Chapter - 9
Advanced Database Concepts.

9.1 Concept of Object oriented and Distributed Database Model.
9.2 Properties of parallel and Distributed Databases.
9.3 Concept of Data warehouse Database.
9.4 Concept of spatial Database.

Object Oriented Database:

Overview of object-oriented concepts.

↳ object oriented database try to maintain a direct correspondance between read-world and database objects so that objects donot lose their integrity and identity and can easily be identified and operated upon.

↳ object has two components
state (value) and behaviour (operations)

↳ In OO databases, objects may have an object structure of arbitrary complexity in order to contain all of the necessary information that describes the object.

↳ In contrast, in traditional database systems, information about a complex object is often scattered over many relations or records, leading

to loss of direct correspondance between a read world object and its database representation

↳ The internal structure of an object in OOPLs includes the specifications of instance variables, which hold the values that define the internal state of the object.

↳ An instance variable is similar to the concept of an attribute, except that instance variables may be encapsulated within the object and thus are not necessarily visible to external users.

↳ Some OO models insist that all operations a user can apply to an object must be predefined. This forces a complete encapsulation of objects.

Object - oriented Data Model (OODM):

This data model is another method of representing real world objects. It considers each object in the world as objects and isolates it from each other. It groups its related functionalities together and allows inhesiting its functionality to other related sub groups.

Element of Object Oriented Data model.

objects:

The real world entities and situations are represented as objects in the object oriented database model.

Attributes and Method.

Every object has certain characteristics. These are represented using attributes. The behaviour of the object is represented using Methods.

Class.

Similar attributes and methods are grouped together using a class. An object can be called as an instance of the class.

Inheritance.

A new class can be derived from the original class. The derived class contains attributes and methods of the original class as well as its own.

Example;

An Example of the object oriented data model is.

Shape, Circle, Rectangle and Triangle are all objects in this model.

Circle has the attributes center and Radius.

Rectangle has the attributes Length and Breath.

Triangle has the attributes Base and Height.

The objects circle, Rectangle and Triangle inhesit from the object shape.

Object-oriented Database (OODB)

A persistent and sharable collection of objects defined by an OODM.

Object-oriented Database Management system (OODBMS)

The manager of an OODB.

Advantages:

↳ Because of its inheritance property, we can re-use the attributes and functionalities. It reduces the cost of maintaining the same data multiple times.

↳ Also, these informations are encapsulated and there is no fear being misused by other objects. If we need any feature we can easily add new class inhesited from parent class and adds new features. Hence it reduces the overhead and maintenance costs.

↳ It becomes more flexible in case of any changes.

↳ Codes are re-used because of inhesitance

↳ Since each class binds its attributes and

its functionality. it is same as representing the real world object. We can see each object as a real entity. Hence it is more understandable.

Disadvantages.

↳ It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.

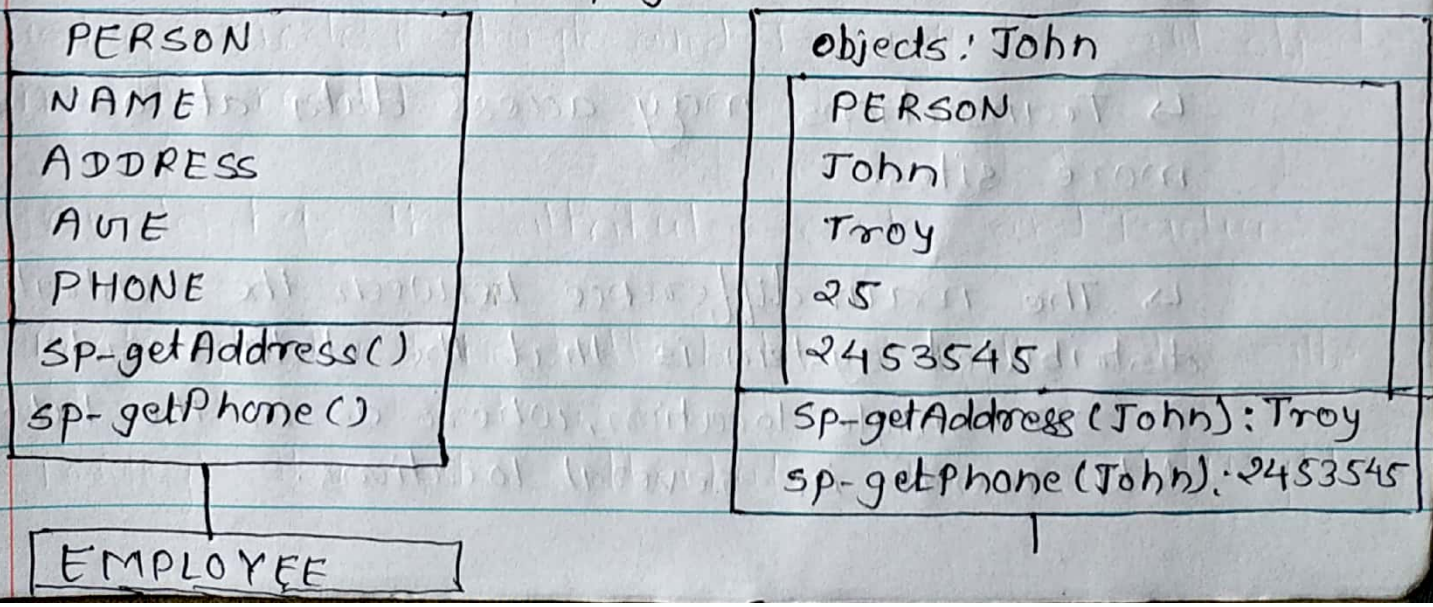
↳ It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.

