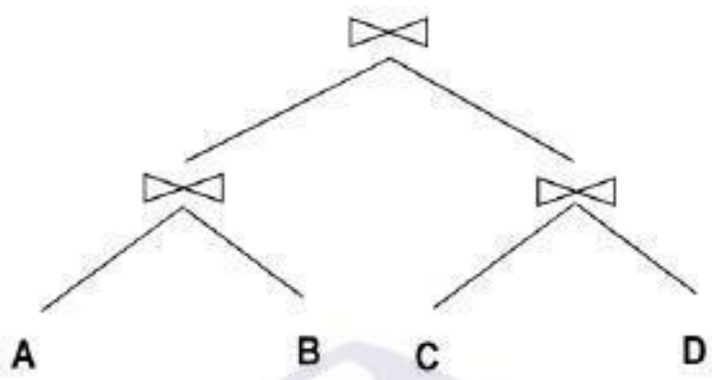
and for outer block is $\prod_{lname,\ fname}(\sigma_{salary>c}(employee))$

The query optimizer would then choose an execution/evaluation plan for each block.

3.17.2 Evaluation of Relational Algebra Expressions

A. Materialized evaluation

- evaluate one operation at a time.
- evaluate the expression in bottom-up manner.
- store intermediate results to temporary files.



Store the result of A ⋈ B in a temporary file.

Store the result of C ⋈ D in a temporary file.

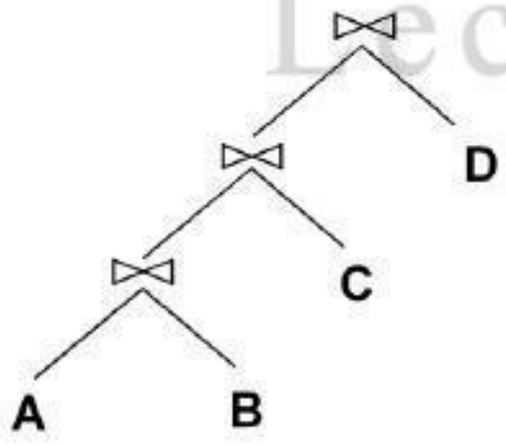
Finally, join the results stored in temporary files.

Overall cost = Sum of costs of individual operations + cost of writing intermediate results to disk

Cost of writing results to results to temporary files and reading them back is quite high.

B. Pipelined evaluation

- evaluate several operation simultaneously. Result of one operation is passed to the next operation.
- evaluate the expression in bottom-up manner.
- don't store intermediate results to temporary files.



Don't store the result of A ⋈ B in a temporary file. Instead, the result is passed directly for projection with C and so on.

Join Ordering : first compute the join that yields smaller intermediate result.

example**:** we know $(R_1 \bowtie R_2) \bowtie R_3 = R_1 \bowtie (R_2 \bowtie R_3)$

If $R_2 \bowtie R_3$ is quite large then $R_1 \bowtie R_2$, we choose $(R_1 \bowtie R_2) \bowtie R_3$ because we have to compute and store a smaller temporary relation.

3.18 Heuristic Optimization

- ✓ Cost-based optimization is expensive. Heuristics are used to reduce the number of choices that must be made in a cost-based approach.
- ✓ Heuristic optimization transforms the expression-tree by using a set of rules (but not in all cases) which improve the performance:
  - Perform selection early (reduces the number of tuples)
  - Perform projection early (reduces the number of attributes)
  - Perform most restrictive selection and join operations before other similar operations.
- ✓ Some systems use only heuristics; others combine heuristics with partial cost-based optimization.

Steps in Heuristic Optimization

1. Deconstruct conjunctive selections into a sequence of single selection operations.
2. Move selection operations down the query tree for the earliest possible execution.
3. First Execute those selection and join operations that will produce smallest relations.
4. Replace Cartesian product operation followed by selection operation with join operation.
5. Deconstruct and move the tree as far as down possible (listing of projection attributes, creating new projections where needed)
6. Identify those subtrees whose operations can be pipelined.

Chapter 4

Transaction Processing

A Transaction is a unit of database processing which contains a set of operations.

Example: deposit of money, balance enquiry, reservation of ticket etc.

Transaction has two basic operations

(i) read(x) :- load the data-item x from database to memory [database is in the hard-disk]

(ii) write(x) :- update the data-item x in memory and store in database

4.1 Properties of Transaction

Database system ensures **ACID** property

1. **A**tomicity:- Either all or none of the transaction operation is done.
2. **C**onsistency:- A transaction transfer from one consistent(correct) state to another consistent state.
3. **I**solation:- A transaction is isolated from other transaction. i.e. A transaction is not affected by other transaction. Although multiple transaction execute concurrently it must appear as if the transaction are running serially (one after other)
4. **D**urability:- The result of a transaction are permanent i.e. The result will never be lost with subsequent failure. durability refers to long lasting/all time lasting i.e. permanency

Example:-Transaction to transfer Rs50 from Account A to Account B

1. Read A
2. A=A–50
3. Write(A)
4. Read(B)
5. B=B+50
6. Write(B)

Atomicity requirement (why atomicity is required):- if the transaction fails at step4 then there is inconsistency due to partial modification (since value of B is not updated)

Consistency requirement:- the sum of A and B should be same before and after transaction.

Isolation requirement:- if another transaction access data during step4 then there is inconsistency since value of B is not updated till now.

4.2 States of transaction

A transaction is broken into states to handle various situation such as failure. Following are the states of a transaction

Active : Transaction is executing.

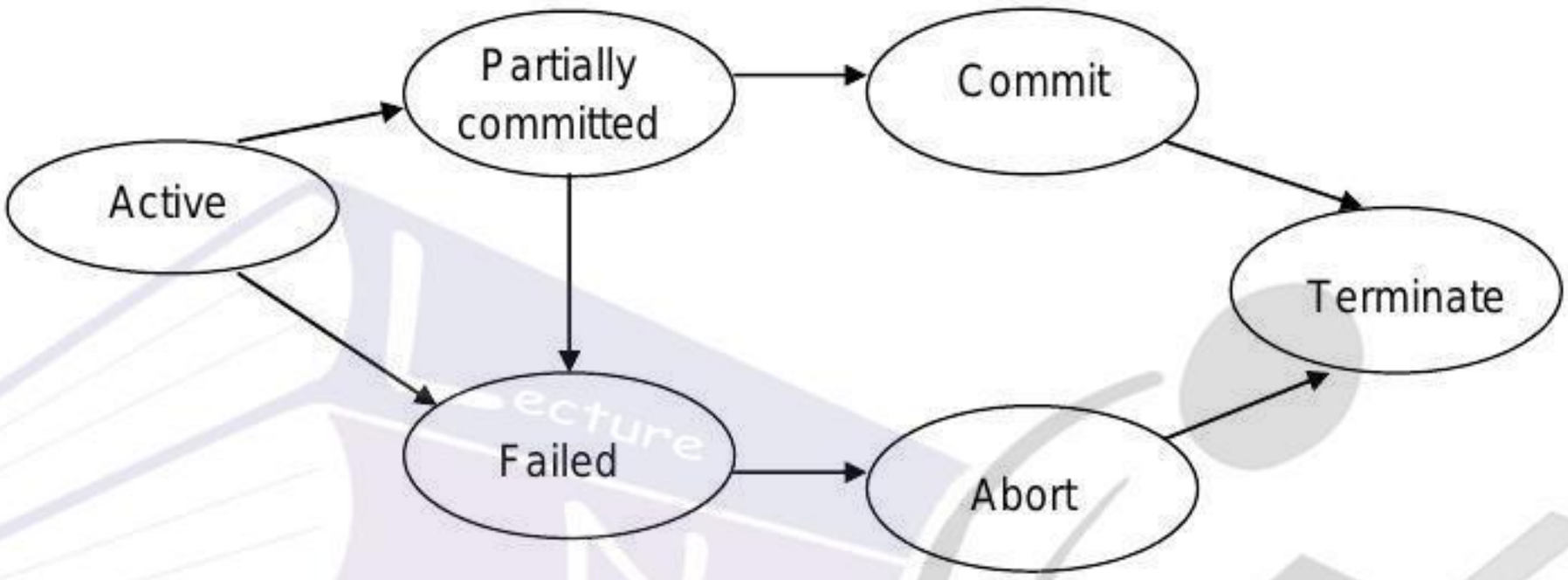
Failed : Transaction fails to complete successfully.

Abort : Changes made by transaction are cancelled (roll back).

Partially commit : Final statement of transaction is executed.

Commit: Transaction completes its execution successfully

Terminated: Transaction is finished.



4.3 Concurrent transaction

Multiple transactions are executed concurrently(simultaneously) in the system.

Advantages:

1. Increases throughput

   [throughput = no of transaction completed per unit time]

2. Reduces waiting time

   Ex:- T1=90sec T2=500sec T3=5sec

   If we execute serially by T1→T2→T3 then Transaction T3 wait for 590 sec. so we go for nonserial/concurrent transaction to reduce waiting time. (i.e. T3→T1→T2)

Disadvantages:

Execution of concurrent transaction may result inconsistency.

4.5.1 Conflict Serializability :- Conflict serializable schedule orders any conflicting operations in same way as some serial execution.

A pair of operation is said to conflict if they operate on same data item and one of them is a write operation. That means

1. $read_i(X)$ $read_j(X)$ – non conflict rr
2. $read_i(X)$ $write_j(X)$ – conflict rw
3. $write_i(X)$ $read_j(X)$ – conflict wr
4. $write_i(X)$ $write_j(X)$ – conflict ww

where i and j denotes two different transaction T_i and T_j

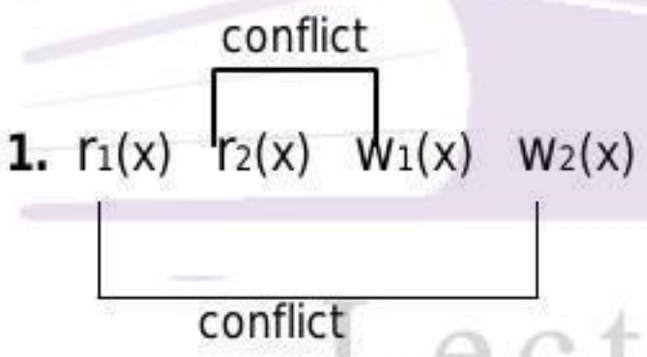
Precedence graph:- It is used to check conflict serializability

Steps:-

1. For each transaction T_x put a node/vertex in the graph.
2. For each conflicting pair, put a edge from T_i to T_j
3. if there is a cycle in graph then schedule is not conflict serializable else schedule is conflict serializable

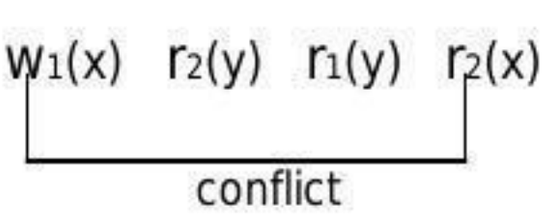
Example:

1. $r_1(x)$ $r_2(x)$ $w_1(x)$ $w_2(x)$



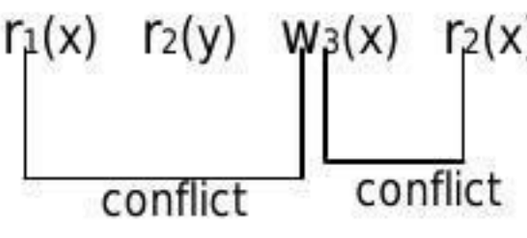
Cycle is present so it is not conflict serializable

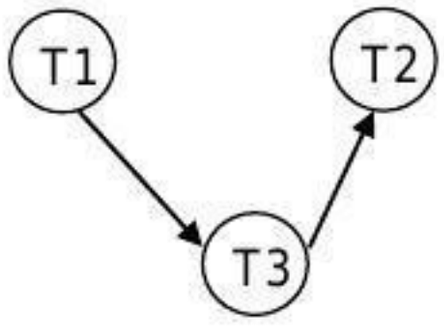
2. $w_1(x)$ $r_2(y)$ $r_1(y)$ $r_2(x)$



Cycle is not present so it is conflict serializable

3. $r_1(x)$ $r_2(y)$ $w_3(x)$ $r_2(x)$ $r_1(y)$





Cycle is not present so it is conflict serializable

4.5.2 View Serializability: A schedule is view-serializable if it is view equivalent to a serial schedule.

A schedule is view serializable if following 3 rules are satisfied:

Rule1: if T_i reads data initially,after this T_j writes the same data, in the given schedule. This sequence must be followed in the transaction combination. [**rw**]

Rule2: if T_i writes data initially ,after this T_j reads the same data, in the given schedule. This sequence must be followed in the transaction combination. [**wr**]

Rule3: if T_i writes data, after this T_j writes the data finally. This sequence must be followed in the transaction combination. [**ww**]

Example1: R1(X) W2(X) W1(X)

create all possible combinations of the Transactions. we have 2 transactions, so the combinations are : <T1, T2> and <T2, T1>

Rule1: T1 reads initially, after this T2 writes the same data.that means transaction sequence must be "T1 followed by T2". So remove following combinations where "T1 is not followed by T2" (i.e.T2 occur before T1)

<T2, T1>

Rule2: T2 writes initially, after this no transaction reads the same data. so we keep all the transaction combinations, that means Rule2 does not remove any combinations.

Rule3: T1 writes data finally, that means T1 must occur at last, So remove following combinations where "T1 does not occur at last".

<T1, T2>

Hence, no combinations left to satisfy the view serializability.

Conclusion: Given schedule is not view-serializable.

Example2: W1(X) R2(Y) R1(Y) R2(X)

create all possible combinations of the Transactions. we have 2 transactions, so the combinations are <T1, T2>

<T2, T1>

Rule1: T2 reads initially, after this no transaction writes the same data. so we keep all the transaction combinations, that means Rule2 does not remove any combinations.

Rule2: T1 writes initially ,after this T2 reads the same data X, that means transaction sequence must be "T1 followed-by T2". So remove following combinations where "T1 is not followed by T2" (i.e.T2 occur before T1)

<T2, T1>

Rule3: T1 writes data finally, but no other write operation is present. so we keep all the transaction combinations, that means Rule3 does not remove any combinations.
Hence, one combinations left to satisfy the view serializability is <T1, T2 >
Conclusion: Given schedule is view-serializable.

Consider two schedules S1 and S2 which contain transactions T1 and T2
S1 and S2 are view equivalent if following Rules are satisfied

1. If in S1, T1 reads the initial value of X(data item), then in S2 also, T1 should read the initial value of X.
2. If in S1, T1 writes in X which is read by T2, then in S2 also, T1 should write in X before T2 reads it.
3. If in S1, T1 performs the final write operation on that data item, then in S2 also, T1 should perform the final write operation on that data item.

Ex:- S_1 : $r_1(x)\ w_2(y)\ r_1(y)$

S_2 : $r_2(x)\ w_1(x)\ r_2(y)$.

check S1, S2 view-equivalent or not .

S_1 = T1	T2	S_2 = T1	T2
$r_1(x)$			$r_2(x)$
	$w_2(y)$	$w_1(x)$	
$r_1(y)$			$r_2(y)$

S_1, S_2 are not view-equivalent since x is initially read by T_1 in S_1 but x is initially read by T_2 in S_2 (i.e. Rule1 is not satisfied)

Blind write:- if a transaction T_i performs a write operation without performing a read operation then it is called blind write.

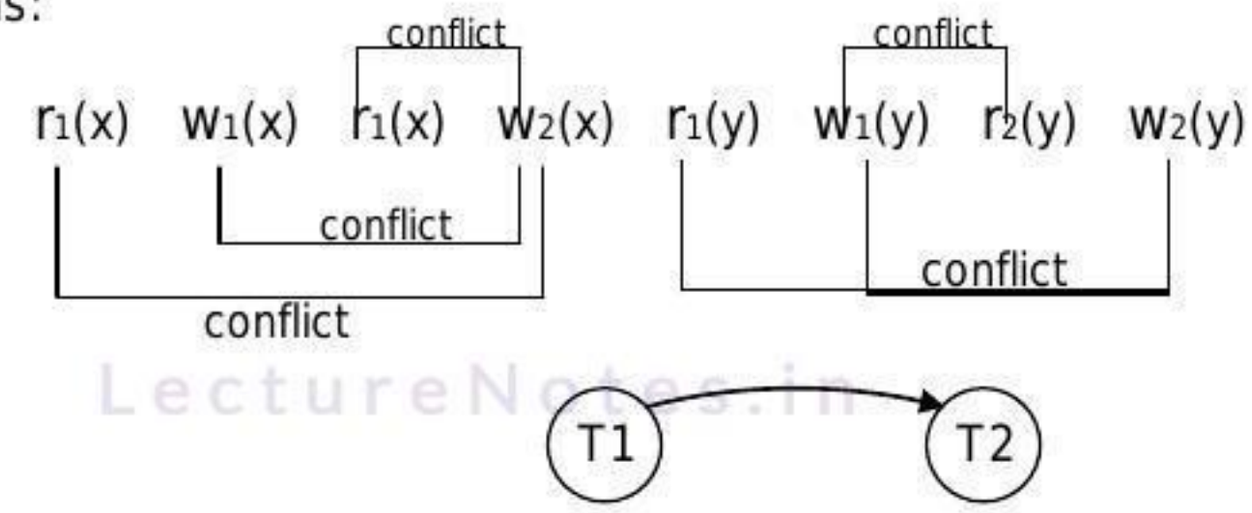
T1	T2
$r_1(x)$	
	$w_2(x)$
$w_1(x)$	

If a schedule is view-serializable but not conflict-serializable, then the schedule must have a blind write.

Exercises

Q:- check the schedule given below is conflict serializable or not.

Ans:



Cycle is not present so it is conflict serializable

Q: R1(X) R2(X) W1(X) W2(X).

Check whether the given schedule is view-serializable ?

Ans : create all possible combinations of the Transactions. we have 2 transactions, so the combinations are

<T1, T2 >

<T2, T1 >

Rule1: T1 reads initially, after this T1 writes the same data. That means transaction sequence must be "T1 followed by T1",which is always false. So, Rule1 remove all transaction combinations. Hence the schedule is not view serializable

Rule2: we will not check Rule2, because we've concluded that schedule is not view serializable.

Rule3: we will not check Rule3, because we've concluded that schedule is not view serializable,

Conclusion: Given schedule is not view-serializable.

Q: R1(X) R2(Y) W3(X) R2(X) R1(Y)

Check whether the given schedule is view-serializable ?

Ans : create all possible combinations of the Transactions. We have 3 transactions, so the combinations are

<T1, T2, T3 >
<T1, T3, T2 >
<T2, T1, T3 >
<T2, T3, T1 >
<T3, T1, T2 >
<T3, T2, T1 >

Rule1: T1 reads initially, after this T3 writes the same data, that means transaction sequence must be "T1 followed by T3". So remove following combinations where "T1 is not followed by T3" (i.e.T3 occur before T1)

<T2, T3, T1 >
<T3, T1, T2 >
<T3, T2, T1 >

Rule2: T3 writes initially, after this T2 reads the same data X, that means transaction sequence must be "T3 followed by T2". So remove following combinations where "T3 is not followed by T2" (i.e.T2 occur before T3)

<T1, T2, T3 >

<T2, T1, T3 >

Rule3: T3 writes data finally, but no other write operation is present, that means we will keep all transaction combination. so Rule3 does not remove any transaction combination.

Hence, one combinations left to satisfy the view serializability is:

<T1, T3, T2 >

Conclusion: Given schedule is view-serializable.

Q: R1(X), R2(Y), R2(Y), W2(X), W3(Y), R1(X)

Check whether the given schedule is view-serializable ?

Ans : create all possible combinations of the Transactions. we have 3 transactions, so the combinations are

<T1, T2, T3 >
<T1, T3, T2 >
<T2, T1, T3 >
<T2, T3, T1 >
<T3, T1, T2 >
<T3, T2, T1 >

Rule1: T1 reads initially, after this T2 writes the same data. that means transaction sequence must be "T1 followed by T2". So remove following combinations where "T1 is not followed by T2" (i.e.T2 occur before T1)

<T2, T1, T3 >
<T2, T3, T1 >
<T3, T2, T1 >

Rule2: T2 writes data, after this T1 reads the same data X, that means transaction sequence must be "T2 followed by T1". Now this rule violates rule1. Because of this, none of the rules can be applied. so it is not view serializable.

Rule3: we will not check Rule3, because we've concluded that schedule is not view serializable,

Conclusion: Given schedule is not view-serializable.

Q: W3(Z), R2(X), W2(Y), R1(Z), W3(Y), W1(Y)

Check whether the given schedule is view-serializable ?

Ans: we have 3 transactions, so the transaction combinations are

<T1, T2, T3 >
<T1, T3, T2 >
<T2, T1, T3 >
<T2, T3, T1 >
<T3, T1, T2 >
<T3, T2, T1 >

Rule1: T2 reads initially, after this no transaction writes the same data. so we keep all the transaction combinations.

Rule2: T3 writes initially, after this T1 reads the same data Z, that means transaction sequence must be "T3 followed by T1". So remove following combinations where "T3 is not followed by T1" (i.e.T1 occur before T3)

<T1, T2, T3 >
<T1, T3, T2 >
<T2, T1, T3 >

Rule3: T1 writes data finally, that means T1 must occur at last, So remove following combination where "T1 does not occur at last".

<T3, T1, T2 >

Hence, two combinations left to satisfy the view serializability are:

<T2, T3, T1 >
<T3, T2, T1 >

Conclusion: Given schedule is view-serializable.

4.6 CONCURRENCY CONTROL

When many transactions are executed simultaneously then it is called as concurrent transactions. Concurrency is required to increase time efficiency. If many transactions try to access the same data, then inconsistency arises. Concurrency control is required to maintain consistency of data.

In order to run transactions concurrently, we interleave their operations.

Each transaction gets a share of the computing time.

This leads to following problems

a. Lost update problem (ww conflict)

b. temporary update / dirty read problem (wr conflict)

c. unrepeatable read / incorrect analysis problem (rw conflict)

All arise because isolation is broken.

a. Lost Update problem

Two transactions T1 and T2 read , modify ,write to the same data item in an interleaved fashion for which an incorrect value is stored in x. T2 reads the value of X before T1 changes it hence the updated value resulting from T1 is lost.

		X	
T1	T2	T1	T2
Read(X)		10	
X = X - 5		5	
	Read(X)		10
	X = X + 5		15
Write(X)		5	
	Write(X)		15

The final value of x is 15 , which is incorrect.

b. Temporary update problem/dirty read problem

T2 read the updated value of X made by T1, but T1 fails and roll back.

So, T2 reads an incorrect value of X.

T1	T2
Read(X)	
X = X - 5	
Write(X)	
	Read(X)
	X = X + 5
	Write(X)
ROLLBACK	
	COMMIT

c. unrepeatable read / incorrect analysis problem

T1	T2
Read(X)	
X = X - 5	
Write(X)	
	Read(X)
	Read(Y)
	Sum = X+Y
Read(Y)	
Y = Y + 5	
Write(Y)	

T1 consists of two parts – subtract 5 from X and add 5 to Y.

In T2, the value of x is updated but the value of y is not updated.

The sum variable stores an incorrect value.

Following protocols are used to control concurrency and preserve consistency.

4.7 Lock Based Protocol

Locking is a mechanism so that no two transactions can access the same data-item at same time. When one transaction accesses the data-item it issues a lock for that data-item so that no other transaction can access this data till the lock is released.

Locks are of two types.

a. Shared Lock:

- ✓ A transaction uses shared lock to read a data-item.
- ✓ It is also called read lock.
- ✓ When one transaction use this lock then other transactions can also use this lock to read a data-item. Hence the lock is shared.
- ✓ It is denoted by Lock-S( ) .

b. Exclusive Lock:

- ✓ A transaction uses exclusive lock to read and write a data item.
- ✓ It is also called read-write lock.
- ✓ When one transaction uses this lock then other transactions can't use any type of lock. Hence the lock is exclusive.
- ✓ It is denoted by Lock-X( ) .

The relationship between Shared and Exclusive Lock can be represented by the following table which is known as Lock Matrix.

	Shared	Exclusive
Shared	TRUE	FALSE
Exclusive	FALSE	FALSE

How Should Lock be Used?

In a transaction, a data item which we want to read/write should first be locked before any read/write operation. After the operation is over, the transaction unlocks the data item so that other transaction can lock that same data item for their use.

Example:- Let us see how locking mechanisms help us to create error free schedules.

An erroneous schedule is shown below

T1	T2
Read A;	
A = A +50;	
	Read A;
	A = A-50;
	Write A;
Write A;	

incorrect

Here, T2 reads A, before A is modified in T1. This will result inconsistency.

Now we use locking mechanism in the above schedule which is shown below

T1	T2
Lock-X(A) .	
Read A;	
A = A +50;	
Write A;	
Unlock(A)	
	Lock-X(A)
	Read A;
	A = A-50;
	Write A;
	Unlock(A)

correct

until T1 performs Unlock(A) T2 can't access A . so, the inconsistency is removed automatically and the schedule becomes a very correct schedule.

The use of lock helps us to create correct concurrent schedule.

4.7.1 Two Phase Locking (2PL) Protocol

Two phase locking has two phases.

Growing Phase:

- ✓ All the locks are issued in this phase. No lock is released.
- ✓ After all changes to data-items are committed and then second phase (shrinking phase) starts.

Shrinking Phase:

- ✓ No locks are issued in this phase.
- ✓ All the changes to data-items are noted (stored) and then locks are released.

In Growing Phase Transaction reaches at a point where all the lock it may need has been acquired. This point is called **Lock Point**.

After Lock Point has been reached, the transaction enters to shrinking phase.

2PL is of two types

1. Strict Two Phase Locking Protocol

- ✓ A transaction can release shared lock after the Lock point, but it cannot release any exclusive lock until the transaction commits.
- ✓ This protocol creates cascade less schedule.

Cascading Schedule: In this schedule one transaction is dependent on another transaction .so if one has to rollback then other has to rollback.

2. Rigorous Two Phase Locking Protocol:-

- ✓ A transaction is cannot release any lock (either shared or exclusive) until it commits.

2PL protocol guarantees serializability, but can't guarantee that deadlock will not happen.

Ex:- Let T1 and T2 are two transactions.

T1=A +B and T2=B +A

T1	T2
Lock-X(A) .	Lock-X(B)
Read A;	Read B;
Lock-X(B)	Lock-X(A)

can't execute Lock-X(A) since A is locked by T1

can't execute Lock-X(B) since B is locked by T2

In the above situation T1 wait for B and T2 wait for A. The waiting time never ends. Both the transaction can't proceed further. This situation is called deadlock.

The wait for graph is shown below

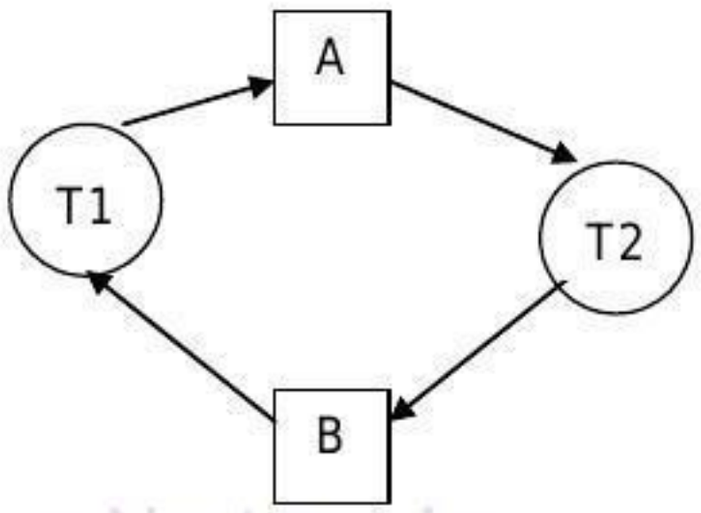


Fig: wait for graph

Wait For Graph (WFG):

- ✓ Used in deadlock detection method.
- ✓ Create a node for each transaction.
- ✓ Create an edge Ti to Tj , if Ti waiting to lock item locked by Tj.
- ✓ A cycle in WFG indicates deadlock has occurred.
- ✓ WFG is created at regular intervals.

4.7.2 Timestamp Ordering Protocol

- ✓ A **timestamp** is a value/time attached to a transaction (or data item). It denotes the starting(relative) time of transaction.
- ✓ Sequential ordering of time stamps ensures serializability. Hence the name time stamp ordering.
- ✓ A conflict occurs when an older transaction (transactions with smaller timestamps) tries to read/write a value already read/written by a younger transaction.
- ✓ Read/write proceeds only if last update on that data item was carried out by an older transaction. Otherwise, transaction requesting read/write is restarted and given a new timestamp.
- ✓ Here, No locks is used so no deadlock.

Timestamp of a data item X is of two types:

Read Time Stamp, **RTS(X)** : Time Stamp of last transaction to read item

Write Time Stamp, **WTS(X)** : Time Stamp of last transaction to write item

These timestamps are updated after a successful read/write operation on data item X.

How should timestamps be used?

Older transaction gets priority over younger transaction in the event of conflict operation.

Conflict is resolved by rolling back and restarting transaction.

To ensure serializability following Rules are used

Rule1 : (for read operation)

```
If TS(Ti)<WTS(X) then
        Rollback Ti
Else
{
        perform read(x)
        set RTS(X) =max{TS(Ti) , RTS(X)
}
```

The value of current transaction(T_i) is greater than the value of transaction who has done the latest write operation(WTS). That means we can read the latest value not any old value.

Rule2 : (for write operation)

```
    If TS(Ti)<RTS(X) then
        Rollback Ti
    Else if TS(Ti) <WTS(X) then
        Rollback Ti      // ignored in Thomas write rule
Else
{
        perform write(x)
        set WTS(X) =TS(Ti)
}
```

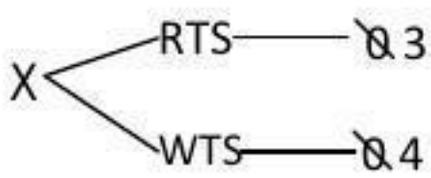
The value of current transaction is greater than the latest read,write operation (i.e. RTS,WTS). That means we can write upon the latest value not any old value.

Ex1:- S : $r_1(X)$ $w_2(X)$ $w_1(X)$

	TS
T1	3
T2	4

check whether time stamp ordering protocol allows schedule S.

Ans:-



for $r_1(X)$:- $TS(T_i)<WTS(X)$ i.e. $TS(T_1)<WTS(X)$

$3<0$ (false)

$\Rightarrow$goto else and perform read operation $r_1(X)$ and $RTS(X)=3$

for $w_2(X)$:- $TS(T_i)<RTS(X)$ i.e. $TS(T_2)<RTS(X)$

$4<3$ (false)

$TS(T_i)<WTS(X)$ i.e. $TS(T_2)<WTS(X)$

$4<0$ (false)

$\Rightarrow$goto else and perform write operation $w_2(X)$ and $WTS(X)=4$

for $w_1(X)$:- $TS(T_i)<RTS(X)$ i.e. $TS(T_1)<RTS(X)$

$3<3$ (false)

$TS(T_i)<WTS(X)$ i.e. $TS(T_1)<WTS(X)$

$3<4$ (true)

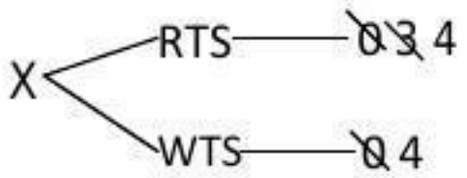
$\Rightarrow$rollback

Ex2:- S : $r_1(X)$, $r_2(X)$, $w_1(X)$, $w_2(X)$.

	TS
T1	3
T2	4

check whether time stamp ordering protocol allows schedule S.

Ans:- Initially for a data-item X , $RTS(X)=0$, $WTS(X)=0$



for $r_1(X)$:- $TS(T_i)<WTS(X)$ i.e. $TS(T_1)<WTS(X)$

$3<0$ (false)

$\Rightarrow$goto else and perform read operation $r_1(X)$ and $RTS(X)=3$

for $r_2(X)$:- $TS(T_i)<WTS(X)$ $TS(T_2)<WTS(X)$

$4<0$ (false)

$\Rightarrow$goto else and perform read operation $r_2(X)$ and $RTS(X)=4$

for $w_1(X)$:- $TS(T_i)<RTS(X)$ i.e. $TS(T_1)<RTS(X)$

$3 < 4$ (true)

$\Rightarrow$ rollback

for $W_2(X)$:- $TS(T_i) < RTS(X)$ i.e. $TS(T_2) < RTS(X)$

$4 < 4$ (false)

$TS(T_i) < WTS(X)$ i.e. $TS(T_2) < WTS(X)$

$4 < 0$ (false)

$\Rightarrow$goto else and perform write operation $w_2(X)$ and $WTS(X)=4$

Note: rollback has occured for $w_1(X)$,that means all the operation in Transaction T1 is cancelled.Transaction T1 will be executed in later time. but in thomas write rule instead of roll-back we wait for our execution.

Thomas Write Rule:- is a modified version of time stamp ordering protocol.

This protocol is same as time stamp ordering except the following modification

if $TS(T_i) < RTS(X)$ then

ignore write(X)

In the above condition T1 is trying to write an obsolete(old)value in X . so, write(X) can be ignored .

Thomas Write Rule allows greater potential concurrency. It allows some view serailzable schedules that are not conflict serializable.

4.7.3 Optimistic Concurrency Control

- ✓ All data items are updated at the end of the transaction. At the end of the transaction, if any data item is found inconsistent with respect to the value in, then the transaction is rolled back.
- ✓ Check for conflicts at the end of transaction. No checking while the transaction is executing.
- ✓ Checks are all made at once, so low transaction execution overhead.
- ✓ Updates are not applied until end_transaction. Updates are applied to local copies in a transaction space.

It has 3 phases.

a. **Read phase :** various data items are read and stored in temporary variables(local copies). All operations are performed in these variables without updating the database.

b. **Validation phase :** All concurrent data items are checked to ensure serializability will not be validated if the transaction updates are actually applied to the database. Any change in the value causes the transaction rollback.

   transaction timestamps are used. write_sets and read_sets are maintained

   To check that TransA does not interfere with TransB the following must hold:

   - ✓ TransB completes its write phase before TransA starts its reads phase
   - ✓ TransA starts its write phase after TransB completes its write phase, and the read set of TransA has no items in common with the write set of TransB
   - ✓ Both the read set and the write set of TransA have no items in common with the write set of TransB, and TransB completes its read phase before TransA completes its read phase.

c. **Write phase :** if validation is successful, transaction updates applied to database; otherwise updates are discarded and transaction is aborted and restarted..

   It doesn't use any locks hence deadlock free. However starvation problem of popular data item may occur.

Exercises

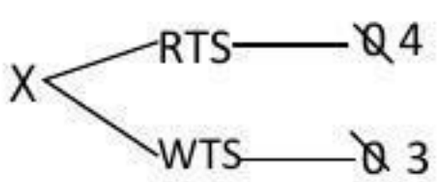
Q:- S : $w_1(X)$, $r_2(Y)$, $r_1(Y)$, $r_2(X)$.

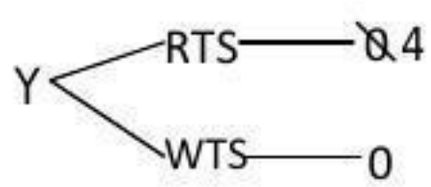
	TS
T1	3
T2	4

check whether time stamp ordering protocol allows schedule S.

Ans:- initially for a data-item X , RTS(X)=0 , WTS(X)=0

initially for a data-item Y , RTS(Y)=0 , WTS(Y)=0





for $w_1(X)$:- $TS(T_i) < RTS(X)$ i.e. $TS(T_1) < RTS(X)$

$3 < 0$ (false)

$TS(T_i) < WTS(X)$ i.e. $TS(T_1) < WTS(X)$

$3 < 0$ (false)

$\Rightarrow$ goto else and perform write operation $w_1(X)$ and $WTS(X)=3$

for $r_2(Y)$:- $TS(T_i) < WTS(Y)$ i.e. $TS(T_2) < WTS(Y)$

$4 < 0$ (false)

$\Rightarrow$ goto else and perform read operation $r_2(Y)$ and $RTS(Y)=4$

for $r_1(Y)$:- $TS(T_i) < WTS(Y)$ i.e. $TS(T_1) < WTS(Y)$

$3 < 0$ (false)

$\Rightarrow$ goto else and perform read operation $r_1(Y)$

for $r_2(X)$:- $TS(T_i) < RTS(X)$ i.e. $TS(T_2) < WTS(X)$

$4 < 3$ (false)

$\Rightarrow$ goto else and perform read operation $r_2(X)$ and $RTS(X)=4$

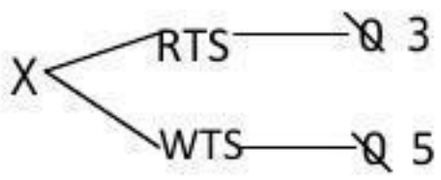
Q:- S : $r_1(X)$, $r_2(Y)$, $w_3(X)$, $r_2(X)$, $r_1(Y)$

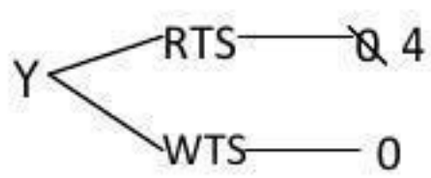
	TS
T1	3
T2	4
T3	5

check whether time stamp ordering protocol allows schedule S.

Ans:- initially for a data-item X , $RTS(X)=0$, $WTS(X)=0$

initially for a data-item Y , $RTS(Y)=0$, $WTS(Y)=0$





for $r_1(X)$:- $TS(T_i) < WTS(X)$ i.e. $TS(T_1) < WTS(X)$

$3 < 0$ (false)

$\Rightarrow$ goto else and perform read operation $r_1(X)$ and $RTS(X)=3$

for $r_2(Y)$:- $TS(T_i) < WTS(Y)$ $TS(T_2) < WTS(Y)$

$4 < 0$ (false)

$\Rightarrow$ goto else and perform read operation $r_2(Y)$ and $RTS(Y)=4$

for $w_3(X)$:- $TS(T_i) < RTS(X)$ i.e. $TS(T_3) < RTS(X)$

$5 < 3$ (false)

$TS(T_i) < WTS(X)$ i.e. $TS(T_3) < WTS(X)$

$5 < 0$ (false)

$\Rightarrow$ goto else and perform write operation $w_3(X)$ and $WTS(X)=5$

for $r_2(X)$:- $TS(T_i) < WTS(X)$ $TS(T_2) < WTS(X)$

$4 < 5$ (true)

$\Rightarrow$ rollback

for $r_1(Y)$:- $TS(T_i) < WTS(Y)$ $TS(T_1) < WTS(Y)$

$3 < 0$ (false)

$\Rightarrow$ goto else and perform read operation $r_1(Y)$

4.8 Recovery System

Recovery means to get back the information, which is lost. It is the component of database system that ensure the atomicity and durability property of transaction. It is responsible to maintain database consistency in case of failure.

Types of failure

1. **Transaction failure**:- A transaction fails due to logical error or system error.
   logical error:- error due to invalid input, overflow etc.
   system error:- error due to a critical condition such as deadlock.
2. **System failure/system crash**:- database system fails due to hardware or software problem. Example: problem due to virus.
3. **Disk failure**:- failure due hard disk problem. It is also called catastrophic failure. The main technique used to handle such failure is data backup onto secondary storage such as magnetic tapes.

Recovery Techniques are used to recover from failure. Various techniques are Log based recovery , check-pointing , shadow paging etc.

4.9 Log based recovery

- ✓ Log is nothing but a file which contain sequence of records (log records). Each log record refer to a write operation(update operation)
- ✓ All the log records are recorded step by step in log file. We can say, log file store the history of all update activities.
- ✓ Log contains start of transaction, transaction no, record no, old value, new value, end of transaction etc. For example mini statement in bank ATM.
- ✓ If within an ongoing transaction, the system crashes, then by using log files, we can return back to the previous state as if nothing has happened to the database.
- ✓ The log is kept on disk so that it is not affected by failures except disk and catastrophic failures.

