PS02CMCA23

(DBMS-II)

UNIT - 4

Introduction Nested Relational Databases

```
CREATE OR REPLACE TYPE employee_type AS OBJECT
(     EmpNo           Number(3),
      EmpName         VARCHAR(40),
      Desig           VARCHAR(40)   );

CREATE OR REPLACE TYPE employee_table_type AS TABLE OF employee_type;

CREATE TABLE departments
(     DeptNo                    NUMBER(3) PRIMARY KEY,
      DeptName                  VARCHAR(15) NOT NULL UNIQUE,
      Location                  VARCHAR(15),
      Employees                 employee_table_type
) NESTED TABLE Employees STORE AS employees_nested_table;
```

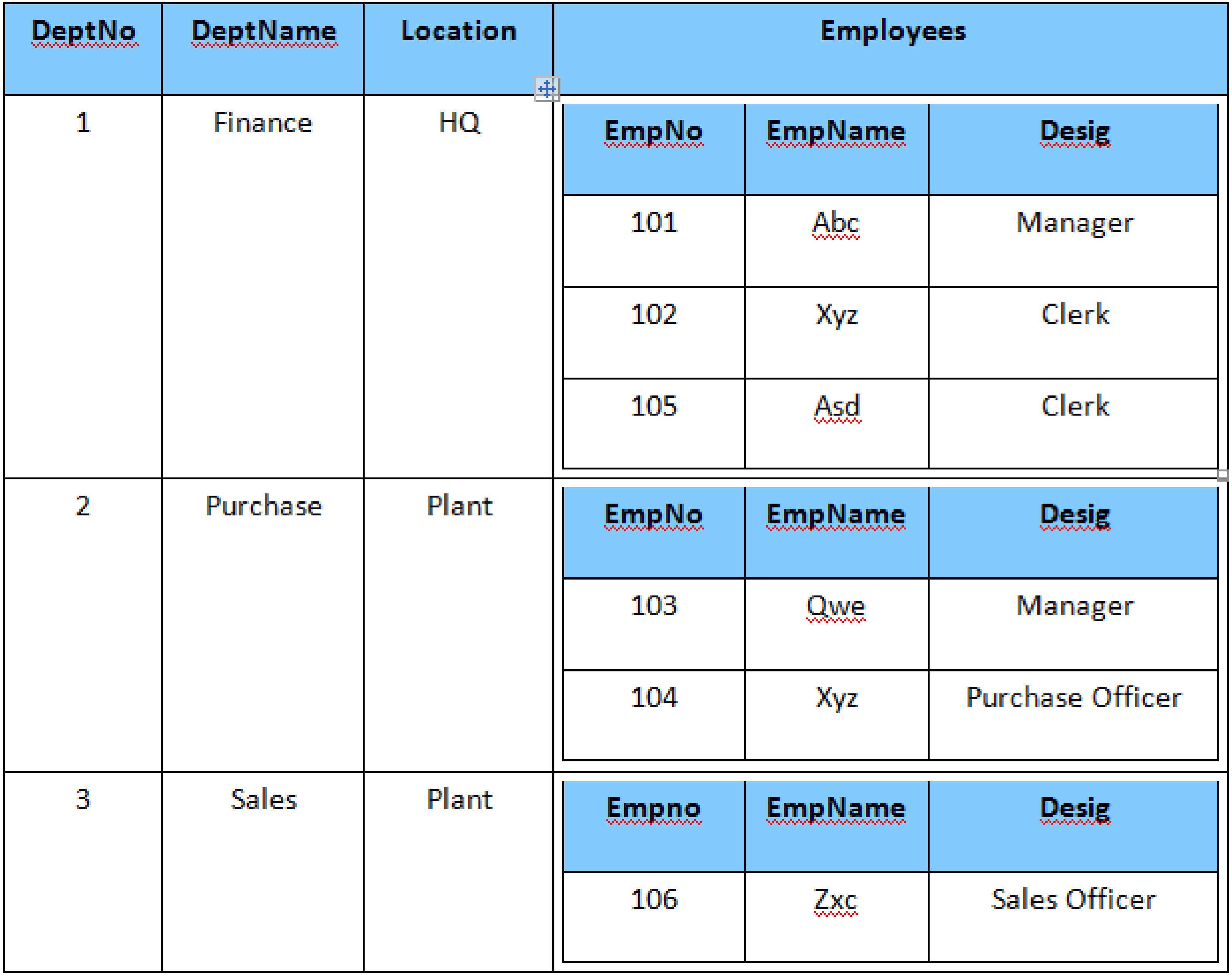
> Desc Departments

```
Name                                      Null?         Type
----------------------------------------- -------- ----------------------------
DEPTNO                                     NOT NULL      NUMBER(3)
DEPTNAME                                   NOT NULL      VARCHAR2(15)
LOCATION                                                 VARCHAR2(15)
EMPLOYEES                                                EMPLOYEE_TABLE_TYPE
```

Nested Table

```
insert into departments values
(      1, 'Finance', 'HQ',
       Employee_table_type (
                              Employee_type(101, 'Abc', 'Manager'),
                              Employee_type(102, 'Xyz', 'Clerk'),
                              Employee_type(105, 'Asd', 'Clerk')
                              )
)
/
insert into departments values
(  2, 'Purchase', 'Plant',
  Employee_table_type (
              Employee_type(103, 'Qwe', 'Manager'),
              Employee_type(104, 'Xyz', 'Purchase Officer')
              )
)
/
insert into departments values
(      3, 'Sales', 'Plant',
       Employee_table_type (
                              Employee_type(106, 'Zxc', 'Sales Officer')   )
)
/
```

Nested Table



<table>
<tr><th>DeptNo</th><th>DeptName</th><th>Location</th><th>Employees</th></tr>
<tr><td>1</td><td>Finance</td><td>HQ</td><td><table><tr><th>EmpNo</th><th>EmpName</th><th>Desig</th></tr><tr><td>101</td><td>Abc</td><td>Manager</td></tr><tr><td>102</td><td>Xyz</td><td>Clerk</td></tr><tr><td>105</td><td>Asd</td><td>Clerk</td></tr></table></td></tr>
<tr><td>2</td><td>Purchase</td><td>Plant</td><td><table><tr><th>EmpNo</th><th>EmpName</th><th>Desig</th></tr><tr><td>103</td><td>Qwe</td><td>Manager</td></tr><tr><td>104</td><td>Xyz</td><td>Purchase Officer</td></tr></table></td></tr>
<tr><td>3</td><td>Sales</td><td>Plant</td><td><table><tr><th>Empno</th><th>EmpName</th><th>Desig</th></tr><tr><td>106</td><td>Zxc</td><td>Sales Officer</td></tr></table></td></tr>
</table>

Materialized Views

- A regular view does not store any data. On the other hand, a materialized view actually **stores data** in it
- The power of materialized views comes from the fact that, once created, Oracle can **automatically synchronize** a materialized view's data **with its source information** as required with little or no programming effort.
- It is **used to summarize, pre-compute, replicate, and distribute data.**
- Best suitable in environment such as **data warehousing, decision support, and distributed, or mobile computing**
- You can use materialized views to **increase the speed of queries** on very large databases. Queries to large databases often involve joins between tables, aggregations such as SUM, or both. These operations are expensive in terms of time and processing power. The type of materialized view you create **determines how the materialized view is refreshed and used by query rewrite.**

Syntax of Materialize View

-
  ```
  CREATE MATERIALIZED VIEW view-name
  BUILD [IMMEDIATE | DEFERRED]
  REFRESH [FAST | COMPLETE | FORCE ]
  ON [COMMIT | DEMAND ]
  [[ENABLE | DISABLE] QUERY REWRITE] AS SELECT ...;
  ```
- Build Option
  - BUILD DEFERRED : Create the materialized view definition but **without data.**
  - BUILD IMMEDIATE : Create the materialized view and then populate it **with data**.