Class diagram.

↳ Class diagram shows the static structure of an object-oriented model; object classes, internal structure, relationships.

↳ Class diagram showing classes.

CLASS: Person and Employee



PERSON()
EMPLOYEE-ID
EMPLOYEE-TYPE
DEPARTMENT_ID
SP-getDeptDetails()

EMPLOYEE
John()
12121
Engineer
100
SP-getDeptDetails(John)
Manufacture

Distributed Database Management System (DDBMS)

↳ In a distributed database, there are a number of databases that may be geographically distributed all over the world.

↳ A distributed DBMS manages the distributed database in a manner so that it appears as on single database to users.

↳ A distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

↳ Transactions may access data at one or more sites.

↳ The main difference between the centralized & distributed system is that the data reside in on single location, where as in the later, the data reside in several locations.

↳ A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features:

↳ It is used to create, retrieve, update and delete distributed databases.

↳ It synchronizes the database periodically and provides access mechanism by the virtue of which the distributed becomes transparent to the users.

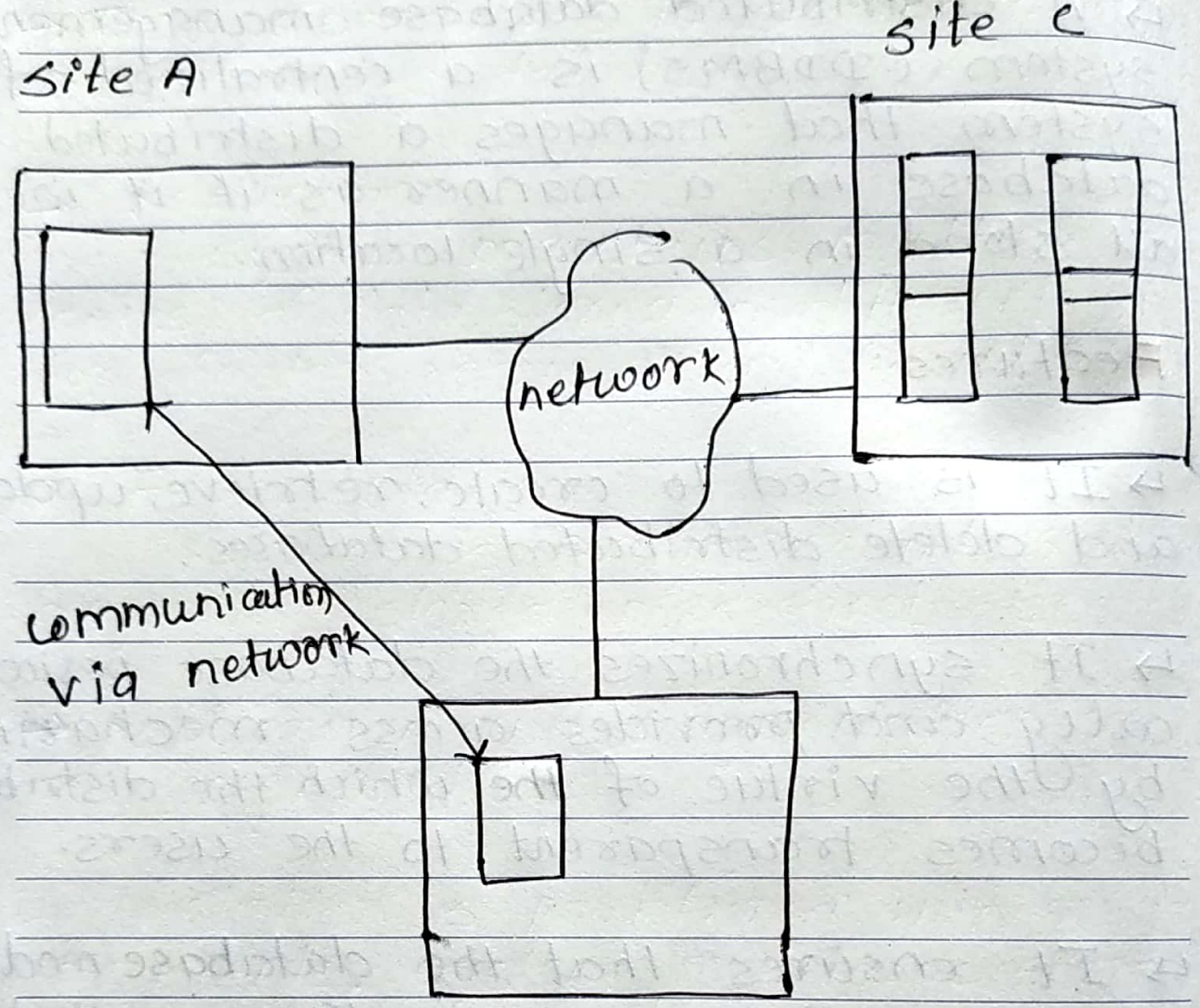
↳ It ensures that the data modified at any site is universally updated.