Example: Different types of log records are shown below

- $<T_i, X_i, V1, V2>$ - update log record , where T_i=transaction , X_i=data , V1=old value , V2=new value
- $<T_i, start>$ - Transaction Ti is starts execution
- $<T_i, commit>$ - Transaction T_i is committed
- $<T_i, abort>$ - Transaction Ti is aborted

Create a log for the given transactions T1 and T2

T1	T2
Read A	Read A
A = A-2000	A = A+5000
Write A	Write A
Read B	Read B
B = B+2000	B = B+7000
Write B	Write B

Log

<T_1,start>
<T_1, A, 5000 , 3000>
<T_1, B, 8000 , 10000>
<T_1,commit>
<T_2,start>
<T_2, A, 3000 , 8000>
<T_2, B, 10000 , 17000>
<T_2,commit>

Log based recovery uses one of the following technique

a. **Deferred database modification:- (**modify database after completion of transaction**)**
   - ✓ Database modification is deferred (delayed) until the last operation of transaction is executed.
   - ✓ Update log record maintain the new value of data item.
   - ✓ Recovery system uses one operation
     i) Redo(T_i) : all data items updated by the transaction Ti are set to new value.
   - ✓ In case of failure the recovery system checks the log. The transaction Ti is redo if log contains both <T_i,start> and <T_i,commit> .
b. **Immediate database modification:- (**modify database after a write operation**)**
   - ✓ Database modification is immediately done when a transaction perform update/write operation.
   - ✓ Update log record maintain both old value and new value of data item.
   - ✓ The recovery system uses two operations
     i) Undo(T_i) :- all data items updated by the transaction T_i are set to old value.
     ii) Redo(T_i) :- all data items updated by the transaction T_i are set to new value.

     undo means don't do (Rollback)
     redo means do again (Roll forward)
   - ✓ In case of failure the recovery system checks the log. Transaction Ti is redo if log contains both <T_i,start> and <T_i,commit>. Transaction Ti is undo if log contains <T_i,start> and does not contain <T_i,commit>

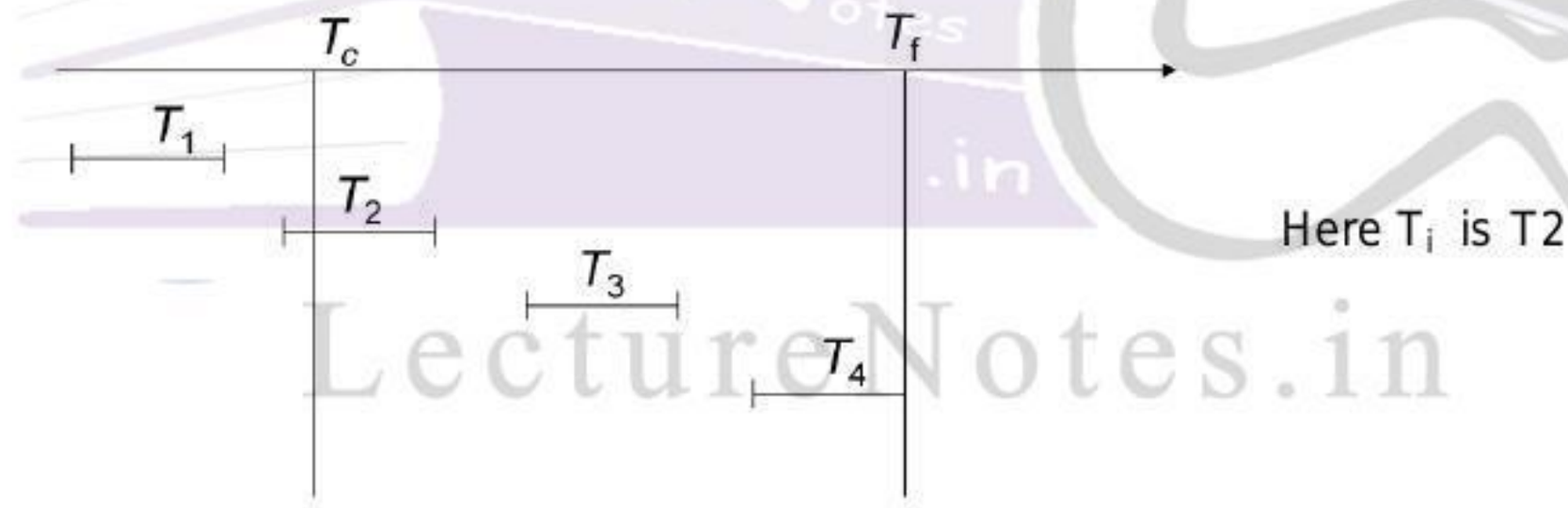
4.10 Check Pointing

- ✓ Searching the entire log (to analyze the history) is time-consuming. We might unnecessarily redo transactions which have already output their updates to the database. So we use check-point record in log file.
- ✓ Check point refers to a particular point of time.
- ✓ At the check point, the contents of database are copied to some storage. Storage can be secondary storage such as hard disk or tertiary storage such as external hard-disk.
- ✓ Database administrator will look after the period of check-point (say after 10 minutes).

Steps:

1. Scan backwards to find the most recent <checkpoint> record
2. Consider the most recent transaction T_i that started before the checkpoint.
3. Scan forward in the log
   (a) If a transaction has <T_i commit>, perform redo(T_i).
   (b) If a transaction has no <T_i commit>, perform undo(T_i).

Example:



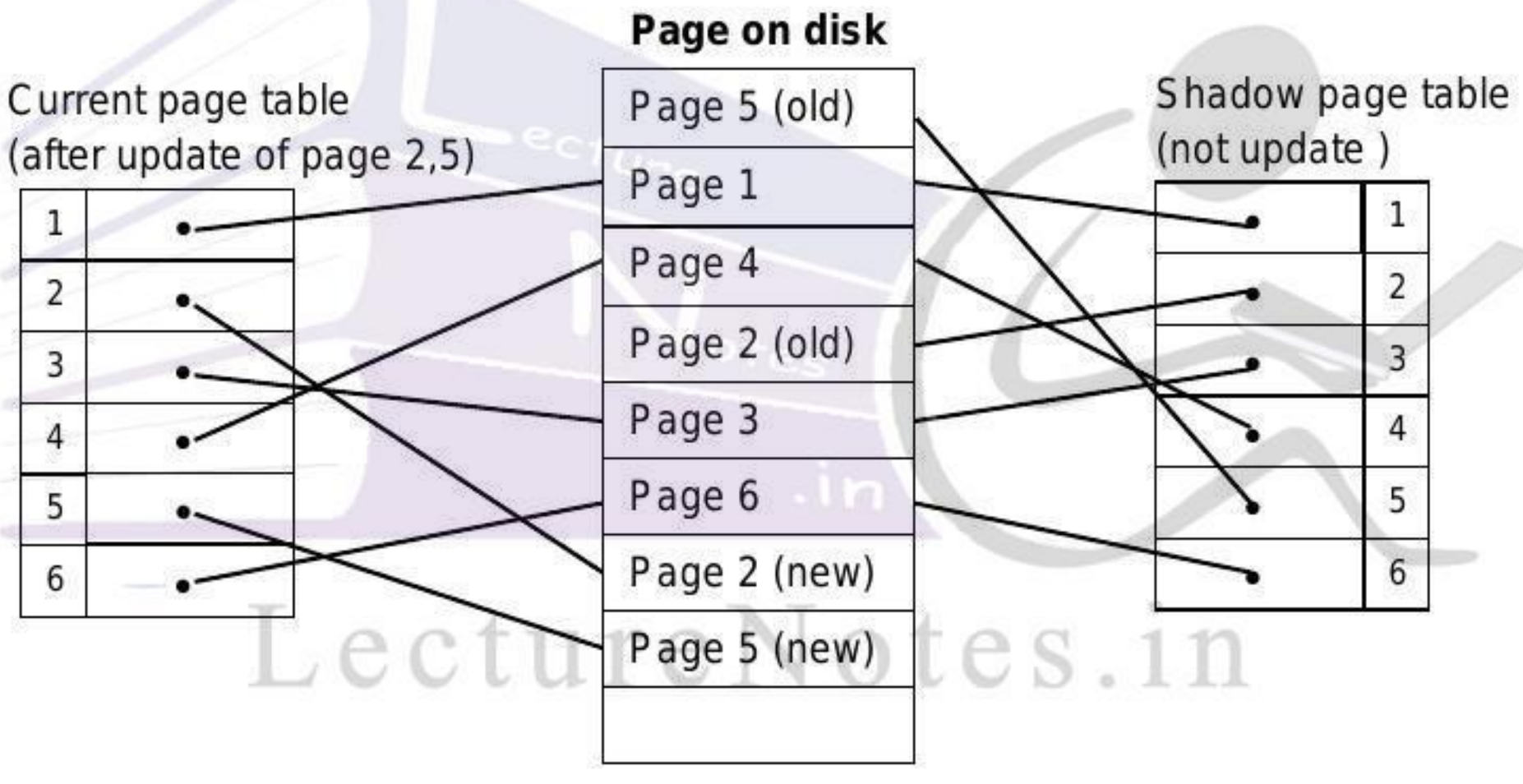
T1 can be ignored (updates already output to disk due to checkpoint)

T2 and T3 redone.

T4 undone

4.11 Shadow Paging

- ✓ Page is a segment of memory. Page table is an index of pages (pages on hard-disk are kept in this table). Each table entry points to a page on the disk.
- ✓ Two page tables are used during the life of a transaction: the current page table and the shadow page table. Shadow page table is a copy of current page table.
- ✓ When a transaction starts, both tables are identical. current page table is updated for each write operation.
- ✓ The shadow page is never changed during the life of the transaction.
- ✓ When the transaction is committed, the shadow page entry becomes a copy of the current page table entry and the disk block with the old data is released.
- ✓ The shadow page table is stored in nonvolatile memory. If system crash occurs, then the shadow page table is copied to the current page table.



Advantages:

- ➢ No need of log records.
- ➢ No Undo / Redo algorithm.
- ➢ Recovery is faster.

Disadvantages :

- ➢ Data are fragmented or scattered.
- ➢ Garbage collection problem (garbage is a unused memory location). Database pages containing old version of modified data need to be garbage collected after every transaction.
- ➢ Concurrent transactions are difficult to execute.

Chapter 5

Storage Structure

5.1 Magnetic disk (Hard disk)

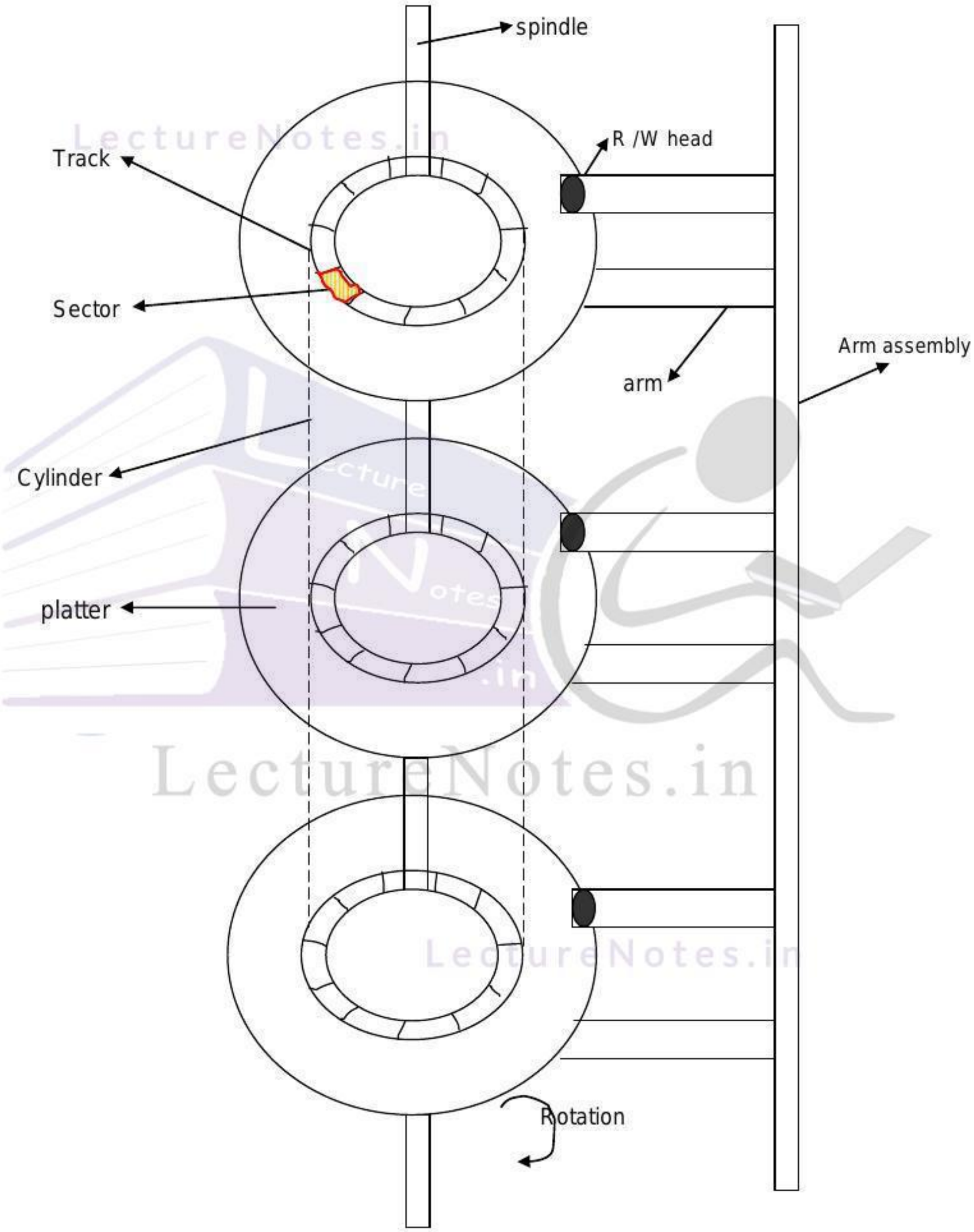


Fig: Components of hard-disk

Magnetic disk supports direct access to a desired location.

Disk blocks: The unit data transfer between disk and main memory is a block. A disk block is a contiguous sequence of bytes.

Track: Blocks are arranged in concentric rings called Tracks.

Sectors: A sectors is the smallest unit of information that can be read from or written to disk. Ex - sector size of 512 bytes.

Platter: The surface of platter is covered with a magnetic material. Information is recorded on this surface. The set of all tracks with same diameter is called cylinder. Typical platter diameters are 3.5 inch or 5.25 inch.

Read write head: Each platter has a read-write head on both side. It is used for reading and writing the data on platter.

Disk controller: A disk controller interfaces a disk drive to the computer.

Checksum:- Disk controller attach checksums to each sector. A checksum is computed when data is written to a sector. Checksum is computer again when the data or the sector read back. For a bad sector, the controller computes checksums & if it detects an error, it tries to read the sector again.

Performance

The time to read or write a block varies depending on the location of data.

Access time = seek time + rotational delay + transfer time

Seek time is the time to move the disk-head to the track on which a desired block is located.

Rotational delay is the waiting time for the desired block to rotate under disk head.

Transfer time is the time to read or write the data in the block once the head is positioned.

5.2 RAID (Redundant Array of Independent Disk)

It combines multiple small, inexpensive disk drives into an array of disk drives which yields performance more than that of a Single Large Expensive Drive (SLED). RAID is also called Redundant Array of Inexpensive Disk. Storing same data in different disk (thus redundantly) increases fault-tolerance.

Mean Time Between Failure (MTBF) of the array = MTBF of an individual drive, divided by the number of drives in the array. Because of this, the MTBF of an array of drives would be too low for many application requirements.

Various types of RAID are discussed below

RAID-0

RAID Level 0 is not redundant. Since no redundant information is stored, performance is very good, but the failure of any disk in the array results in data loss(no fault tolerance). A single record is divided into stripes(typically 512bytes) and is stored across all disk. The record can be accessed quickly by reading all disks at same time. This is called striping.

RAID-1

RAID Level 1 provides redundancy by writing all data to two or more drives. The performance is faster on reads and slower on writes compared to a single drive. If any one drive fails, no data is lost. This is called mirroring.

RAID-2

In RAID Level 2 Hamming error correction codes is used with drives which do not have built-in error detection.

RAID-3

RAID Level 3 stripes data at a byte level across several drives, with parity stored on one drive. Byte-level striping requires hardware support for efficient use.

RAID-4

RAID Level 4 stripes data at a block level across several drives, with parity stored on one drive. Parity information allows recovery from the failure of any single drive. The performance of a level 4 array is very good for reads (the same as level 0). Writes, however, require that parity data be updated each time. Because only one drive in the array stores redundant data. The cost per megabyte is low.

RAID-5

RAID Level 5 is similar to level 4, but distributes parity among the drives. This can speed up small writes in multiprocessing system. The performance for reads is lower than a level 4 array. The cost per megabyte is the same as level 4.

Summary:

- RAID-0 is the fastest and most efficient array type but offers no fault-tolerance.
- RAID-1 is the array of choice for critical, fault-tolerant environments.
- RAID-2 is seldom used today since ECC is embedded in almost all modern disk drives.
- RAID-3 can be used in data intensive or single-user environments which access long sequential records to speed up data transfer. However, RAID-3 does not allow multiple I/O operations to be overlapped and requires synchronized-spindle drives in order to avoid performance degradation with short records.
- RAID-4 offers no advantages over RAID-5 and does not support multiple simultaneous write operations.
- RAID-5 is the best choice in multi-user environment. However, at least three drives are required for RAID-5 array.

5.3 Index

The concept of index in database is similar to index of a book. Using the index of book we can quickly search for a topic. Similarly, using the index of table we can quickly search for a record.

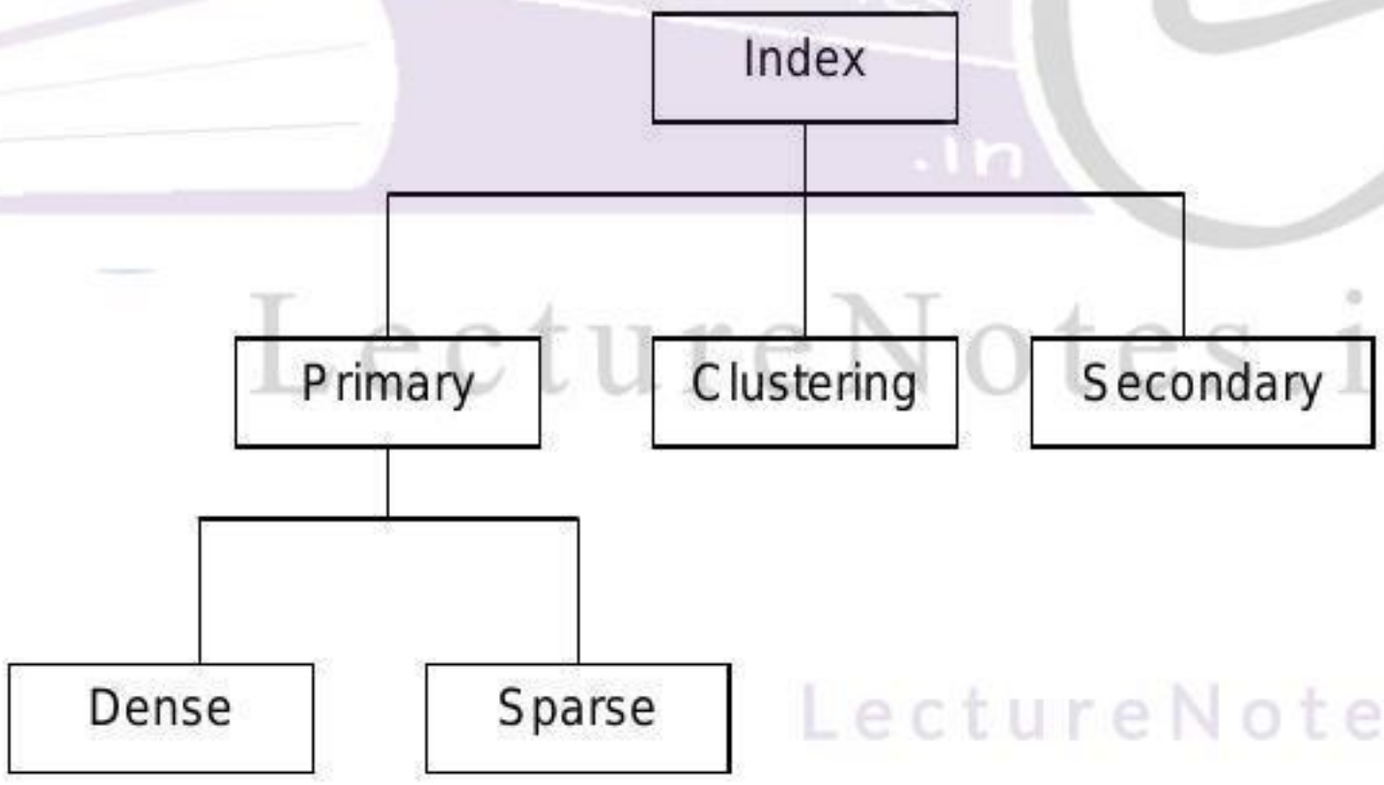
An index is a small table having only two columns. The first column contains the primary key of the main table and the second column contains a set of pointers holding the address of the disk block where that particular key value can be found.

Advantage:- indexing reduces searching time of a record.

Suppose a table has several rows of data, each row is of 20 bytes. If you want to search for the record number 100, the system must read each and every row and after reading 99x20 = 1980 bytes it will find record number 100. If we have an indexing system then The index contain only two columns, may be just 4 bytes in each of its rows. After reading only 99x4 = 396 bytes of data from the index, the system finds an entry for record number 100, and directly points at the record in the physical storage device. The result is a much quicker access to the record (a speed advantage of 1980:396).

disadvantage:- index takes up additional space in memory. Index needs to be updated periodically for insertion or deletion of records in the main table.

5.4 Types of Index



5.4.1 Primary Index

In primary index, there is a one-to-one relationship between the entries in the index table and the records in the main table. Primary index sometimes used to mean index on a Primary key. Primary index can be of two types:

i) Dense primary index: An index record appears for every search-key value in the main table. i.e. each and every record in the main table has an entry in the index.

Example:-

index

Roll	Pointer
1	
2	
3	
4	

Table

Roll	Name	Branch	Section
1			
2			
3			
4			

A pointer stores the address of a record.

ii) Non-Dense/Sparse primary Index: An index record appears for only some of the values in the file.

For large tables the Dense Primary Index itself begins to grow in size. To keep the size of the index smaller, instead of pointing to each and every record in the main table, the index points to the records in the main table in a gap. See the following example.

index

Roll	Pointer
1	
11	
21	
31	
41	

Table

Roll	Name	Branch	Section
1			
. .			
10			
11			
. .			
20			
. .			

block1

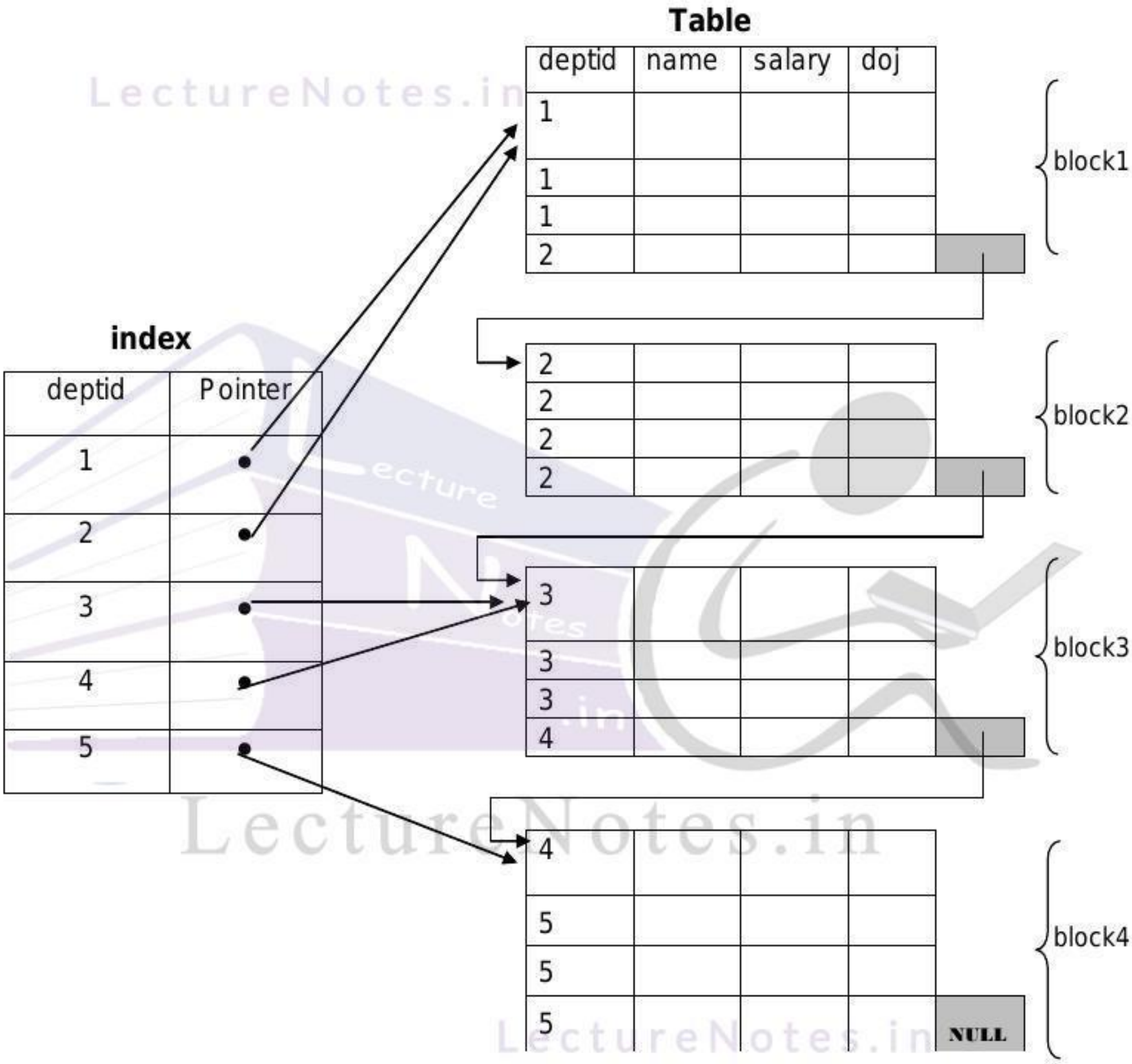
block2

blocks are logically separated from each other

As you can see, the records have been divided into several blocks, each containing a fixed number of records (in our case 10). The pointer in the index table points to the first record of each block. If you are searching for roll 14, the index is first searched to find out the highest entry which is smaller than or equal to 14. We have 11. The pointer leads us to roll 11 where a sequential search is made to find out roll 14.

5.4.2 Clustering Index

sometimes we are asked to create an index on a non-unique key (non-primary key) , such as Dept-id. There could be several employees in each department. Here, all employees belonging to the same Dept-id are considered to be within a single cluster, and the index pointers point to the cluster as a whole.

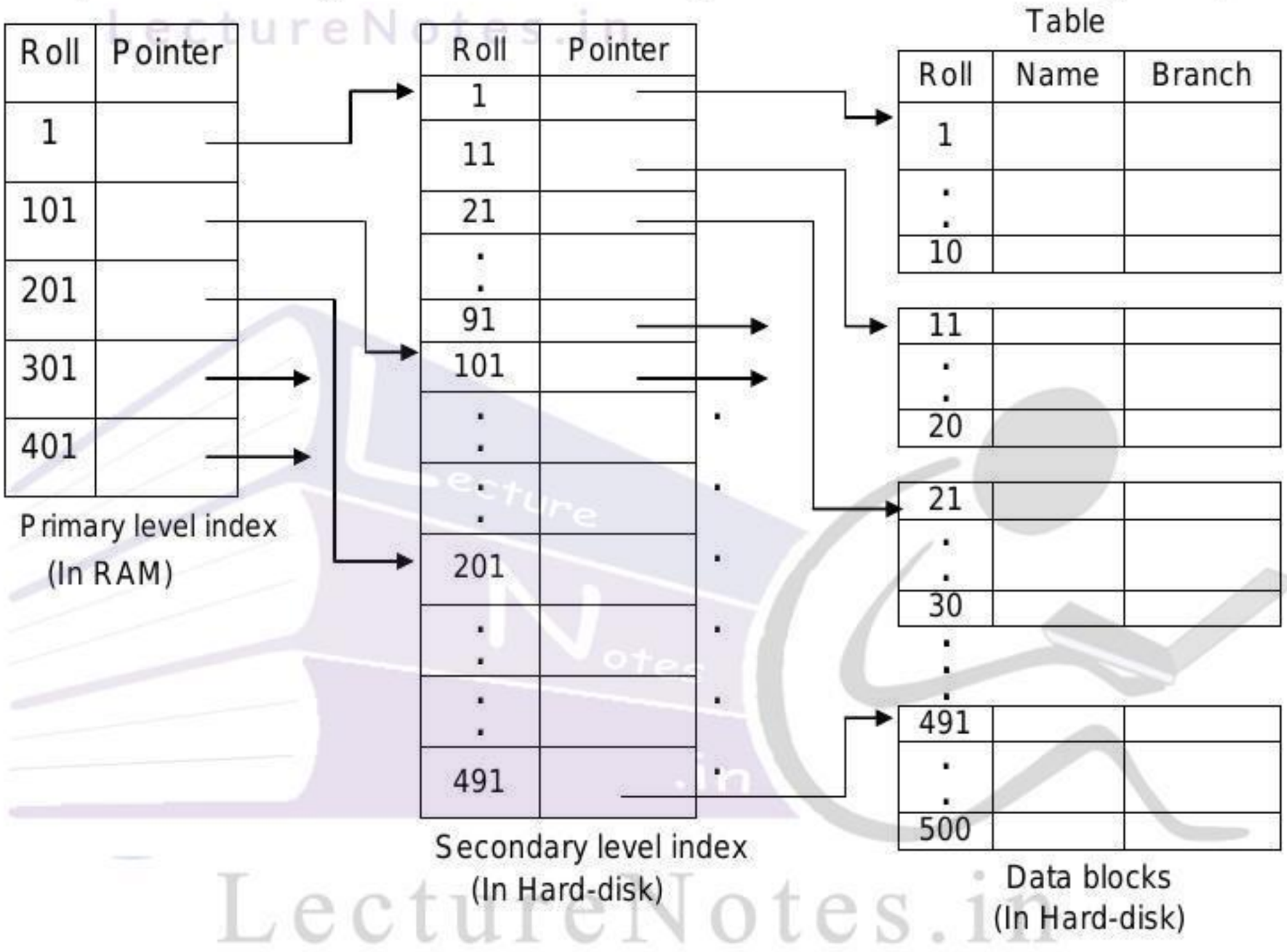


Let us explain the above diagram. The disk blocks contain a fixed number of records (in this case 4 each). The index contains entries for 5 separate departments. The pointers of these entries point to the first record of the block. The blocks themselves may point to the next block in case a cluster overflows a block size. This can be done using a special pointer at the end of each block.

5.4.3 Secondary Index

while creating the index, generally the index table is kept in the primary memory (RAM) and the main table, because of its size is kept in the secondary memory (Hard Disk).

A table may contain thousand of records for which even a sparse index becomes so large in size that we cannot keep it in the primary memory. And if we cannot keep the index in the primary memory, then we lose the advantage of the speed of access. For very large table, it is better to organize the index in multiple levels. See the following example.



Here, the primary level index, (created with a gap of 100 records, and thereby smaller in size), is kept in the RAM for quick reference. If you need to find out the record of roll 14, the index is first searched to find out the highest entry which is smaller than or equal to 14. We have 1. The adjoining pointer leads us to the anchor record of the corresponding secondary level index, where another similar search is conducted. This finally leads us to the actual data block whose anchor record is roll 11. We now come to roll 11 where a sequential search is made to find out roll 14.

Multilevel Index

The Multilevel Index is a modification of the secondary level index system. In this system we may use even more number of levels in case the table is even larger.

5.5 Hashing

Searching time depends upon the number of elements present in memory.

But, Hashing is a searching technique which is independent of number of elements.

- A File has n records. Each record has a key, which uniquely determines the record.

Let K is key and L is memory location (address). Hash function is defined as H: K→ L

Example:-

0		
1		
2	011	Delhi
3		
4	022	Calcutta
5		
6	033	Chennai
7		
8	044	Mumbai
9		

Here STD codes of cities are keys .Hash function maps these keys into the addresses 2,4,6,8, by just adding the digits of the keys.

5.5.1 Hash Function

Following methods are used for Hash function, H(k)

a) Division Method:— The hash function is H (k) = k (mod m)

where m is a prime number (greater than number of keys)

Example: Let 56 students have a unique 10 digit regdno. Here regdno is key.

m=59 (greater than 56)

L consist of 100 memory locations 00, 01, 02 …… 99

for the regdno 1201229109, 1201229128, 1201229402 H(k) is shown below

H (1201229109) = 1201229109 % 59 = 24

H (1201229128) = 1201229128 % 59 = 43

H (1201229402) = 1201229402 % 59 = 22

b) Mid-square Method: — The key k is squared. Then the address is obtained by taking middle digits.

Example: Let 200 employees have a unique 3 digit empid. Here empid is key.

H (k) = 2nd & 3rd digit of k^2

k :	067	048	146
k^2 :	4489	2304	21316
H(k) :	48	30	13

c) Folding Method:—

The key k is partitioned into number of parts k_1, k_2, ……. k_n. Then the parts are added together (ignoring the carry) i.e. $H(k) = k_1 + k_2 + \ldots\ldots\ldots\ldots + k_n$.

Example: Let 2000 employees have a unique 4 digit empid. Here empid is key.

H (1643) = 16 + 43 = 57

H (1999) = 19 + 99 = 18 (The leading digit 1 is ignored)

H (1176) = 11 + 76 = 87

H (1374) = 13 + 74 = 87

When more than one key refer to same memory location. That problem is called collision.

5.5.2 Collision avoidance technique

Two techniques are used to avoid collision.

a) Linear probing

b) Chaining

a) Linear probing:- here array is used.

Table T stores all the records in memory.

If a memory location(h) is already filled then we store the record in the next empty location. We apply linear search in table T to find an empty memory location T(h), T(h+1), T(h+2), T(h+3)…….

Record: A, B, C, D, E, X, Y, Z

H(k): 4, 8, 2, 11, 4, 11, 5, 1

Table T

1	X
2	C
3	Z
4	A
5	E
6	Y
7	
8	B
9	
10	
11	D

Disadvantage of Linear probing:- when records tend to cluster (repeating memory location) then searching time of a record increases. Two methods are used to minimize clustering

i) Quadratic probing:-suppose a record has hash address h, is already

filled then we search the memory locations with address h, h+1,h+4, h+9, h+16……h+i^2, …….to decrease the collision.

ii) Double hashing:-The collision is resolved by hashing the hash address again. so hash function Hash(h)=h′ .we search the memory location with address h, h+ h′, h+ 2h′ , h+ 3h′ , …….

disadvantage:- Suppose a record in collision is deleted from location T[r].While searching for another record in collision if we reach at T[r] then it does not mean an unsuccessful search.

b) Chaining :- here link-list is used.

Each record has two parts

i) Data part to store data

ii) Next part to link the records having same hash address.

Example:-The keys 25, 96, 102, 162, 197 stored in hash Table using chaining method.

Here, H(k): k % 5

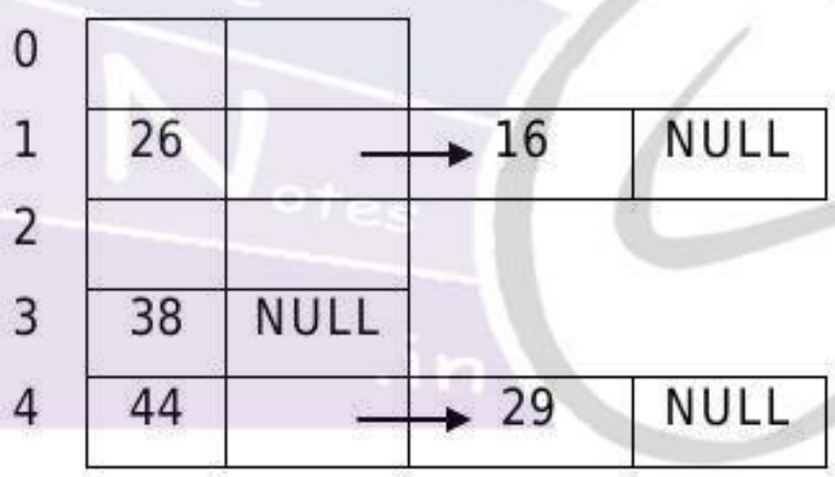
H(26) = 26 % 5 = 1

H(44) = 44 % 5 = 4

H(38) = 38 % 5 = 3

H(29) = 29 % 5 = 4

H(16) = 16 % 5 = 1



Disadvantage:- 1. Memory requirement is more for storing pointers.

2. Extra overhead/coding for manipulating pointers.

5.6 B-Tree

Reasons for using B-Tree:-

- When searching tables on disc, the cost of each disc transfer (disc access) is high but doesn't depend much on the amount of data transferred. So our aim is to minimize disc access (file access).
  - ✓ We know, we can't improve the height i.e. log n for a tree. So, we wish to make the height of tree as small as possible. The solution is to use B-Trees. It has more branches and thus less height. As branching increases, depth decreases so as access time.
  - ✓ In a Btree, one node can hold many elements/items.
    Assume that to access one node of Btree, we need one disc read operation.
    A B-tree of order 101 and height 3 can hold $101^4 - 1$ items (approximately 100 million). So, any item can be accessed with max^m 3 disc reads (assuming we hold the root in memory)
  - ✓ B-Trees are called balanced sorted trees, since all the leaves are at the same level. It is also called multi-way search tree. It is a form of multilevel indexing.
  - ✓ B-Trees grow towards root not towards leaf.