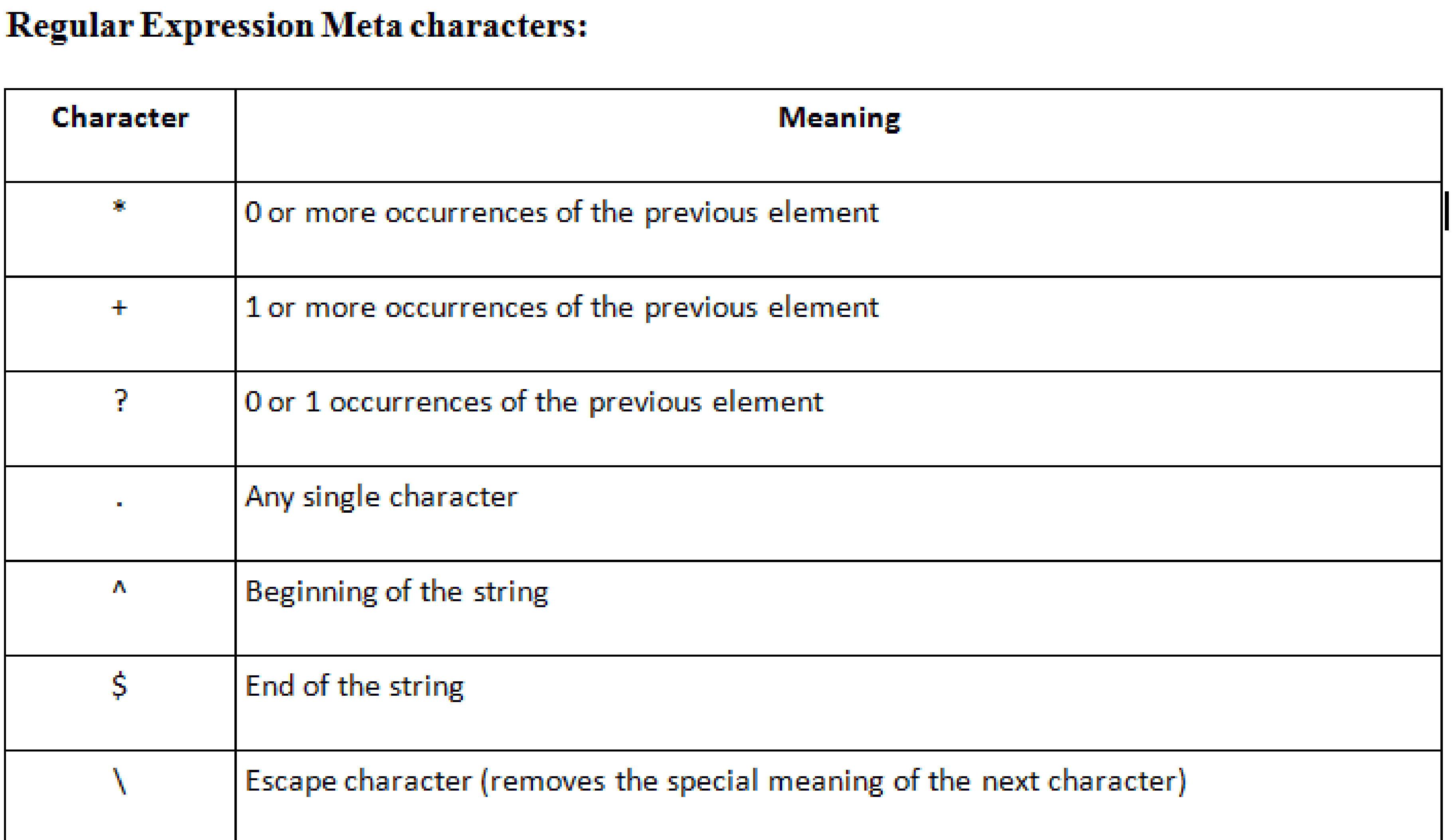
- The following refresh types are available.
  - FAST : **Only changes are reflected** in materialized view data.
  - COMPLETE : **Reading details tables to computer the result**.
  - FORCE : **Default option**. First it tries for Fast than tries for Complete.

- A refresh can be triggered in one of two ways.
  - ON COMMIT : The refresh is triggered by a committed **data change in one of the dependent tables.**
  - ON DEMAND : The refresh is initiated by a manual request or a **scheduled task**.

- The **QUERY REWRITE** clause tells the optimizer if the materialized view should be consider for query rewrite operations. If query rewrite is not enabled, you would only see a performance benefit from using the materialized view if you explicitly queried the materialized view rather than querying the base table

Regular Expressions

- Regular expressions are a pattern-matching language.
- They are far more powerful than the LIKE operator
- Regular expressions are used in a wide variety of tools



Regular Expression Meta characters:

Character	Meaning
*	0 or more occurrences of the previous element
+	1 or more occurrences of the previous element
?	0 or 1 occurrences of the previous element
.	Any single character
^	Beginning of the string
$	End of the string
\	Escape character (removes the special meaning of the next character)

Character Classes

- [abcd] - Any one character from a, b, c or d
- [a-z] - Any one character between a and z, inclusive
- [a-z0-9] - Any one character from between a and z, inclusive **or** between 0 and 9, inclusive
- [a-zA-Z0-9] - Any one character from between a and z, inclusive **or** A and Z, inclusive **or** between 0 and 9, inclusive
- [a-z0-9ABCD] - Any one character from between a and z, inclusive **or** between 0 and 9, inclusive **or** A, B, C or D
- [^a-z0-9] - Any one character other than between a and z, inclusive **or** between 0 and 9, inclusive

Pattern	**Match Results**
ab*c	ac (Y), abc (Y), abbc (Y), abbbc (Y)
ab+c	ac (N), abc (Y), abbc (Y), abbbc (Y)
ab?c	ac (Y), abc (Y), abbc (N), abbbc (N)
a.c	ac (N), abc (Y), acc (Y), adc (Y), abbc (N)

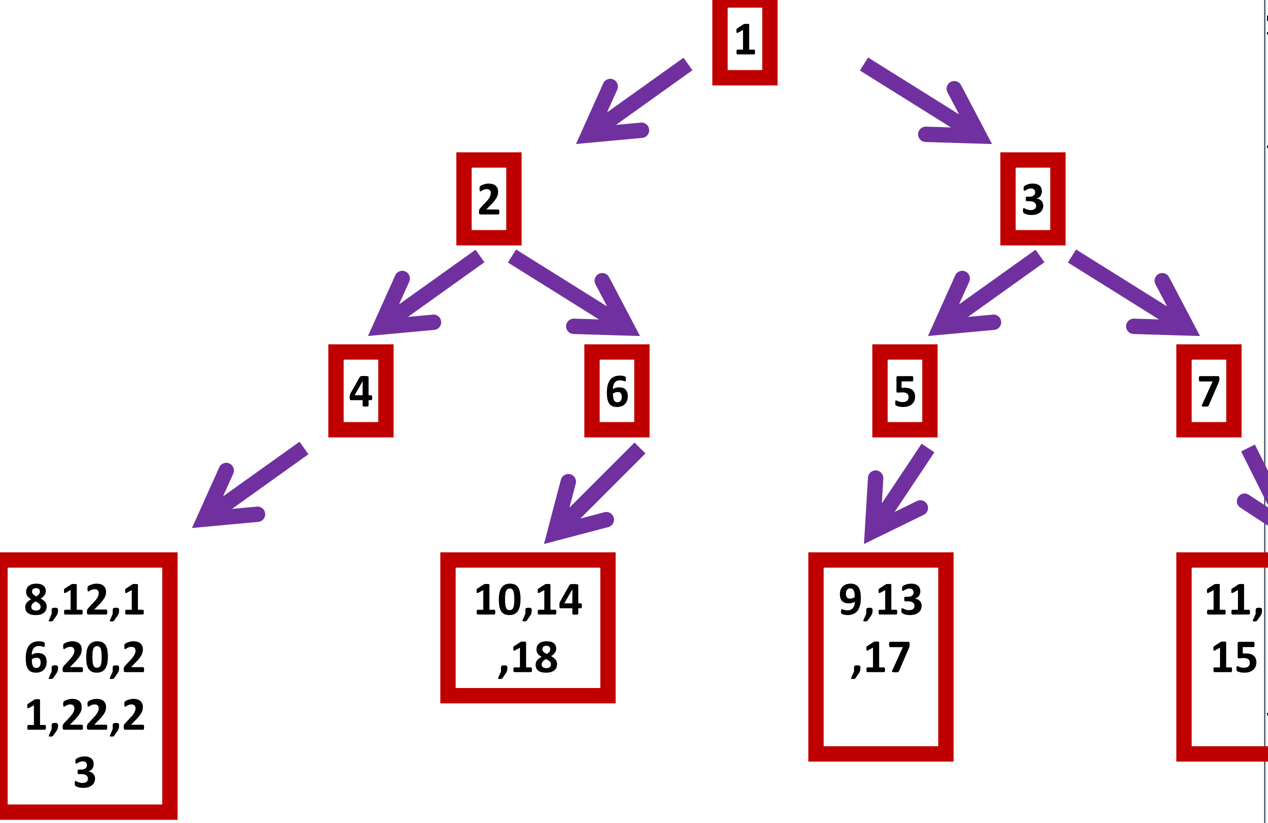
Hierarchical Queries

- Hierarchical queries are used when some kind of hierarchical (tree) structure is stored in a table and the results are desired according to the hierarchy
- Typically, hierarchical data are stored in a table using self-reference, with the child node(s) containing a reference to the parent node
- Examples
  - Every employee in an organization has some superior to whom they report
  - A menu hierarchy stored in a database, with main menu, sub-menu, sub-sub-menu, and so on.