↳ It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.

↳ It is designed for heterogenous database platforms.

↳ It maintains confidentiality and data integrity of the databases.

Notes



Mainly two type

Heterogeneous Database
Homogenous Database

Homogeneous Database.

↳ All sites have identical software

↳ Are aware of each other and

agree to co-operate in processing user requests.

↳ Each site surrenders part of its autonomy in terms of right to change schemas or software

↳ Appears to user as a single system.

Heterogenous distributed database.

↳ Diffesent sites may use diffesent schemas and softwdre.

↳ Diffesence in schema is a major problem for query processing.

↳ Diffesence in software is a major problem for transaction processing.

↳ Sites maynot be aware of each other and may provide only limited facilities for cooperation in transaction processing.

Distributed Data storage.

↳ Assume relational data model stored in the database.

Notes

↳ Two approaches to storing relation in distributed database

1. Replication.
   System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.

2. Fragmentation.
   Relation is partitioned into several fragments stored in distinct sites

↳ Replication and fragmentation can be combined.

Relation is partitioned into several fragments; system maintains several identical replicas of each such fragment.

Data Replication.

↳ A relation or fragment of a relation is replicated if it is stored redundantly in two or more sites.

↳ Full replication of a relation is the case where the relation is stored at all sites.

↳ Fully redundant databases are

those in which every site contains a copy of the entire database.

Advantages of Replication.

Availability: failure of site containing relation r doesnot result in unavailability of r if replicas exist

Parallelism: queries on r may be processed by several nodes in parallel.

Reduced data transfer: relation r is available locally at each site containing a replica of r.

Disadvantages of Replication.

Increased cost of updates: each replica of relation r must be updated.

Increased complexity of concurrency control: concurrent updates to distinct replicas may lead to inconsistent data unless special concurrency control mechanisms are implemented.

One solution may be to choose

one copy as primary copy and apply concurrency control operations on primary copy.

Data Fragmentation.

↳ Data fragmentation is division of relation r into fragments $r_1, r_2, \ldots, r_n$ which contains sufficient information to reconstruct relation r.