Definition of a B-tree:-

- A B-tree of order m is an m-way tree (i.e. a node can have max^m m children) which satisfy following properties

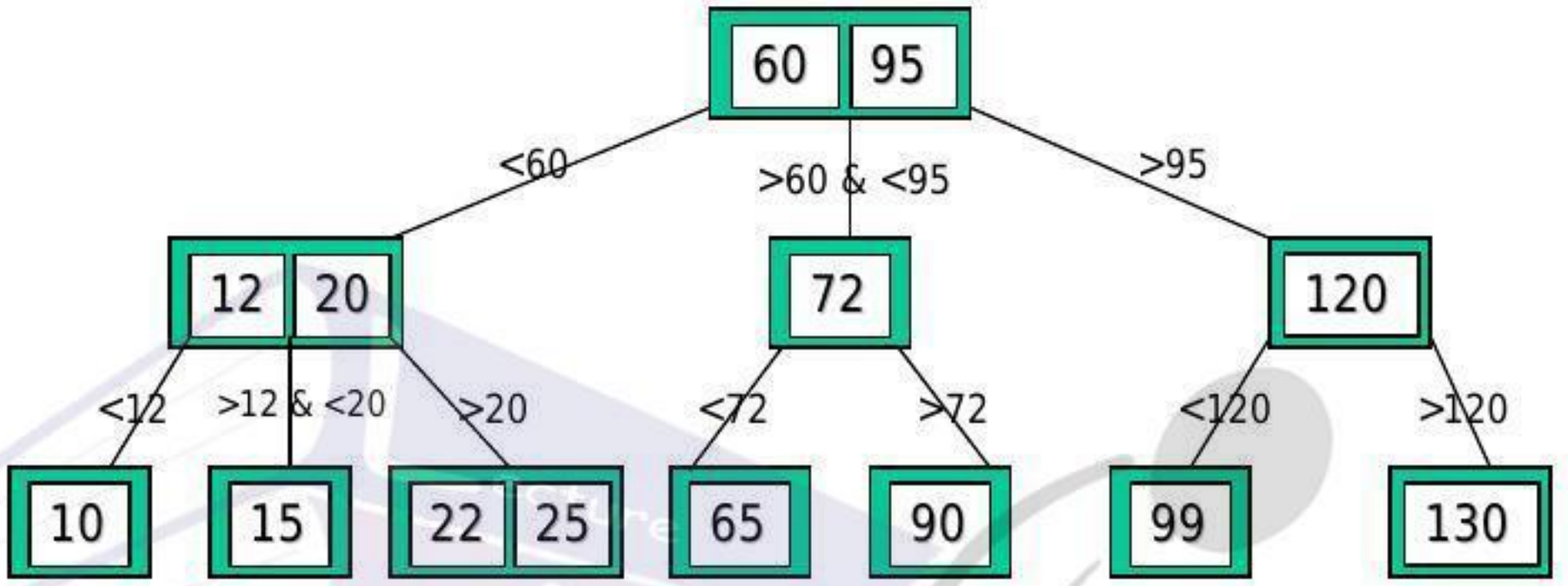
1. The number of keys in a node (non-leaf node) is one less than the number of its children (and these keys partition the keys of children like Binary Search Tree).
2. The root has minimum one key to maximum (m –1) keys. So, root has minimum two children to maximum m children.
3. Any node (non-leaf node) except the root has min^m $(\lceil m/2 \rceil - 1)$ to max^m (m–1) keys
   So, they have $min^m \lceil m/2 \rceil$ children to max^m m children
4. All leaf nodes are on the same level

Note:- non-leaf node are called internal node (i.e. node having child)
leaf node are called external node (i.e. node having no child)

Example:-

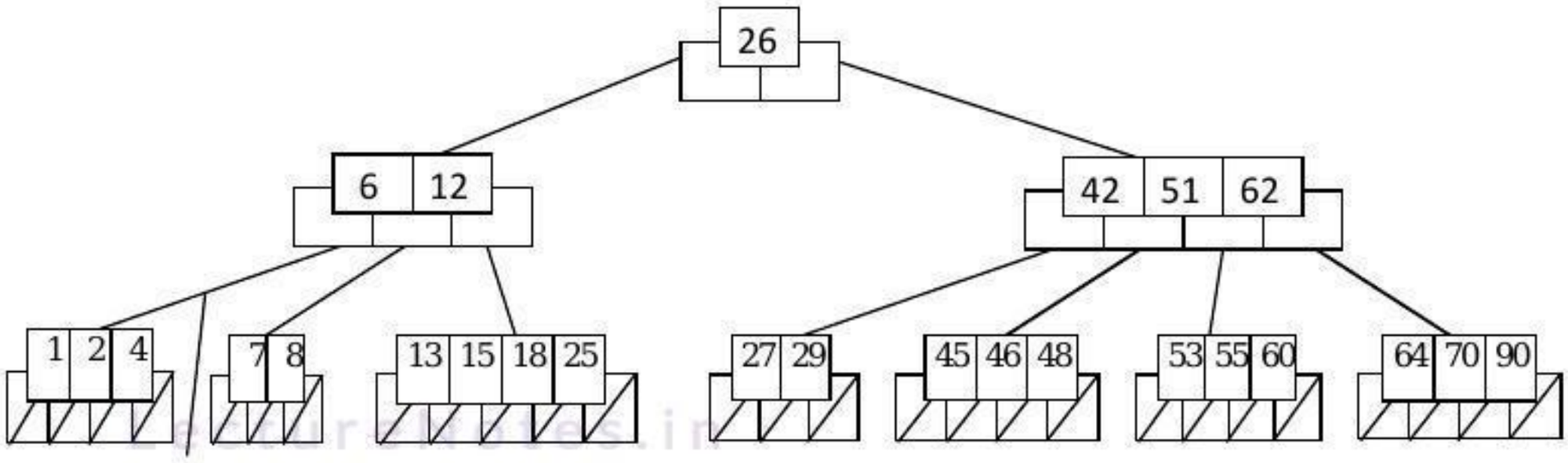
A B-tree of order 3 (i.e. a node can have max^m 3 children) satisfy following properties

1. The root has minimum one key to maximum 2 keys. So, root has minimum two children to maximum 3 children.
2. Any node (non-leaf node) except the root has min^m 1 to max^m 2 keys. So, they have min^m 2 children to max^m 3 children.



Note:- duplicate key values are not allowed.

Example:-



A B-tree of order 5 containing 26 items/keys

- B-Tree is called Balanced Tree because every path from the root node to a leaf node is of same length. (That means all searches for individual values/keys require the same number of nodes to be read from the disc**)**

Analysis of B-Trees:-

The maximum number of items in a B-tree of order m and height h:

root $m-1$

level 1 $m(m-1)$

level 2 $m^2(m-1)$

.
.
.

level h $m^h(m-1)$

So, the total number of items is $(1 + m + m^2 + m^3 + \ldots + m^h)(m-1)$

$$= [(m^{h+1} - 1)/(m-1)](m-1) = \mathbf{m^{h+1} - 1}$$

When $m = 5$ and $h = 2$, this gives $5^3 - 1 = 124$

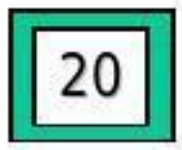
- A B-tree of order 3 is known as 2-3 tree ($\because \min^m \lceil 3/2 \rceil = 2$ children and $\max^m$ 3 children)
- A B-tree of order 5 is known as 3-5 tree ($\because \min^m \lceil 5/2 \rceil = 3$ children and $\max^m$ 5 children)
- A B-tree of order 9 is known as 5-9 tree ($\because \min^m \lceil 9/2 \rceil = 5$ children and $\max^m$ 9 children)

5.6.1 Creating a B-tree

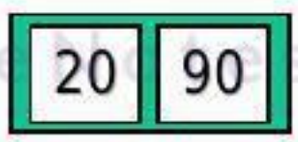
Q:- Create a Btree of order 3 for the following set of key values

20, 90, 15, 14, 2, 11, 1, 73,12

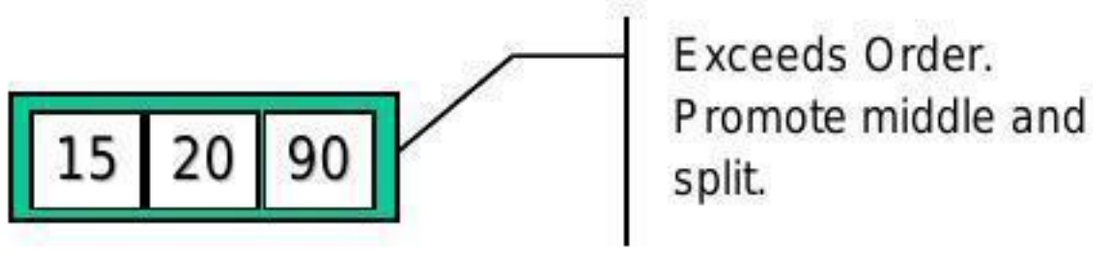
Ans:- insert 20



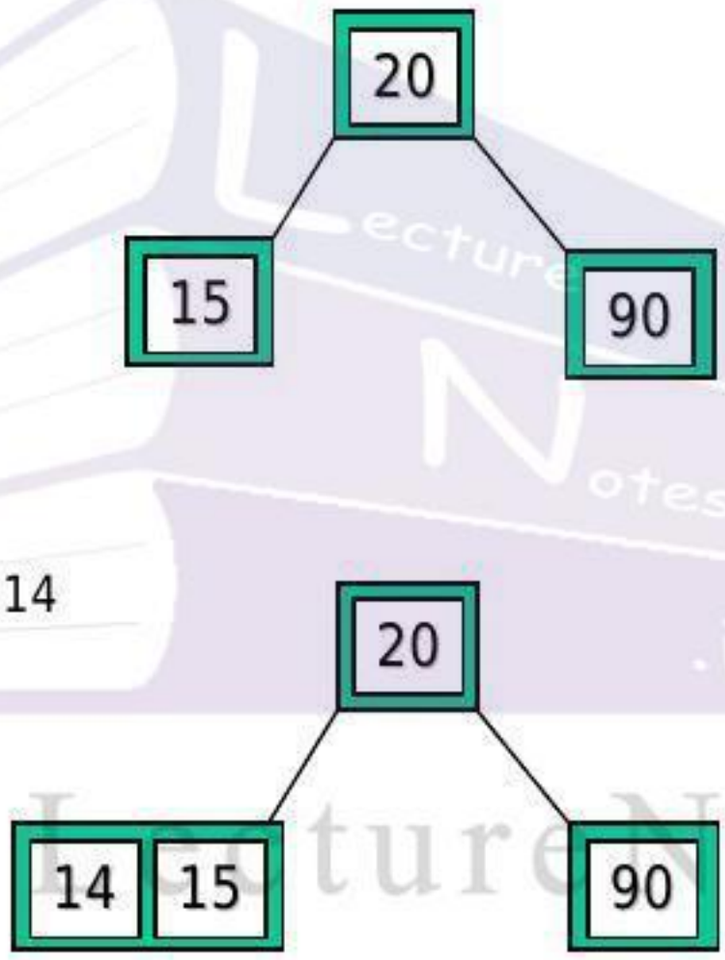
insert 90



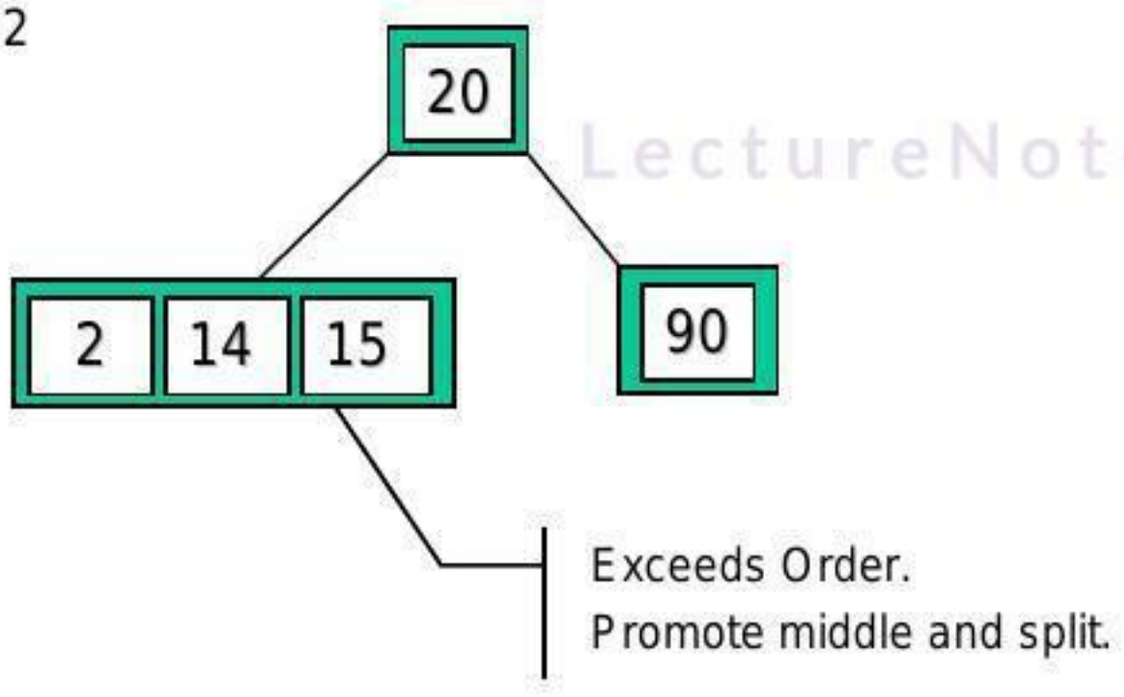
insert 15

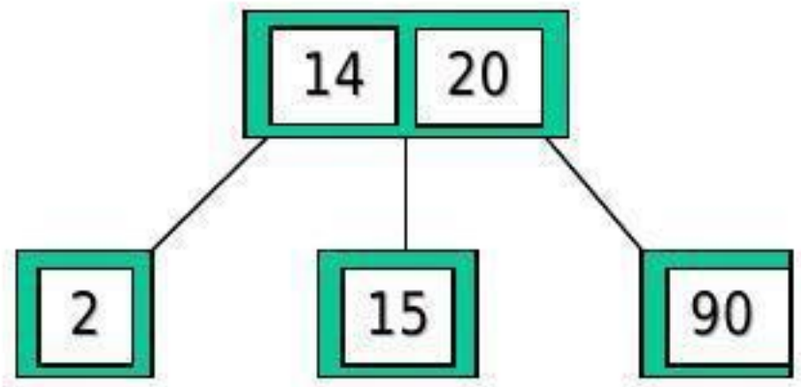


insert 14

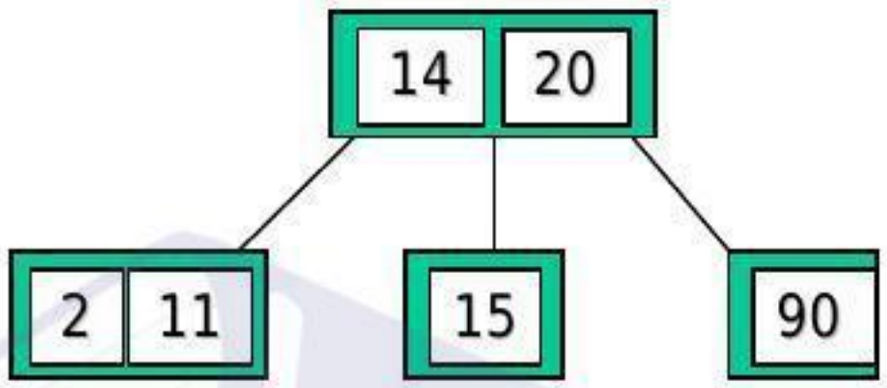


insert 2

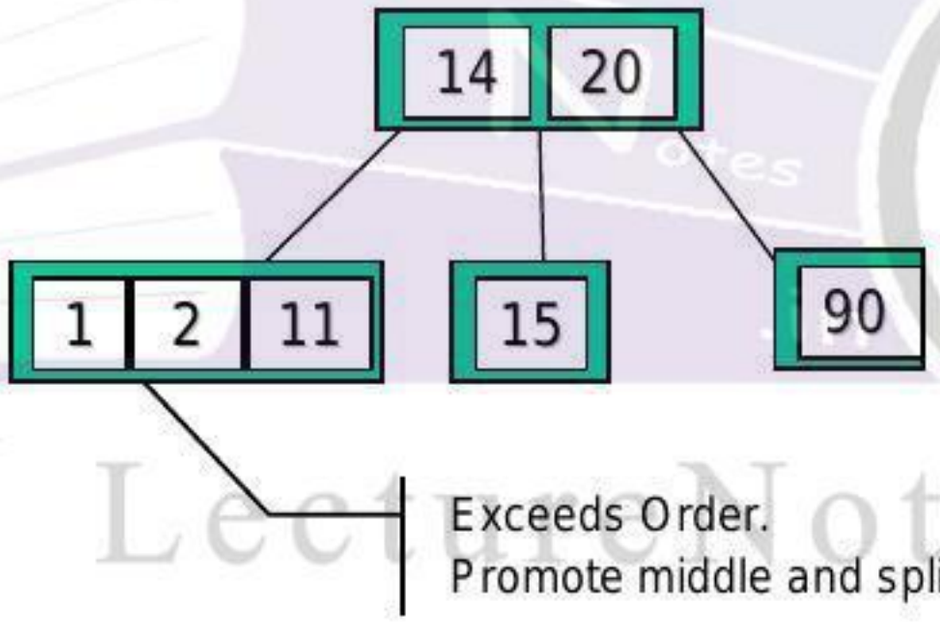


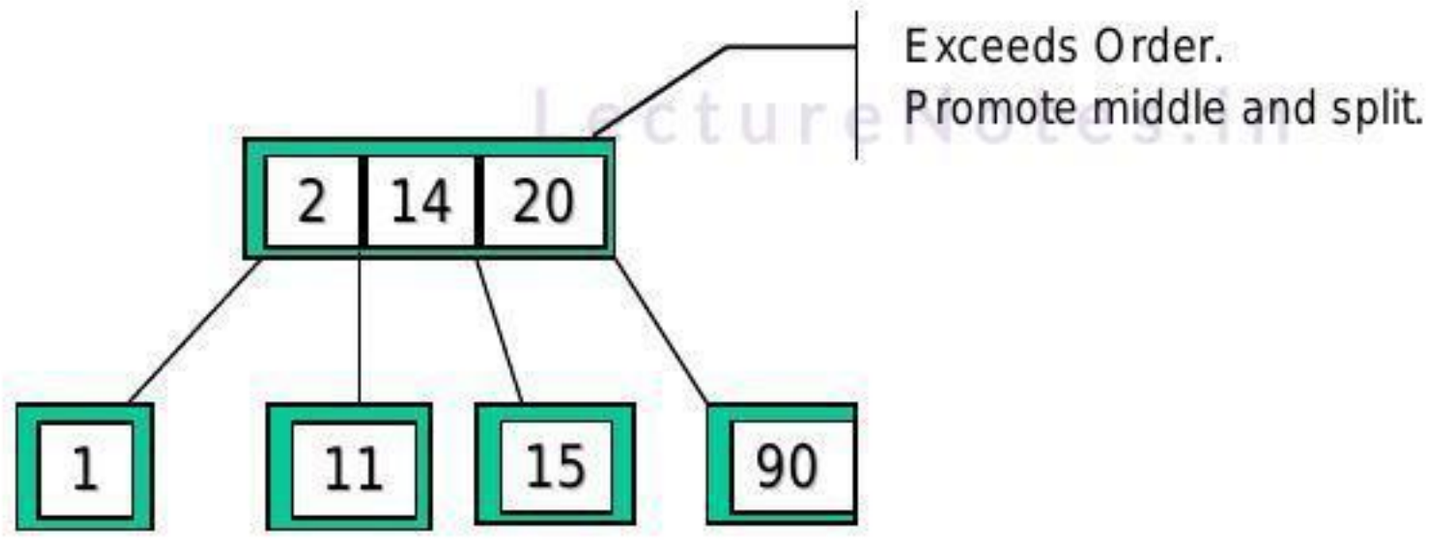


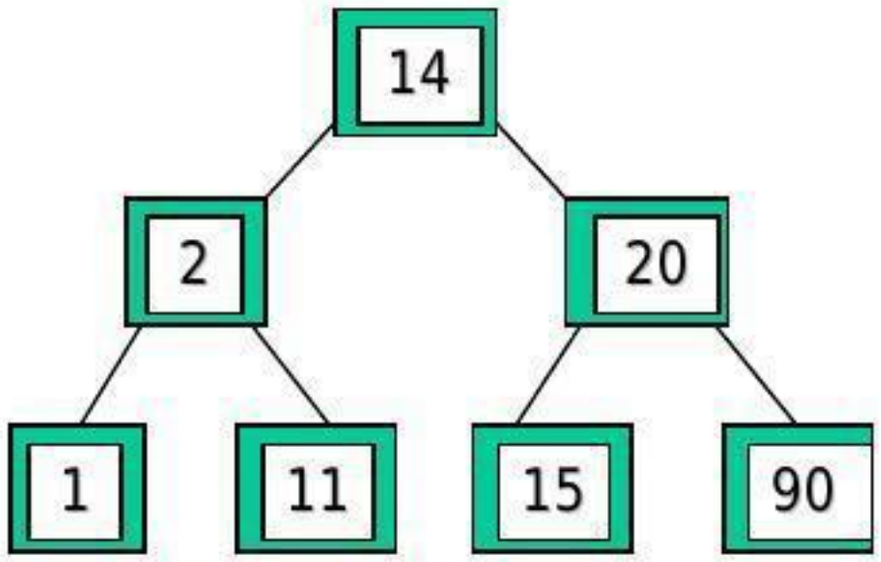
insert 11



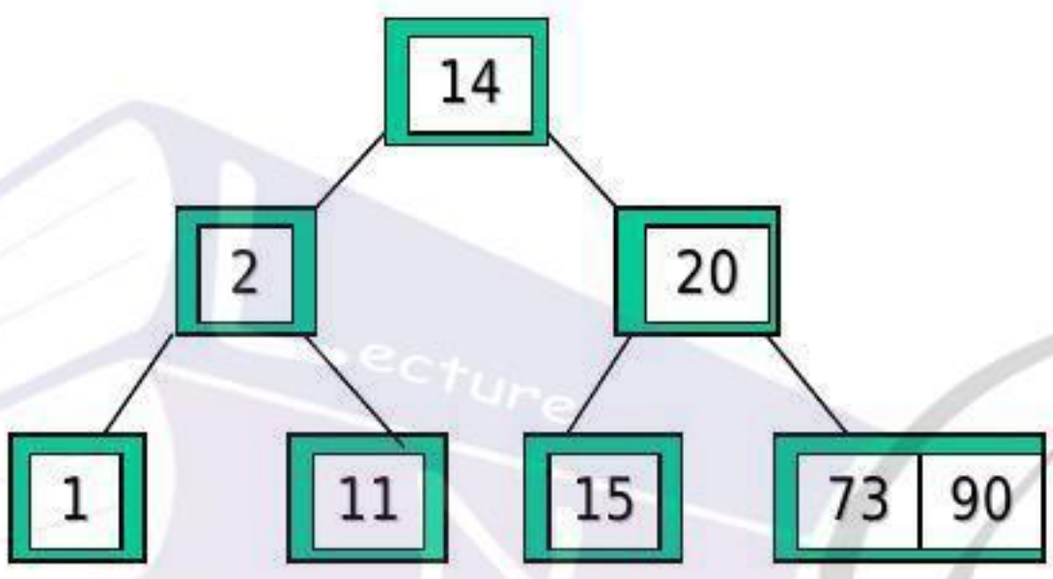
insert 1



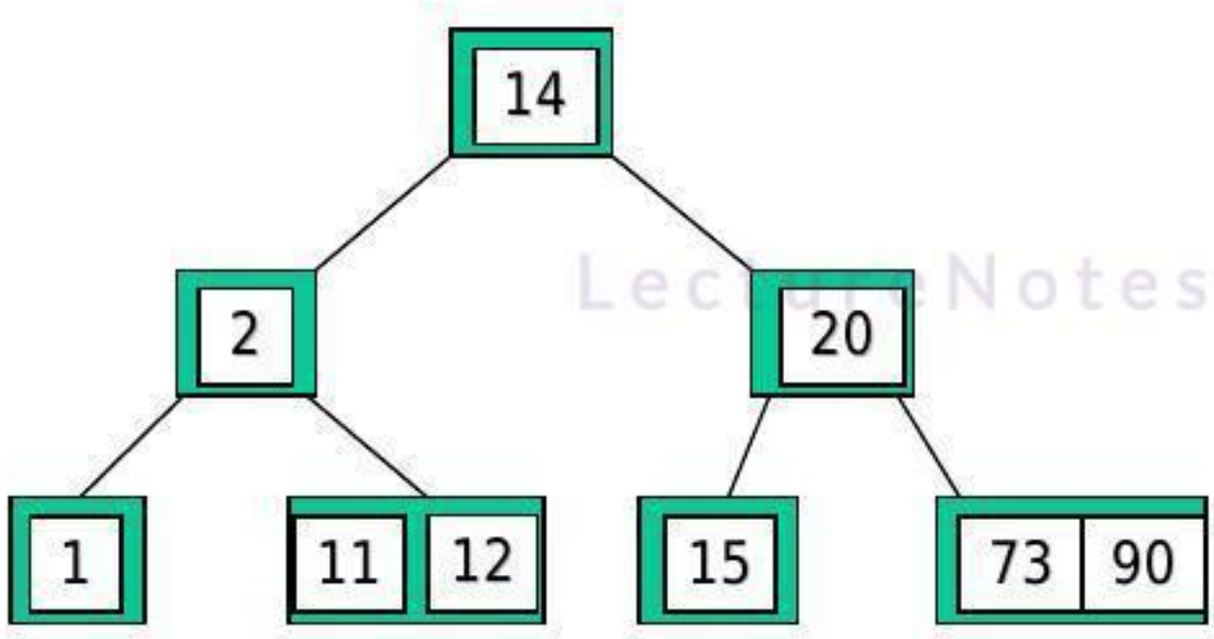




insert 73



insert 12

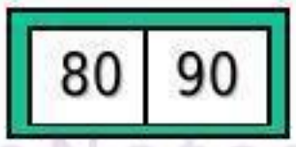


Q:- Create a Btree of order 4 for the following set of key values

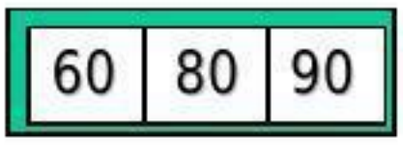
90, 80, 60, 5, 120, 14, 150, 1 , 73, 45

Ans:- insert 90

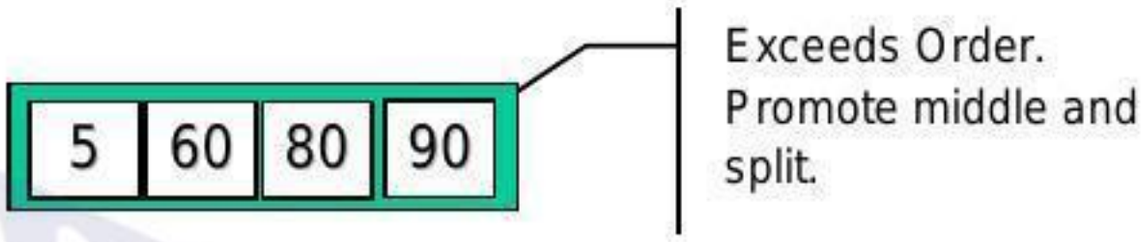
insert 80

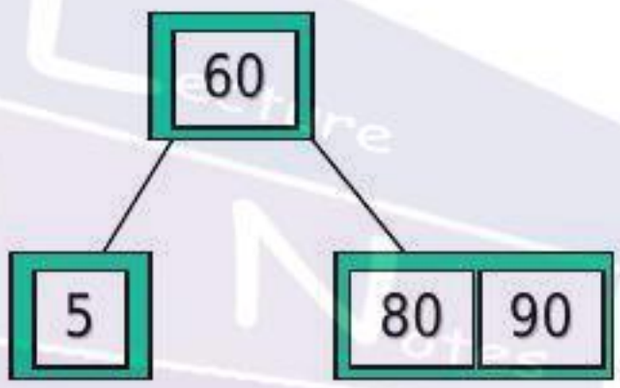


insert 60

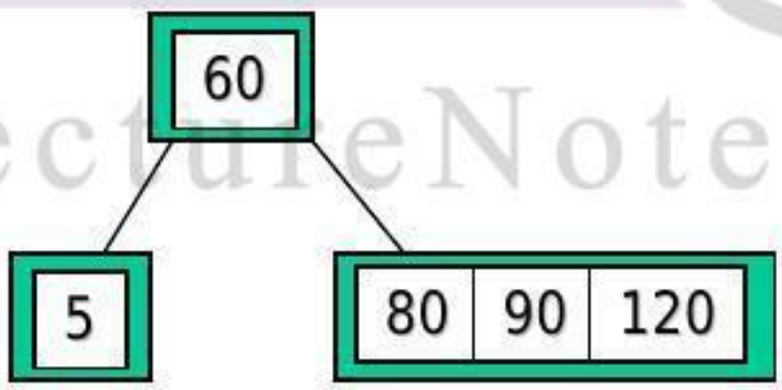


insert 5

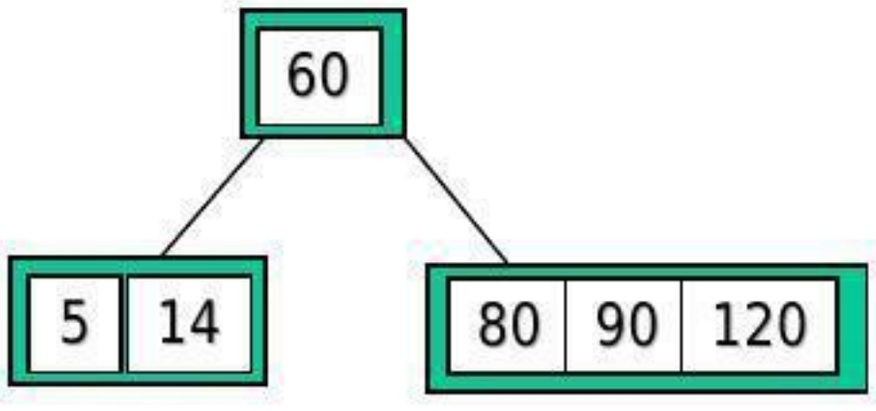




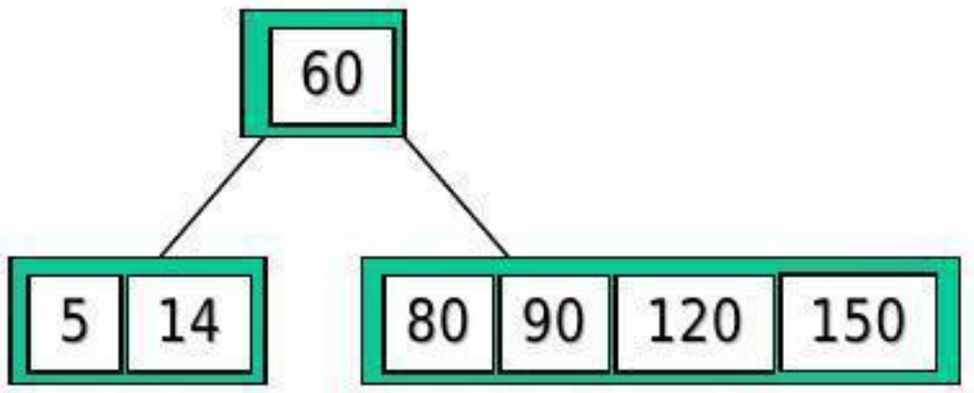
insert 120

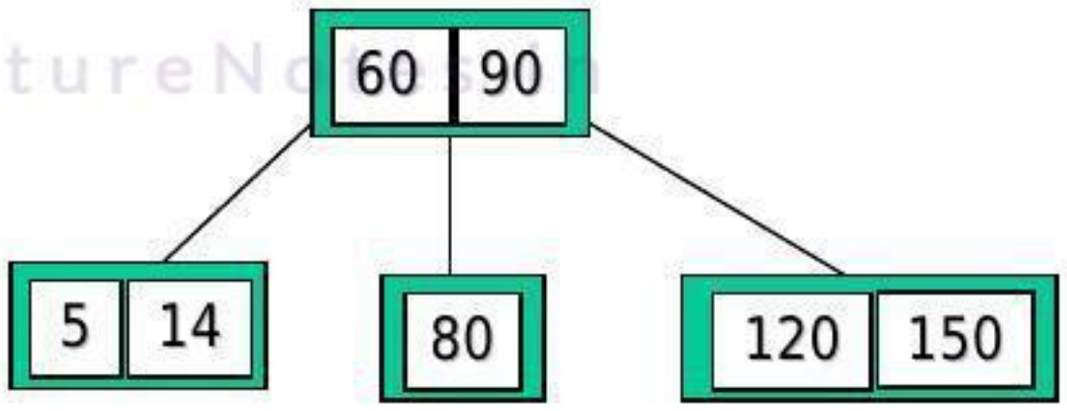


insert 14

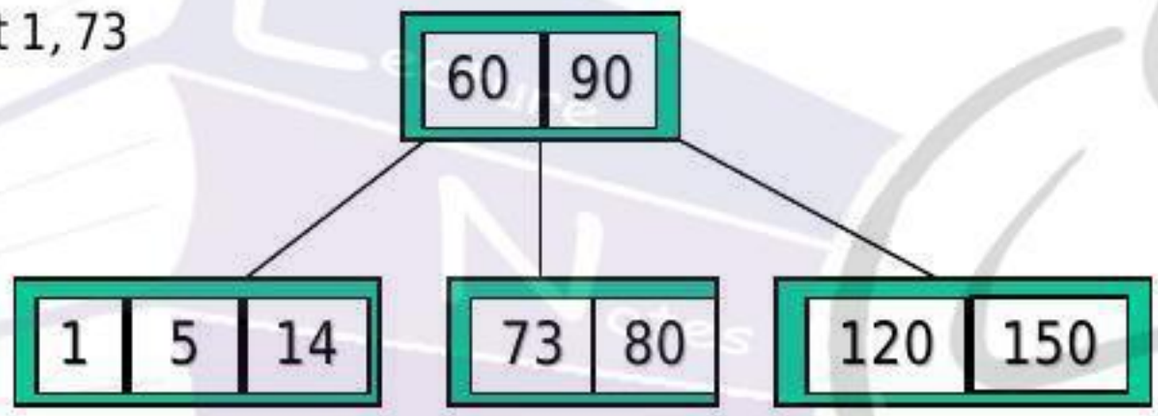


insert 150

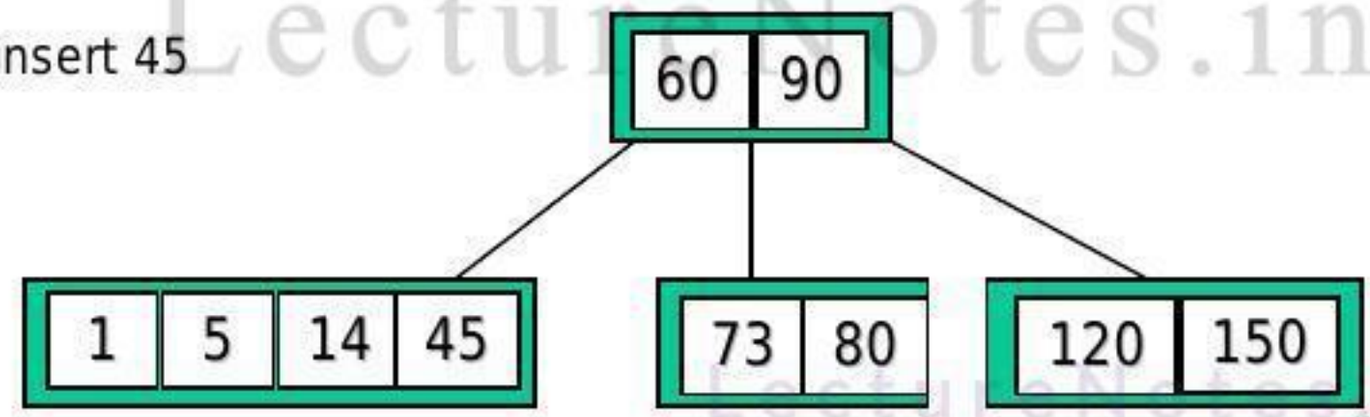


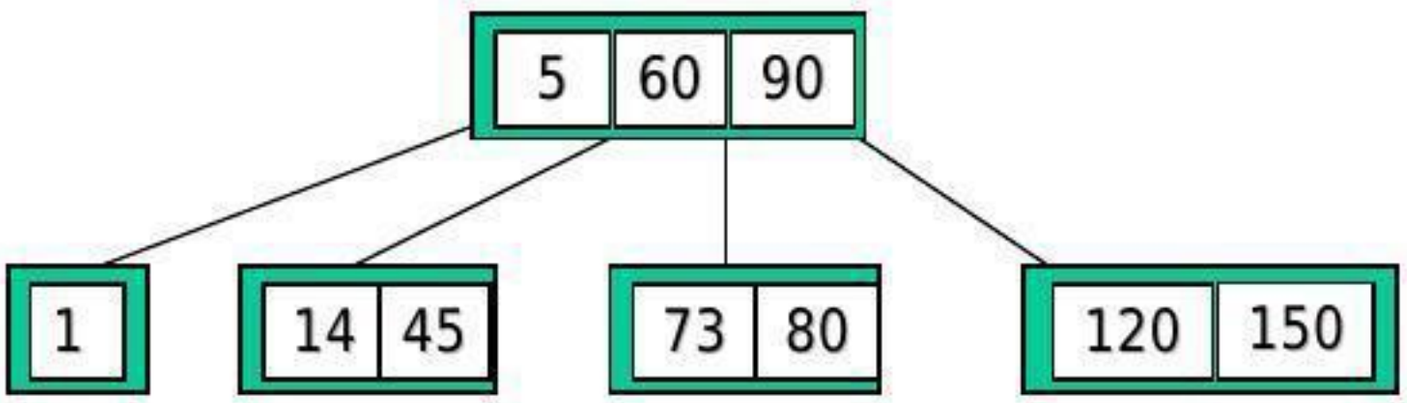


insert 1, 73



insert 45

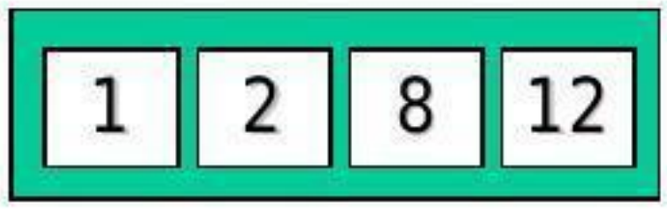




- Create a Btree of order 4 for the following set of key values :

  1 12 8 2 25 6 14 28 17 7 52 16 48 68 3 26 29 53 55 45

- The first four items go into the root:

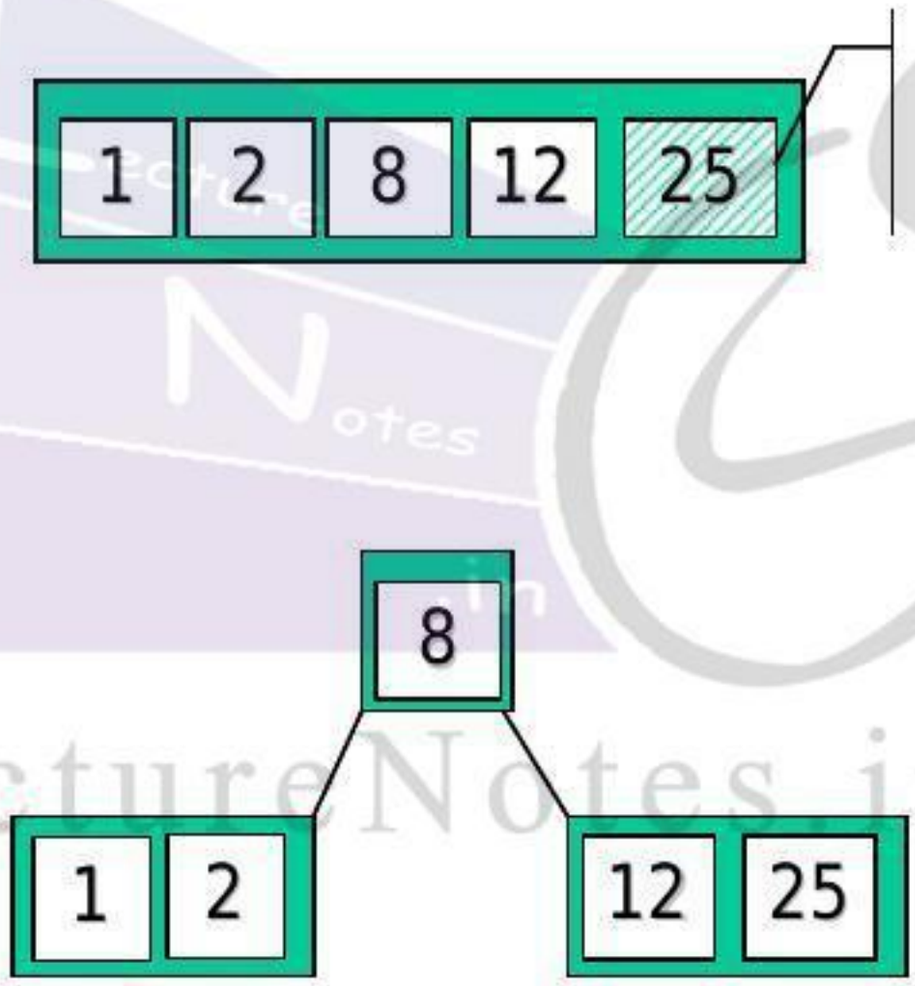


- To put the fifth item in the root would violate condition 5
- Therefore, when 25 arrives, pick the middle key to make a new root