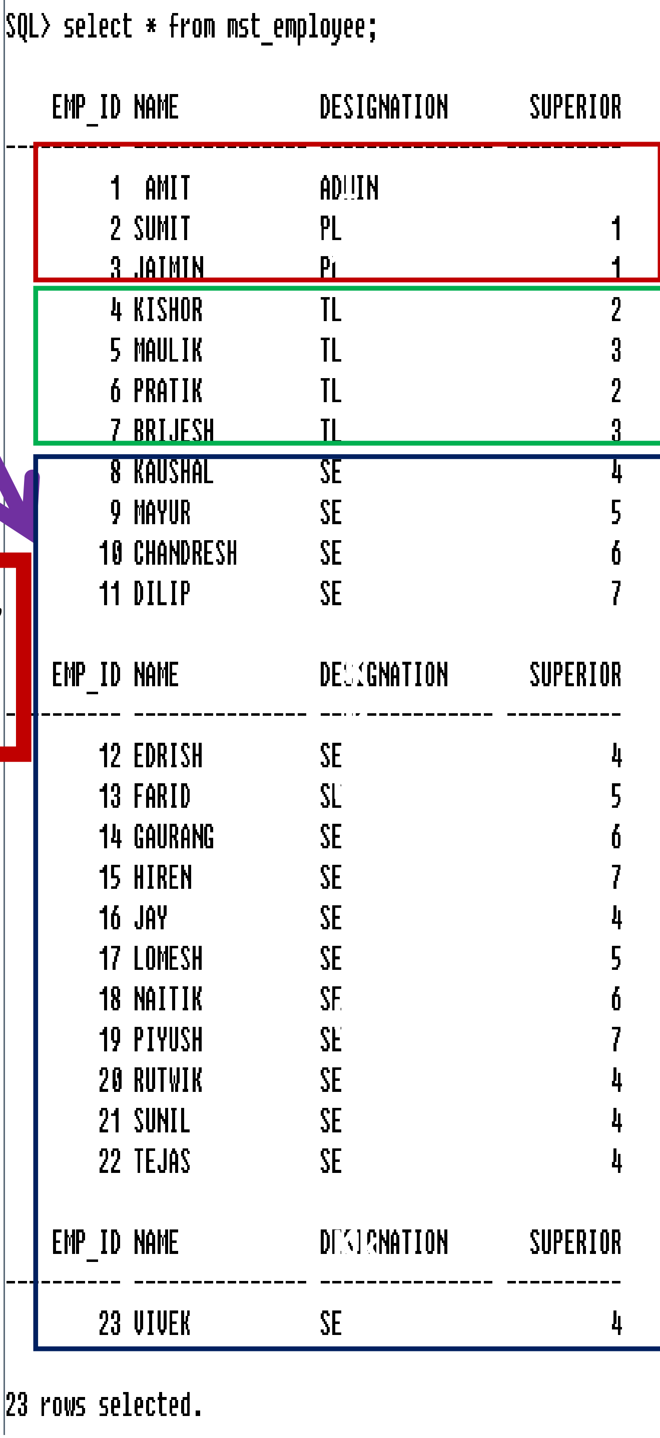
Mst_Employee Table with Data

```
BEGIN
INSERT INTO mst_employee VALUES (1, ' AMIT', 'ADMIN', NULL);
INSERT INTO mst_employee VALUES (2, 'SUMIT', 'PL', 1);
INSERT INTO mst_employee VALUES (3, 'JAIMIN', 'PL', 1);
INSERT INTO MST_employee VALUES (4,'KISHOR','TL', 2);
INSERT INTO MST_employee VALUES (5,     'MAULIK',         'TL', 3);
INSERT INTO MST_employee VALUES (6,     'PRATIK',         'TL', 2);
INSERT INTO MST_employee VALUES (7,     'BRIJESH',    'TL', 3);
INSERT INTO MST_employee VALUES (8,     'KAUSHAL',    'SE', 4);
INSERT INTO MST_employee VALUES (9,     'MAYUR',      'SE', 5);
INSERT INTO MST_employee VALUES (10,    'CHANDRESH',   'SE', 6);
INSERT INTO MST_employee VALUES (11,    'DILIP',      'SE', 7);
INSERT INTO MST_employee VALUES (12, 'EDRISH', 'SE', 4);
INSERT INTO MST_employee VALUES (13, 'FARID', 'SE', 5);
INSERT INTO MST_employee VALUES (14, 'GAURANG', 'SE', 6);
INSERT INTO MST_employee VALUES (15, 'HIREN', 'SE', 7);
INSERT INTO MST_employee VALUES (16, 'JAY', 'SE', 4);
INSERT INTO MST_employee VALUES (17, 'LOMESH', 'SE', 5);
INSERT INTO MST_employee VALUES (18, 'NAITIK', 'SE', 6);
INSERT INTO MST_employee VALUES (19, 'PIYUSH', 'SE', 7);
INSERT INTO MST_employee VALUES (20, 'RUTWIK', 'SE', 4);
INSERT INTO MST_employee VALUES (21, 'SUNIL', 'SE', 4);
INSERT INTO MST_employee VALUES (22, 'TEJAS', 'SE', 4);
INSERT INTO MST_employee VALUES (23, 'VIVEK', 'SE', 4);
END;
/
```

```
Create table mst_employee
(emp_id number(5) primary key,
name varchar2(80),
designation varchar2(80),
superior number(5),
FOREIGN KEY(superior) REFERENCES
mst_employee(emp_id)  )
```



Hirerchical Structure of Employee Id and Parent Id of mst_employee Table



```
SQL> select * from mst_employee;

EMP_ID NAME            DESIGNATION     SUPERIOR
------ --------------- --------------- ----------
     1 AMIT            ADMIN
     2 SUMIT           PL                       1
     3 JAIMIN          PL                       1
     4 KISHOR          TL                       2
     5 MAULIK          TL                       3
     6 PRATIK          TL                       2
     7 BRIJESH         TL                       3
     8 KAUSHAL         SE                       4
     9 MAYUR           SE                       5
    10 CHANDRESH       SE                       6
    11 DILIP           SE                       7

EMP_ID NAME            DESIGNATION     SUPERIOR
------ --------------- --------------- ----------
    12 EDRISH          SE                       4
    13 FARID           SE                       5
    14 GAURANG         SE                       6
    15 HIREN           SE                       7
    16 JAY             SE                       4
    17 LOMESH          SE                       5
    18 NAITIK          SE                       6
    19 PIYUSH          SE                       7
    20 RUTWIK          SE                       4
    21 SUNIL           SE                       4
    22 TEJAS           SE                       4

EMP_ID NAME            DESIGNATION     SUPERIOR
------ --------------- --------------- ----------
    23 VIVEK           SE                       4

23 rows selected.
```

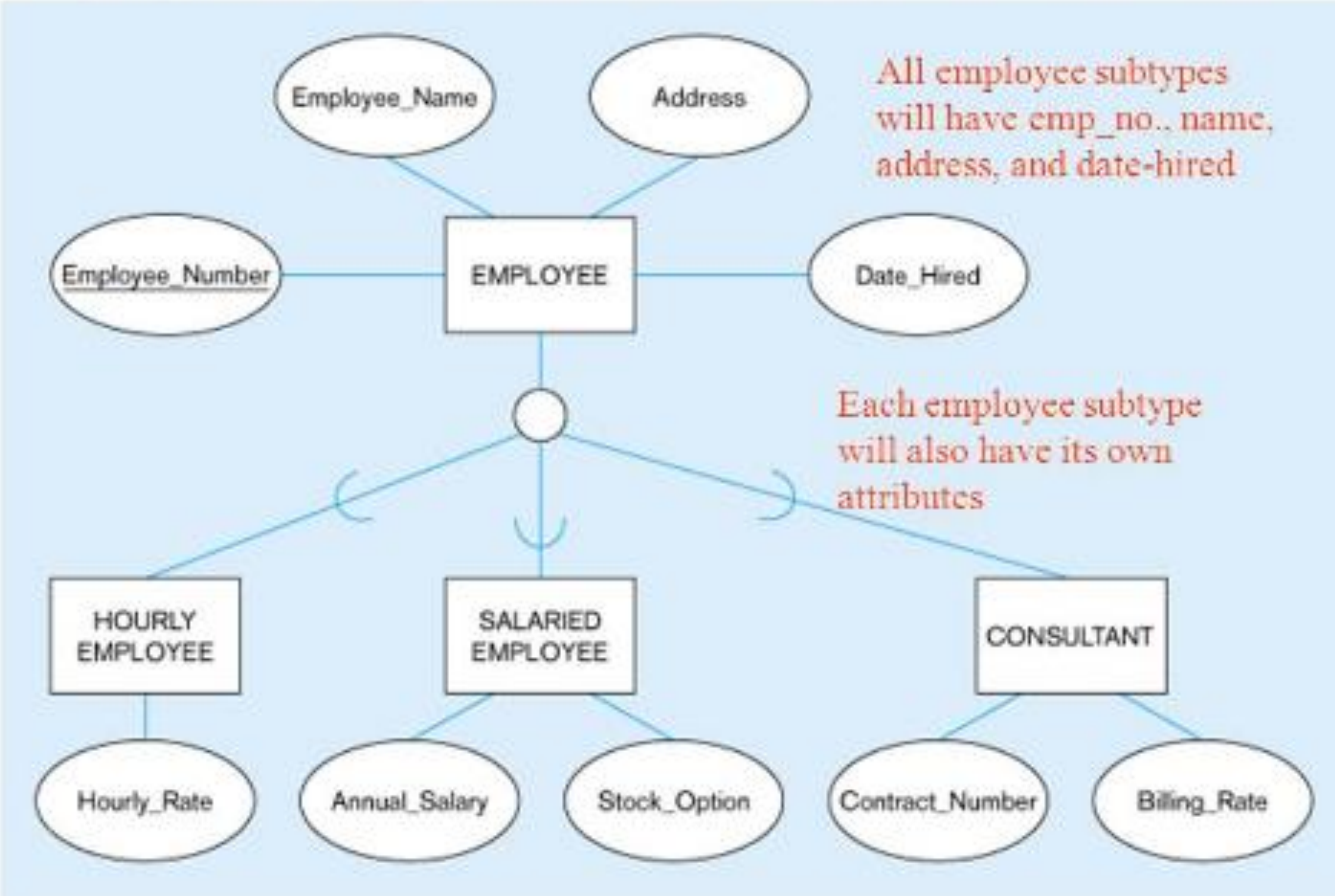
UNIT - 2

EER (Extended Entity Relationship) Model

- <u>**SubType and SuperType**</u>
  - **SuperType** - An entity type that relates to one or more subtypes.
  - **SubType** - A subgroup of entities with unique attributes.
  - One entity type might be a subtype of another (subclasses in OO programming)
    - **Manager** is a subtype of **Employee**
    - **Technician** is a subtype of **Employee**
    - **Engineer** is a subtype of **Employee**
  - This relationship is called ***IsA***
    - The two entities related by **IsA** are always descriptions of the same real-world object
- The **Supertype** entity is connected to the **Subtype** entities through lines and a circle.
- The **U-symbol** indicates the Subtype is a subset of the Supertype.