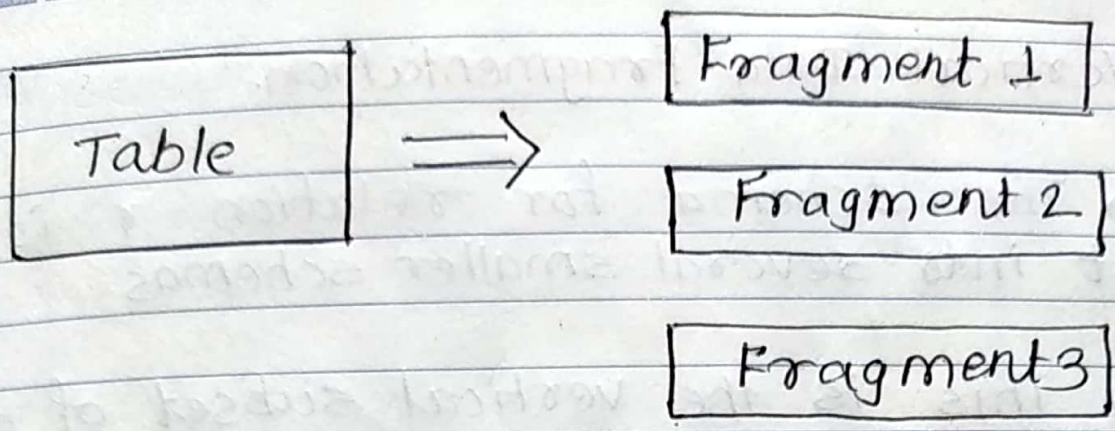
↳ These fragments may be stored at different locations.

↳ The data fragmentation process should be carried out in such way that the reconstruction of original database from the fragments is possible.

Types of data Fragmentation.

1. Horizontal data fragmentation.

↳ Each tuple of r is assigned to one or more fragments.



Considers the relation:

customer-id	Name	Area	PaymentType	sex
1	Bob	London	Credit card	M
2	Mike	Machestes	Cash	M
3	Ruby	London	Cash	F

Horizontal Fragmentation are subsets of tuple (rows)

Fragment 1

customer-id	Name	Area	PaymentType	Sex
1	Bob	London	credit card	Male
2	Mike	Manchestes	Cash	Male

Fragment 2

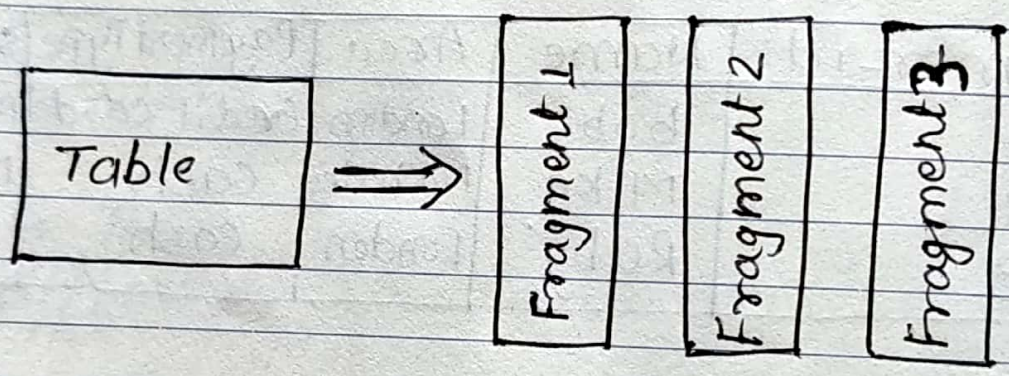
customer-id	Name	Area	Payment Type	Sex
3	Ruby	London	cash	Female

Notes

2. Vertical Data Fragmentation.

↳ The schema for relation r is split into several smaller schemas.

↳ This is the vertical subset of a relation. That means a relation / table is fragmented by considering the columns of it.



Vertical fragmentation are subset of attributes.

Fragment 1:

customer_id	Name	Area	Sex
1	Bob	London	Male
2	Mike	Manchester	Male
3	Ruby	London	Female