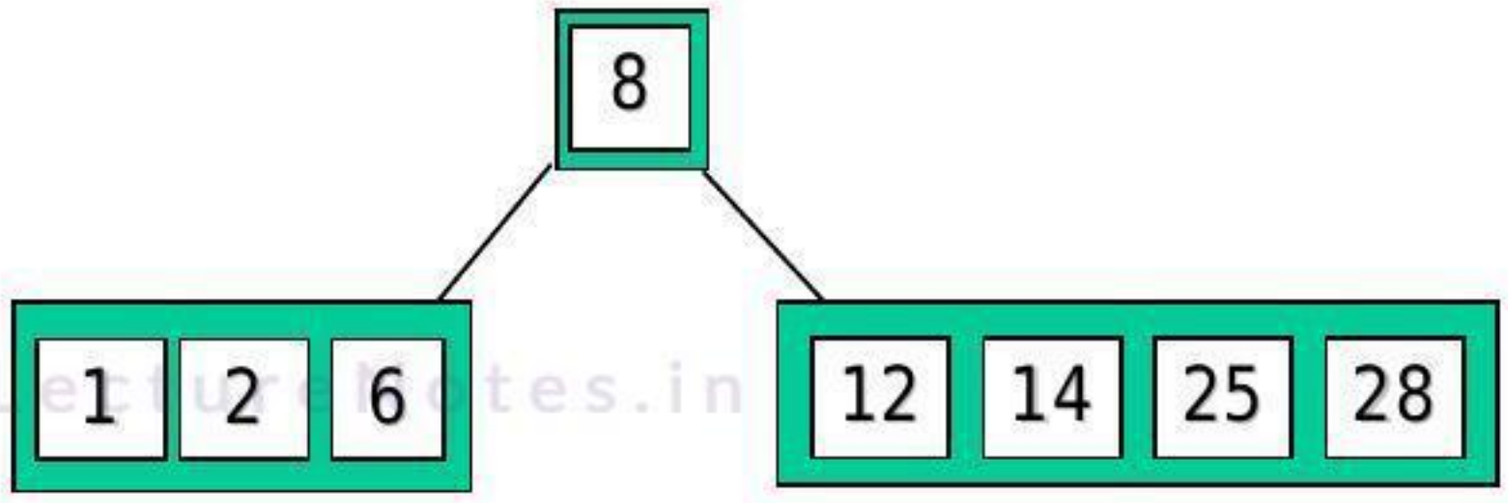
elements

1
12
8
2
25
6
14
28
17
7
52
16
48
68
3
26
29
53
55
45

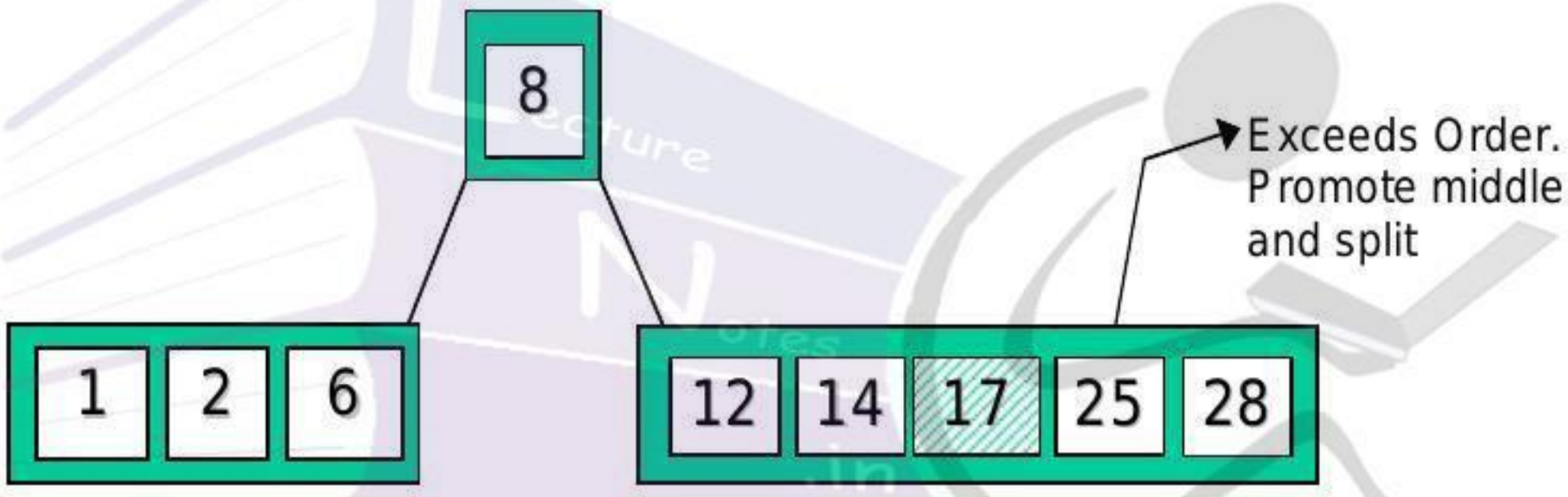
Add 25 to the tree

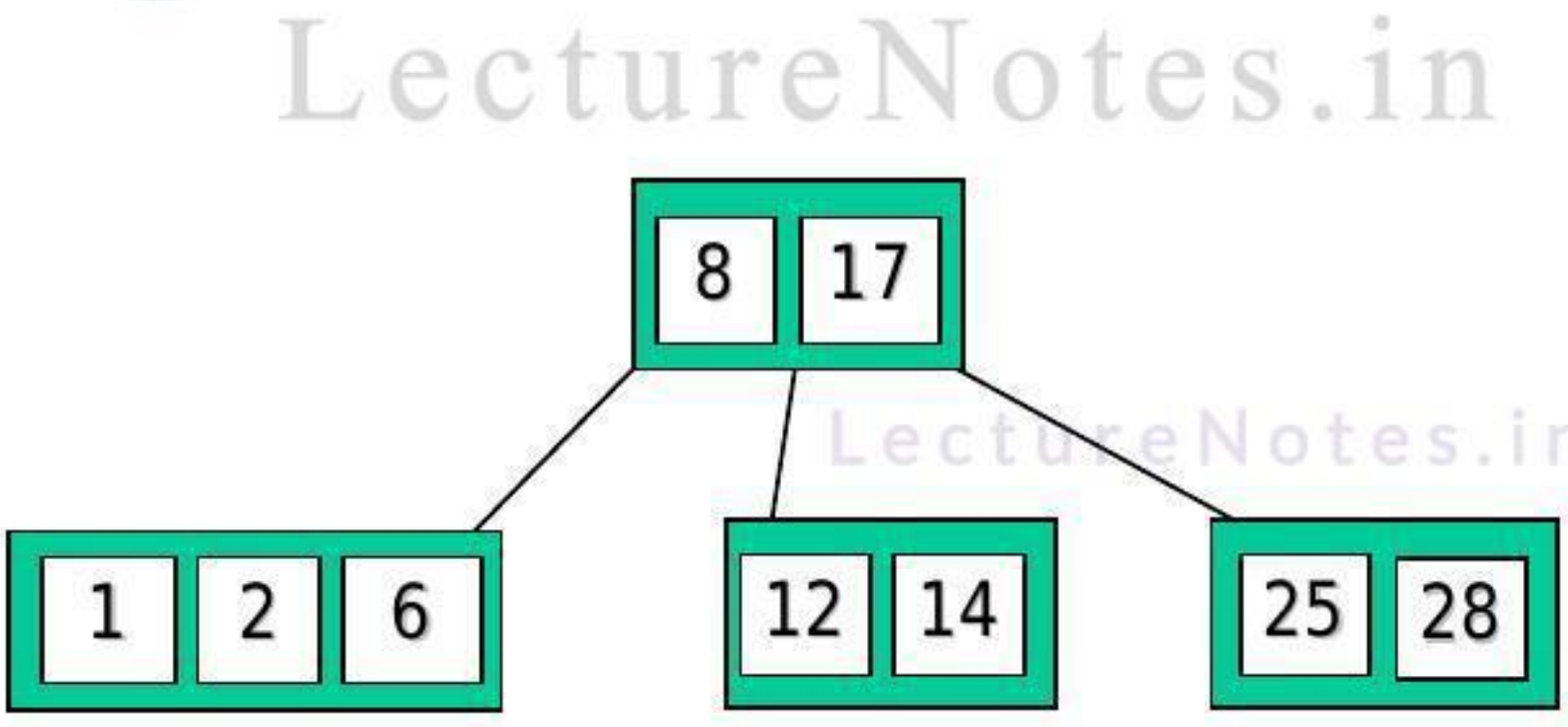


6, 14, 28 get added to the leaf nodes:

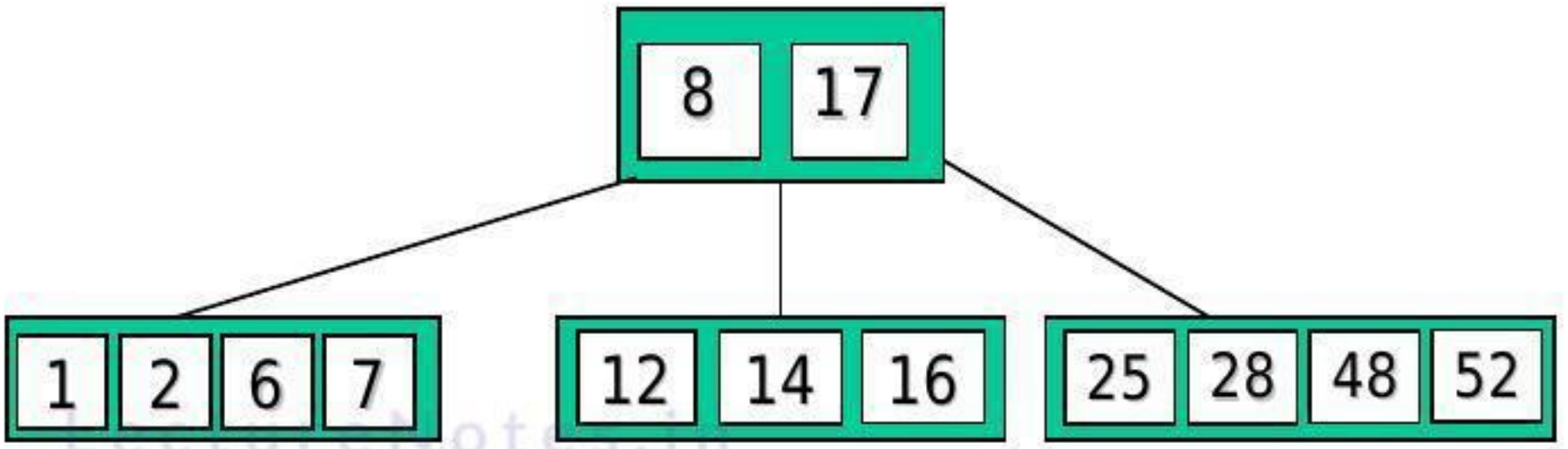


Adding 17 to the right leaf node would over-fill it, so we take the middle key, promote it (to the root) and split the leaf

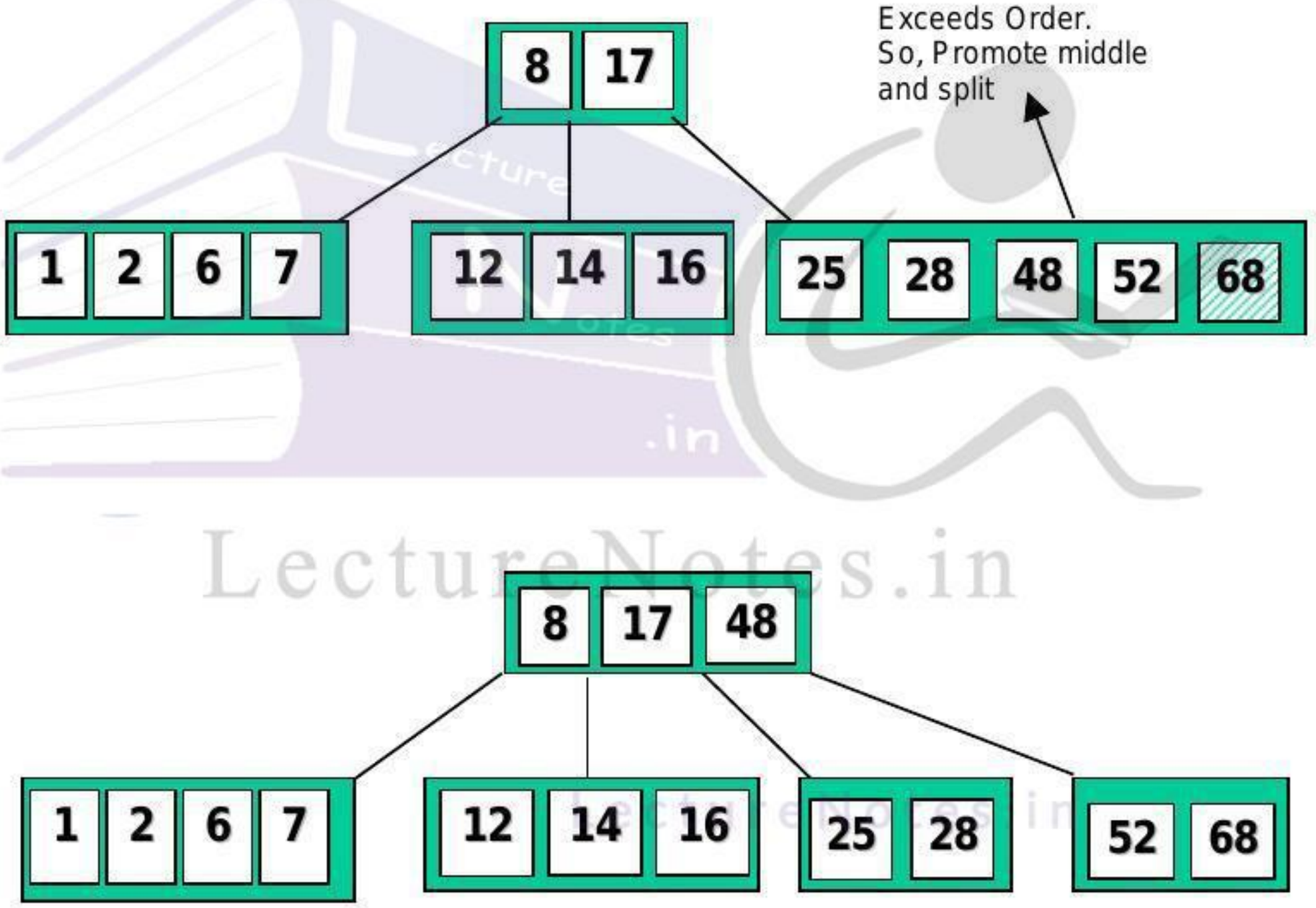




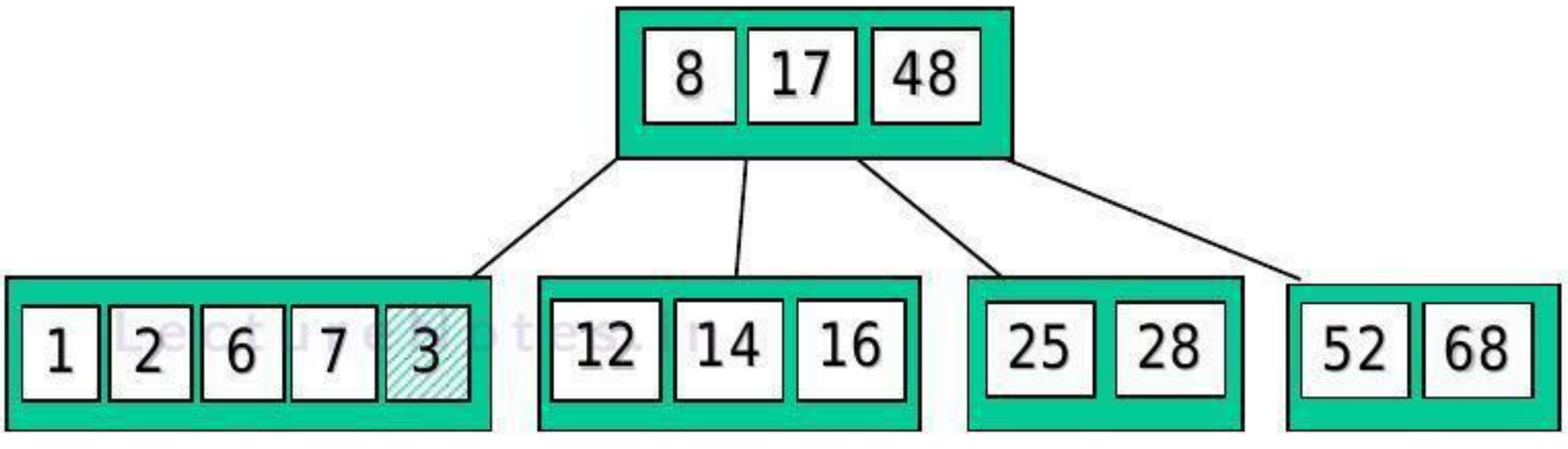
7, 52, 16, 48 get added to the leaf nodes

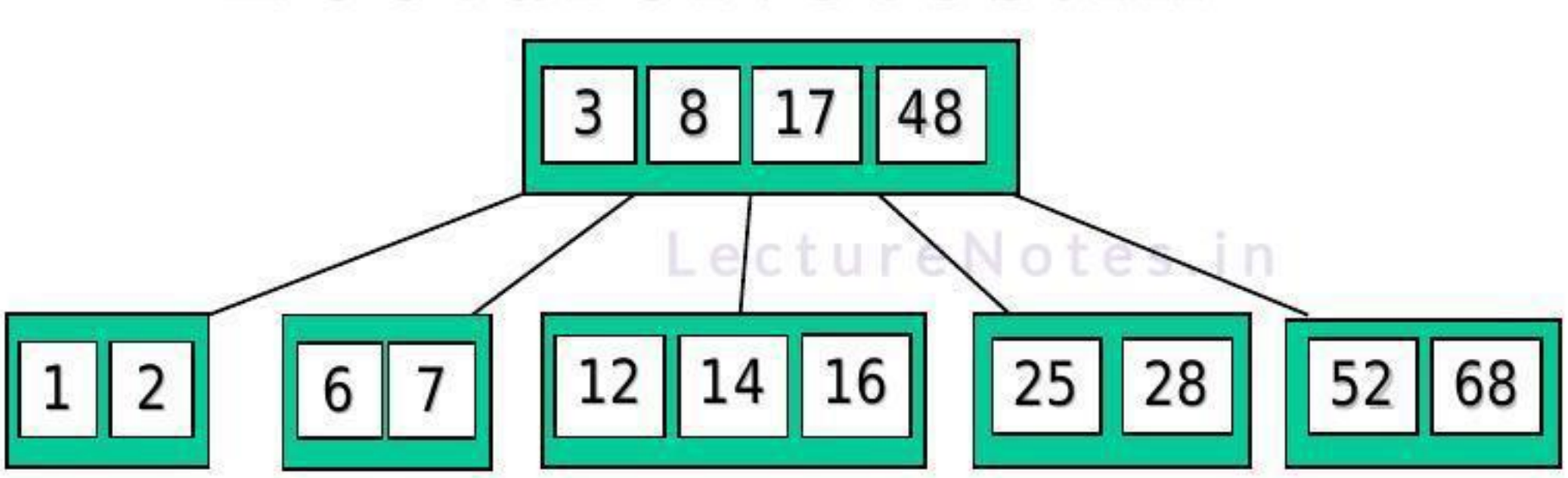


Adding 68 causes us to split the right most leaf, promoting 48 to the root

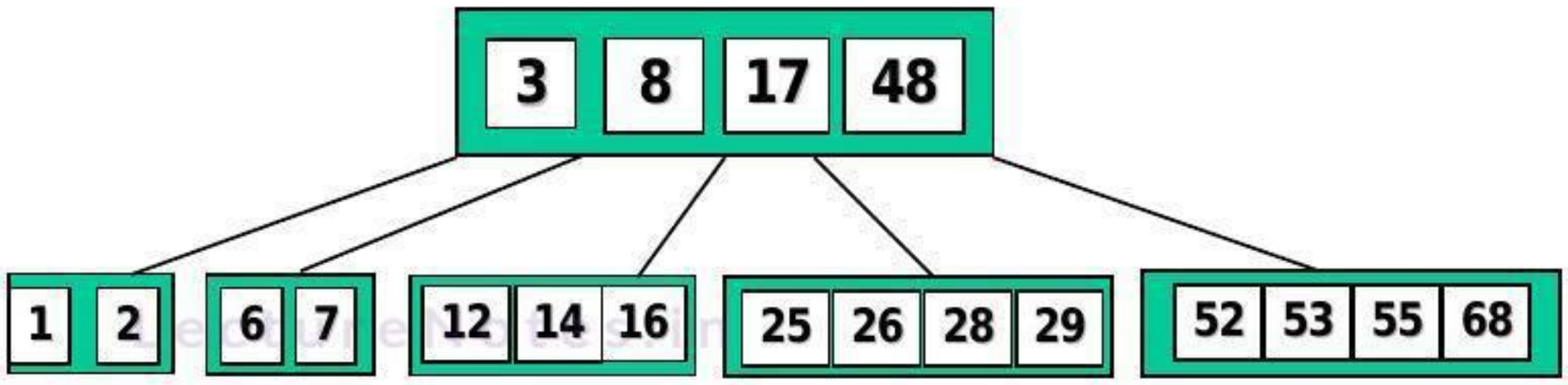


Adding 3 causes us to split the left most leaf

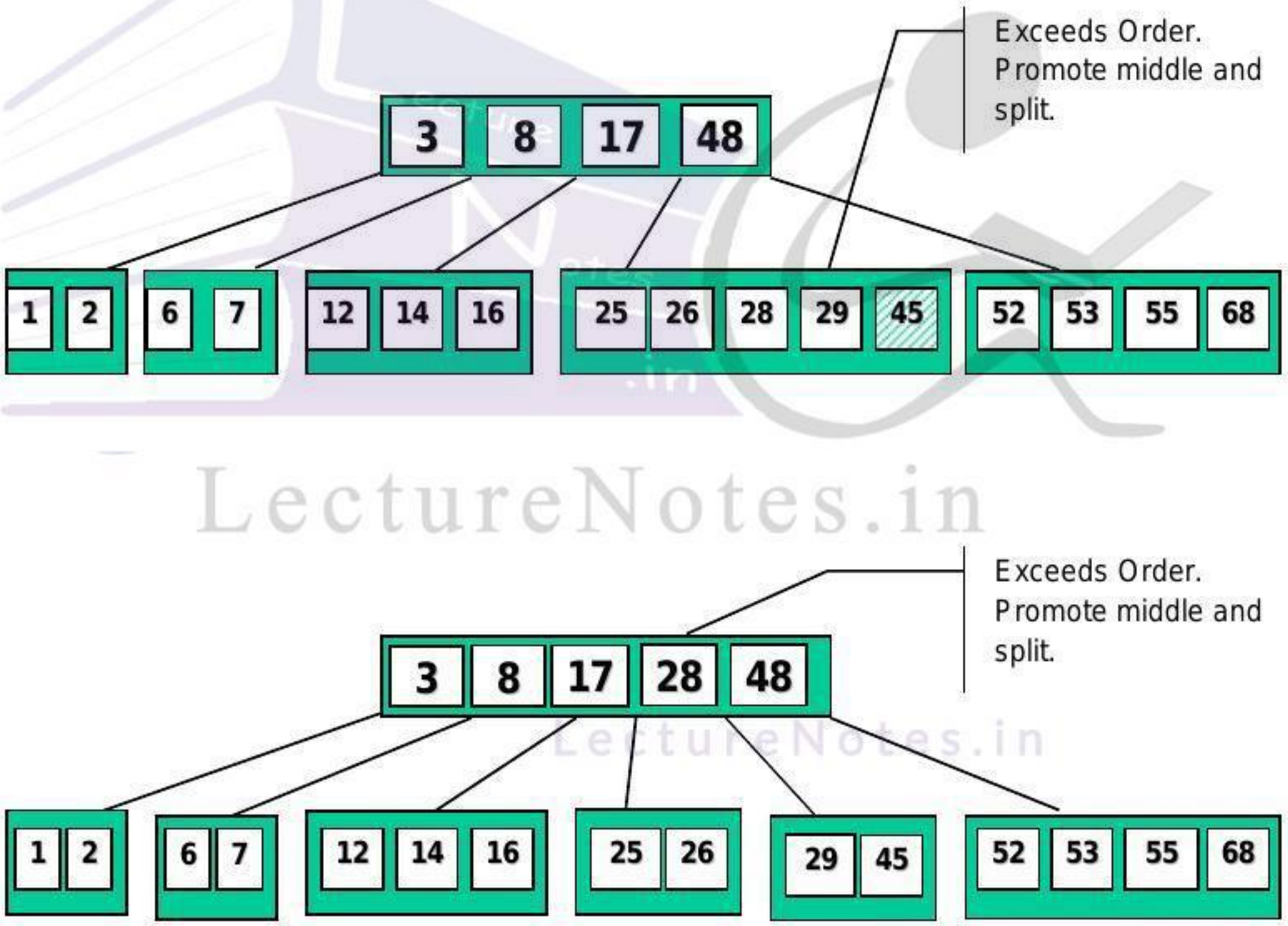


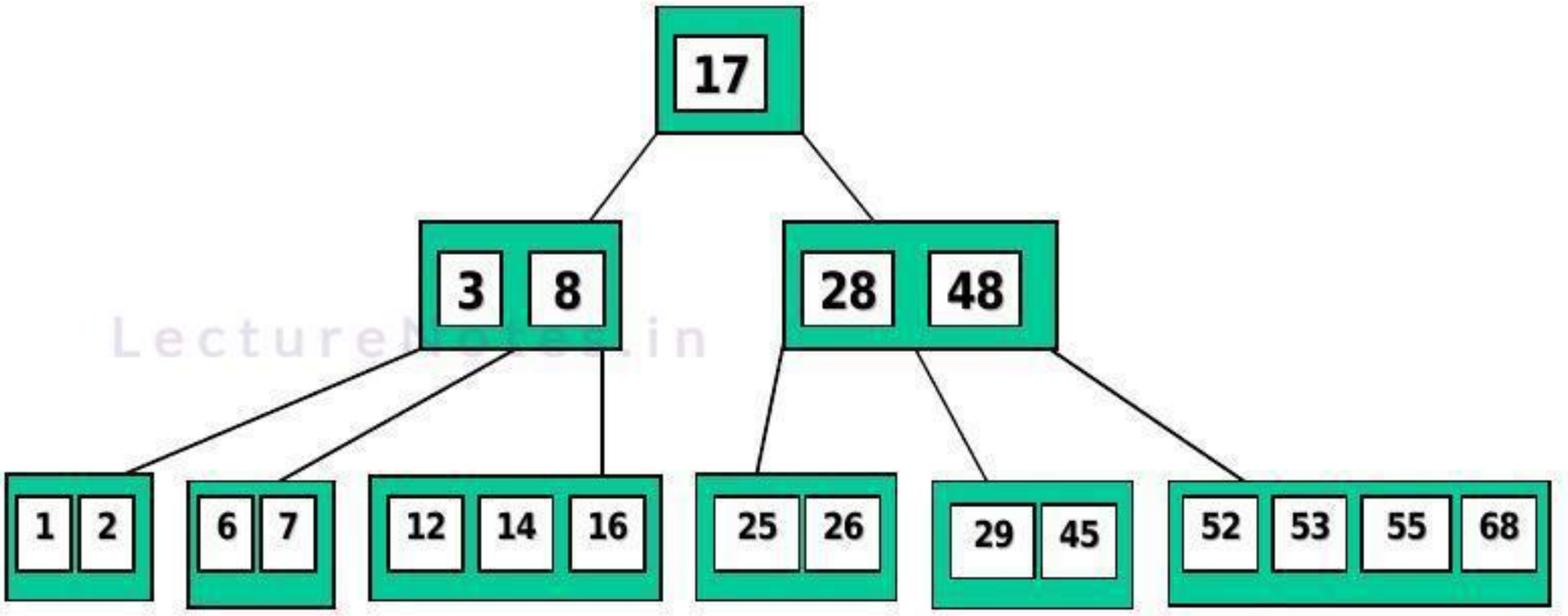


Add 26, 29, 53, 55 then go into the leaves



Add 45 increases the trees level





Notes:-

- Attempt to insert the new key into a leaf.

  i) If this would result in that leaf becoming too big, split the leaf into two, promoting the middle key to the leaf's parent

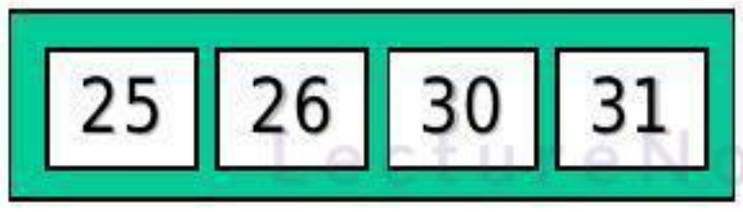
  ii) If this would result in the parent becoming too big, split the parent into two, promoting the middle key

Q:- Construct a B-tree of order 5 having following elements :

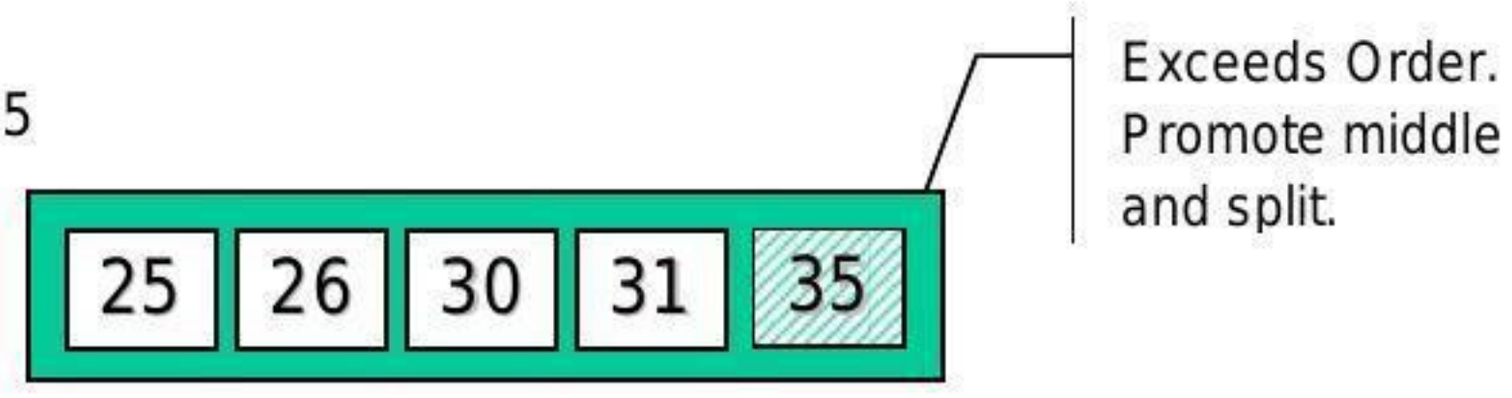
25 31 30 26 35 28 32 37 34 29 43 33 42 48 27 36 38 44 45 40

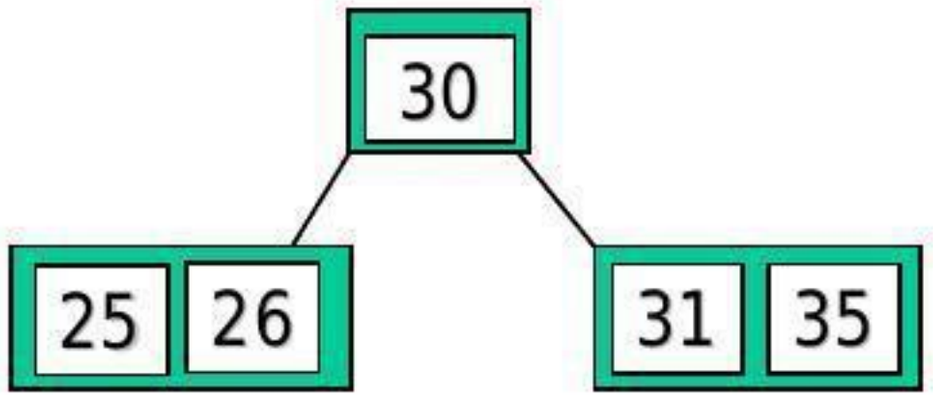
Ans:- insert 25,31,30,26

Ascending order

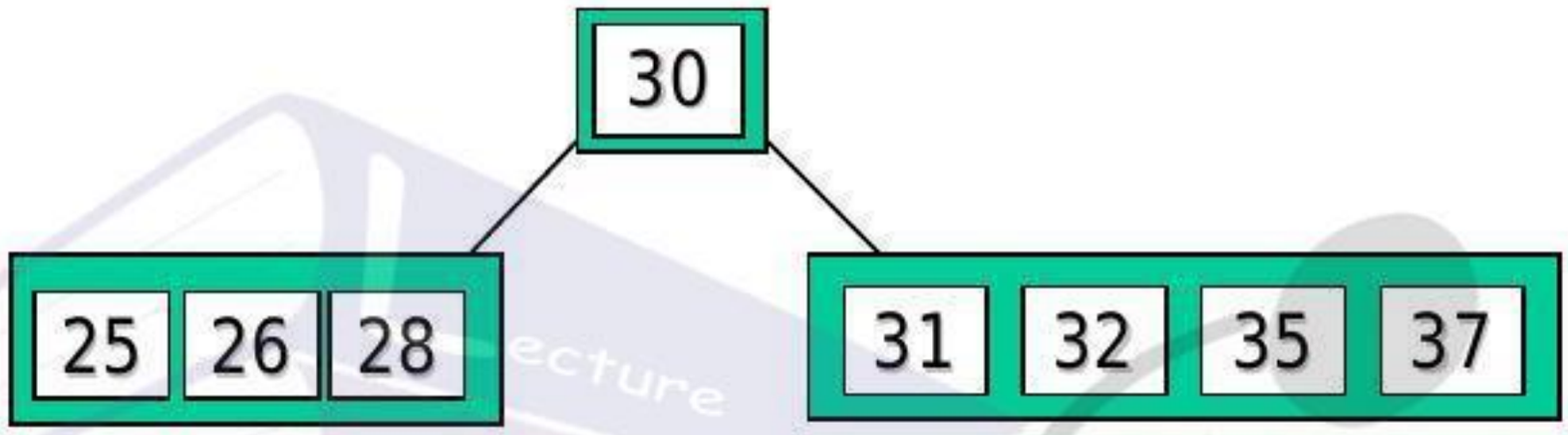


Insert 35

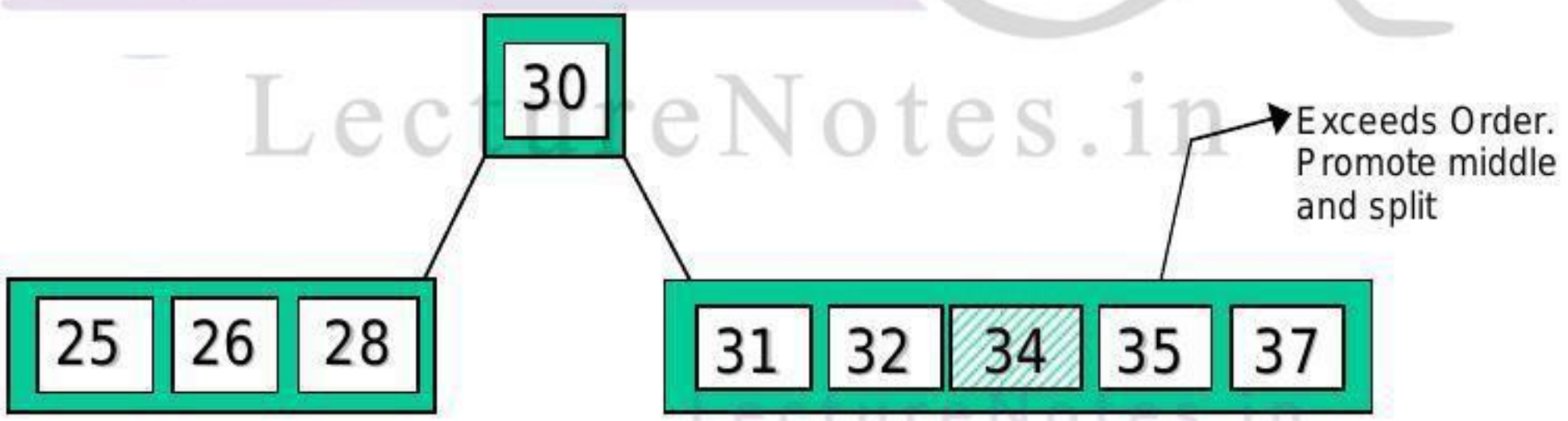


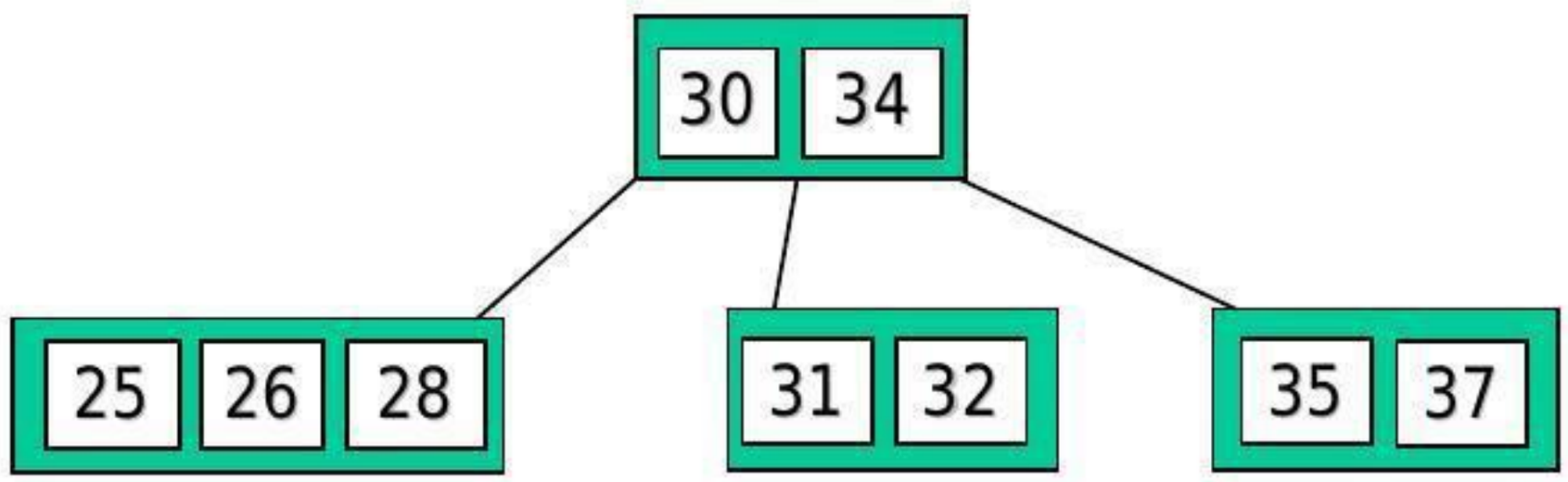


Insert 28, 32, 37:

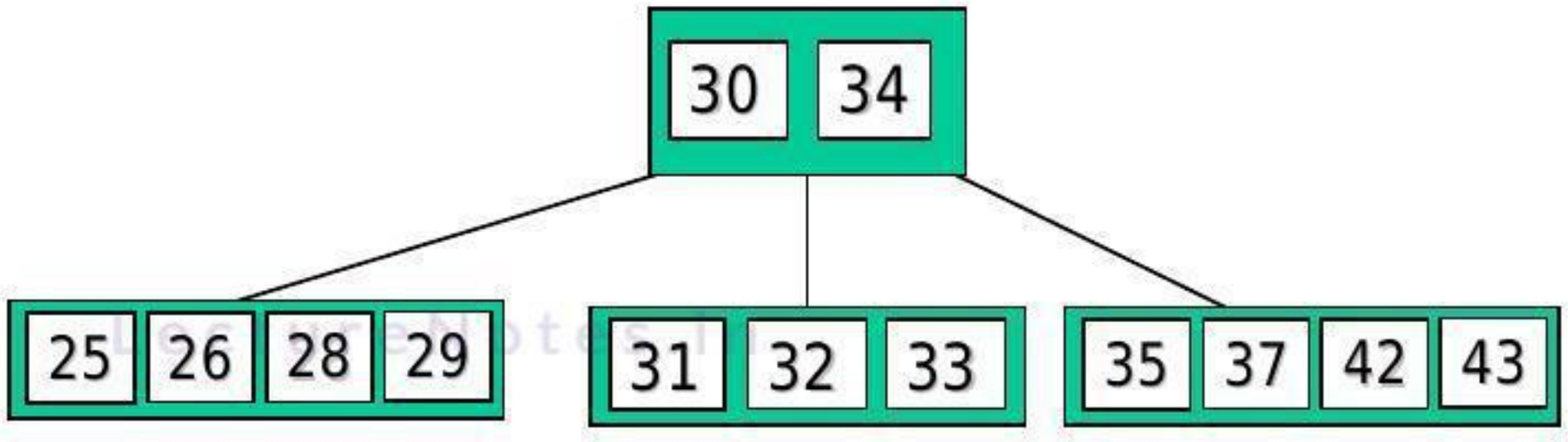


Adding 34 to the right leaf node would over-fill it, so we take the middle key, promote it (to the root) and split the leaf

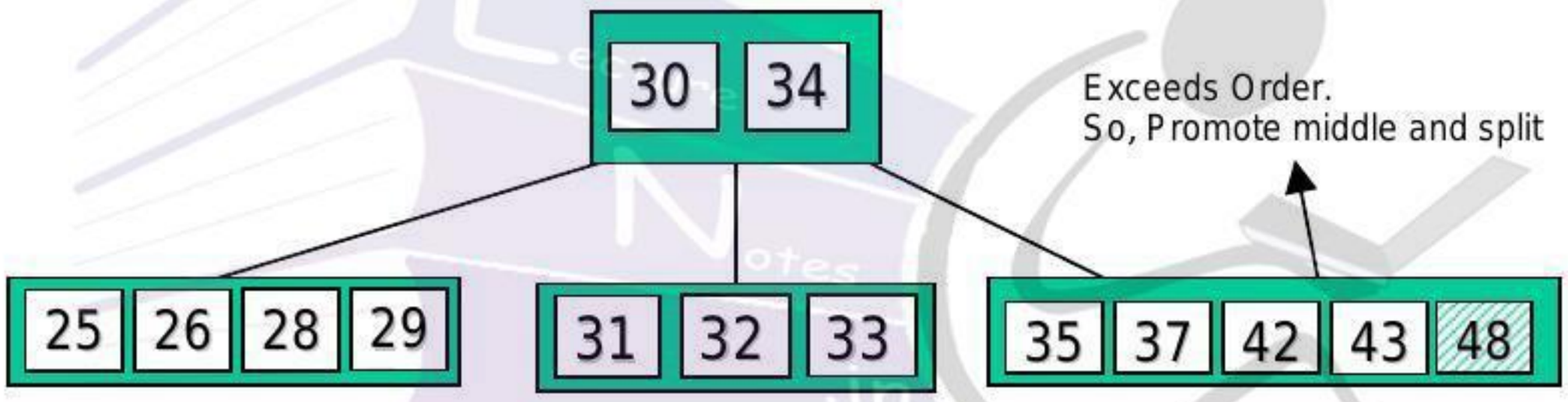


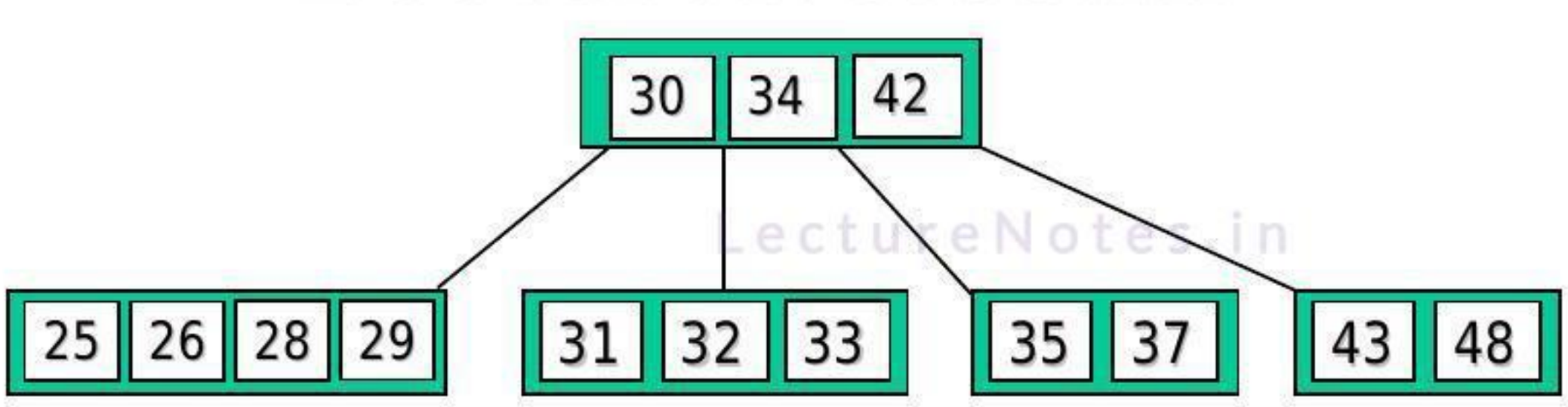


Insert 29, 43, 33, 42

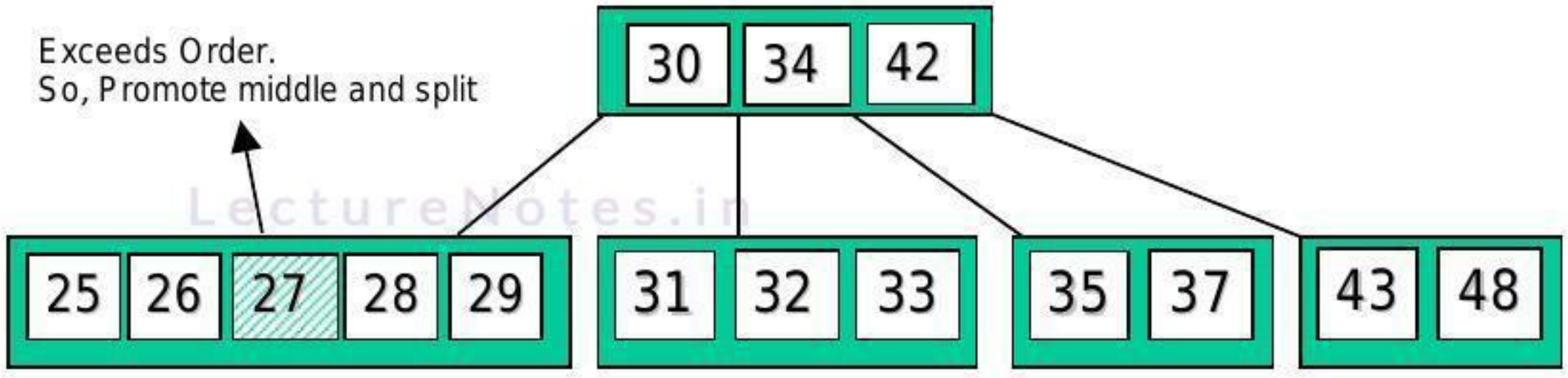


insert 48 causes us to split the right most leaf, promoting 42 to the root

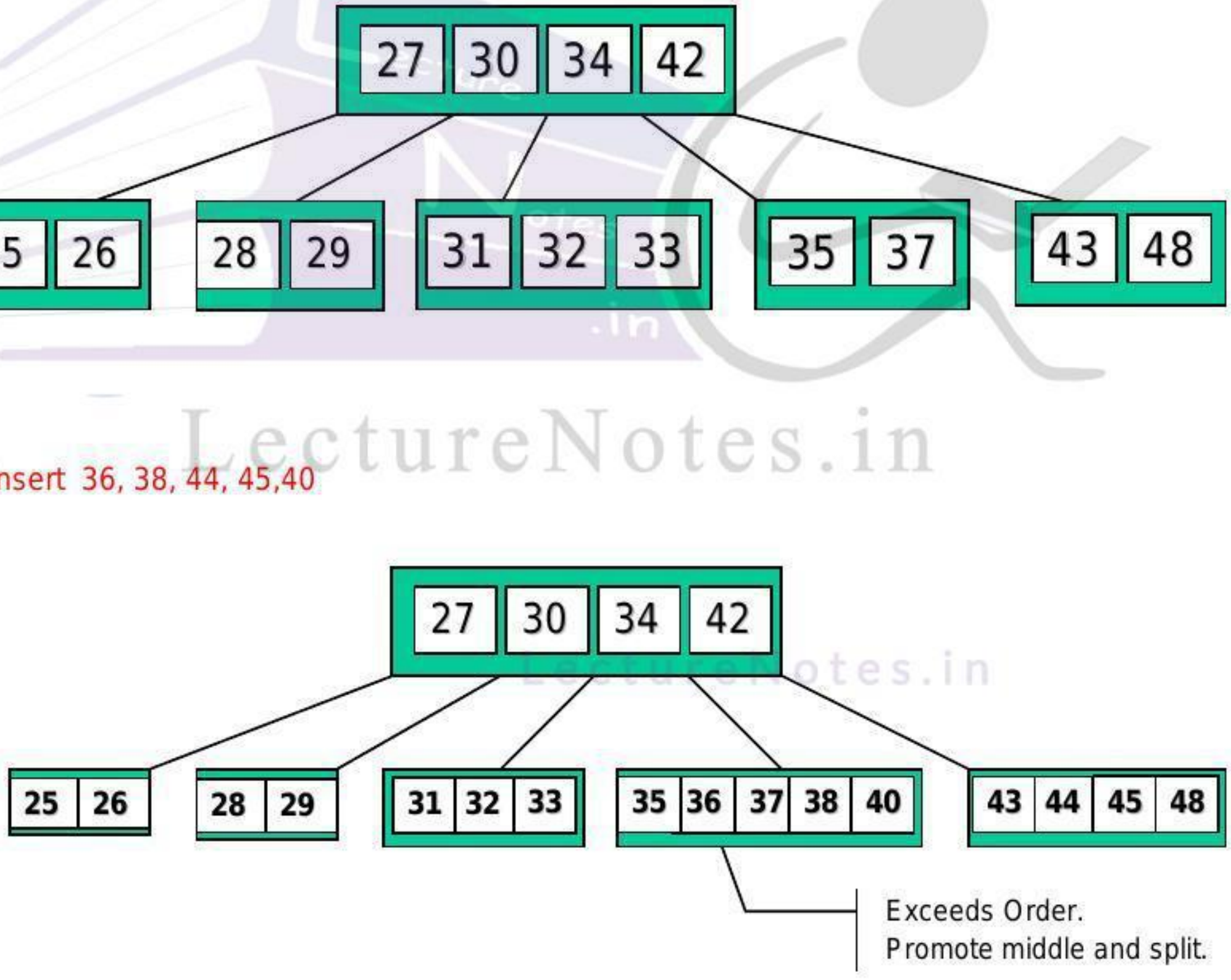


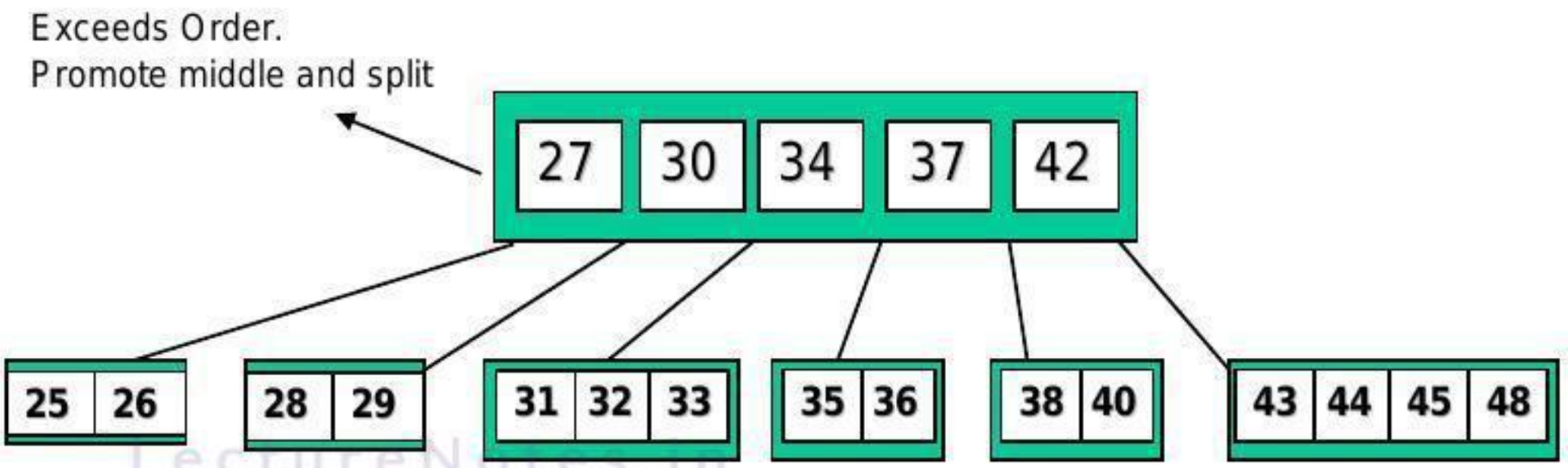


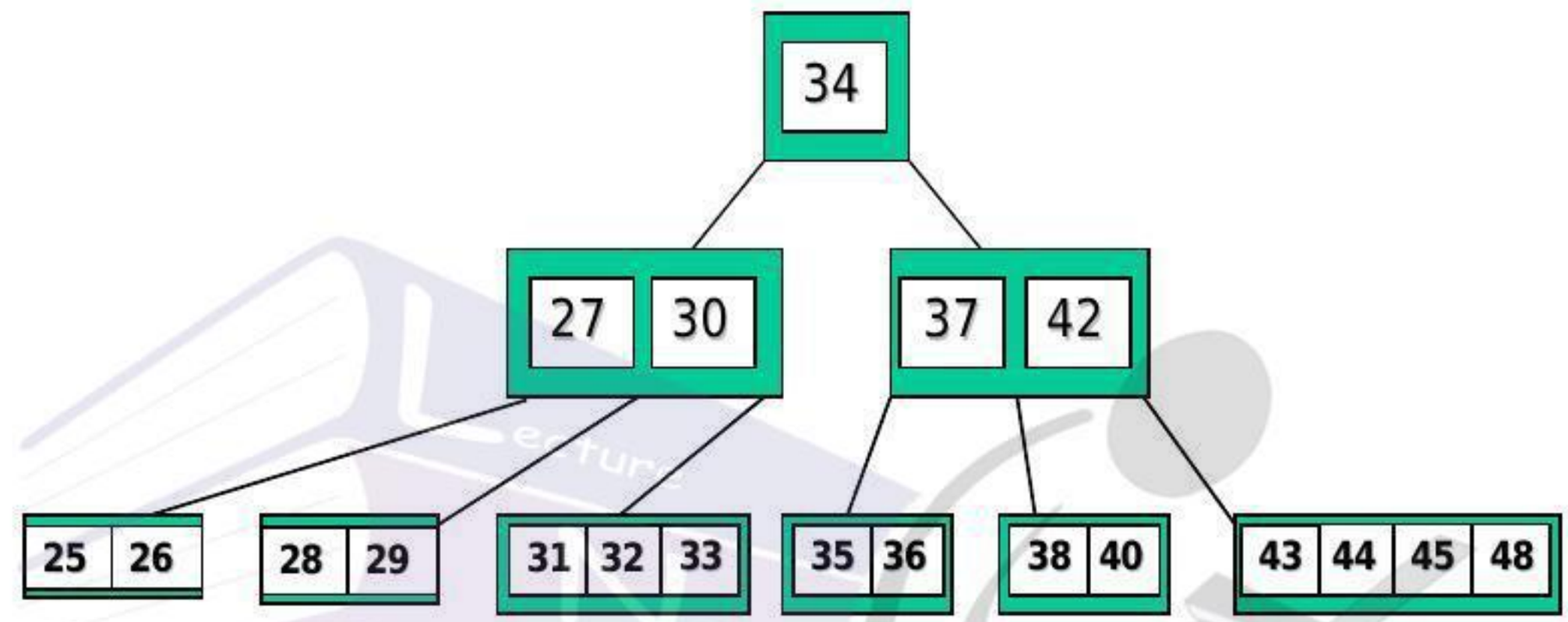
Adding 27 causes us to split the left most leaf



Insert 36, 38, 44, 45,40



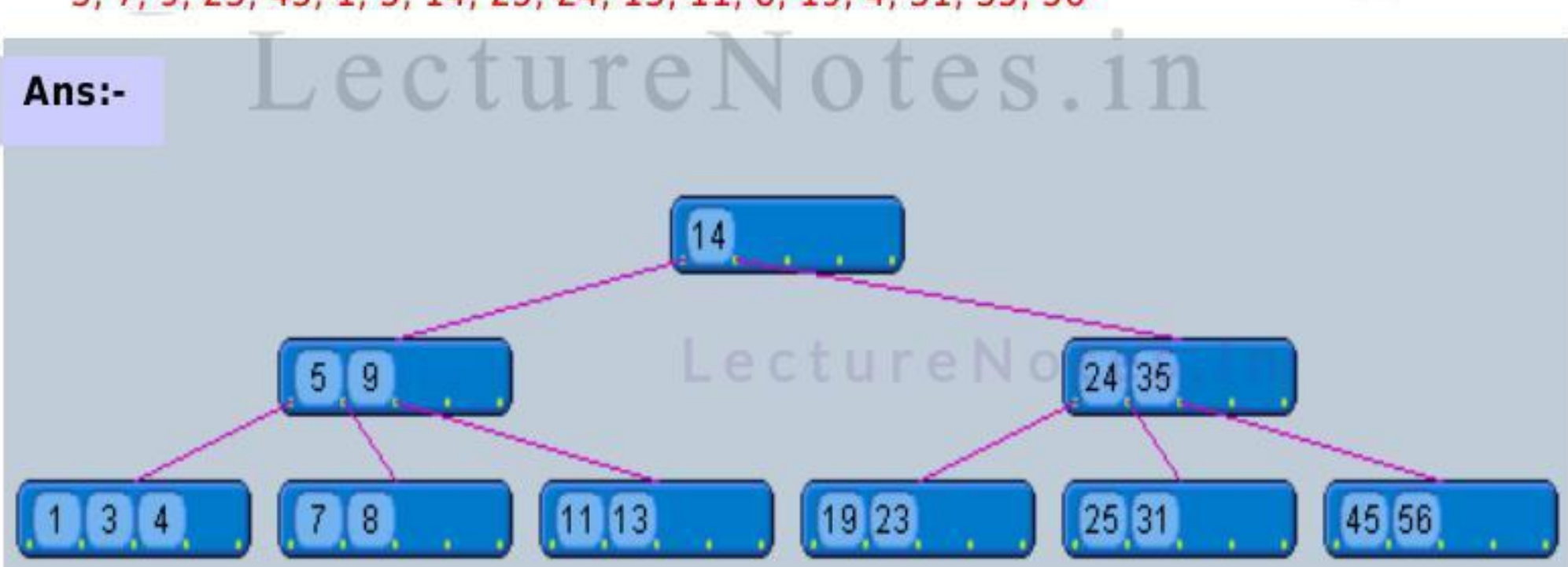




Q:- Insert the following keys to a 5-way B-tree:

3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56

Ans:-



5.6.2 Deletion from a B-tree

The key to be deleted may be present in a leaf/non-leaf node.

deletion of key from leaf node	**deletion of key from non-leaf node**
Step1:- check the following two conditions to take action.	**Step1**:- delete the key by replacing with its successor. **Step2**:- check the following two conditions to take action.

cond^{n}	**leaf node will underflow after deletion?**	**Action**
1	**NO**	1. delete the key
2	**YES**	1. delete the key and check sibling 2. **if** (sibling has > min^{m} no of keys) apply rotation **else** combine the leaf node,sibling & parent key. Continue combine until you reach a node having correct fill factor or root .

There are four possible cases

case1 - If the key is present in a leaf node, then simply remove the key.

case2 - If the key is present in a non-leaf node then we delete the key by replace/promoting the predecessor or successor key to the deleted key's position.

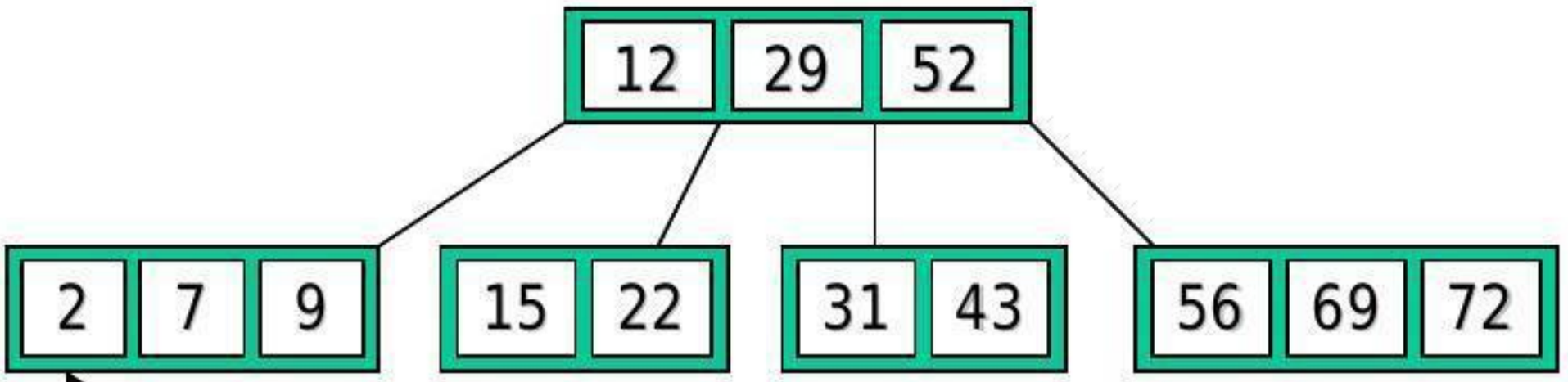
[successor is the next higher valued key. In the above Btree successor of 20 is 73. if a key is present in non-leaf node then its predecessor or successor must be present in a leaf node]

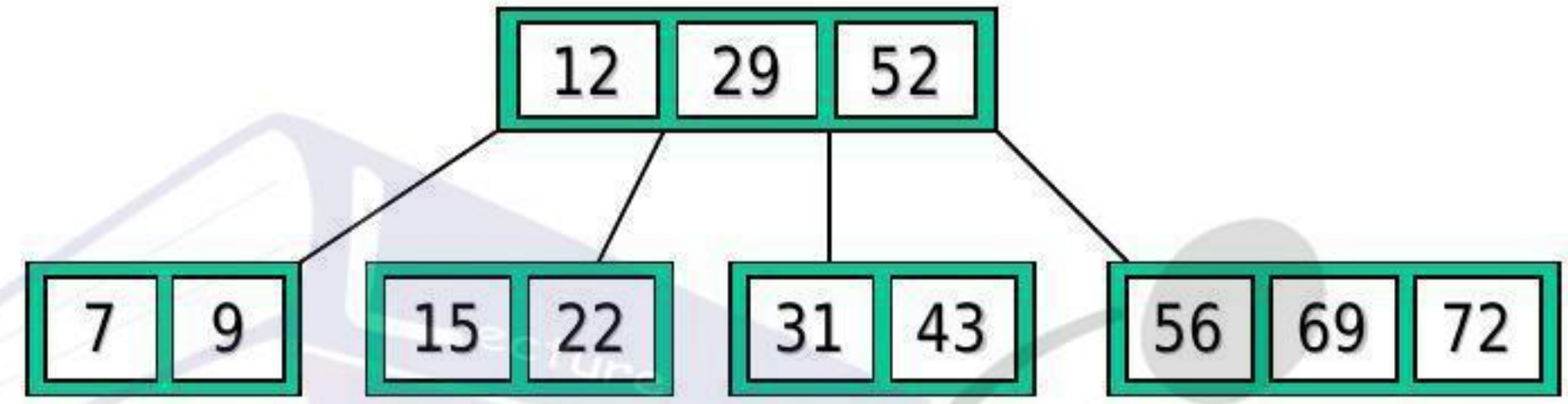
- If case (1) or (2) lead to a leaf node containing less than the minimum number of keys then we check the sibling of leaf node(left or right sibling)

case3 - if one of sibling has more than the min^{m} number of keys then we promote one of the keys of sibling into its parent node and demote (move) the key of parent node into our lacking leaf (this process is known as rotation)

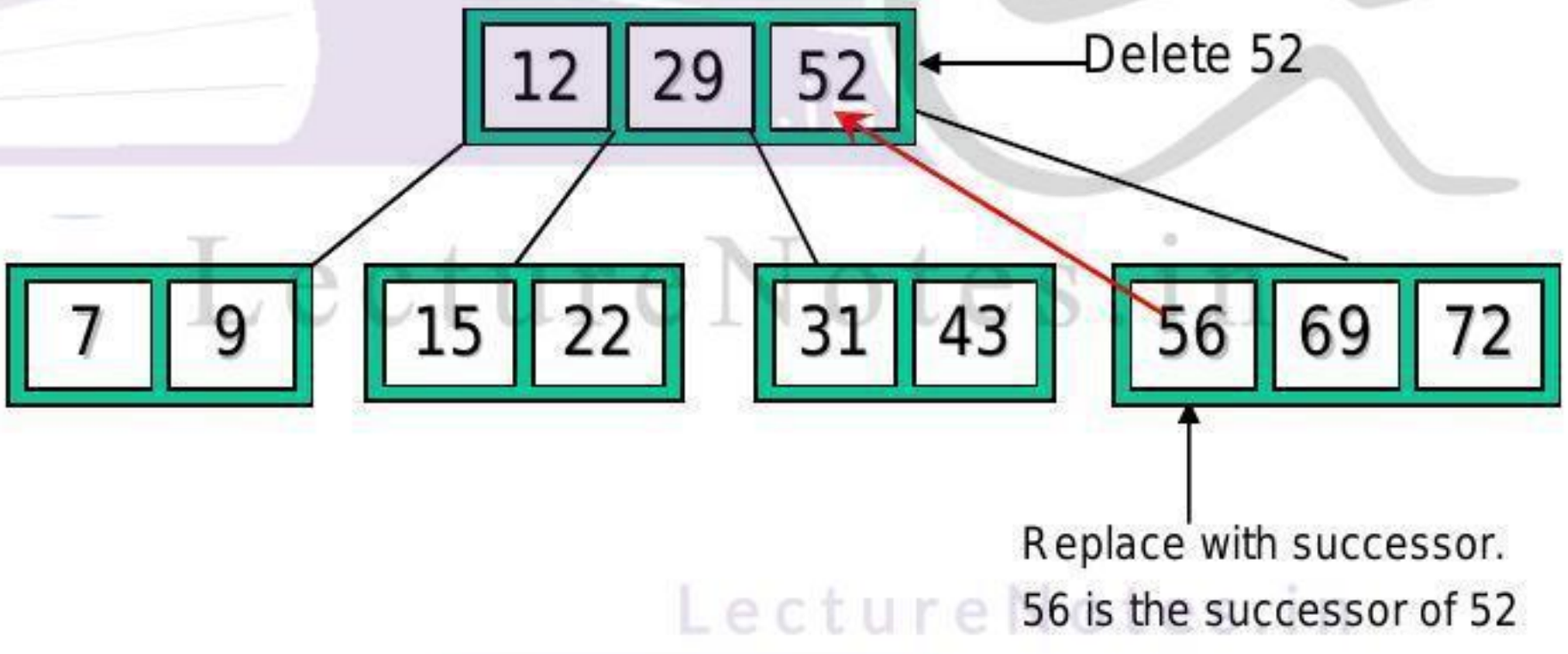
case4 - if neither of the sibling has more than the min^{m} number of keys then the leaf node and one of its sibling can be combined with parent key; if this step leave the parent node with too few keys then we repeat the process up to the root itself, if required.

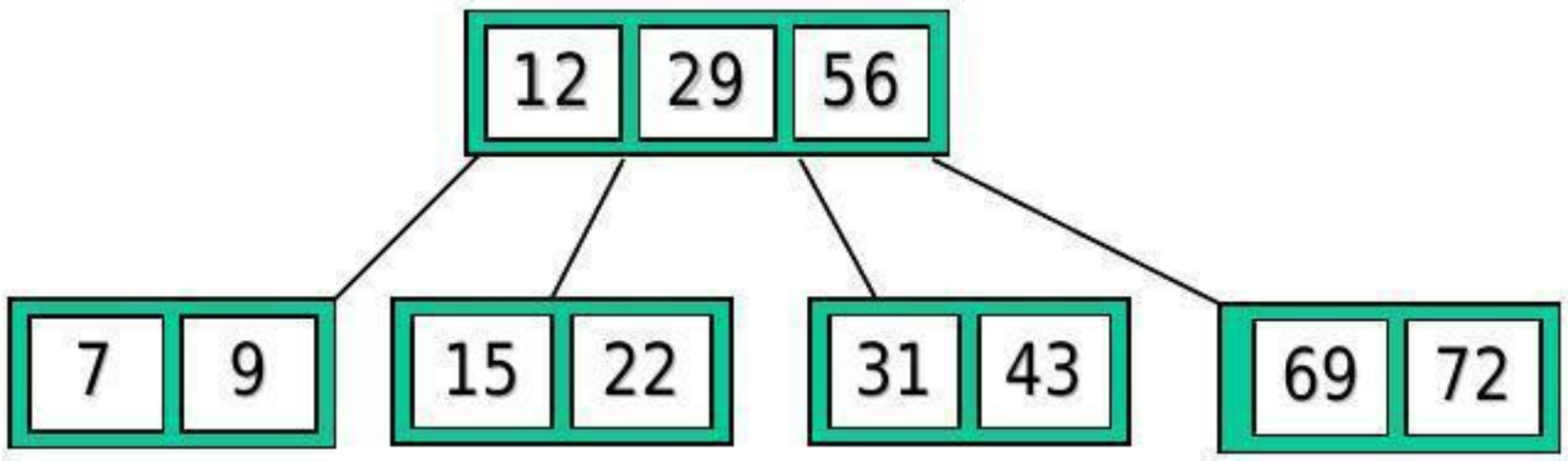
Case 1: Simple leaf deletion :- consider a b-tree of order 5 shown below





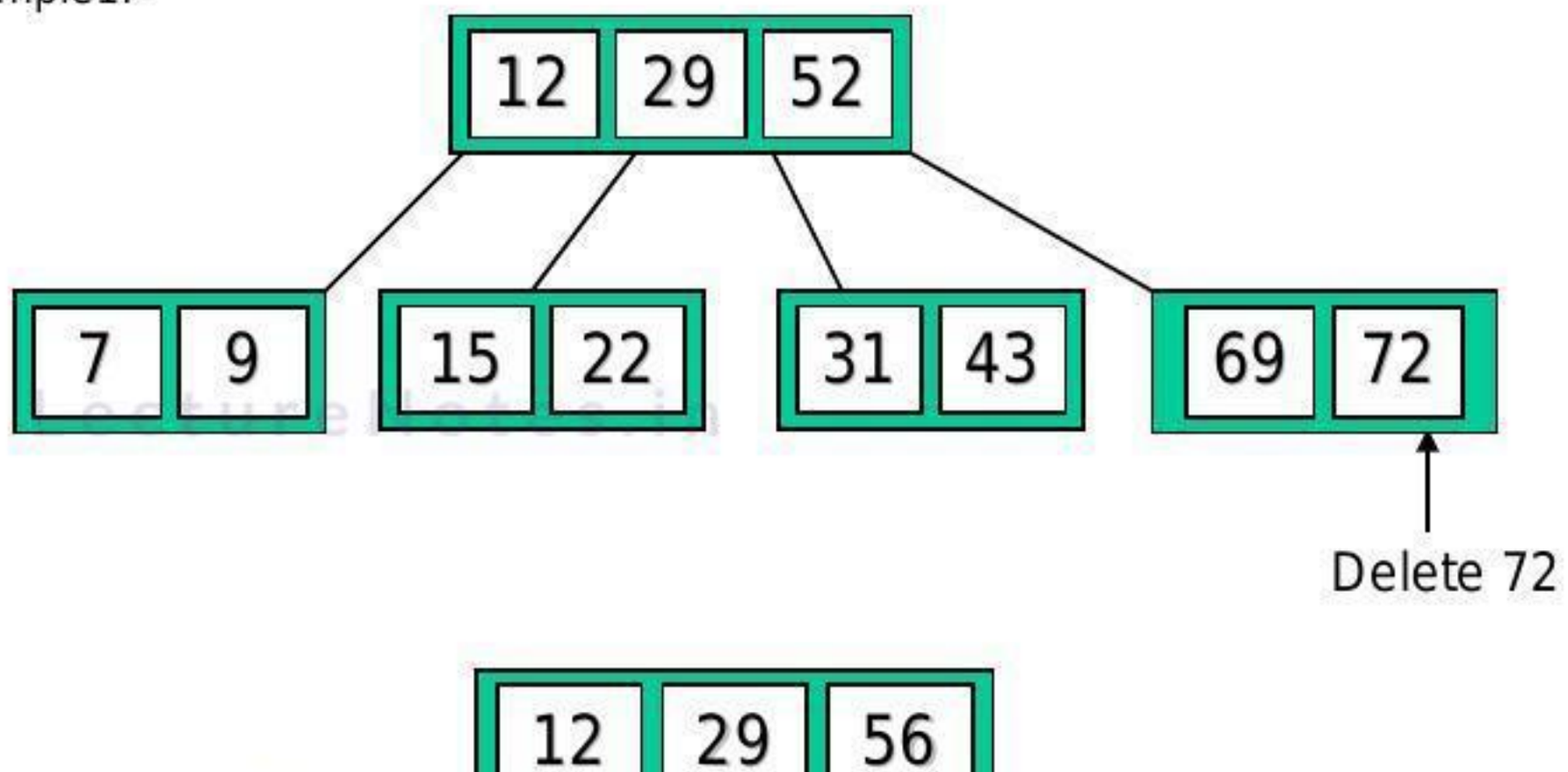
Case 2: Simple non-leaf deletion :-

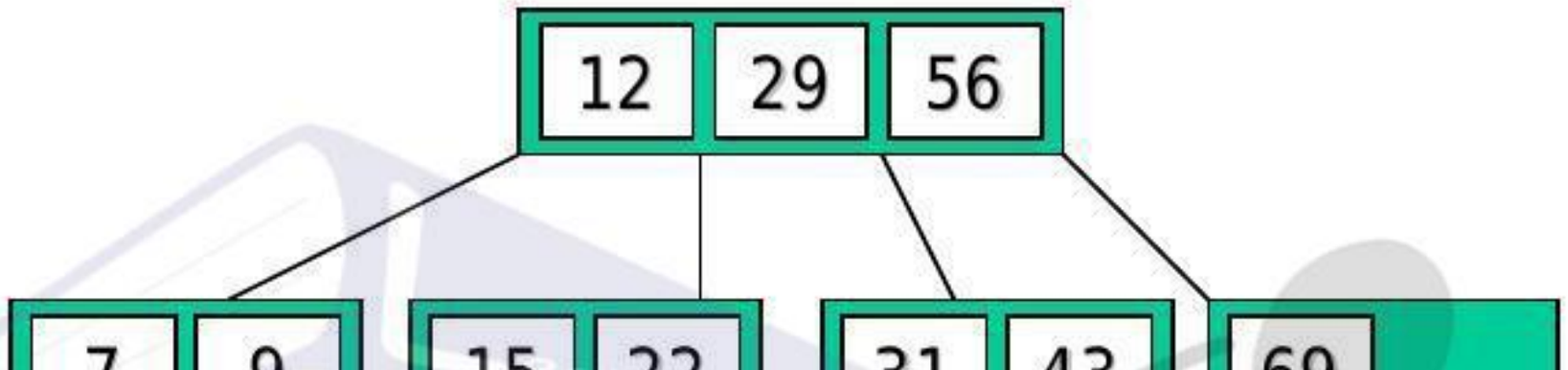


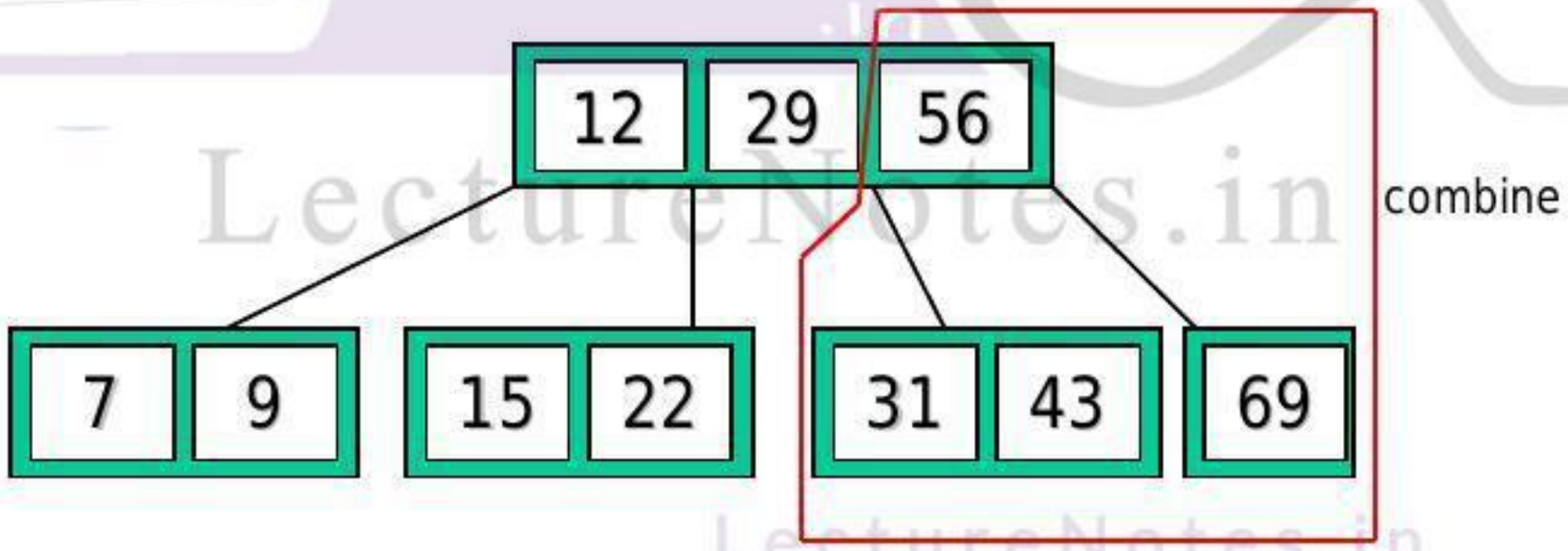


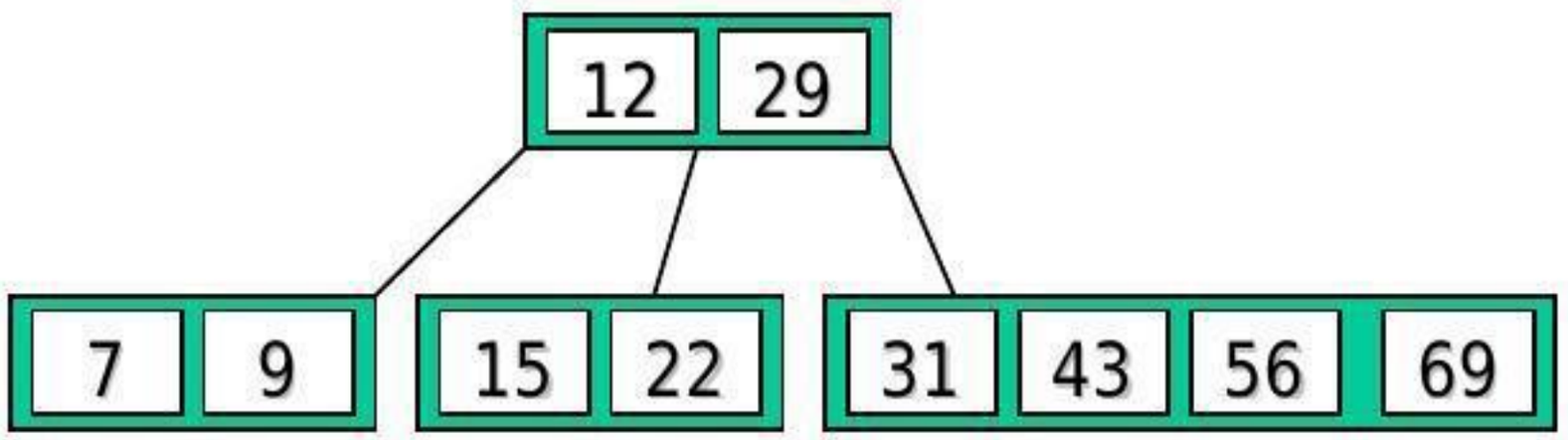
Case 4: Too few keys in node and its siblings :-