EER (Extended Entity Relationship) Model

Employee supertype with three subtypes

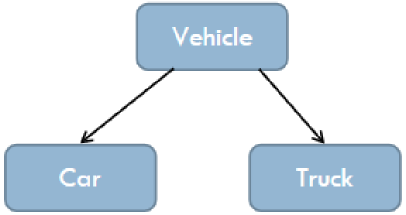


All employee subtypes will have emp_no., name, address, and date-hired

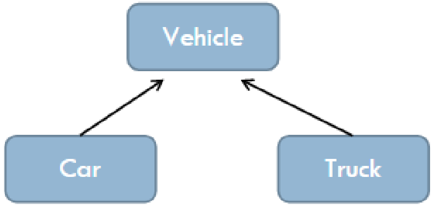
Each employee subtype will also have its own attributes

EER (Extended Entity Relationship) Model

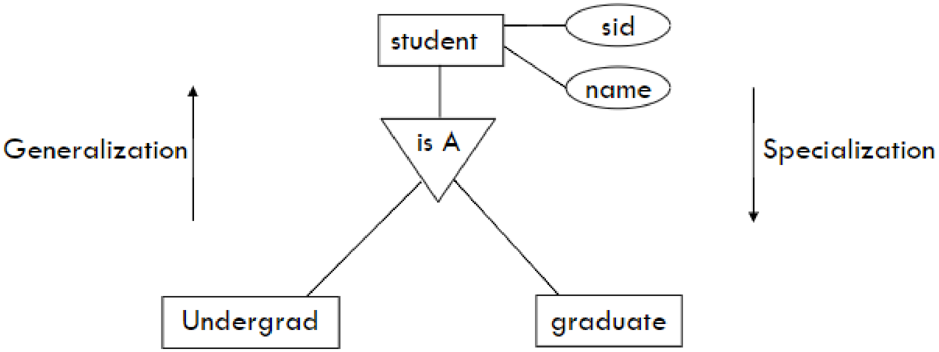
- **Specialization & Generalization**
  - It is the **process of defining a set of subclasses of an entity type**; this entity type is called the superclass of the **specialization**.
  - The set of subclasses that form a specialization is defined on the basis of some **distinguishing characteristic of** the entities in the **superclass**



Arrows pointing to the specialized subclasses represent a specialization

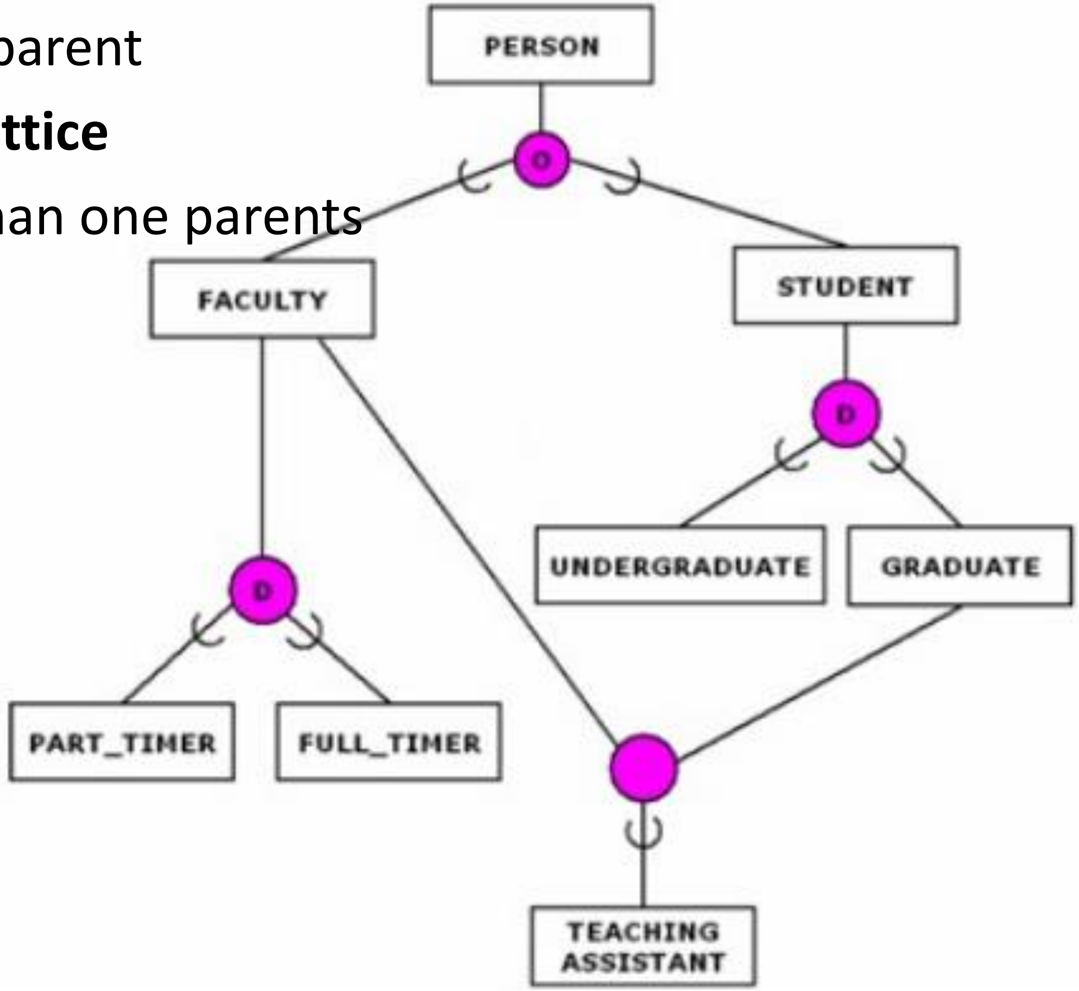


Arrow pointing to the generalized superclass represents a generalization



EER (Extended Entity Relationship) Model

- **Specialization Hierarchy / Class Hierarchy**
  - Each subclass has only one parent
- **Specialization Lattice / Class Lattice**
  - A subclass can have more than one parents
  - Multiple inheritance



Union Type : The subclass represents a collection of objects that is a subset of the **UNION** of distinct entity types. Such a subclass is called Union Type or Category

E.g. The owner entity type of a registered vehicle has an entity set that is a subset of the union of the entity sets of the person, bank and company entity types.

UNIT - 1

Transaction Processing

- A transaction includes one or more database operations like insertion, deletion, updating or retrieval.
- If database operations in a transaction only retrieve the data, the transaction is called a **read-only transaction.**
- If database operations in a transaction changes the data, the transaction is called a **read-write transaction**
- **Basic operations**
  - **X :** Database field **or** Database record.
  - **Assumption:** Database Field and Program Variable Named **X**
  - **Read_item (X) :** Read a database item named X into a program variable X.
  - **Write_item (X) :** Writes the value of program variable X into the database item named X.
- **Basic unit of data transfer from disk to main memory is one block.**