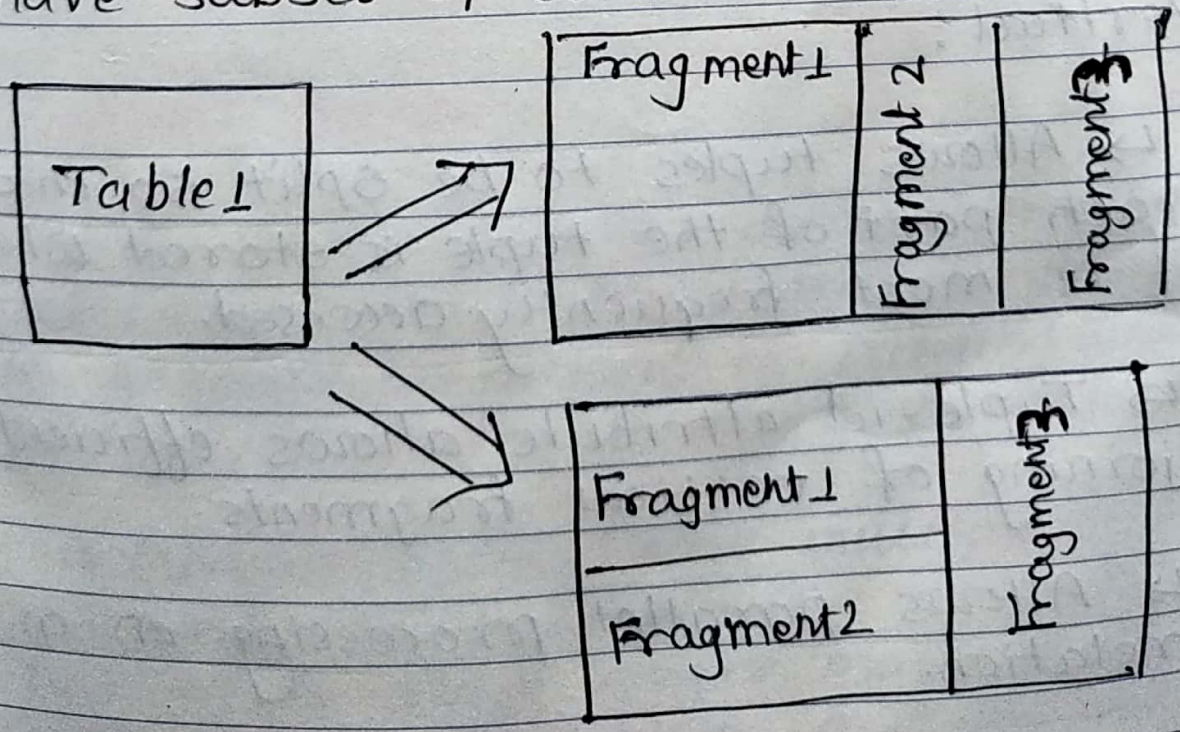
Fragment 2

customer-id	Payment Type
1	credit card
2	cash
3	cash

3. Hybrid Data Fragmentation.

↳ This is the combination of horizontal as well as vertical fragmentation.

↳ This type of fragmentation will ~~be~~ have horizontal fragmentational to have subset of data to be distributed oves the DB, and vertical fragmentation to have subset of columns of the table.



As we observe in above diagram, this type of fragmentation can be done in any order. It doesnot have any pasticulas order. It is solely based on the user requisement. But it should satify fragmentation conditions.

Advantages of Fragmentation.

Horizontal:

↳ Allows parallel processing on fragments of a relation.

↳ allows a relation to be split so that tuples are located where they are most frequently accessed.

Vertical:

↳ Allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed.

↳ Tuple-id attribute allows efficient joining of vertical fragments.

↳ Allows parallel processing on a relation.

Reasons for Distributed Database.

↳ Business unit autonomy and distribution.

↳ Data sharing

↳ Data communication costs.

↳ Data communication reliability & costs.

↳ Multiple application vendors.

↳ Database recovery.

↳ Transaction and analytic processing.

Parallel DBMS:

↳ A DBMS running across multiple processors and disks that is designed to execute operations in parallel, whenever possible, in order to improve performance.

↳ Parallel DBMSs link multiple, smaller machines to achieve the same throughput as a single, larger machine, often with greater scalability and reliability

than single processor DBMSs.

↳ A coarse-grain parallel machine consists of a small number of powerful processors.

↳ A massively parallel or fine grain parallel machine utilizes thousands of smaller processors.

↳ Two main performance measures:

throughput:
the number of tasks that can be completed in a given time interval.

response time
the amount of time it takes to complete a single task from the time it is submitted.