Example1:-

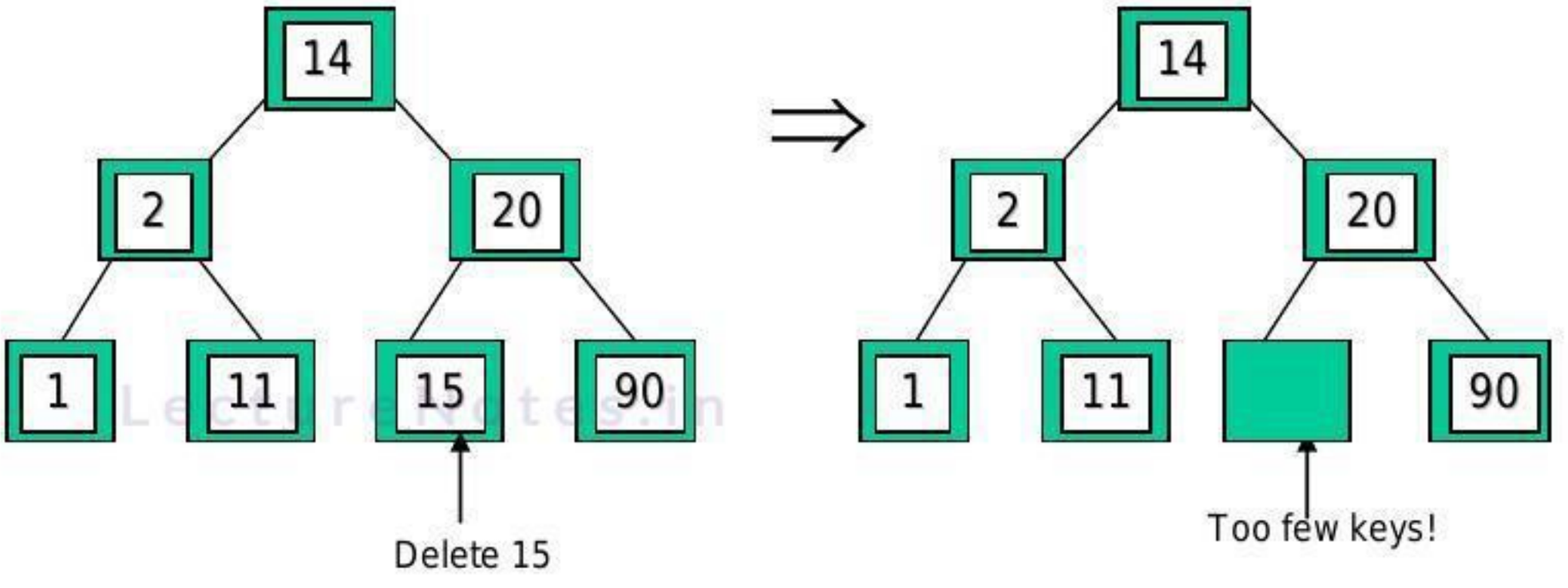


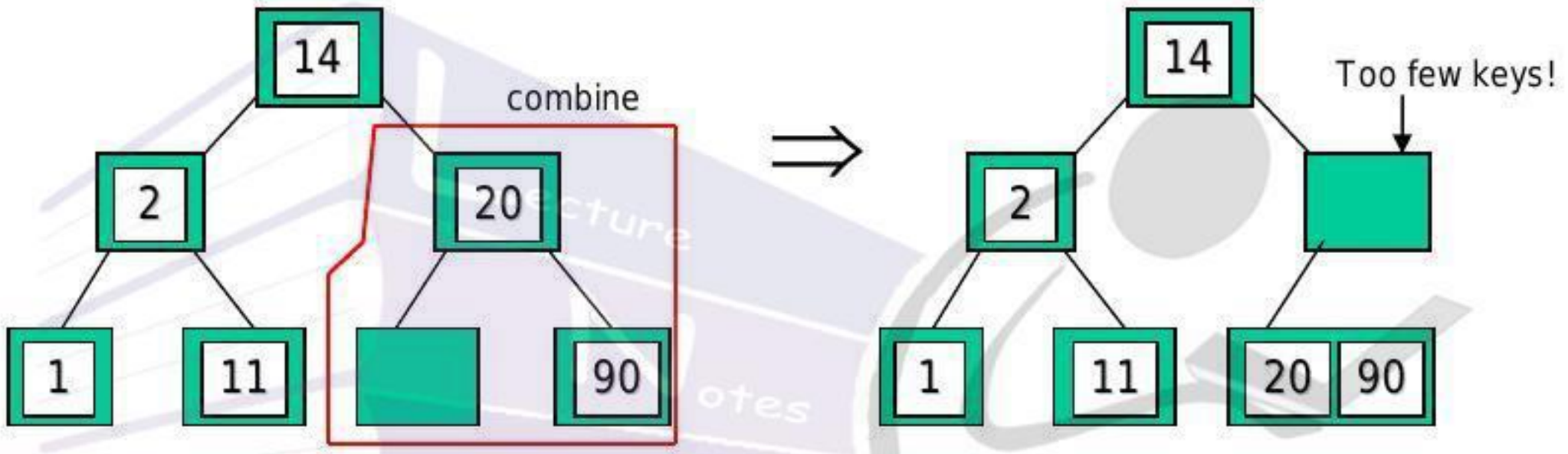


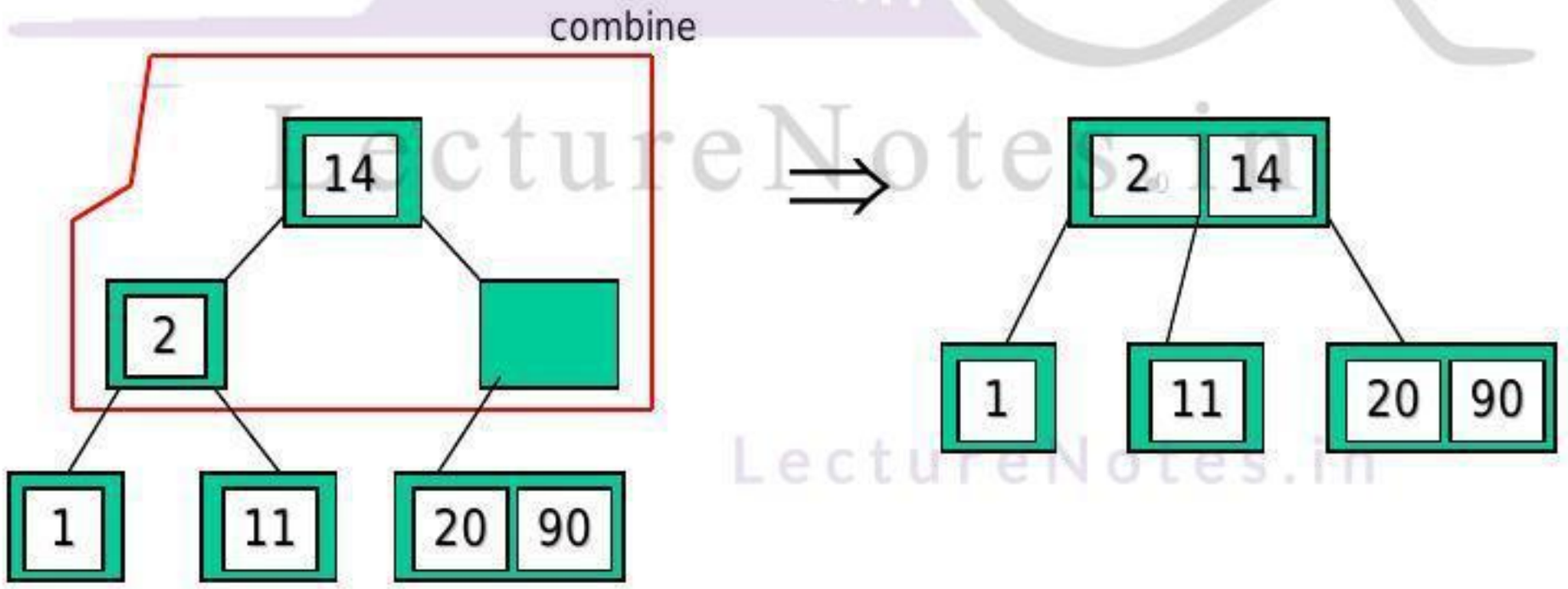




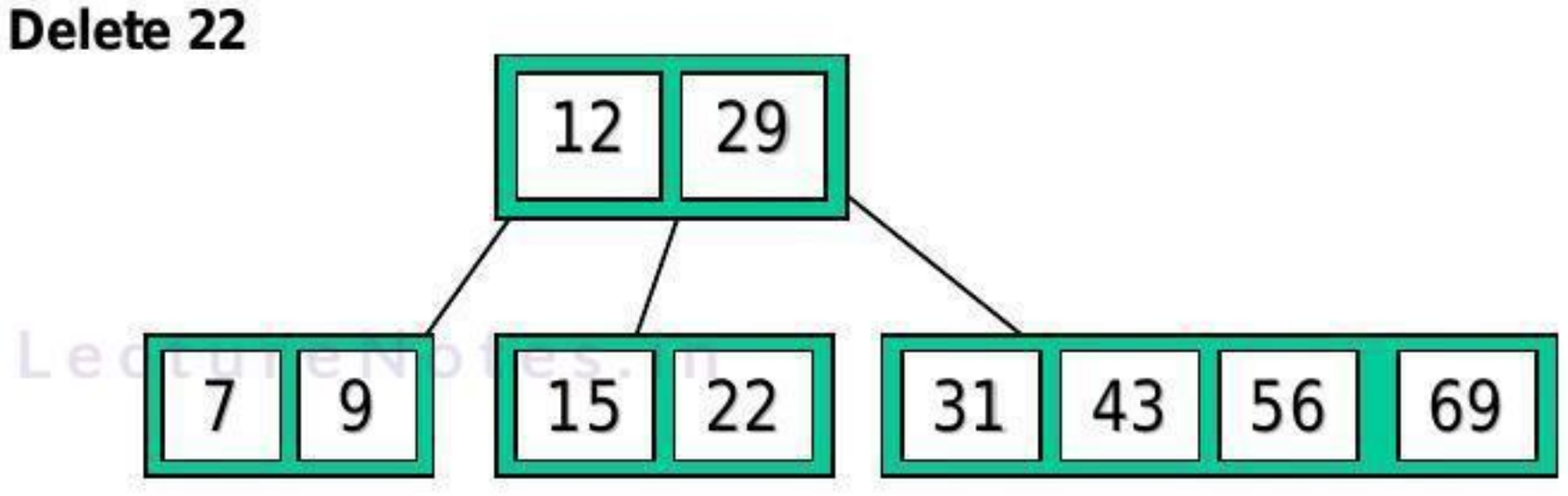
Example2:-

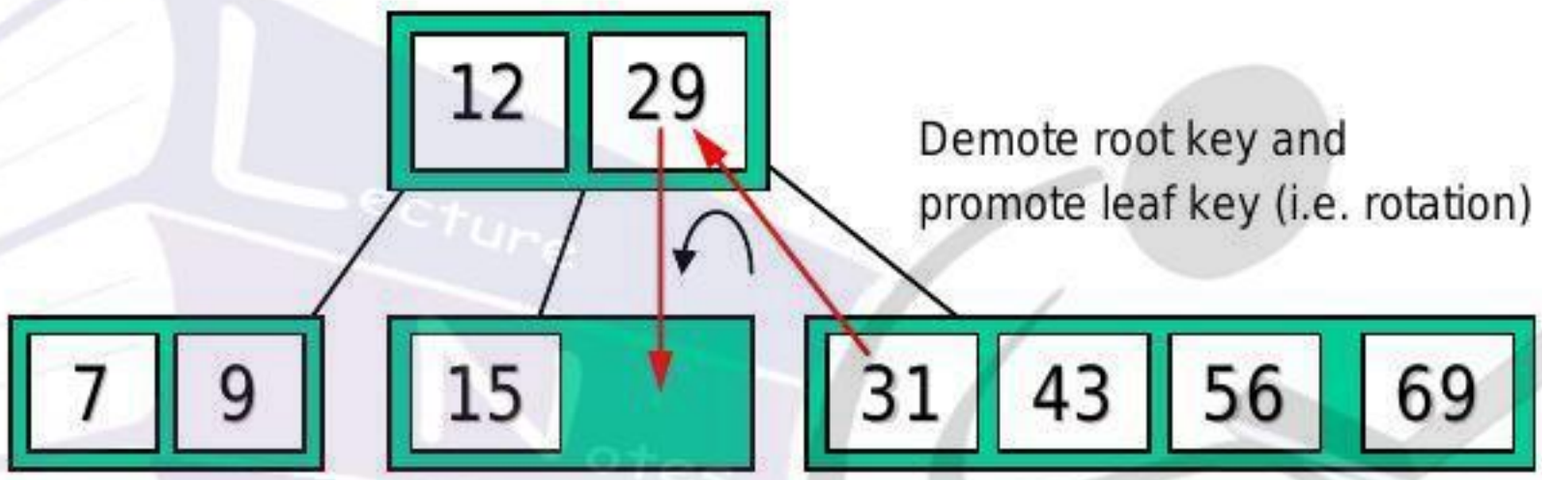


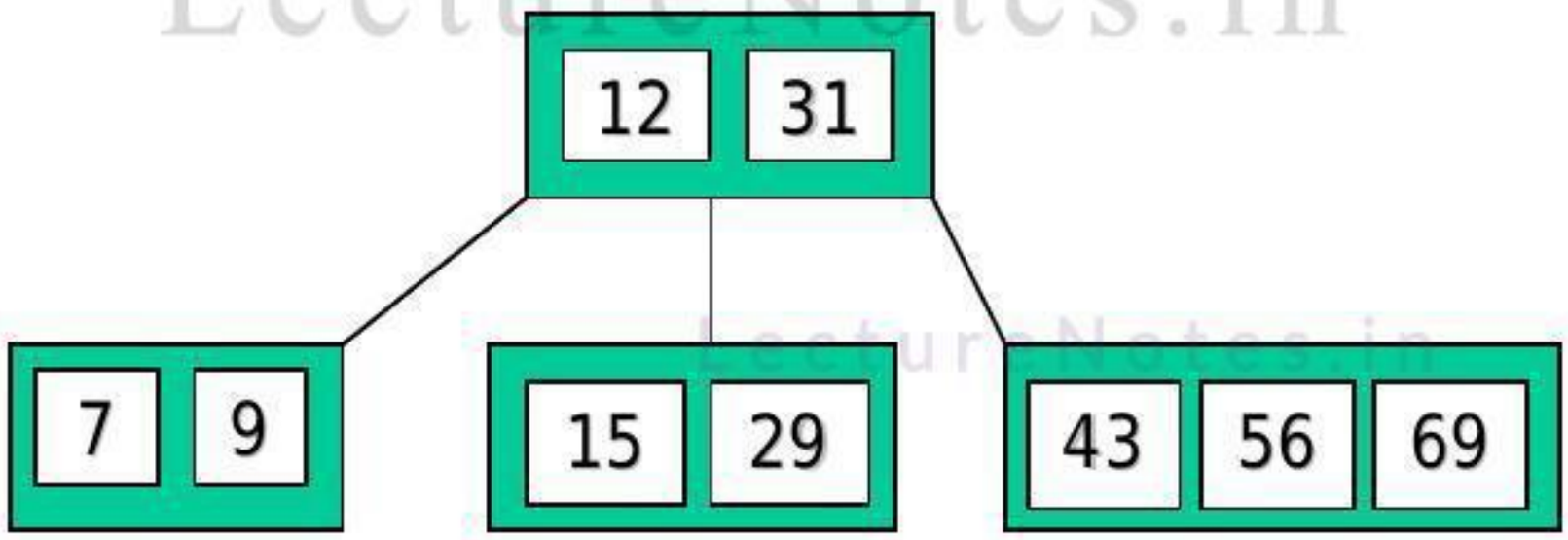




Case 3: Enough siblings :- rotation is done here







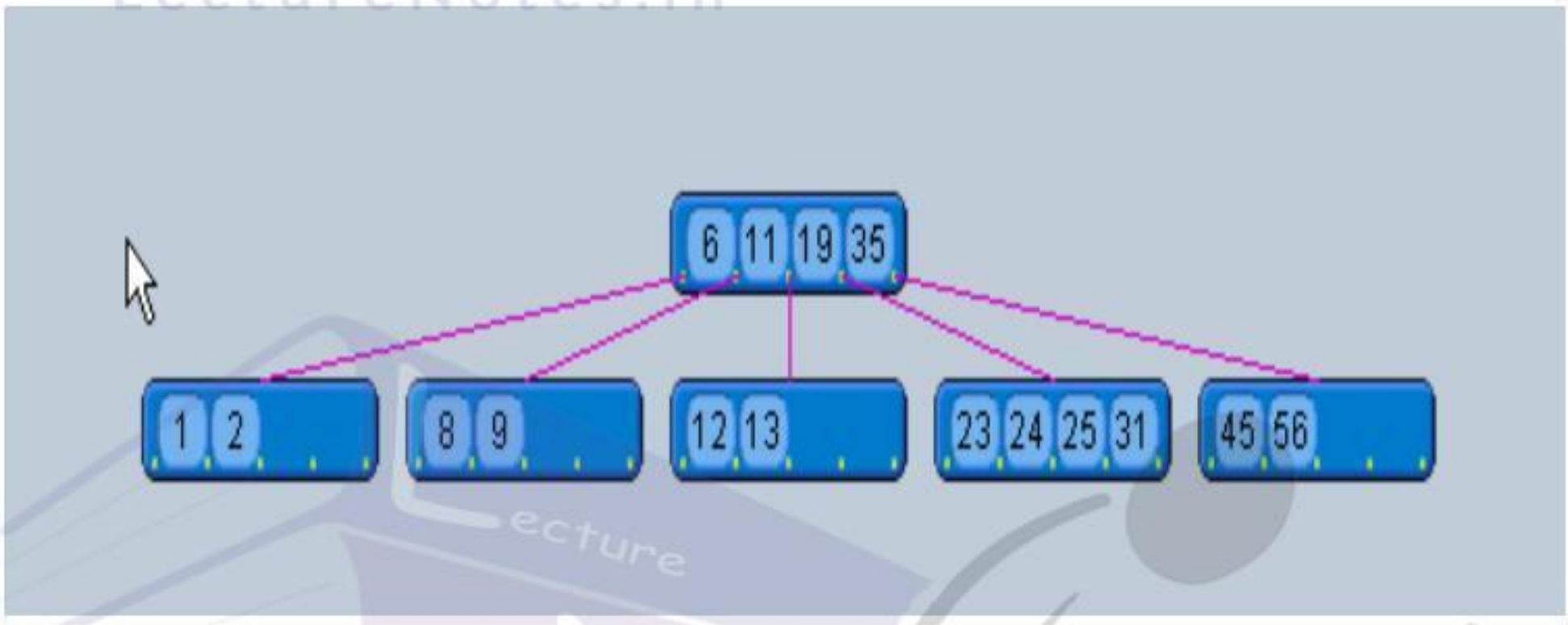
Rotation is done from the sibling which contains enough keys (more than min^m keys). Rotation can be done from right/left sibling.

Q:- Given 5-way B-tree created by these data (last question):

3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56

- Add these further keys: 2, 6,12
- Delete these keys: 4, 5, 7, 3, 14

Ans:-



5.7 B$^+$ Tree

B$^+$ tree is a B tree where

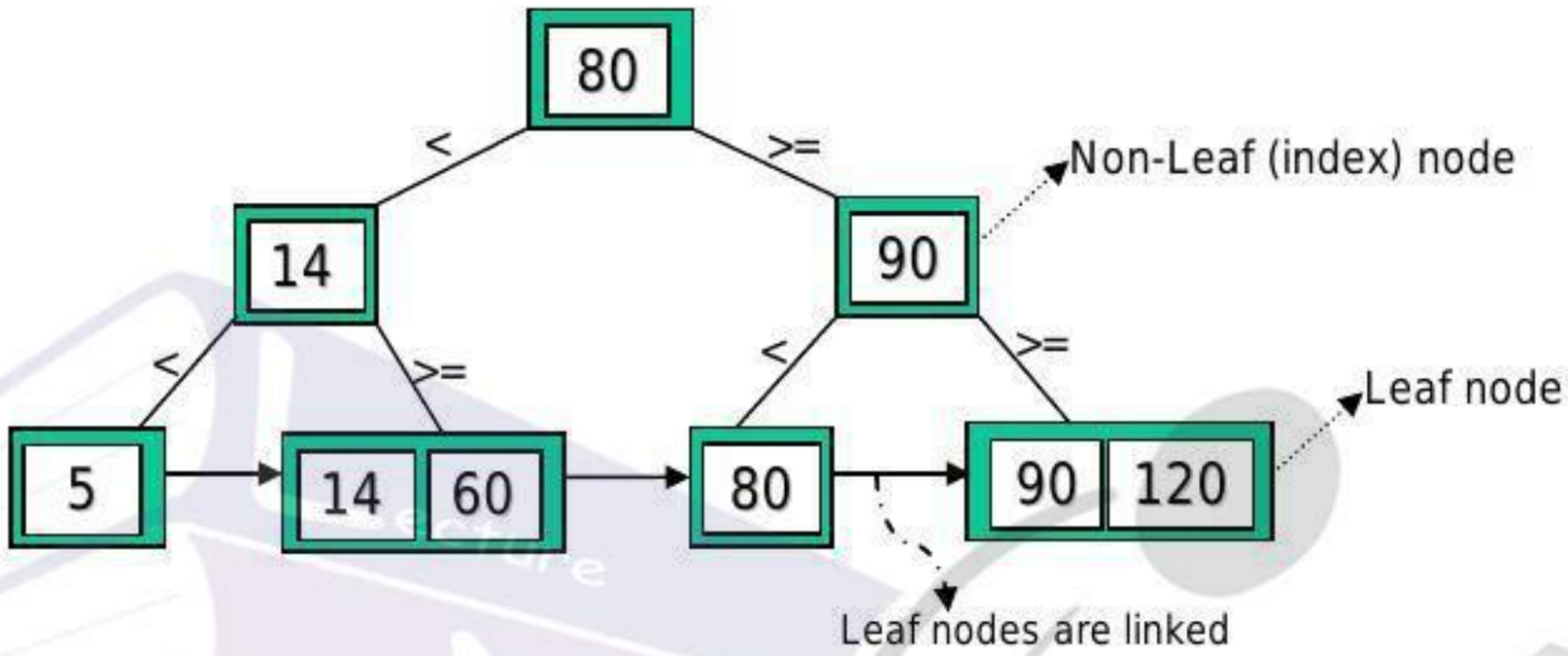
i) All the key values must be at the leaf node.

ii) each leaf node points to the next leaf-node.

In B$^+$ tree all the leaf nodes are linked (i.e. sequential access)

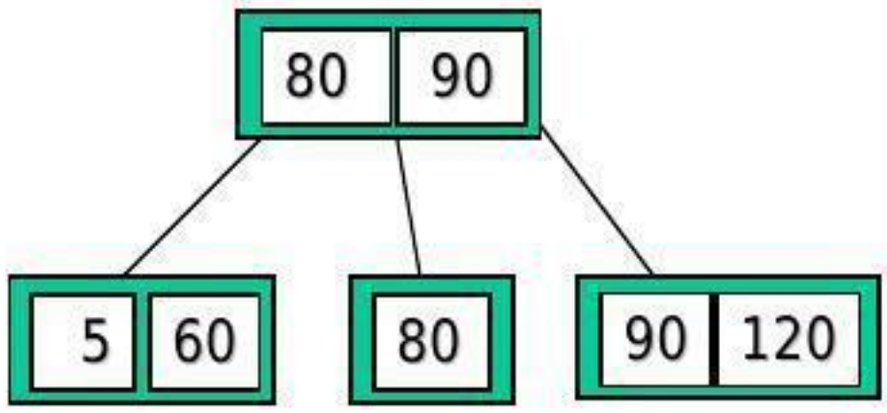
Also a node can be accessed directly by its index. So, B$^+$ tree is called indexed-sequential file. Hence, B$^+$ tree improves the efficiency of accessing/searching.

Example:- B$^+$ tree of order 3

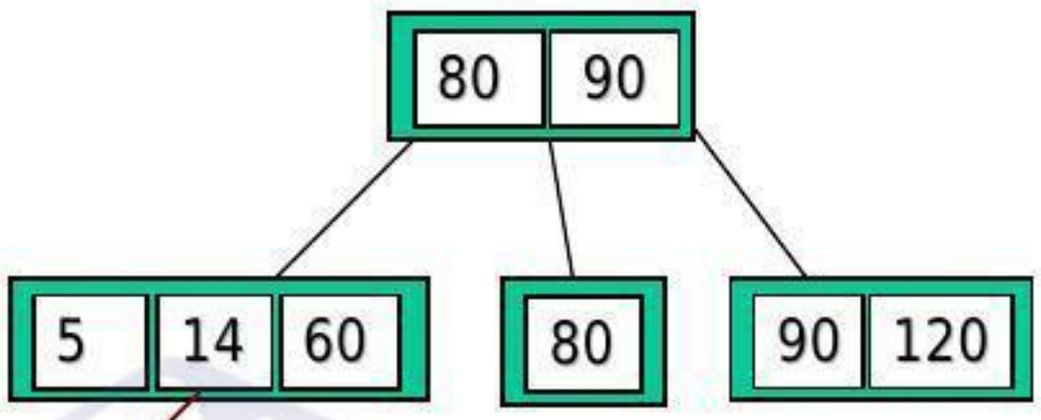


Insertion Algorithm:-

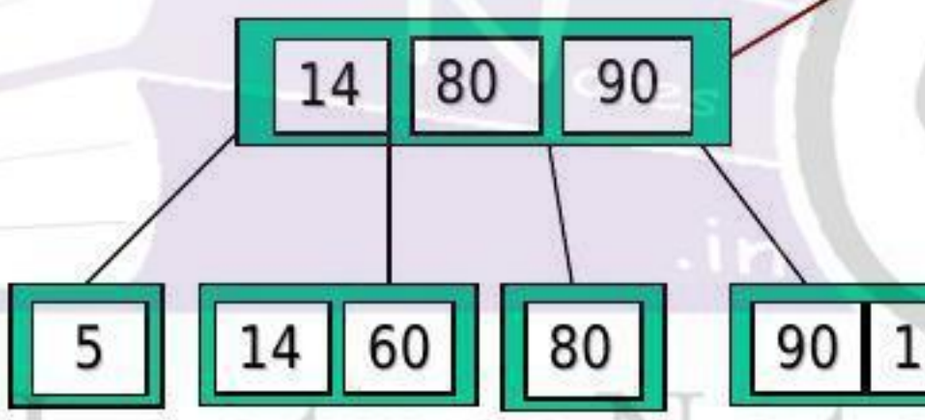
Leaf node Full?	Index node Full?	Action
No	No	Place the key in sorted position in the appropriate leaf node
Yes	No	1. Split the leaf node 2. Place Middle Key in the index node in sorted order. 3. Left leaf node contains keys below the middle key. 4. Right leaf node contains keys >= than the middle key.
Yes	Yes	1.Split the leaf node 2. keys < middle key go to the left leaf node 3. keys >= middle key go to the right leaf node. 4. Split the index node 5. Keys < middle key go to the left index node. 6. Keys > middle key go to the right index node. 7. The middle key goes to the next (higher) level. If the next level index node is full, continue splitting the index nodes.



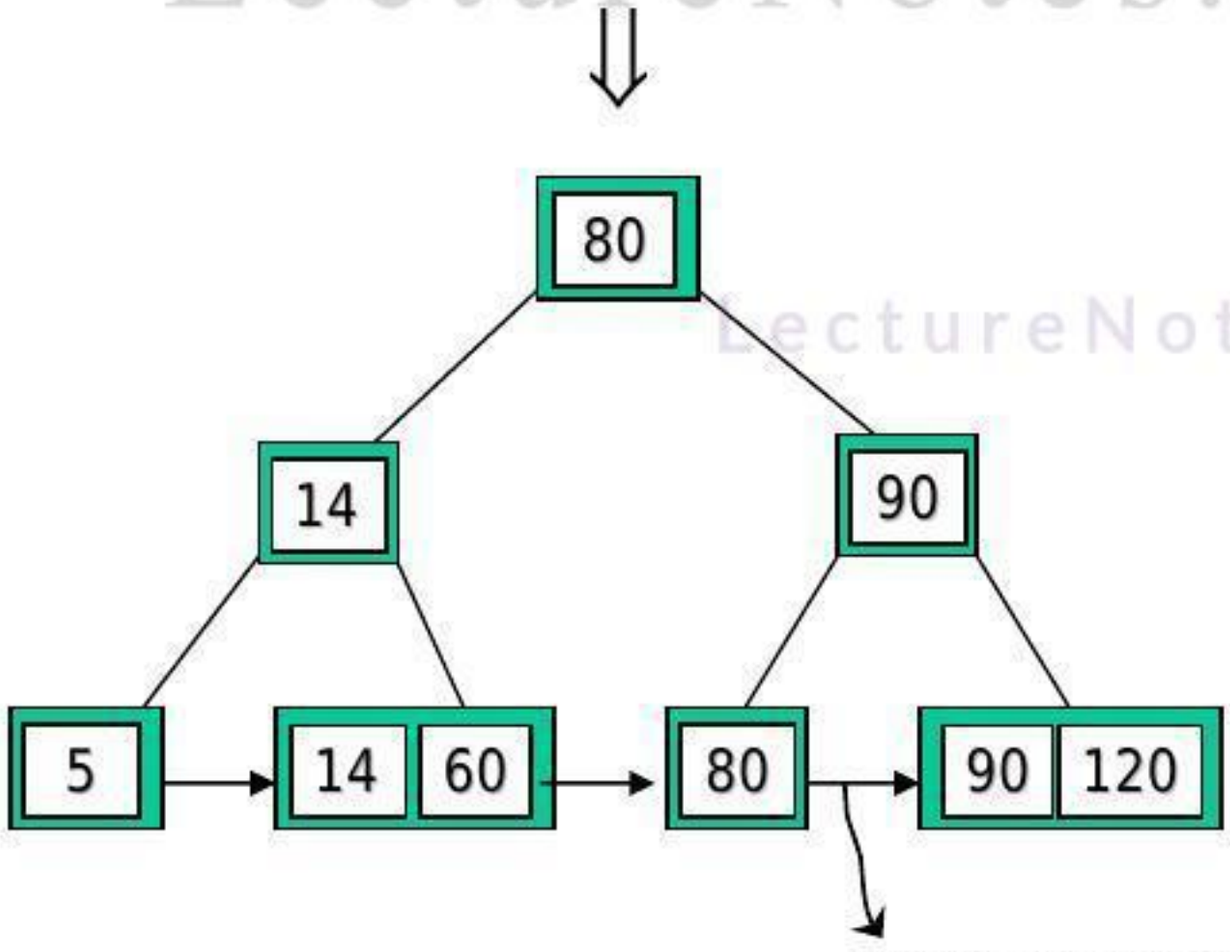
insert 14



Exceeds Order. Promote middle-key and split. copy middle-key in its right-child



Exceeds Order. Promote middle-key and split. But do not copy middle-key in its right-child since it is a non-leaf node.