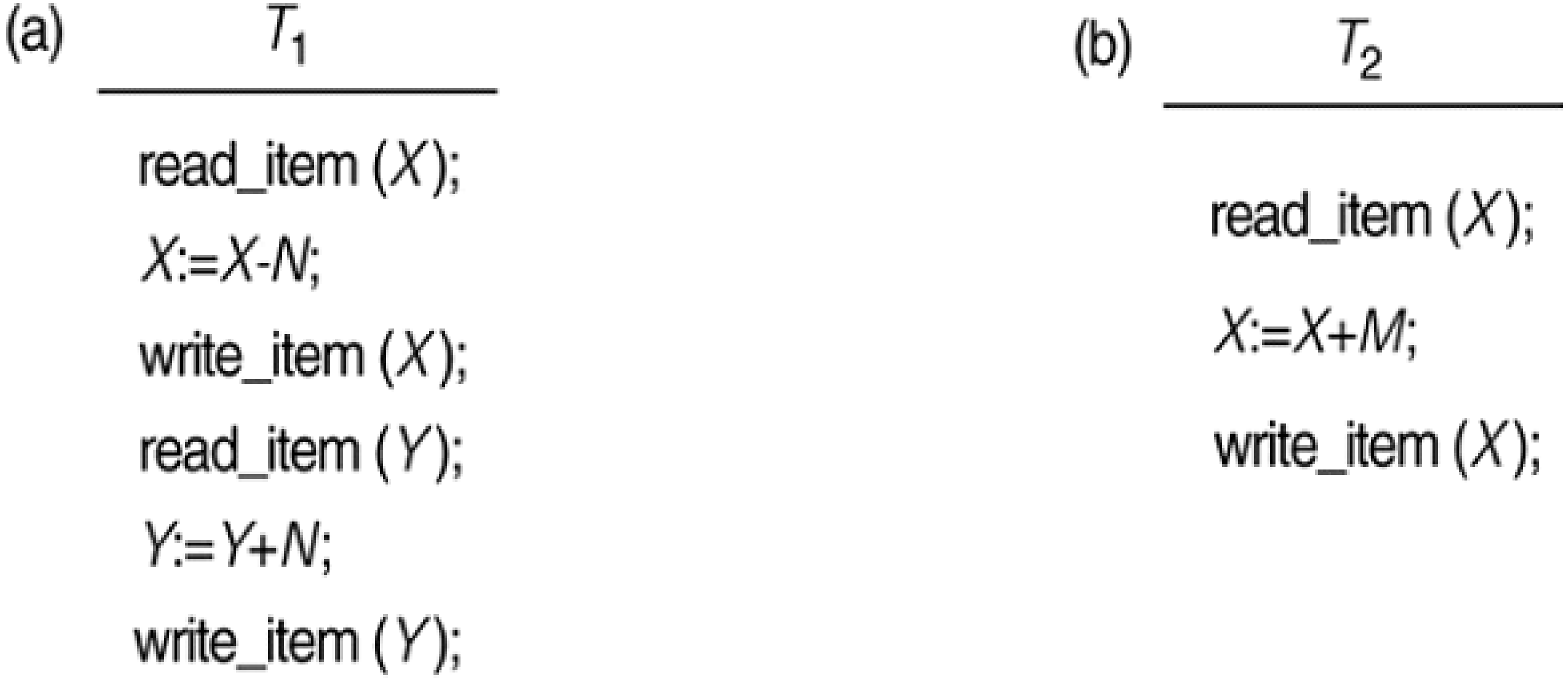
Transaction Processing

- **Two Sample Transaction**

E.g. Airline reservation system.

Transaction T1 transfers N reservations from one flight (X) to another flight (Y)

Transaction T2 Reserves the M seats on the first flight (X)



(a) T_1

```
read_item (X);
X:=X-N;
write_item (X);
read_item (Y);
Y:=Y+N;
write_item (Y);
```

(b) T_2

```
read_item (X);
X:=X+M;
write_item (X);
```

- Transactions submitted by the various users may execute concurrently and may access and update the same database items. **If concurrent execution is uncontrolled, it may lead to inconsistent database.**

Why Transaction Recovery is Needed?

- **Types of Failures**

1. **A computer failure (System crash):**
   - A hardware, software, or network error occurs in the computer during execution. E.g. Memory failure, HDD failure, OS failure
2. **A transaction or system error.**
   - Operations like **integer overflow** or **division by zero**,
   - **Erroneous parameter values**
3. **Exception conditions detected by the transaction**
   - An exception like insufficient account balance may cause a fund withdrawal, to be canceled.
4. **Disk failure:**
   - Some disk blocks may lose their data because of a read or write failure **or** because of a **disk read/write head crash**.
5. **Physical problems and catastrophes**
   - Power failure, Fire, Theft, Overwriting disks or tapes by mistake

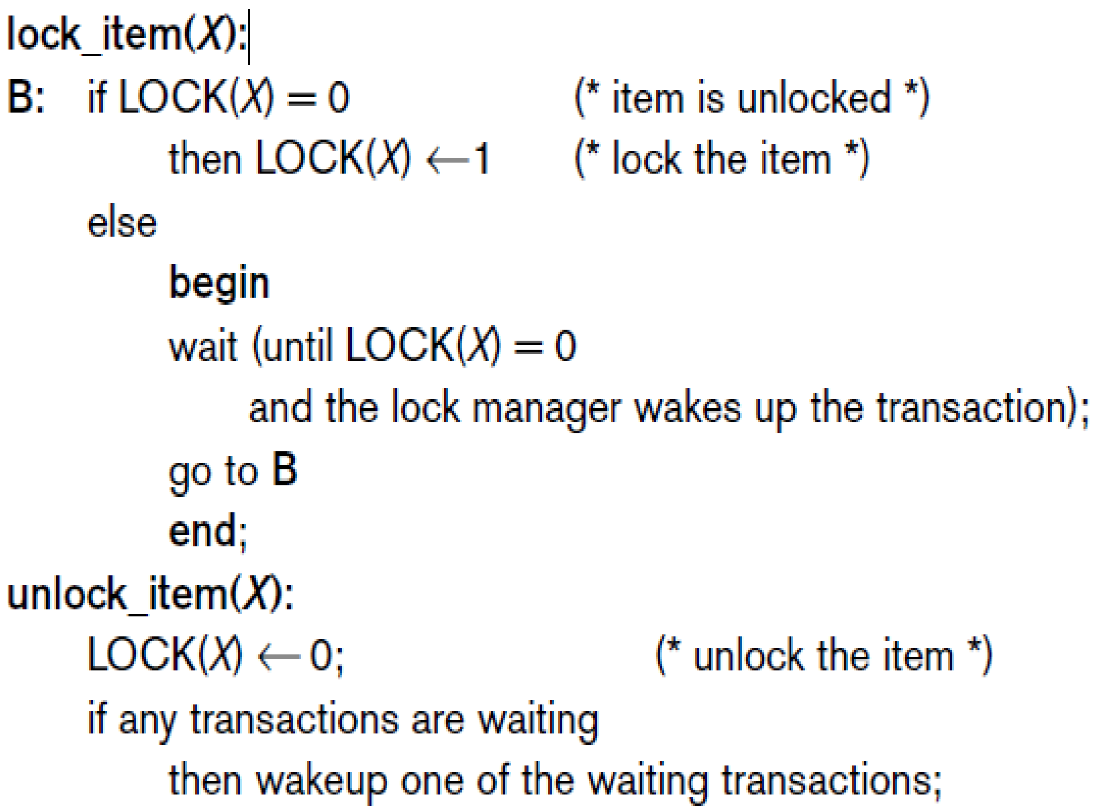
Concurrency Control : : Binary Locking

Technique of locking data items to prevent multiple transactions from accessing the items concurrently.

- **Binary Lock:**
  - It has two states or values: **locked and unlocked.**
  - **A distinct lock** is associated **with each** database **item *X*.**
  - If the **value of** the **lock on X is 1**, **item X cannot be accessed** by a database operation that requests the item.
  - If the **value of** the **lock on X is 0**, **item X can be accessed** when requested, **and** the **lock value is changed to 1**.

Concurrency Control : Binary Locking

- **Two operations : lock_item & unlock_item**
- **A transaction requests lock_item(X) operation**.
  - **If LOCK(X) = 1**, **transaction** is forced to **wait**.
  - **If LOCK(X) = 0**, **transaction locks the item**
- When the transaction operations related to locked item is completed, it issues an **unlock_item(X) operation**.



```
lock_item(X):
B:  if LOCK(X) = 0              (* item is unlocked *)
        then LOCK(X) ← 1        (* lock the item *)
    else
        begin
        wait (until LOCK(X) = 0
            and the lock manager wakes up the transaction);
        go to B
        end;
unlock_item(X):
    LOCK(X) ← 0;                    (* unlock the item *)
    if any transactions are waiting
        then wakeup one of the waiting transactions;
```

Figure 22.1
Lock and unlock operations for binary locks.

Concurrency Control : Shared/Exclusive OR Read/Write Locks

- **Three locking operations:**
  - read_lock(X)
  - write_lock(X) &
  - unlock(X).
- A **read-locked (share-locked)** item read by other transactions.
- A **write-locked** (**exclusive-locked)** item not read by other transaction

Each record in the lock table will have four fields:

<Data_item_name, LOCK, No_of_reads, Locking_transaction(s)>.