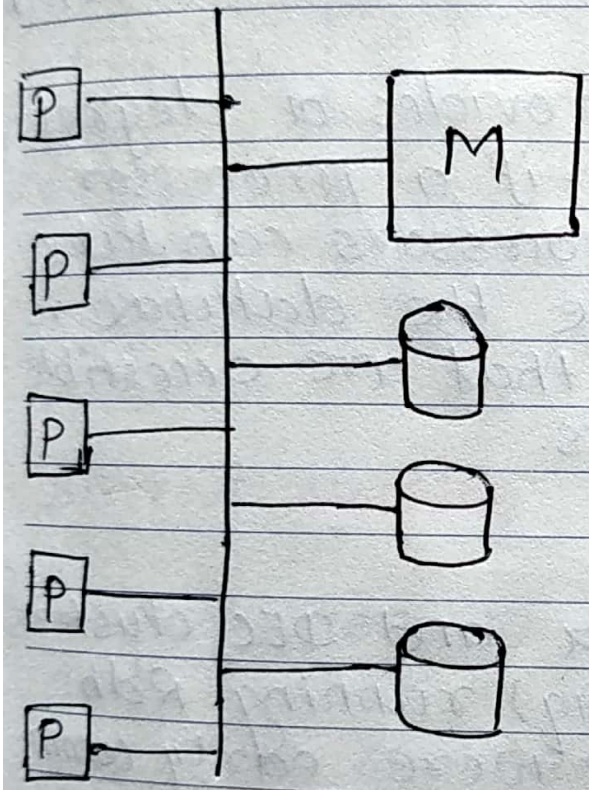
↳ speed up: a fixed-sized problem executing on a small system is given to a system which is N-times larger. Measured by:

$$\text{Speed up} = \frac{\text{small system elapsed time}}{\text{large system elapsed time}}$$

Parallel database Architectures.

Shared Memory:

↳ It is tightly coupled architecture in which multiple processors within a single system share system memory.



↳ Processors and disks have access to a common memory; typically via a bus or through an interconnection network.

↳ Extremely efficient communication between processors.

↳ Widely used for lowes degrees of parallelism.

Notes

Shared Disk.

↳ All processors can directly access all disks via an interconnection network but the processors have private memories

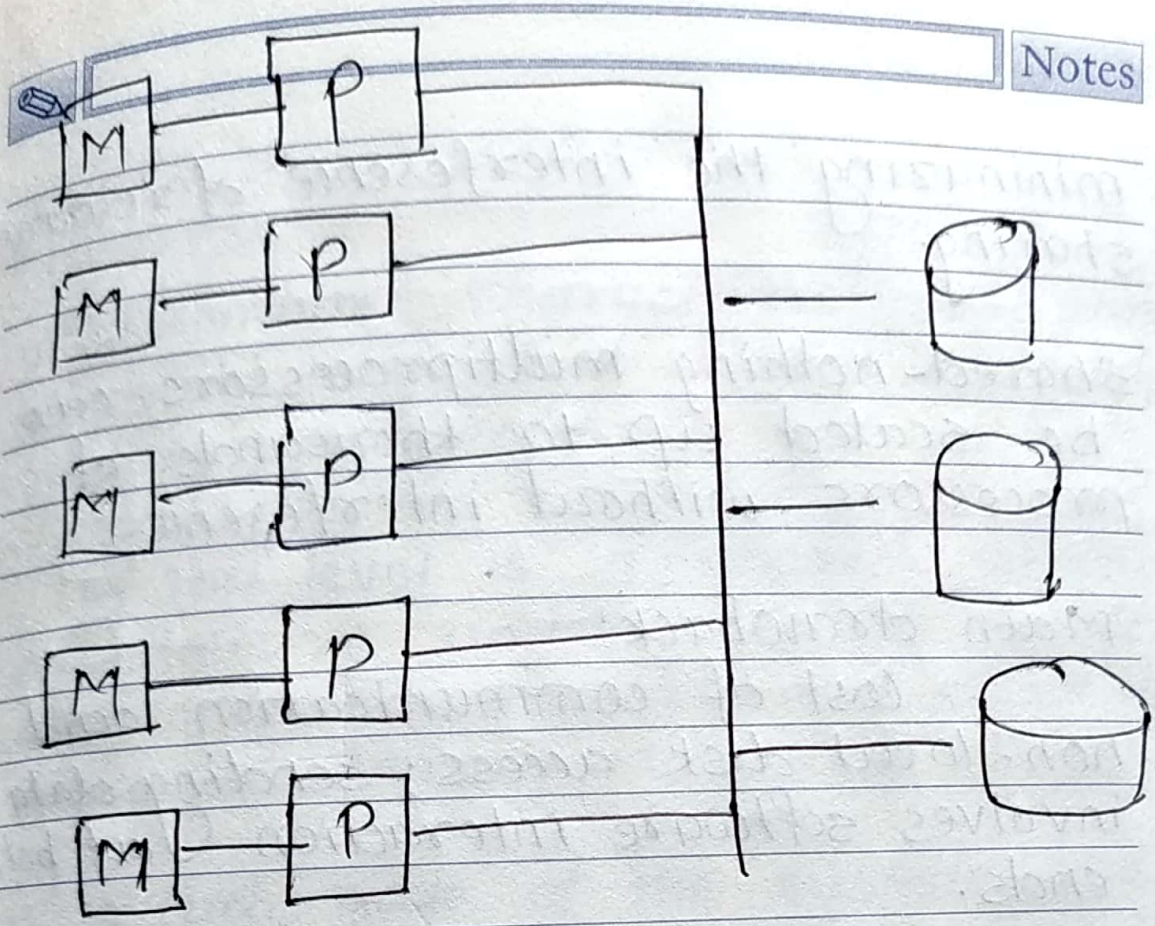
↳ The memory bus is not a bottleneck.