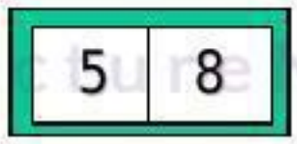
leaves are connected by pointers

Q:- Create a B^+ tree of order 3 for the following set of key values

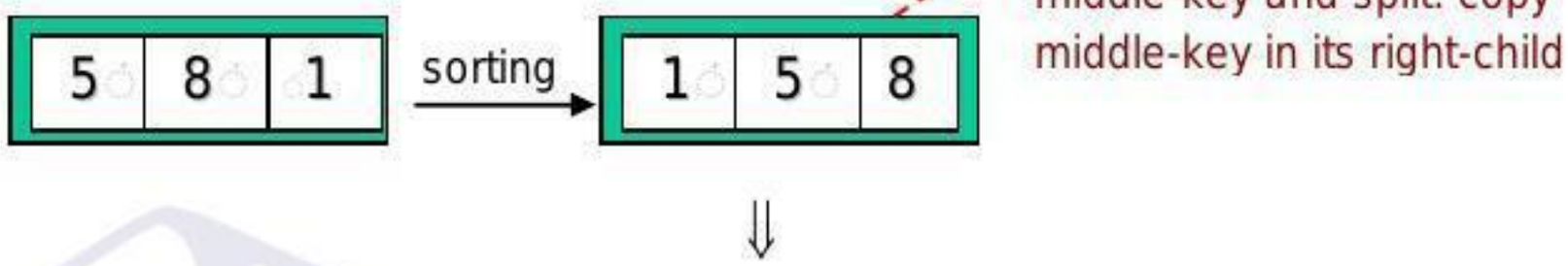
5, 8, 1, 7, 3, 12, 9, 6

insert 5

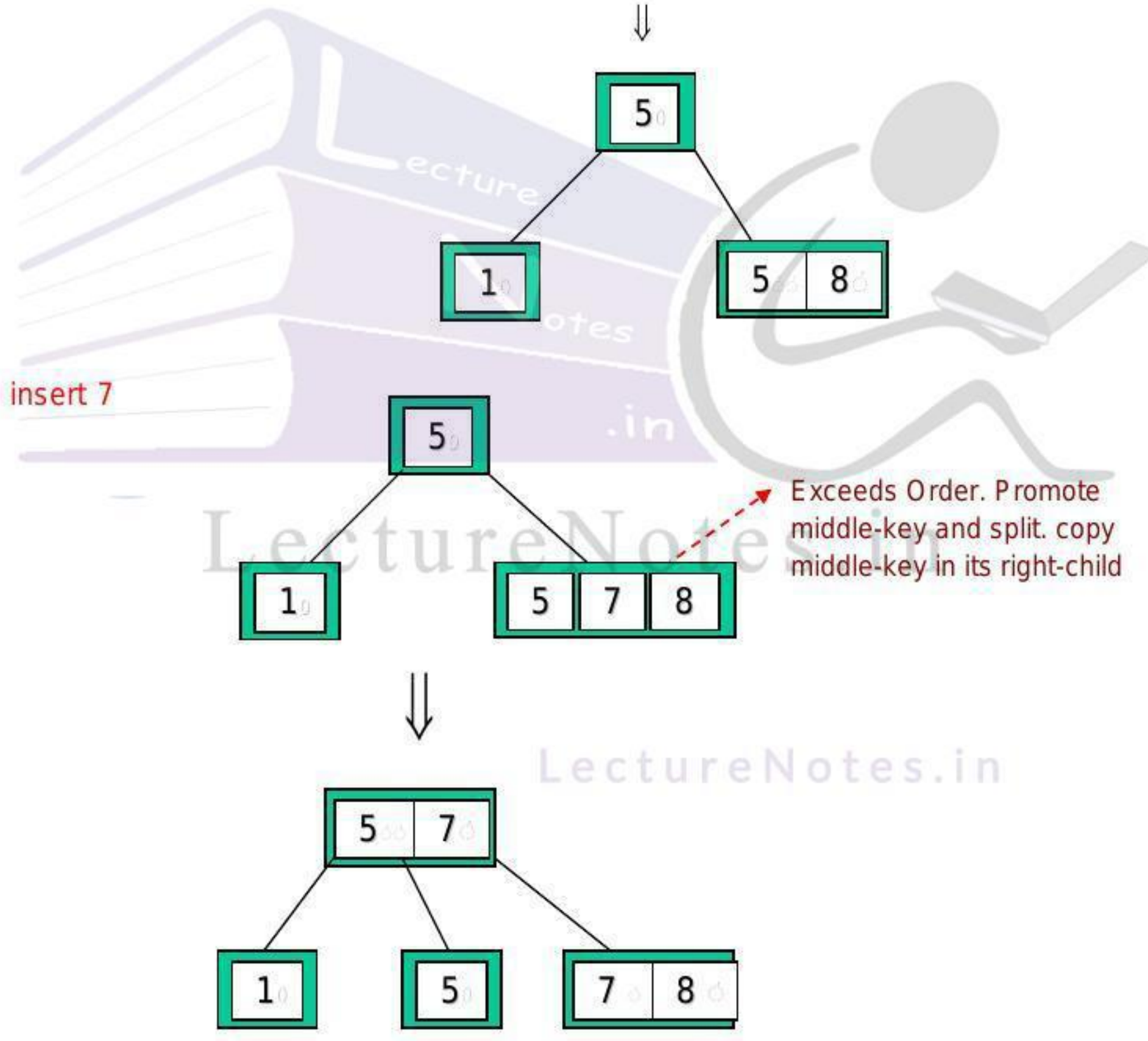
insert 8



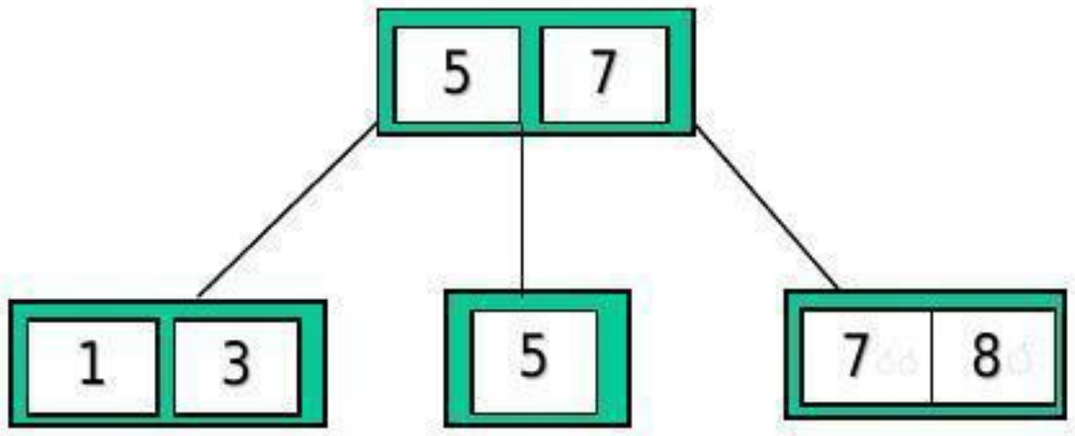
insert 1



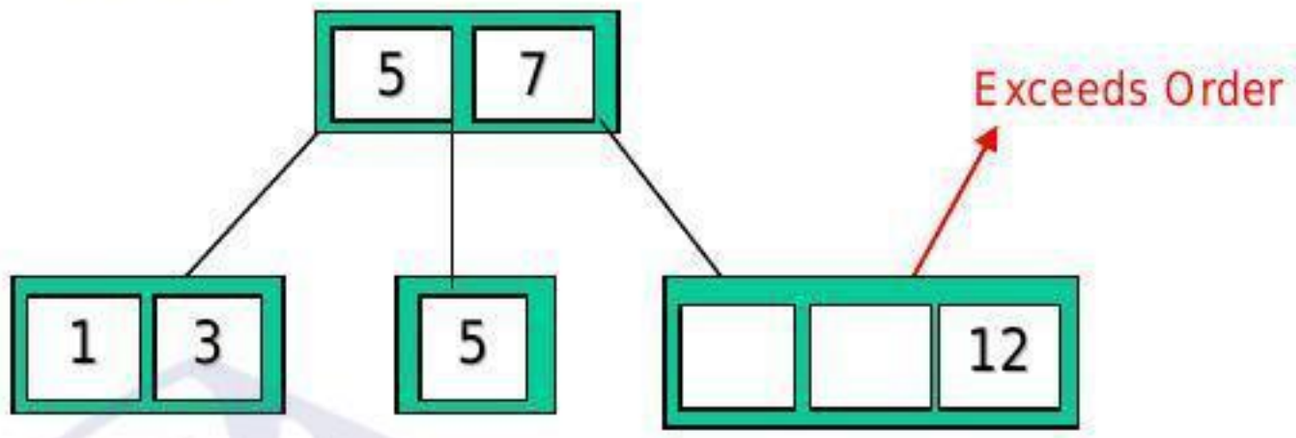
insert 7

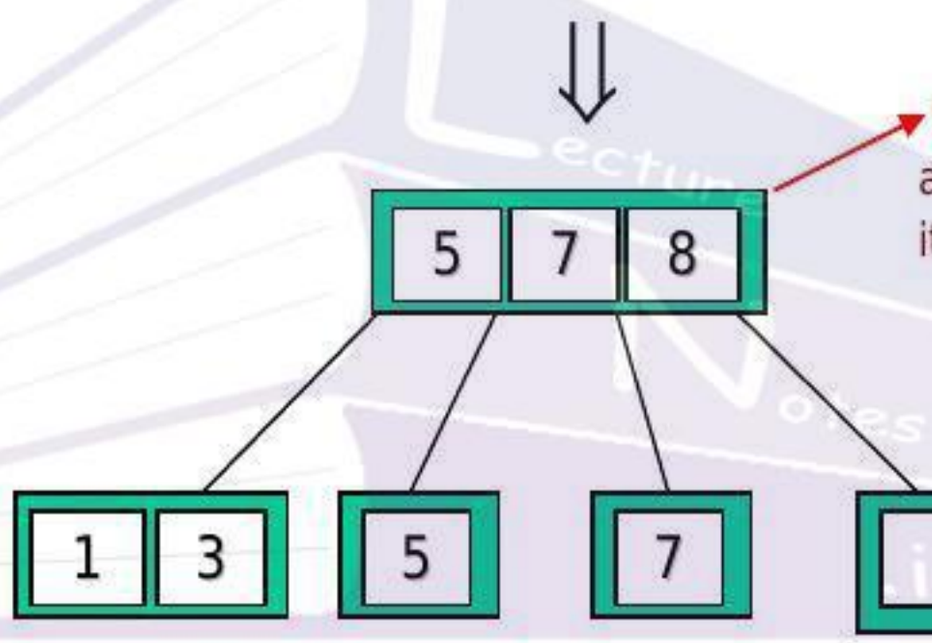


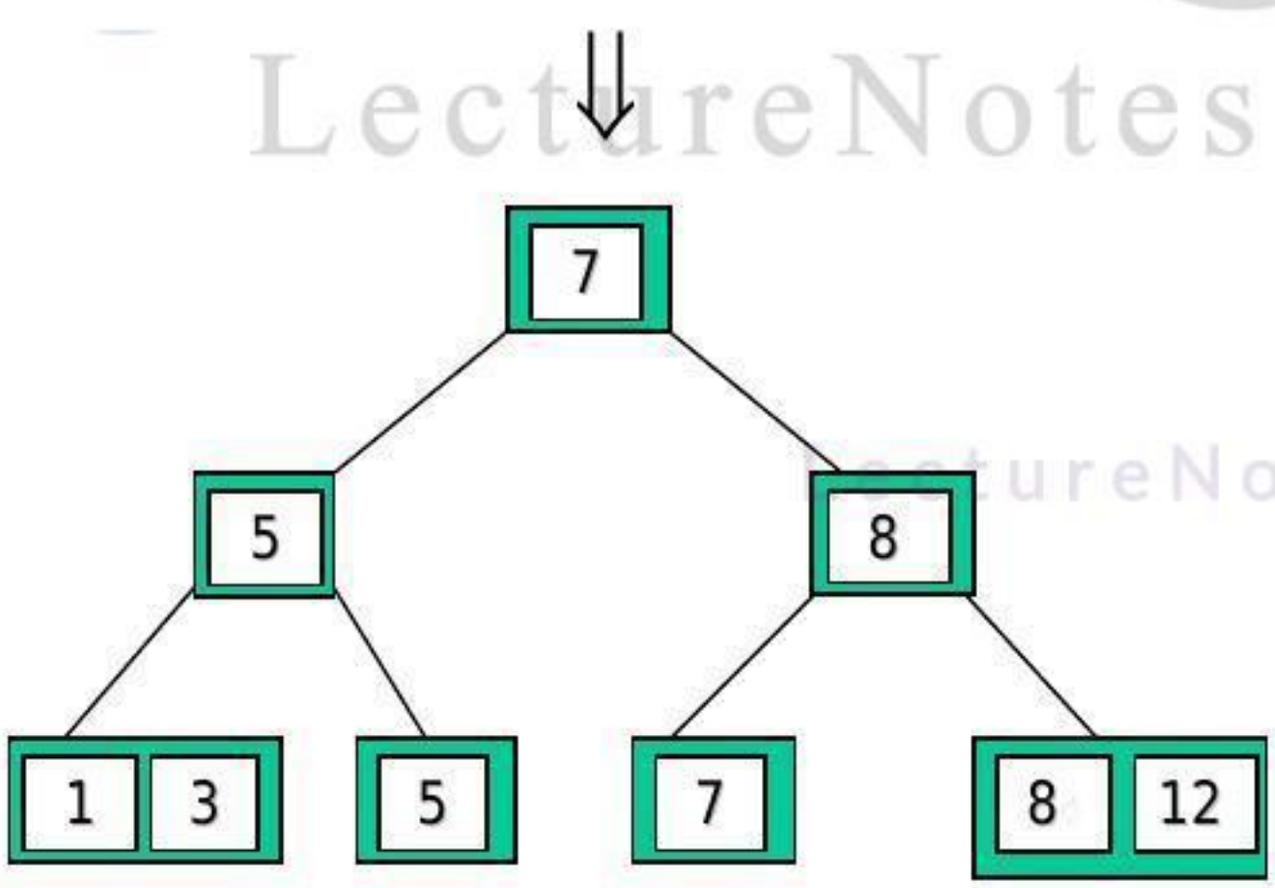
insert 3



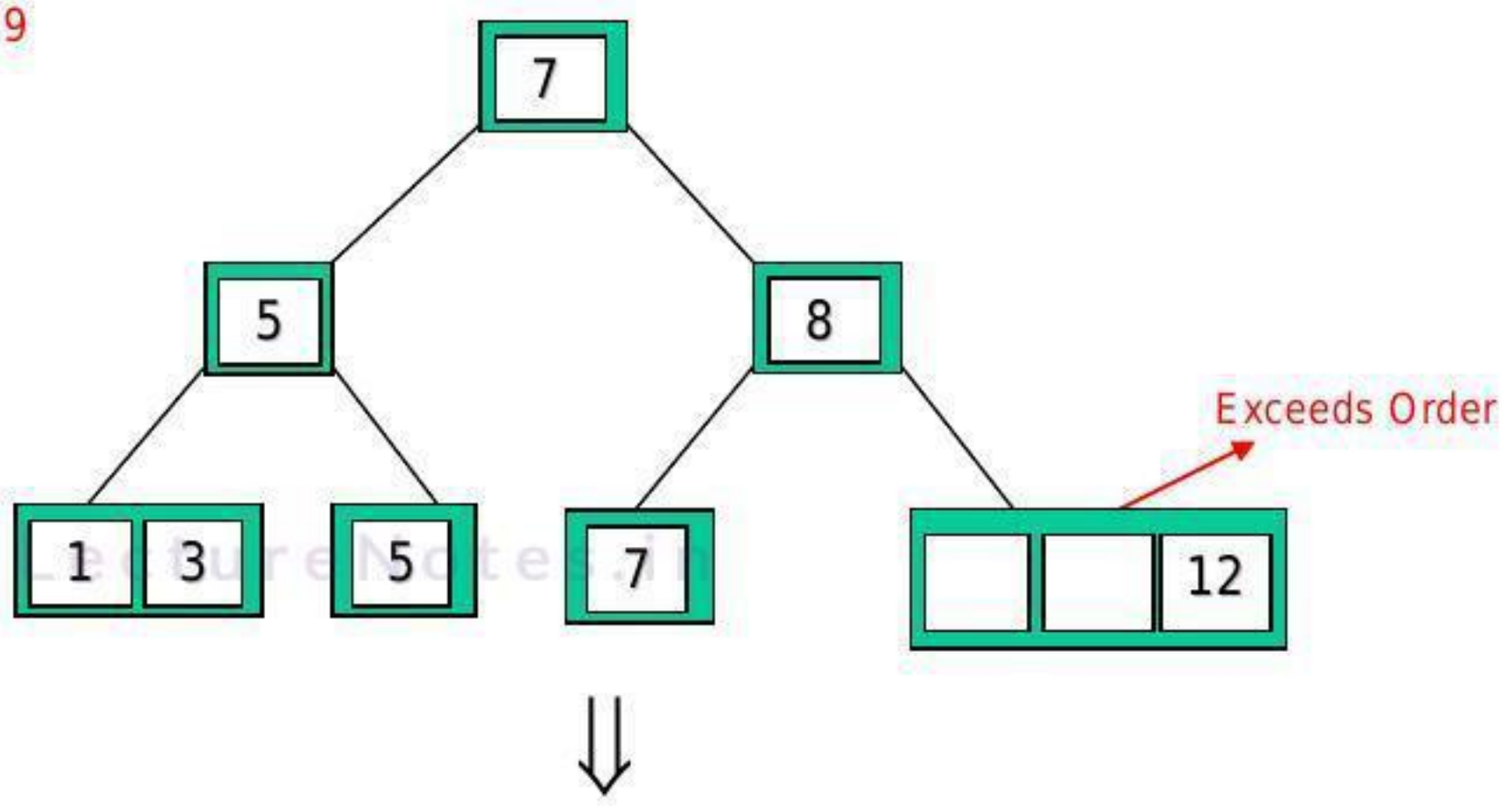
insert 12

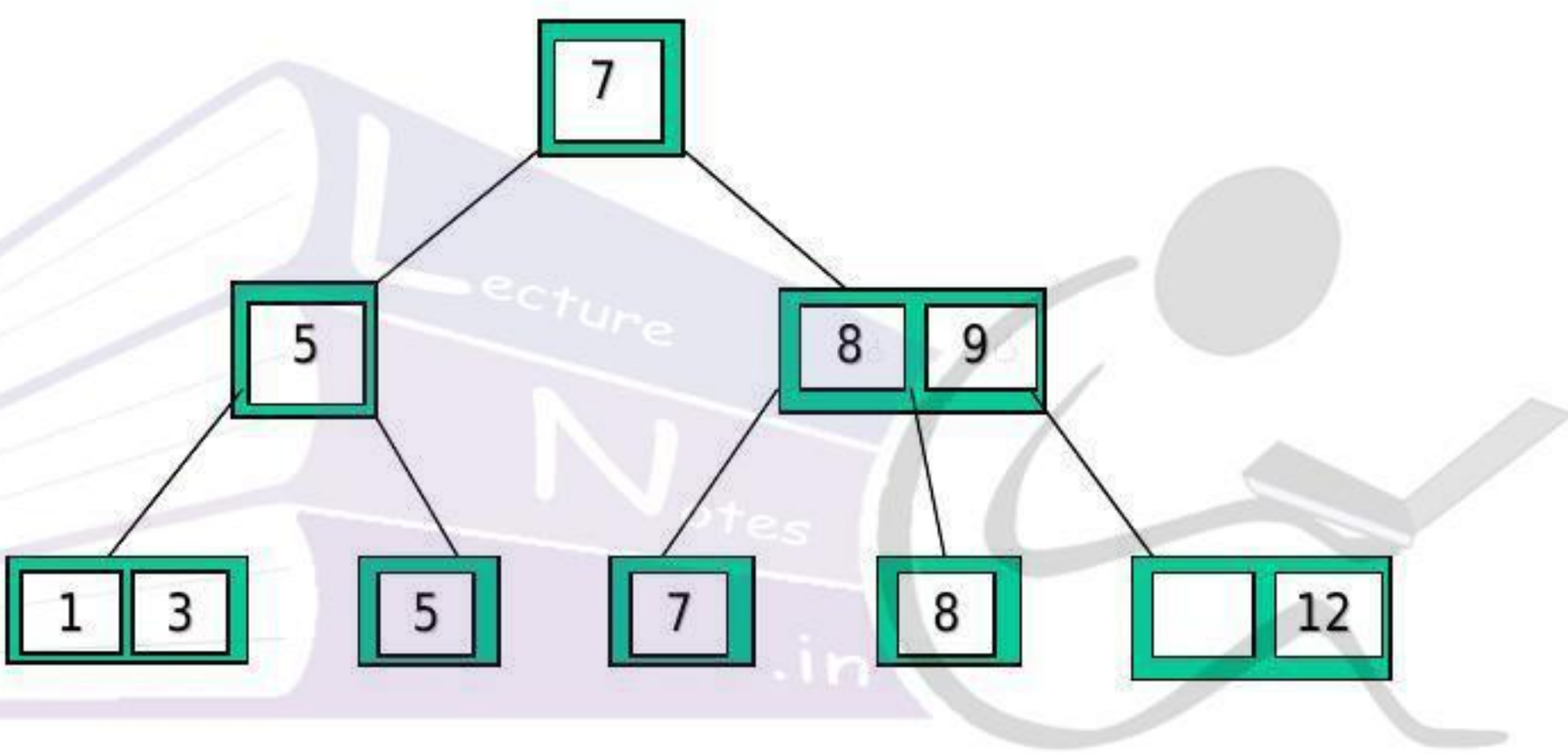




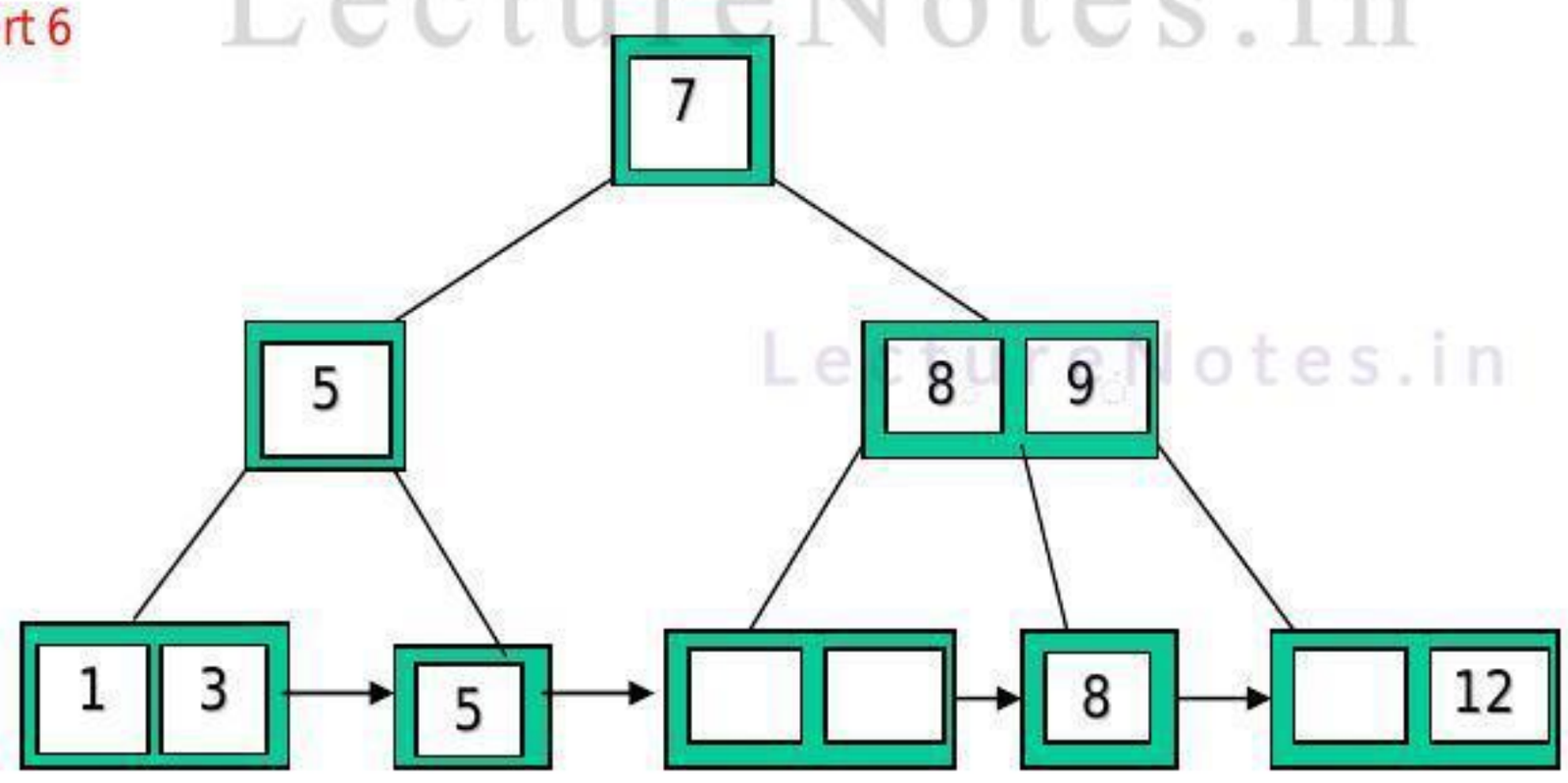


insert 9





insert 6



5.7.2 Deletion from a B^+ tree

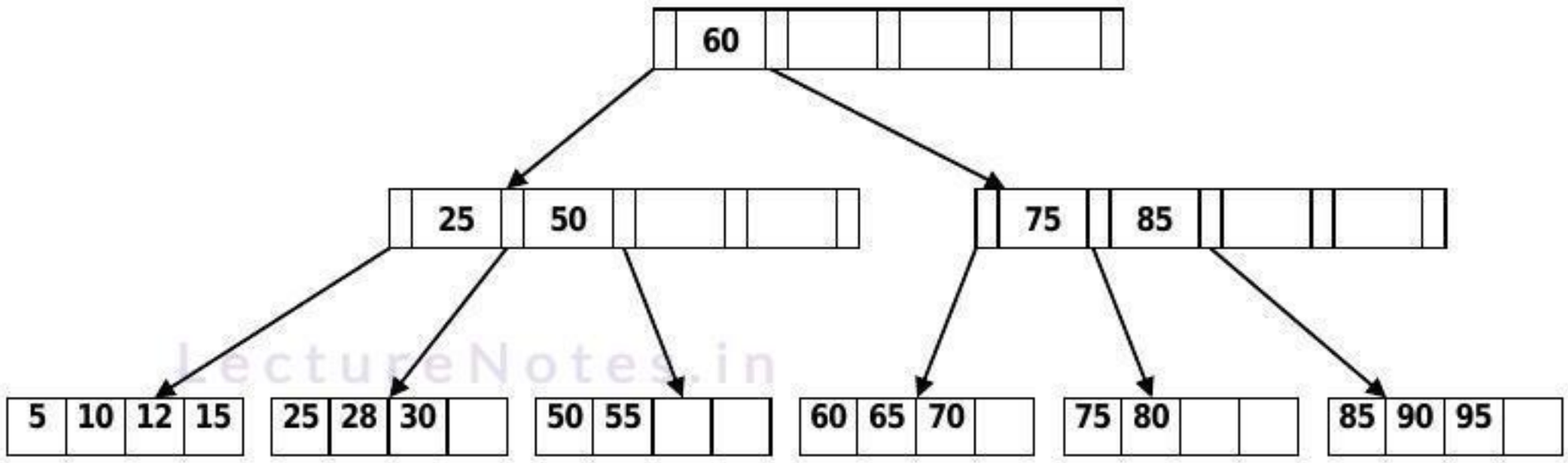
any node is first deleted from leaf then its repetition is deleted from non-leaf (if any)

underflow occurs when node contain less than min^m no of keys.

Deletion Algorithm:- There are **four** cases after we delete a key from B^+ tree

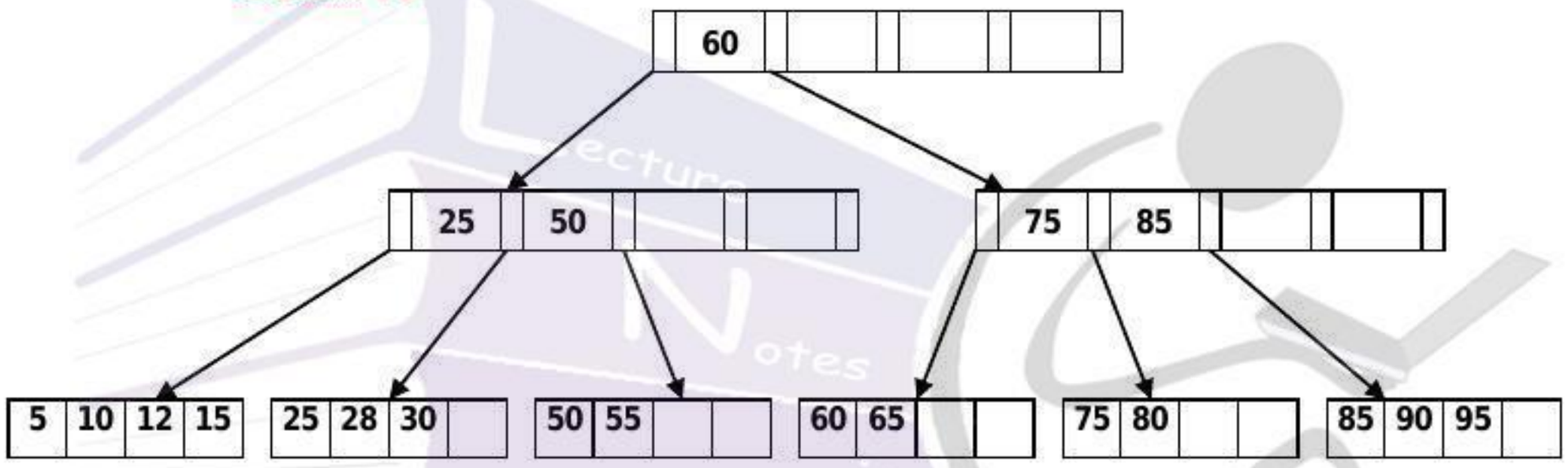
	Leaf node will underflow after deletion?	**Non-leaf node will underflow after deletion?**	**Action**
Case1	NO	NO	delete the key
Case2	NO	YES	delete the key by replacing with a copy of successor.
Case3	YES	NO	1. delete the key and check sibling 2. **if** (sibling has > min^m no of keys) apply rotation **else** combine the leaf node,sibling & parent key . Continue combining non-leaf node with its sibling until you reach a node having correct fill factor or the root node.
Case4	YES	YES	1. delete the key and check sibling 2. **if** (sibling has > min^m no of keys) apply rotation **else** combine the leaf node,sibling & parent key . Continue combining non-leaf node with its sibling until you reach a node having correct fill factor or the root node.

Example:- Consider a B^+ tree of order 5 shown below

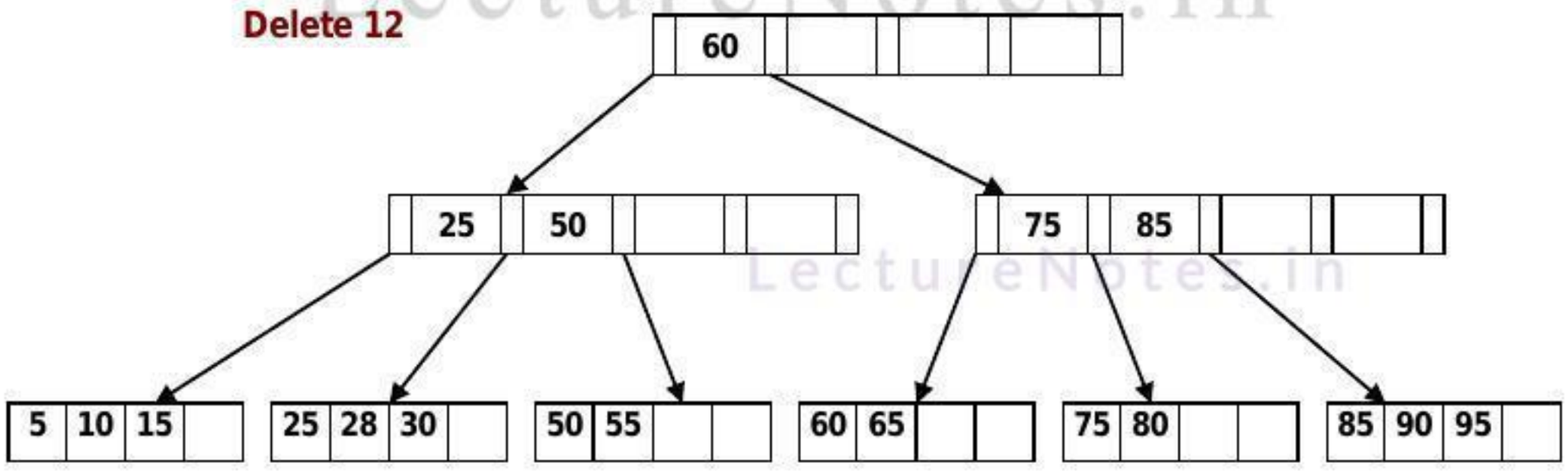


Case1:-

Delete 70



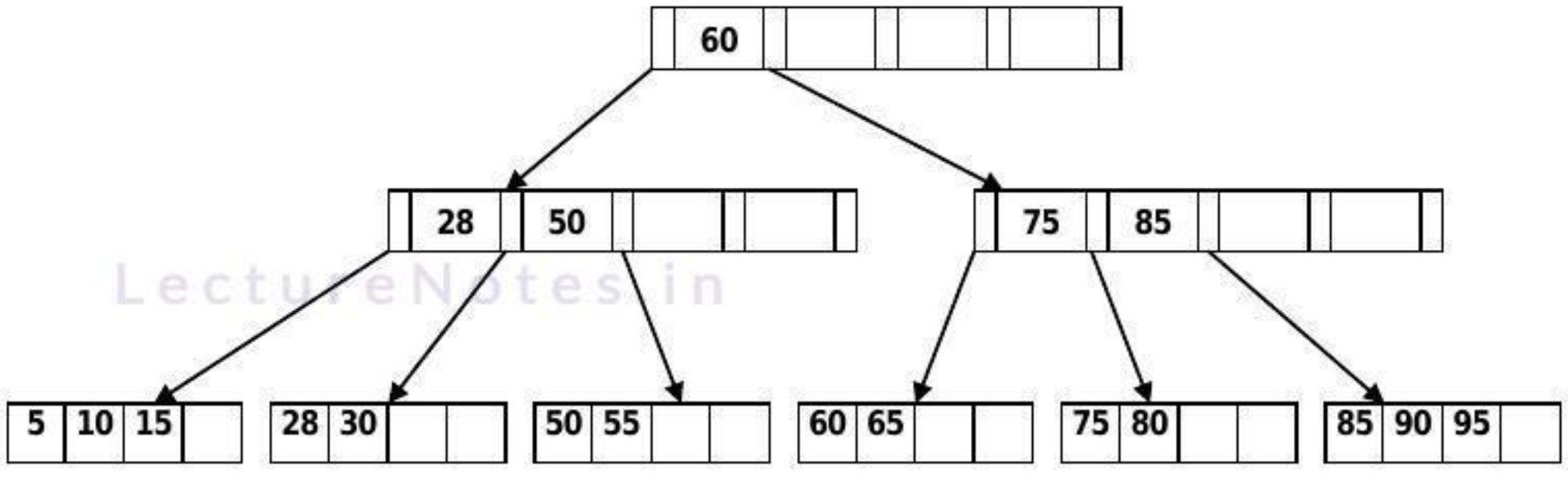
Delete 12



Case2:-

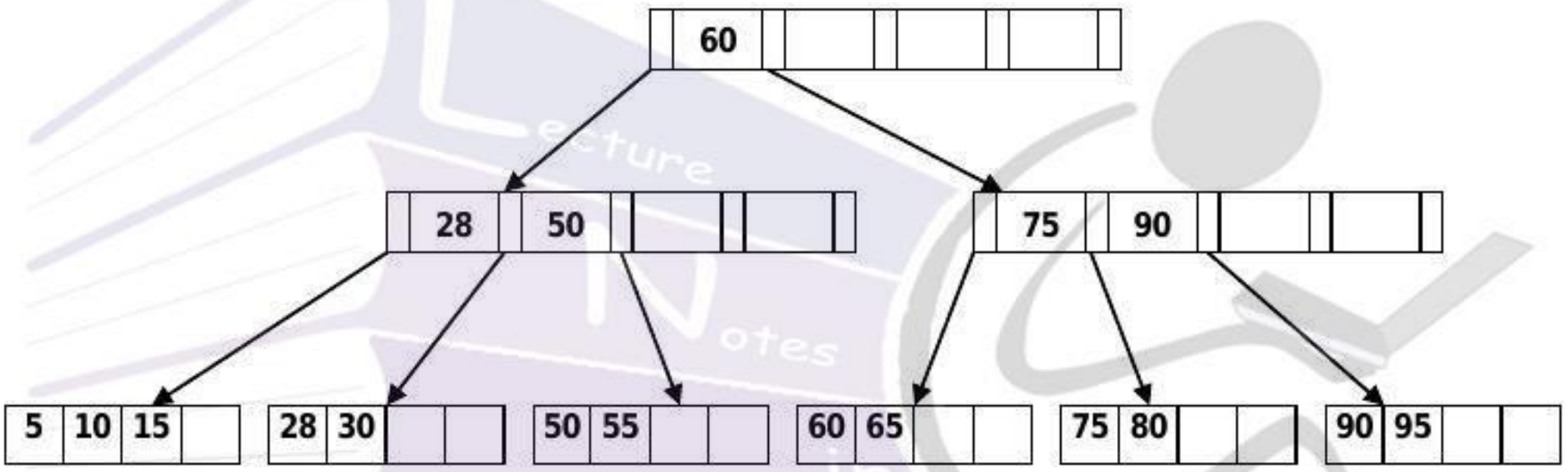
Delete 25

delete 25 from leaf node. Replace 25 with its successor 28 in the non-leaf node.



Delete 85

delete 85 from leaf node. Replace 85 with its successor 90 in non-leaf node.

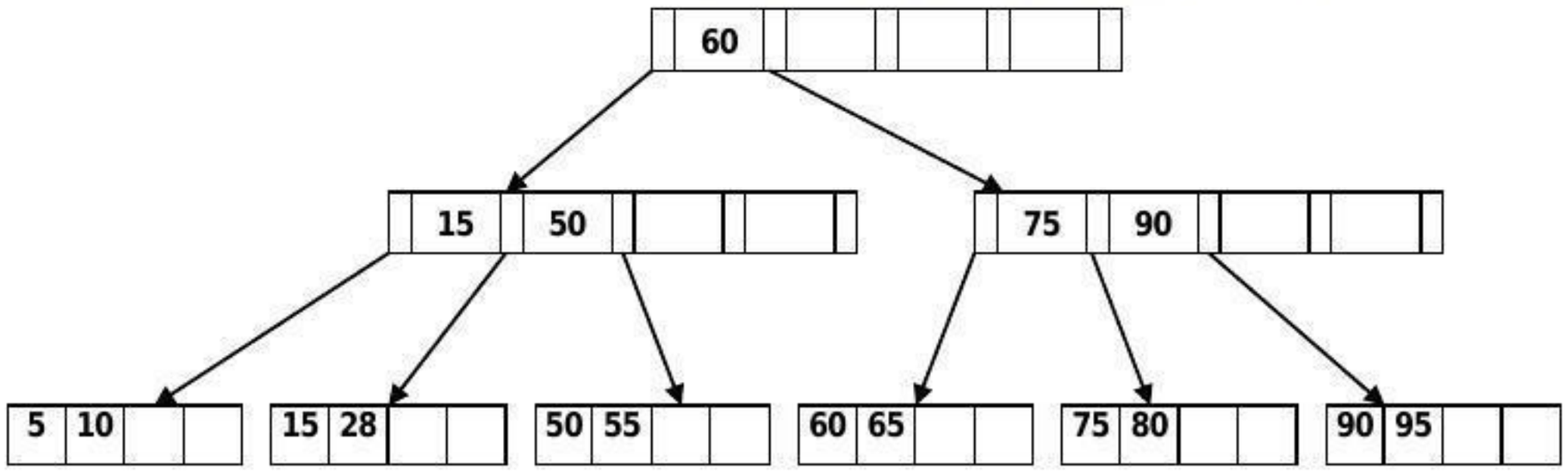


Case3:-

Example1:-

Delete 30

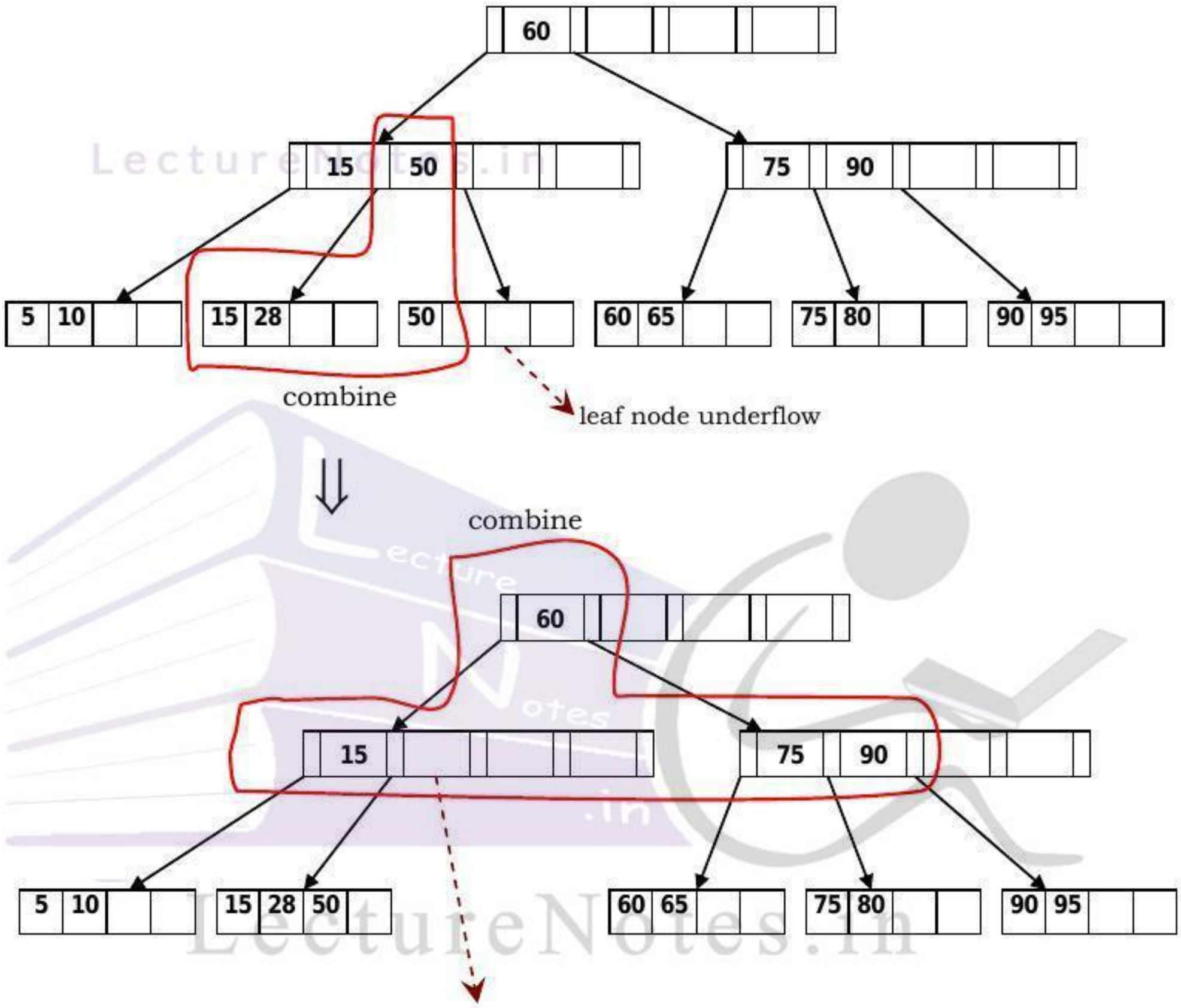
delete the key 30 from leaf node . since sibling has < min^m no of keys, we apply rotation



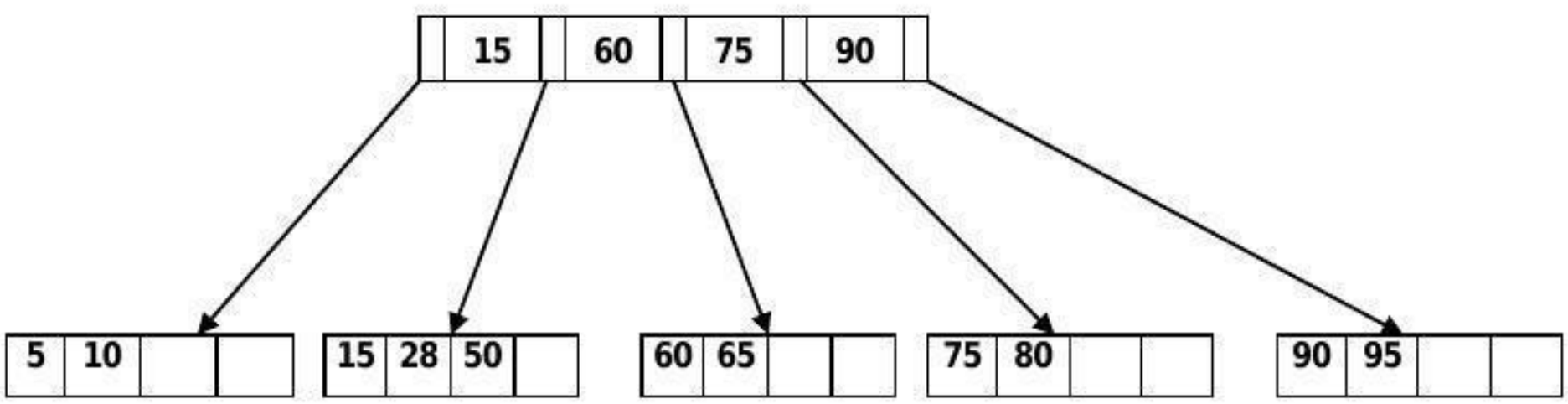
Example2:-

Delete 55

delete the key 55 from leaf node . since the sibling has < min^m no of keys, we combine the leaf node, sibling & parent key .

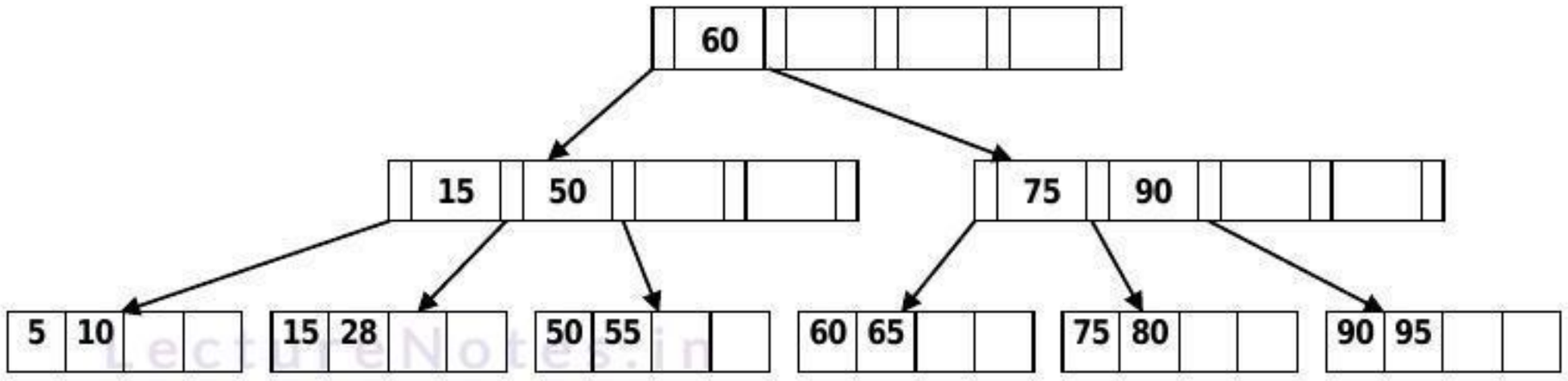


Now, nonleaf node underflow. So, we combine the nonleaf node, sibling & parent key.