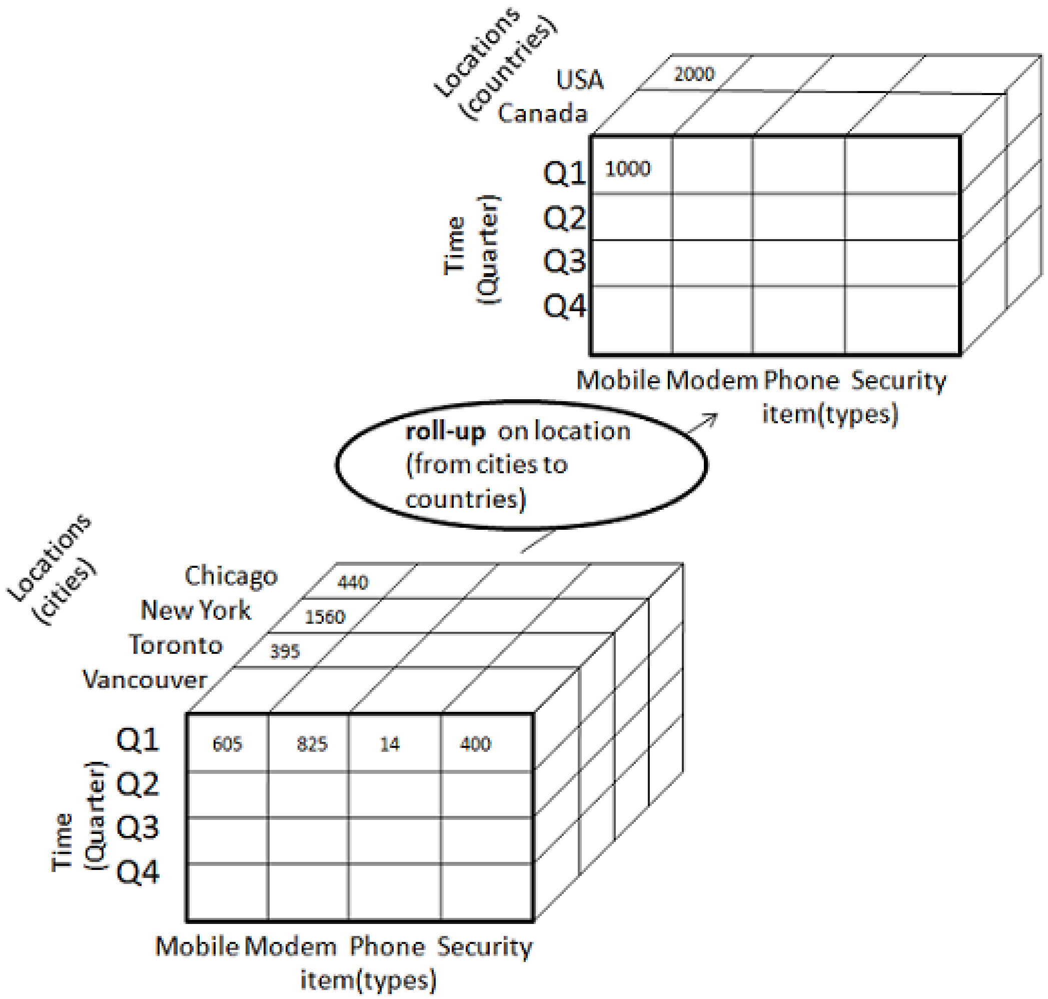
- The **value** (state) **of LOCK** is either **read-locked or write-locked**
- If **write-locked**, **value of locking_transaction(s)** is single transaction.
- If **read-locked**, **value of locking_transaction(s)** may be one or more transaction.

Data Warehousing

- “A data warehouse is constructed by integrating data from multiple heterogeneous sources.
- It is separate from the organization's operational database.
- It possesses consolidated historical data, so data is not frequently updated.
- It helps executives to organize, understand, and use data to take strategic decisions.
- An OLAP (Online Analytical Processing) tool’s query needs only **read only** access of stored data.

OLAP Operations : Roll-up & Drill-down

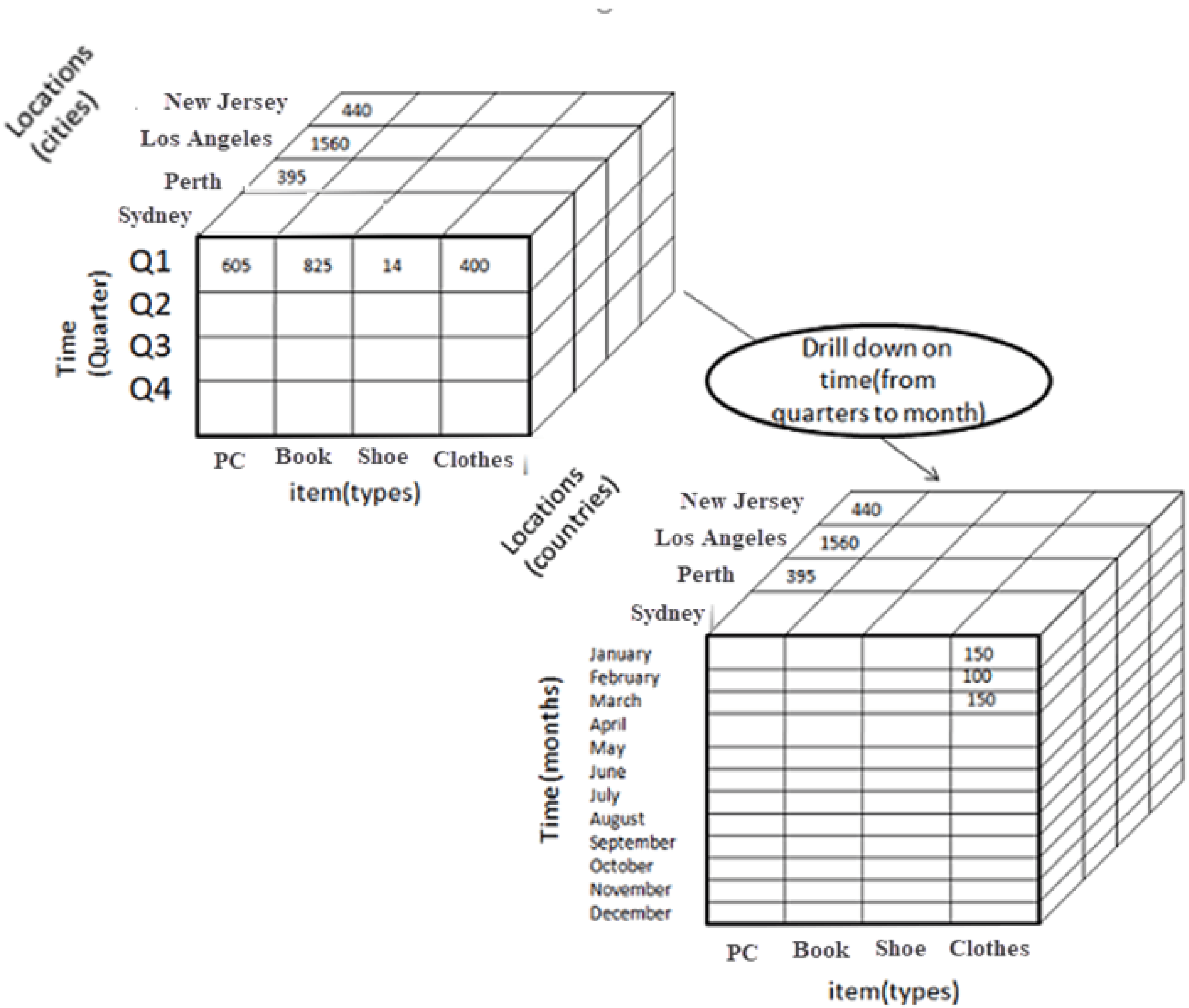
Roll-up : Decreases a number of dimensions - removes row headers.



roll-up on location (from cities to countries)

OLAP Operations : Roll-up & Drill-down

Drill-Down : Increases a number of dimensions - adds new headers

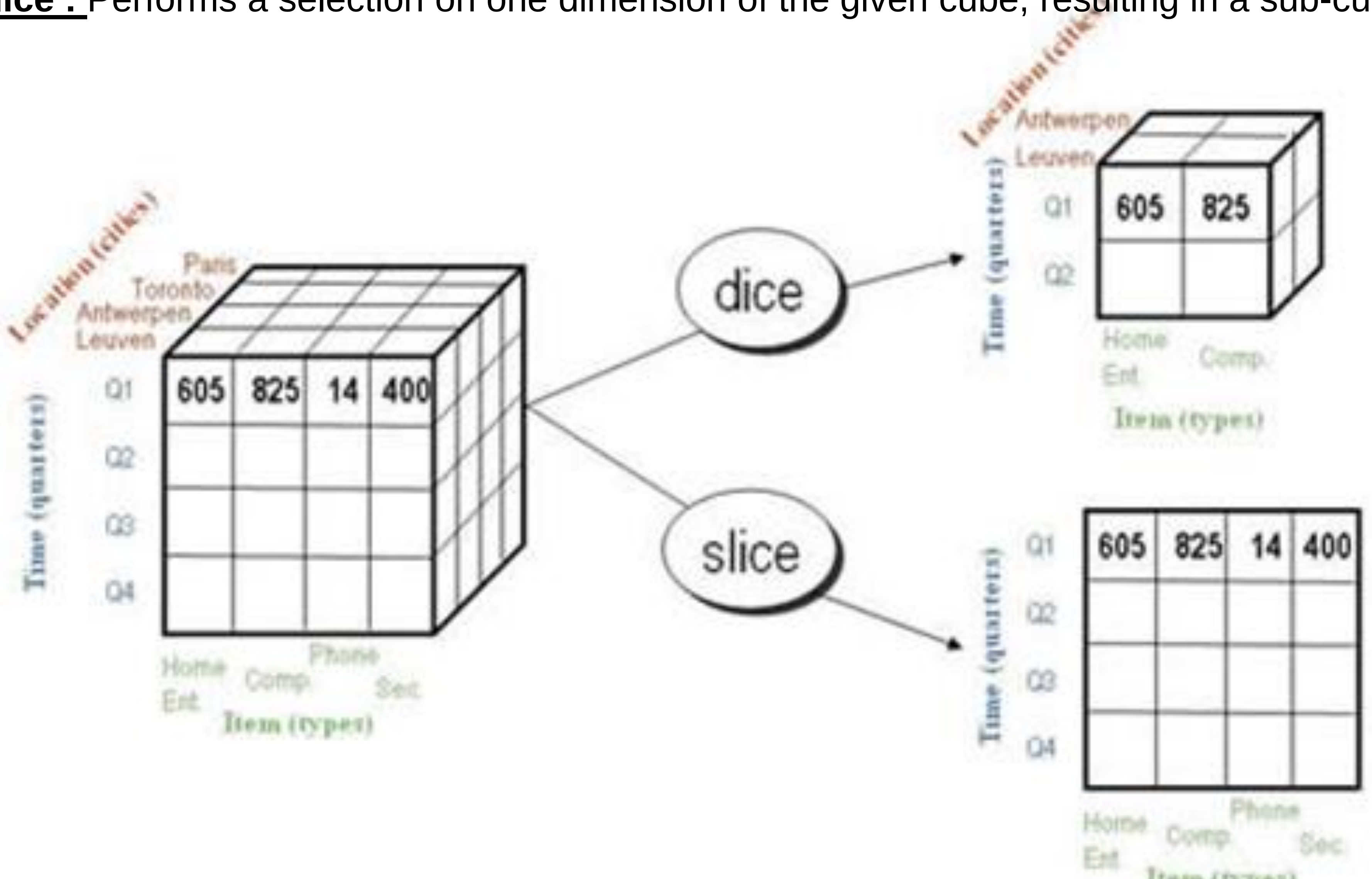


Drill down on time(from quarters to month)

OLAP Operations : Roll-up & Drill-down

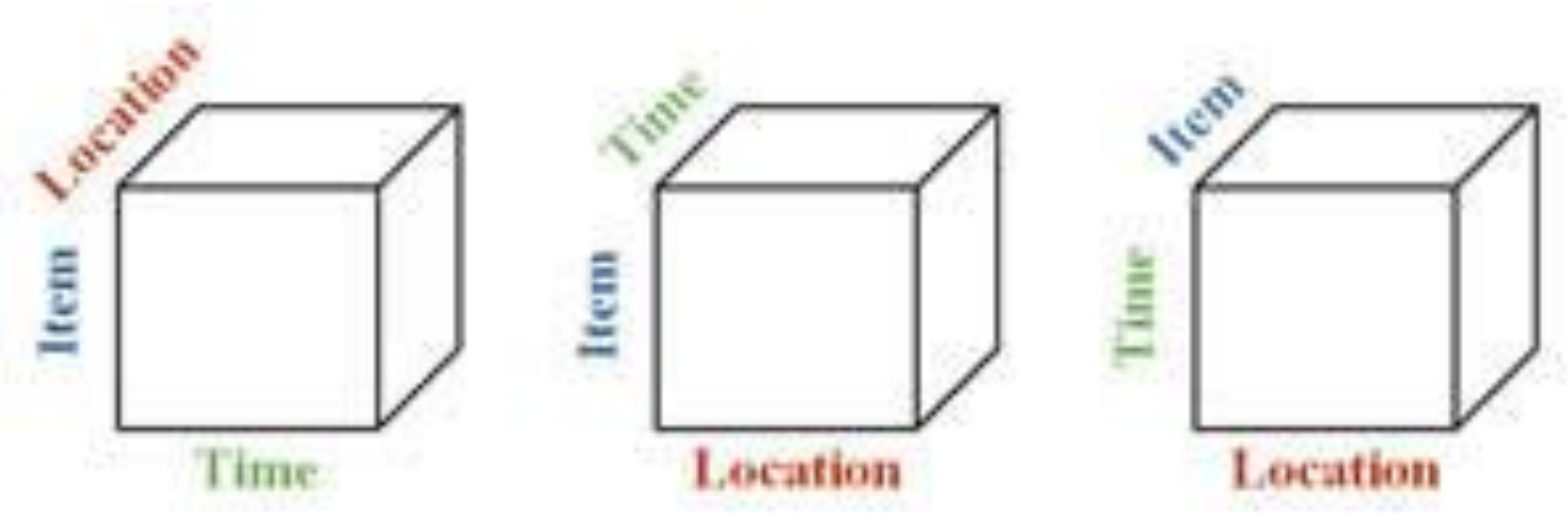
Dice : To select condition on more than one dimension

Slice : Performs a selection on one dimension of the given cube, resulting in a sub-cube



OLAP Operation : Pivot

Pivot or Rotate : Rotates the data axis to view the data from different perspectives.



Data Warehouse Applications

- **Manufacturing & Distribution Industry**
  - Identify profitable product lines
  - Analyze previous data and customer feedback to evaluate the weaker product lines and eliminate them.
- **Banking services**
  - Analysis of card holder’s transactions and introduce some offers.
- **Consumer goods**
  - Prediction of consumer trends
  - Inventory management
  - Market and advertising research
- **Government and Education**
  - The state government to human resources like recruitment and accounting like payroll management.
- **Health care**
  - Generate patient reports, share data with insurance company, medical aid services etc.
- **Telephone Industry**
  - Tracking of customer query
  - Analysis of customer’s data usage patterns for advertising campaign.

Data Mining

- "A process of extracting valid, previously unknown, understandable and actionable information from large databases and using it to make crucial business decisions."

OR

- A procedure of extracting information from huge sets of data.

OR

- It is mining knowledge from data.

Functions Of Data Mining

- **Summarization**
- **Classification**
- **Regression**
- **Association**
- **Clustering**

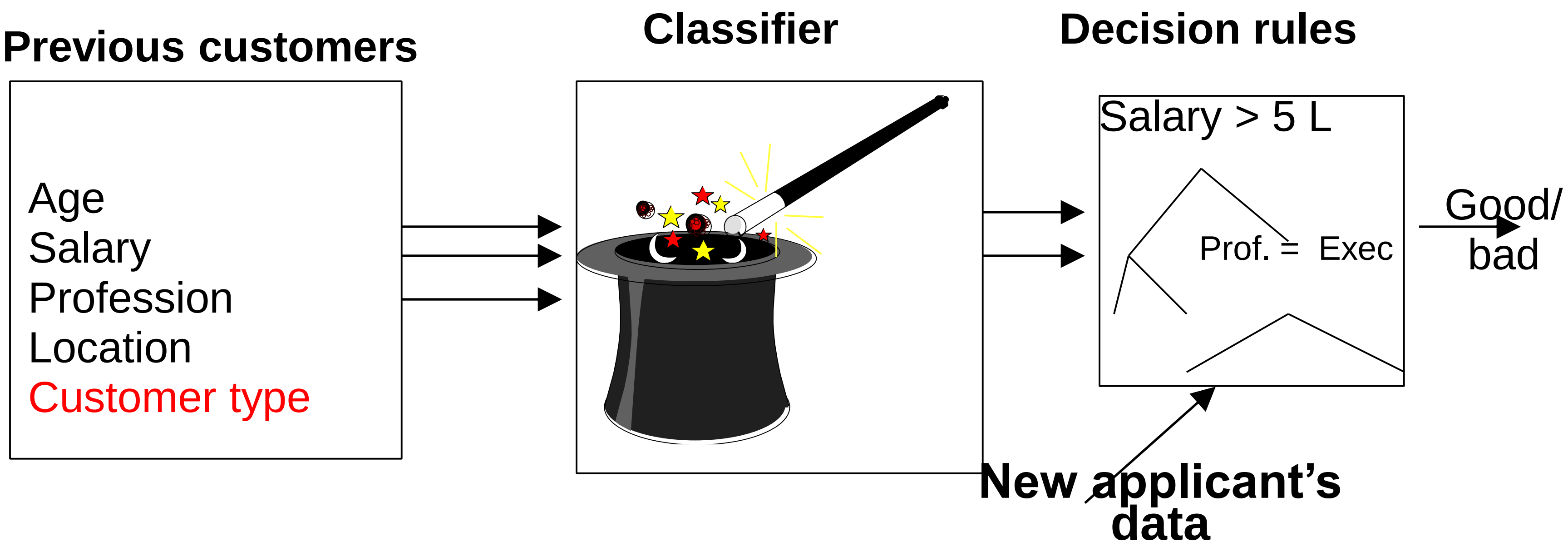
Summarization Task

- It is the process of representing the collected data in an accurate and compact way.

- **E.g.** Display the data as a graph and statistical functions like mean, median, mode, standard deviation etc.

Classification Task (Supervised Learning)

- With the previous history, prediction of future.
- E.g. With customers old data about payments, predict applicant’s loan eligibility.