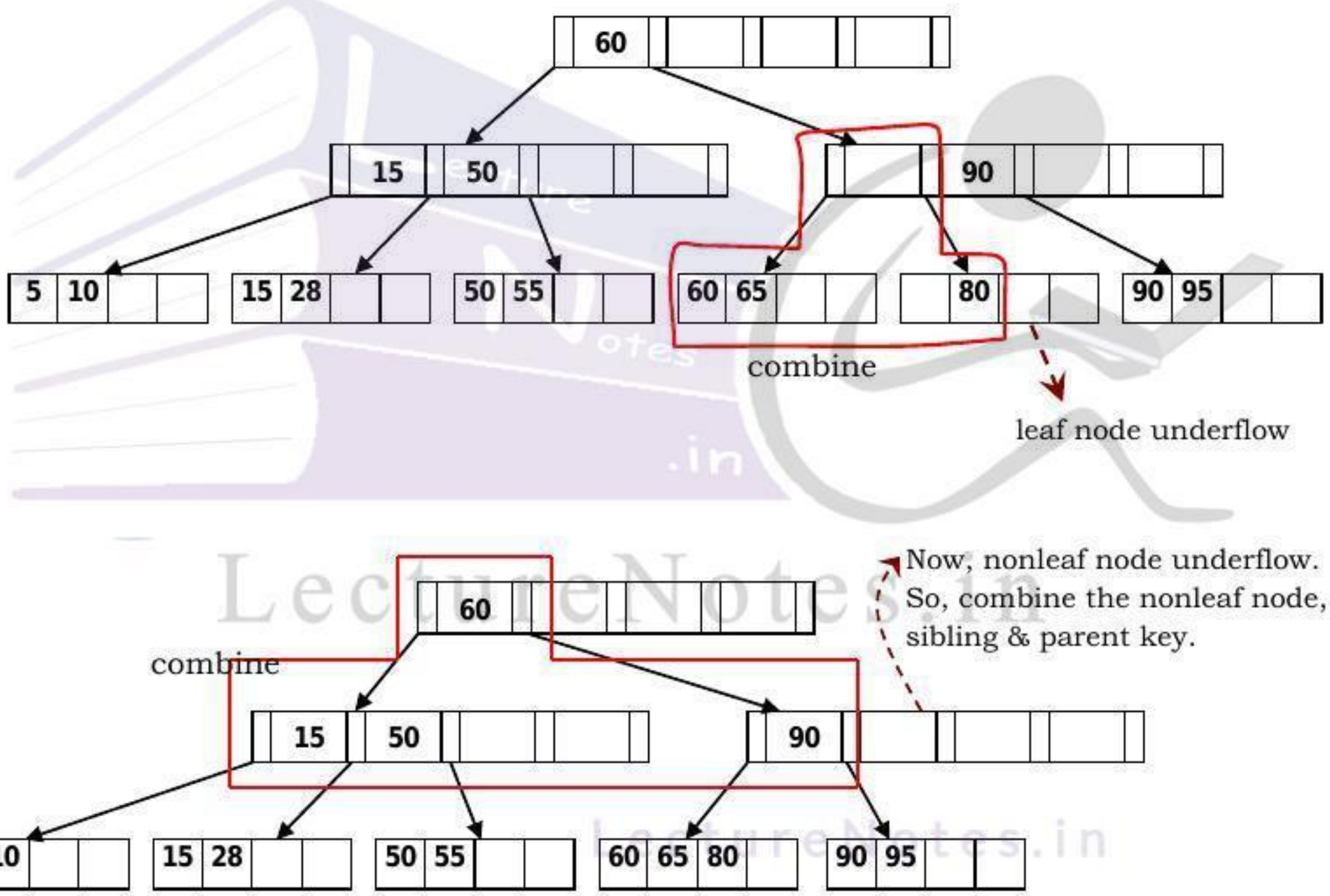
Case4:-

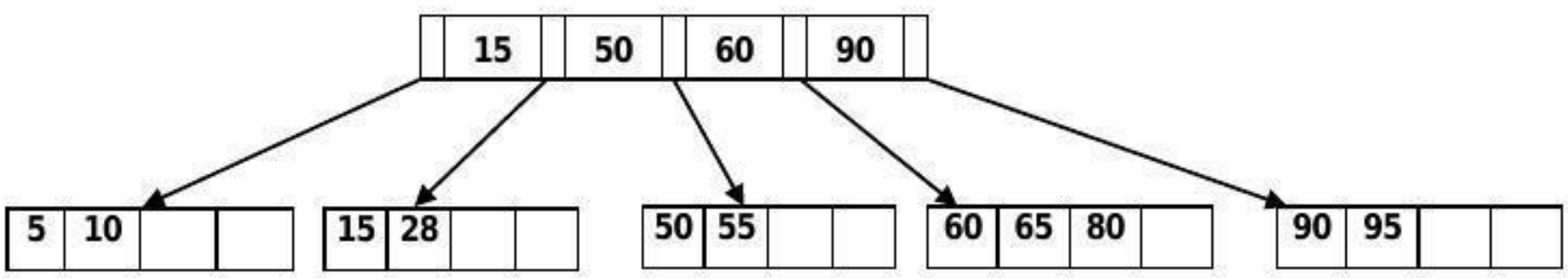
Example:- Consider a B^+ tree of order 5 shown below



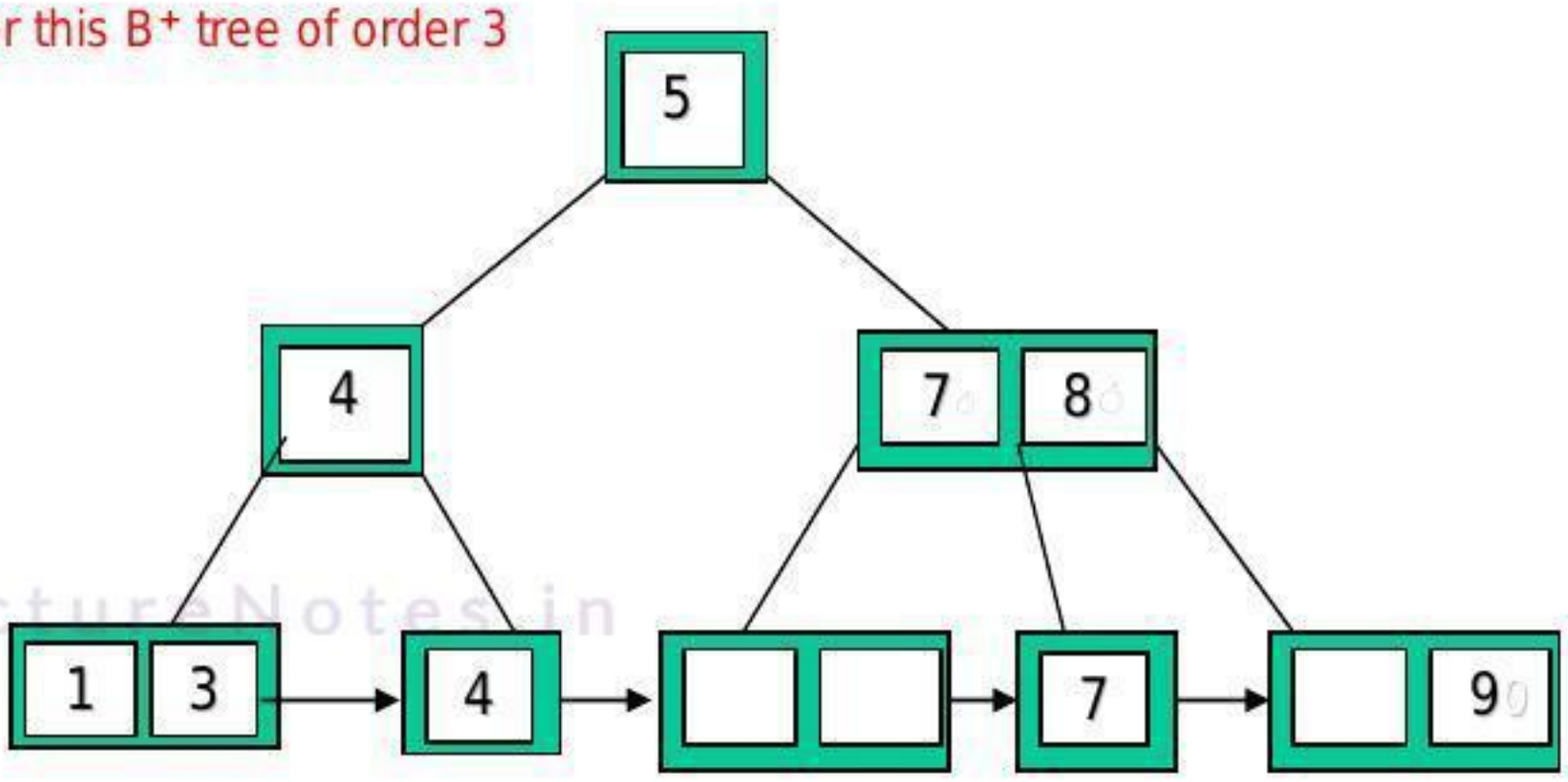
Delete 75

delete the key 75 from leaf and nonleaf node . since the sibling has < min^m no of keys, we combine the leaf node, sibling & parent key .

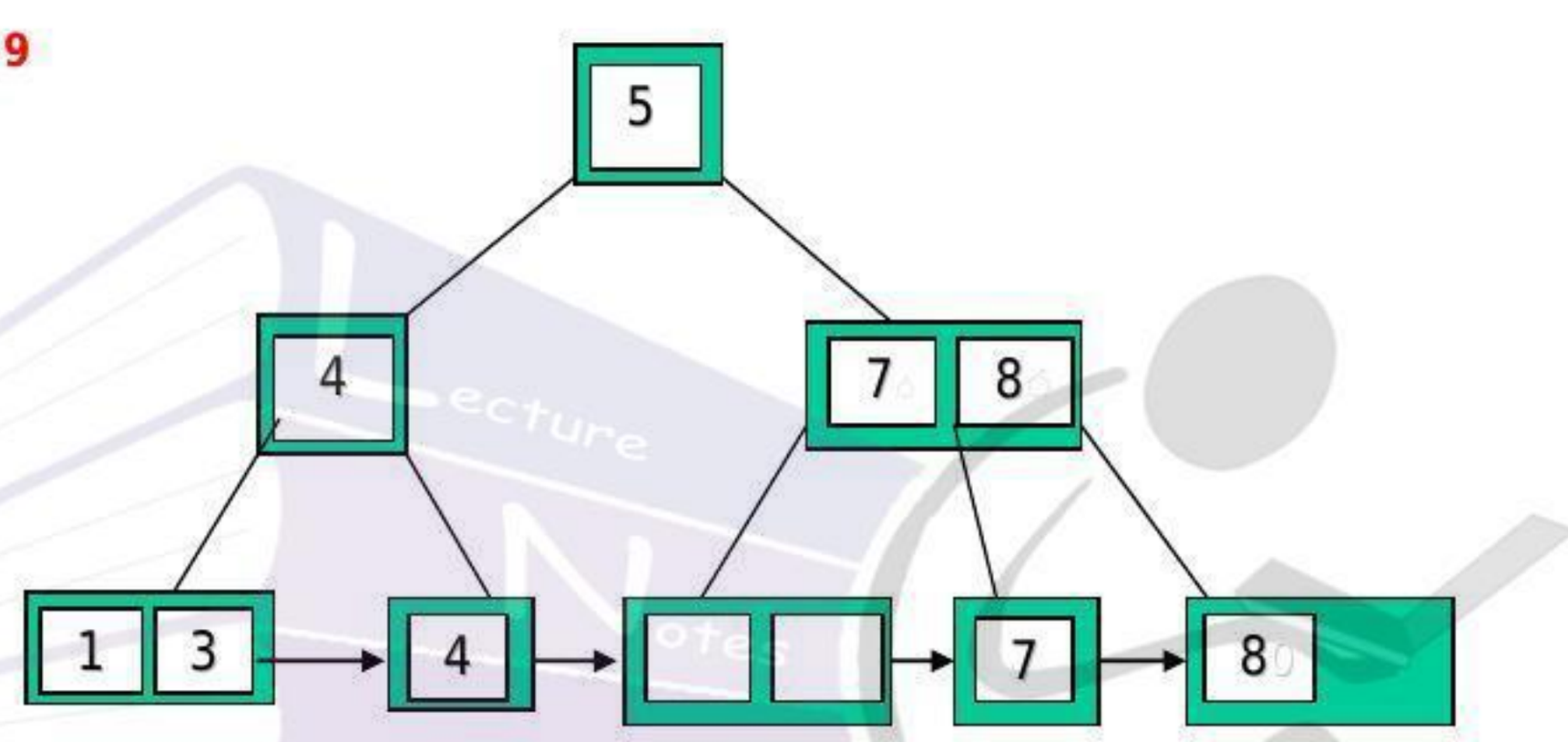




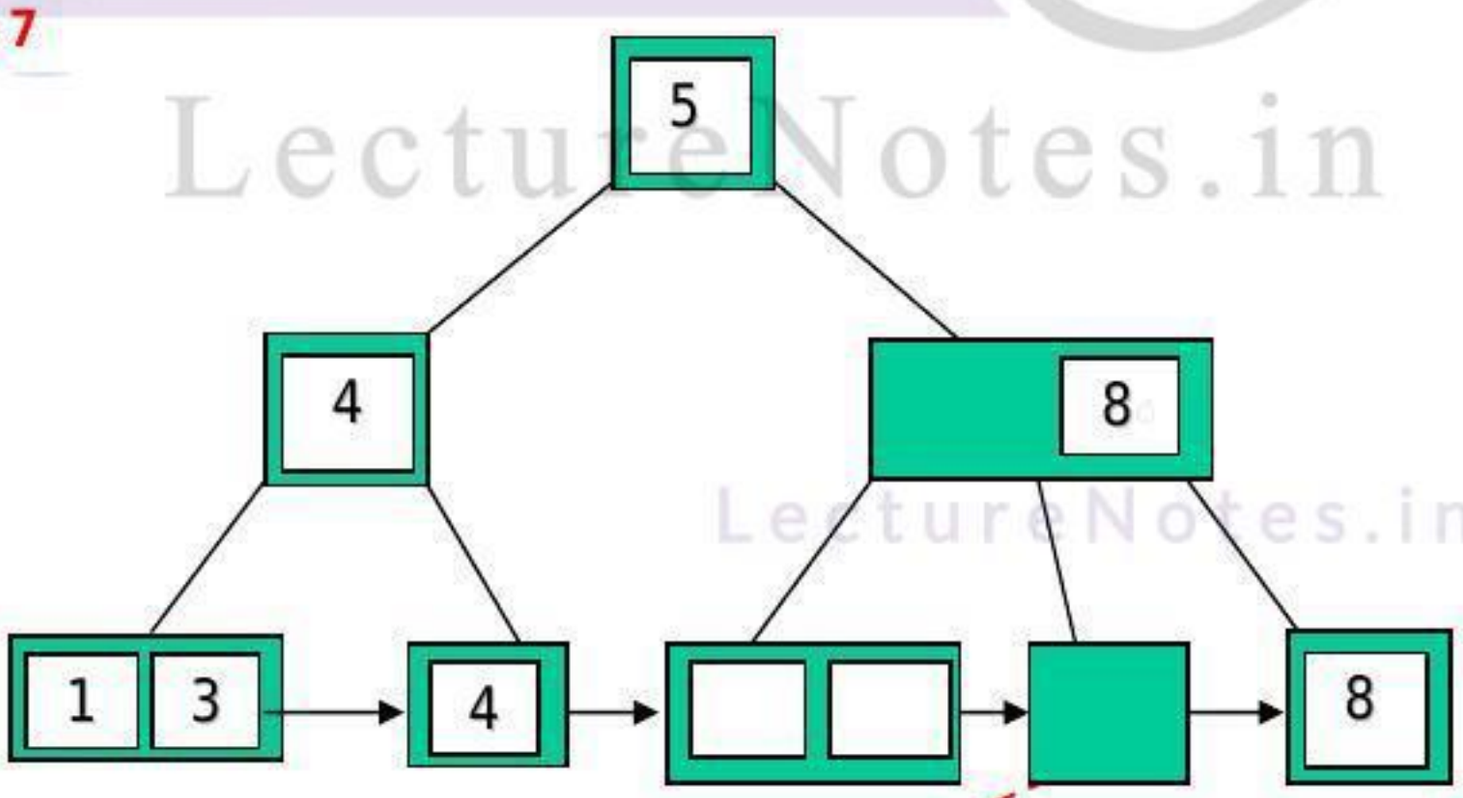
Ex:- Consider this B+ tree of order 3



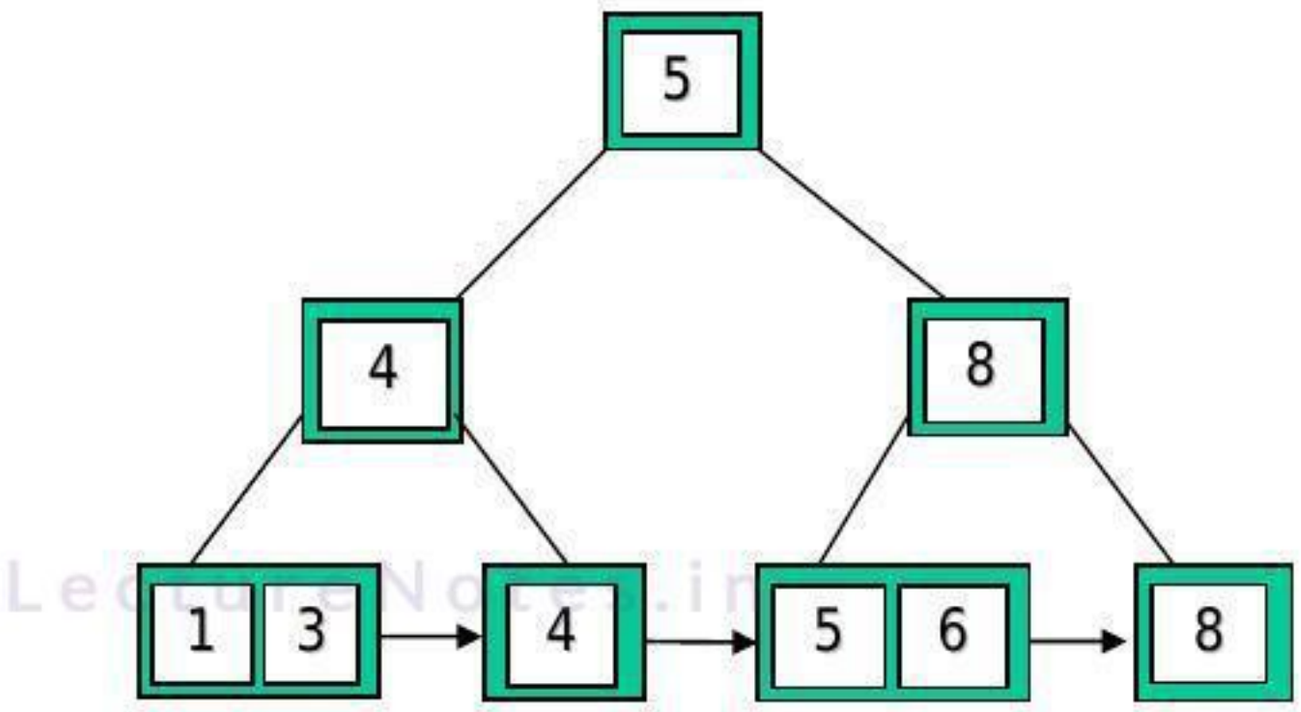
delete 9



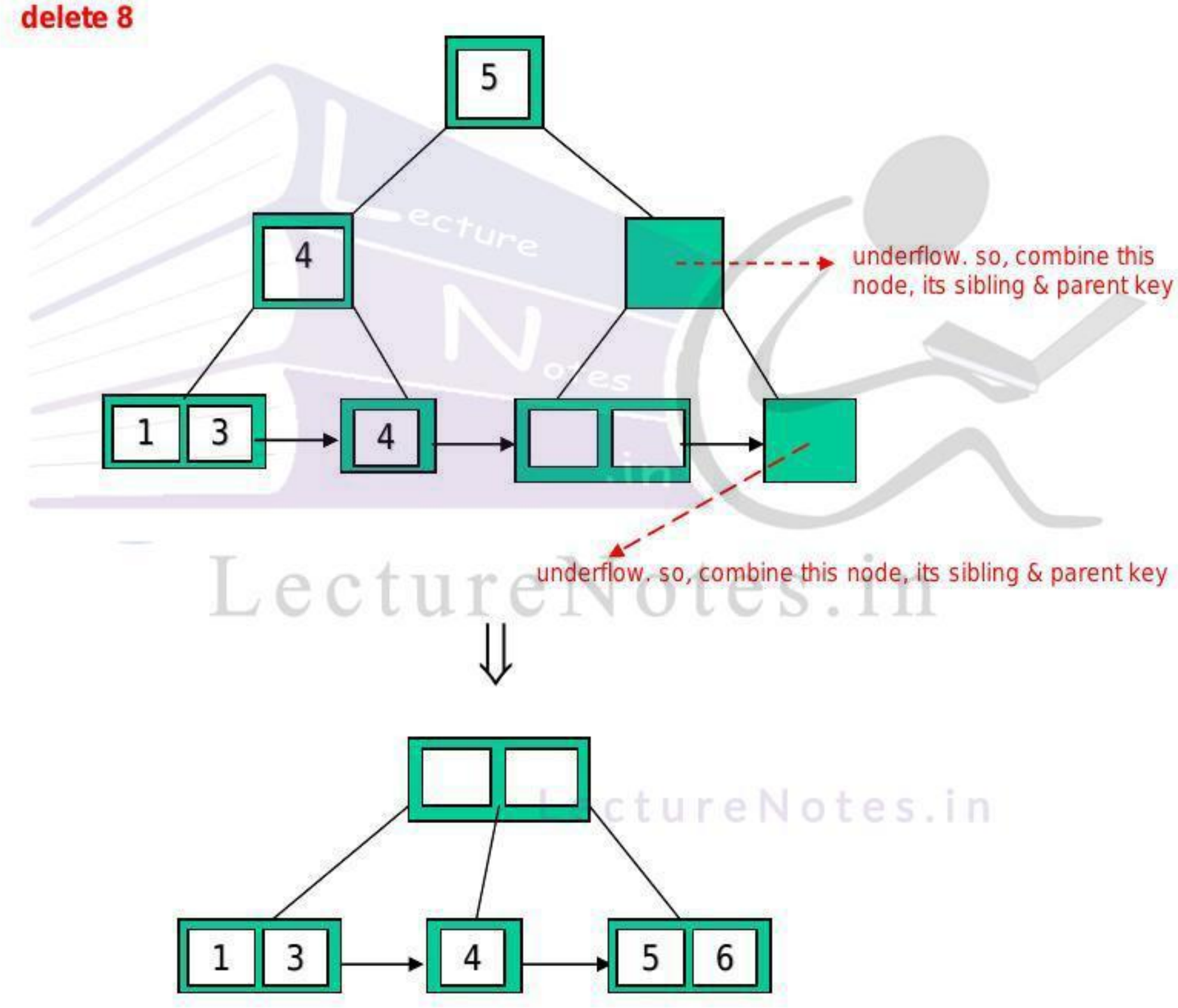
delete 7



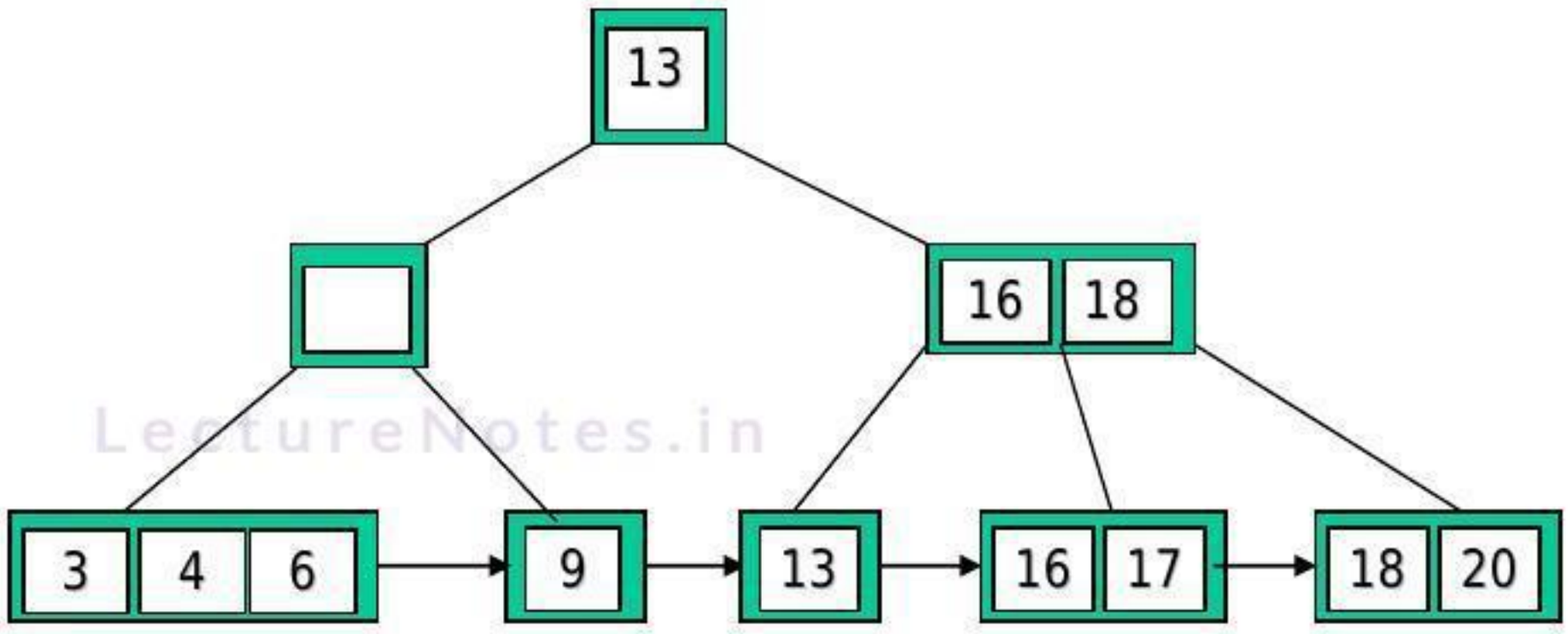
underflow. so, combine the node, sibling & parent key



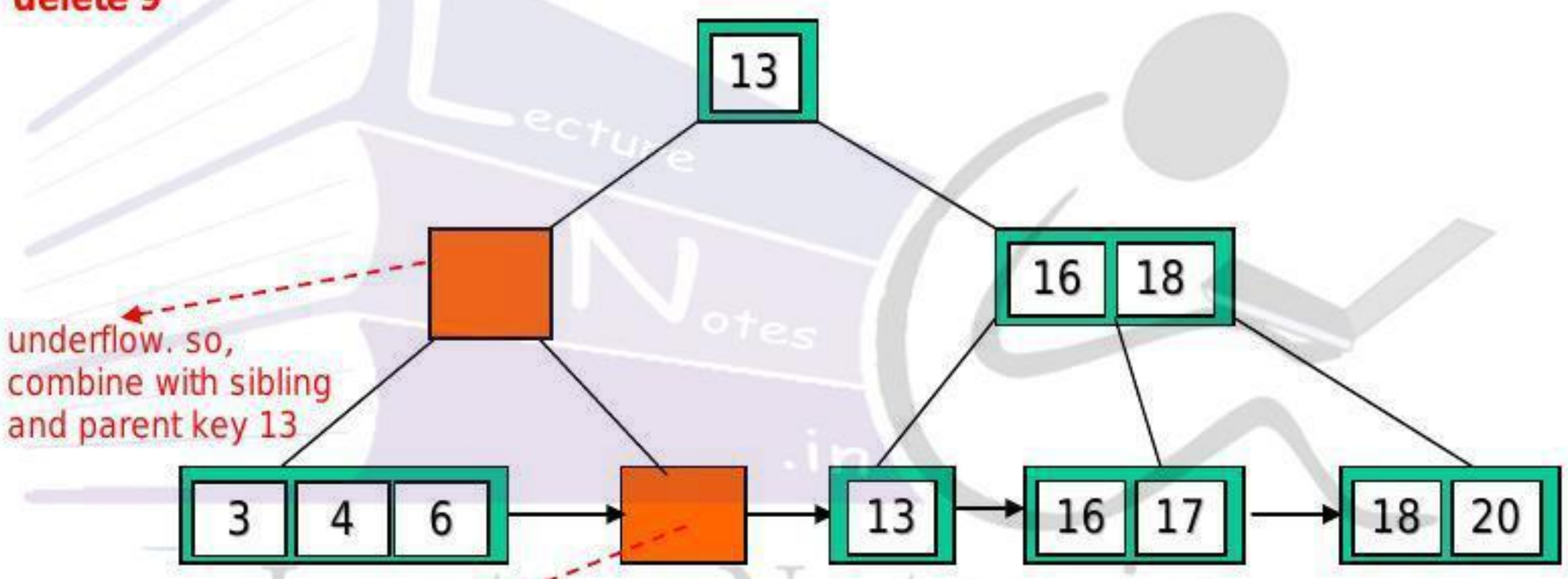
delete 8



Ex:- Consider this B^+ tree of order 4

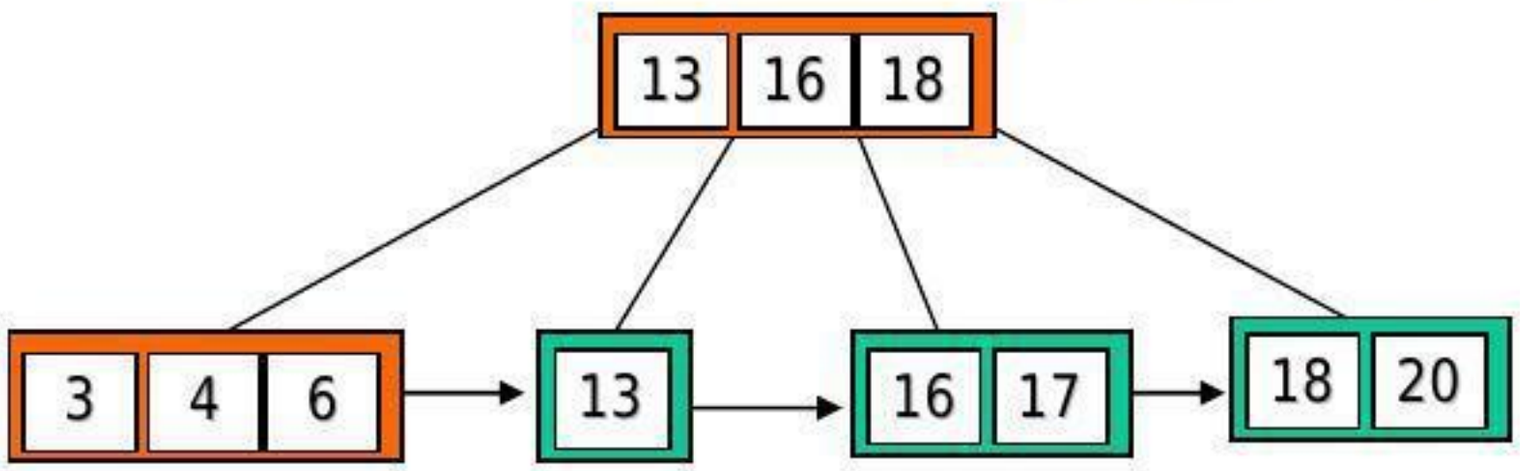


delete 9



underflow. so, combine with sibling and parent key 13

underflow. so, combine with sibling and parent key



5.8 Object oriented database (OODB)

- ✓ Entities are represented as objects.
- ✓ Object is similar to an entity in ER model.
- ✓ Supports all the features of oops.
- ✓ Now a days, database applications use large amounts of complex data, which is difficult to represent in Relational database (since relational attributes store simple, atomic values).
- ✓ OODB are very appropriate for managing complexity (e.g. complex data relationships).
- ✓ OODB designed to support new application areas such as multi-media, CAD, CASE, OIS etc.
  - Computer Aided Design(CAD): A CAD database stores objects related to mechanical and electrical branch such as buildings, IC circuits etc.
  - Computer-aided software engineering(CASE): A CASE database stores objects related to SDLC(software development lifecycle).
  - Office information systems (OIS): An OIS database stores objects related to business information such as emails, invoices etc.

5.9 Object Relational database (ORDB)

- ✓ Supports relational and object-oriented features.
- ✓ Also called as "Extended Relational Database"
- ✓ Support internet-enabled applications found on the Web which include multimedia data types (text, image, audio, video), time series data type, geospatial data type and any data type a user may wish to define.

Features:

- Inheritance
- Polymorphism
- Additional/extended data types, e.g. BLOBs
- User-defined or abstract data types
- Generalization/specialization
- Aggregation
- Multivalued attributes

Example of user defined data type:

create name_type as object (firstname char(25), middlename char(15), lastname char(25));

create table person (name name_type,address char(50));

Example of ORDB: DB2 Universal Database of IBM, Dynamic Server of Informix, Oracle-10g of Oracle, PostgreSQL of PostgreSQL Inc.

5.10 Parallel database

A parallel database system improves performance by parallel execution of various operations, such as loading data, building indexes and evaluating queries. Parallel database improves the processing and I/O speed by using multiple CPU and disk in parallel.

Parallel database is divided into three categories

- Shared memory architecture: multiple processors share the main memory.
- Shared disk architecture: multiple processors share the hard-disk
- Shared nothing architecture: each system has its own memory and hard-disk.

5.11 Distributed database

A distributed database has a centralized DBMS but the data is present in multiple computers located over a network.

Data fragmentation:- Division of relation R into fragments (R_1,R_2,..... ,Rn)

Types of fragments are

1. Horizontal Fragmentation:- tuples of R are divided into fragments.
2. Vertical Fragmentation:- attributes of R are divided into fragments.

Applications: Bank with many branches, retail store with many locations, Library with many branches

Advantages:

1. economical:- less cost to create a network of many smaller computers than to create a single large computer.
2. modularity:- A system is modified, added and removed from the distributed database without affecting other system.
3. protection of valuable data:- if there is catastrophic situation such as fire, earthquake in one location, then can get this data from other location.

5.12 Parallel vs Distributed database

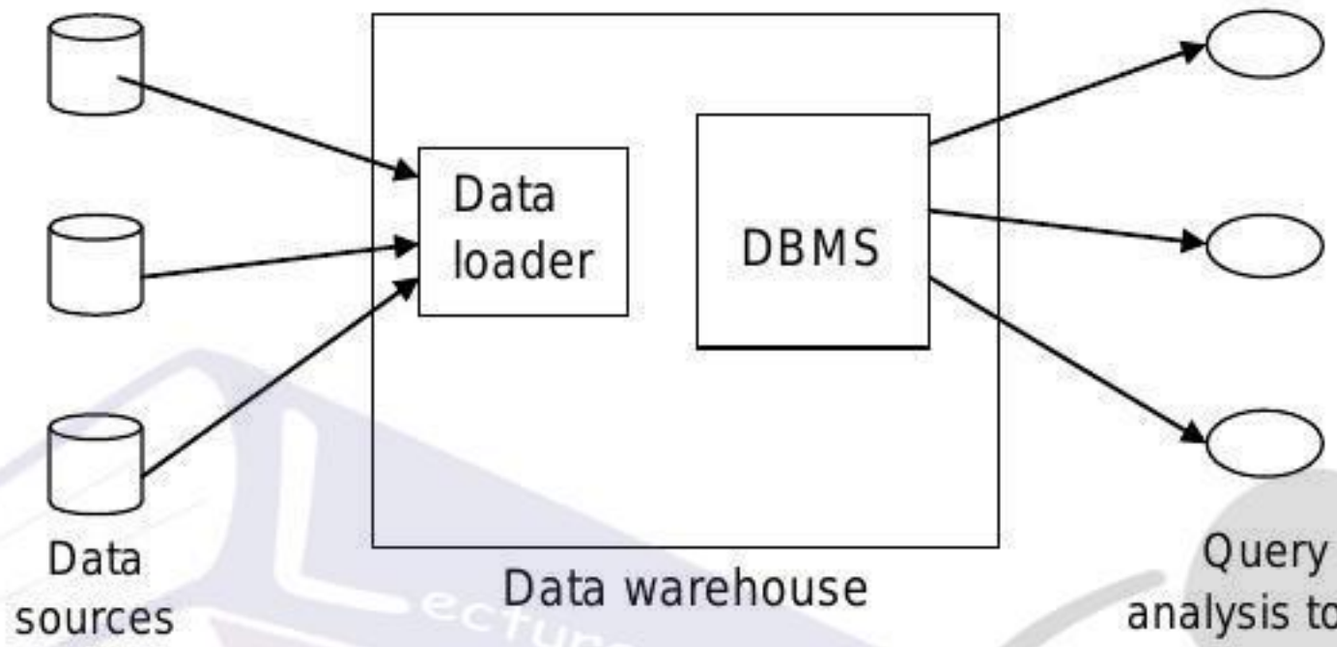
Parallel Database	Distributed Database
1. Machines are physically close to each other, e.g. same server room.	1. Machines can far from each other, e.g. in different locations
2. Machines are connected using LAN.	2. Machines are connected using public-purpose network, e.g. Internet.
3. Communication cost is less.	3. Communication cost is high
4. Can have shared-memory, shared-disk, or shared-nothing architecture	4. Usually shared-nothing architecture.
5. Faster response time for queries	5. Faster response time for queries
6. We can perform parallel processing on a single machine.	6. We can't perform distributed processing on a single machine.

5.13 Data warehouse

Data warehousing is the process of centralizing or aggregating data from multiple sources (databases) into one common repository/storage. Data warehouse supports decision making system.

Example:- Facebook basically gathers all of your data – your friends, your likes, whom you follow etc. And then stores that data into one central repository.

Components of Data warehouse:-



Steps for preparing Data warehouse:-

1. Gathering data:- Collecting information from multiple data sources.

2. Schema definition:- Data collected from different data sources may have different schemas. So we perform schema integration and convert the data to the integrated schema.

Example:-

DRIEMS = | Regdno, name, branch, phoneno |

OEC = | Regdno, name, branch, mailid |

⇒Schema = | Regdno, name, branch, phoneno, mailid |

3. Data cleansing and transformation:- Correcting and preprocessing the data collected from data sources is called data cleansing.

Data collected from different sources may have minor inconsistency. Thus, it is required to transform it to correct data.

4. Propagating updates:- updates made to the relation of data sources must be propagated to the data warehouse.

5.14 Data mining

Data mining is the process of finding patterns(meaningful data) from a data set.

Example1: supermarkets to study their consumers.

Supermarket aggregates(collect) data regarding their consumers. They apply data mining techniques to find meaningful data(e.g. shopping habits). This valuable information is applied in their business policy to gain profit.

Example2: Fraud detection in credit cards.

credit card companies have a history of your purchases and know geographically where those purchases have been made. If all of a sudden some purchases are made in a city far from where you live, the credit card company put an alert for possible fraud since the data mining shows that you don't normally make purchases in that city. Then, the credit card company can disable your card for that transaction or just put a flag on your card for suspicious activity.

Data warehousing vs Data mining

Data warehousing is a process that must occur before any data mining can take place. data warehousing is the process of compiling and organizing data into one common database and data mining is the process of extracting meaningful data from that database. Data mining process relies on the data compiled in the data warehousing phase in order to detect meaningful patterns.