Previous customers

Age
Salary
Profession
Location
Customer type

Classifier

Decision rules

Salary > 5 L

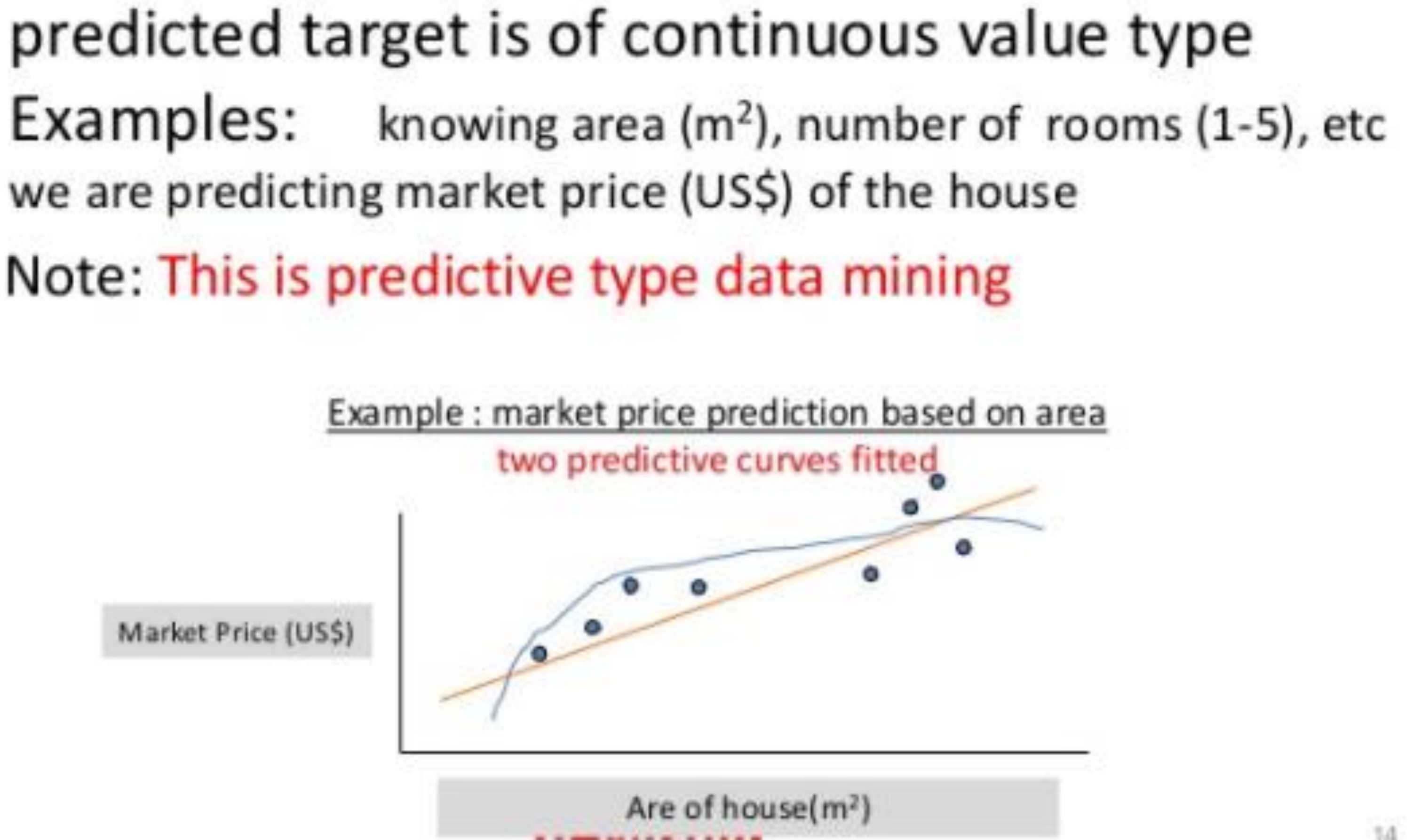
Prof. = Exec

Good/
bad

New applicant’s data

Regression / Prediction

- To determine the strength of the relationship between two variables
- **E.g.** Marketing manager needs to predict how much a given customer will spend during a sale at company.



predicted target is of continuous value type

Examples: knowing area (m^2), number of rooms (1-5), etc

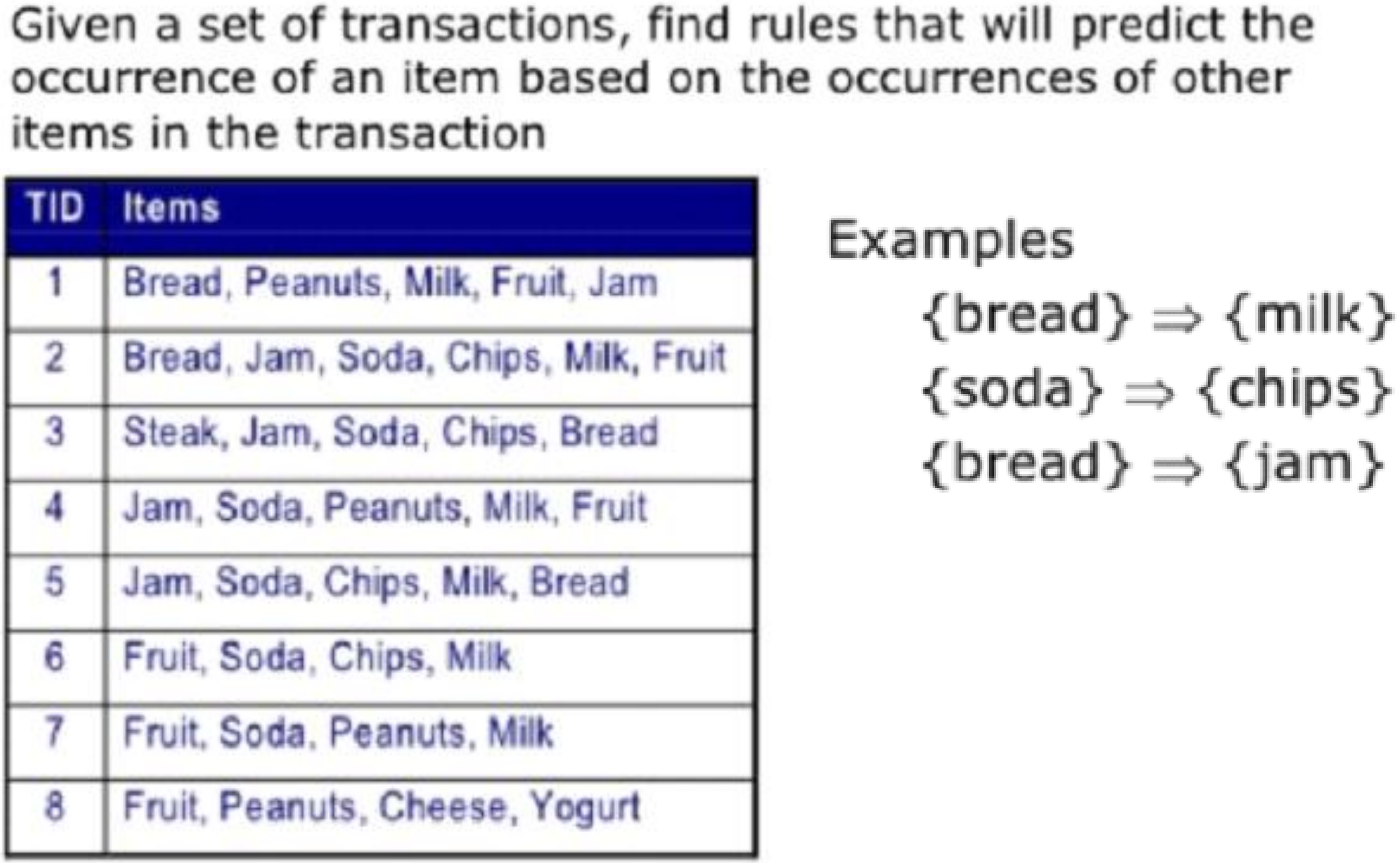
we are predicting market price (US$) of the house

Note: This is predictive type data mining

Example : market price prediction based on area

two predictive curves fitted

Association rules



Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples

$\{bread\} \Rightarrow \{milk\}$

$\{soda\} \Rightarrow \{chips\}$

$\{bread\} \Rightarrow \{jam\}$

Clustering Task (Unsupervised Learning)

Unsupervised learning when old data with class labels not available

- **E.g. 1.** Suppose you have a basket and it is filled with some fresh fruits and your task is to arrange the same type fruits at one place.

  You will select any of following physical characteristics of fruits.
  - Based on Colour
  - Based on Size
  - Based on Colour and Size

2. When introducing a new product.
  - Group existing customers based on payment history
  - Group customers bases on features they used.

Data Mining Application

- **Market Analysis and Management**
  - Determine what kind of people buy what kind of products.
  - Association/correlations between product sales.
  - Predict what factors will attract new customers
- **Corporate Analysis and Risk Management**
  - Monitoring competitors and market directions
  - Summarizing and comparing the resources
- **Fraud Detection (Credit card & Telecommunication services)**
  - To find the call details (date, duration & place)
  - Analyzes the patterns that deviate from expected norms.
- **Financial Data Analysis**
  - Classification and clustering of customers for targeted marketing.
  - Detection of money laundering and other financial crimes.
- **Retail Industry**
  - Analysis of sales, customers, products, time and region.
  - Analysis of effectiveness of sales campaigns.