↳ Architecture provides a degree of fault-tolerence - if a processor fails, the other processors can take over its tasks since the database is resident on disks that are accessible from all processors.

↳ Examples:
IBM Sysplex and DEC clusters (now part of Compaq) running Rdb (now Oracle Rdb) were early commercial users.

↳ Downside:
bottleneck now occurs at interconnection to the disk subsystem.

↳ Shared-disk systems can scale to a somewhat larger numbes of processors, but communication bet^n processors is slower.



Shared Nothing.

↳ Node consists of a processor, memory, and one or more disks. Processor at one node communicate with another processor at another node using an interconnection network. A node functions as the server for the data on the disk or disks the node owns.

↳ Examples:
Teradata, Tandem, oracle-n CUBE

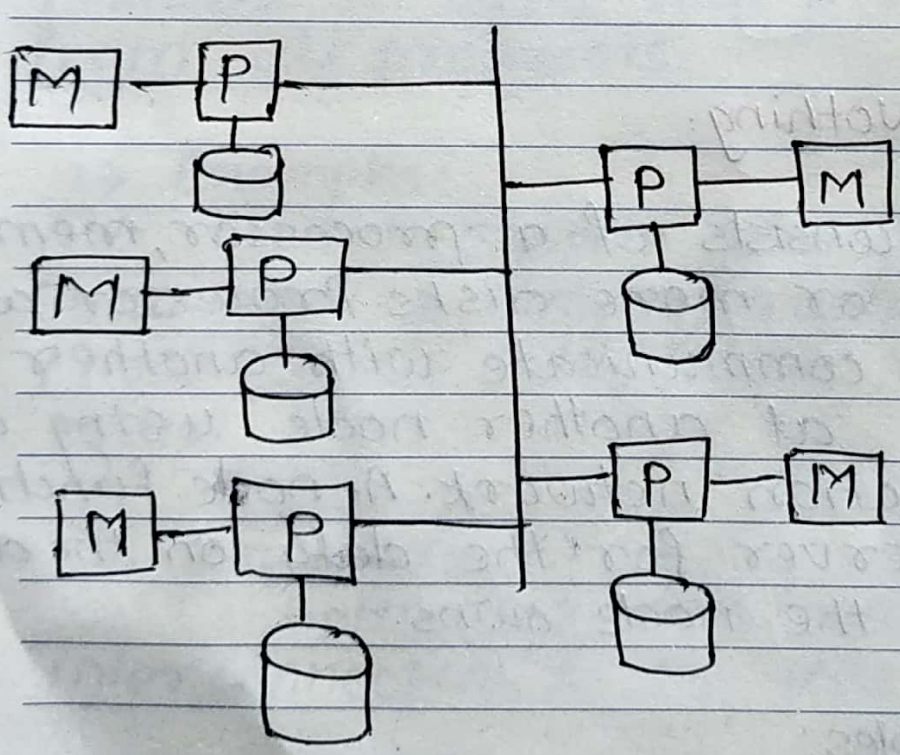
↳ Data accessed from local disks (& local memory accesses) donot pass through interconnection network, thereby

Notes

minimizing the interfesence of resource sharing.

Shared-nothing multiprocessors can be scaled up to thousands of processors without interfesence.

Main drawback:
cost of communication and non-local disk access; sending data involves software interaction at both ends.



Hierarchical:

↳ combines characteristics of shared memory, shared-disk and shared-nothing architectures.

↳ Top level is a shared-nothing architecture - nodes connected by an interconnection network, and do not share disks or memory with each other.

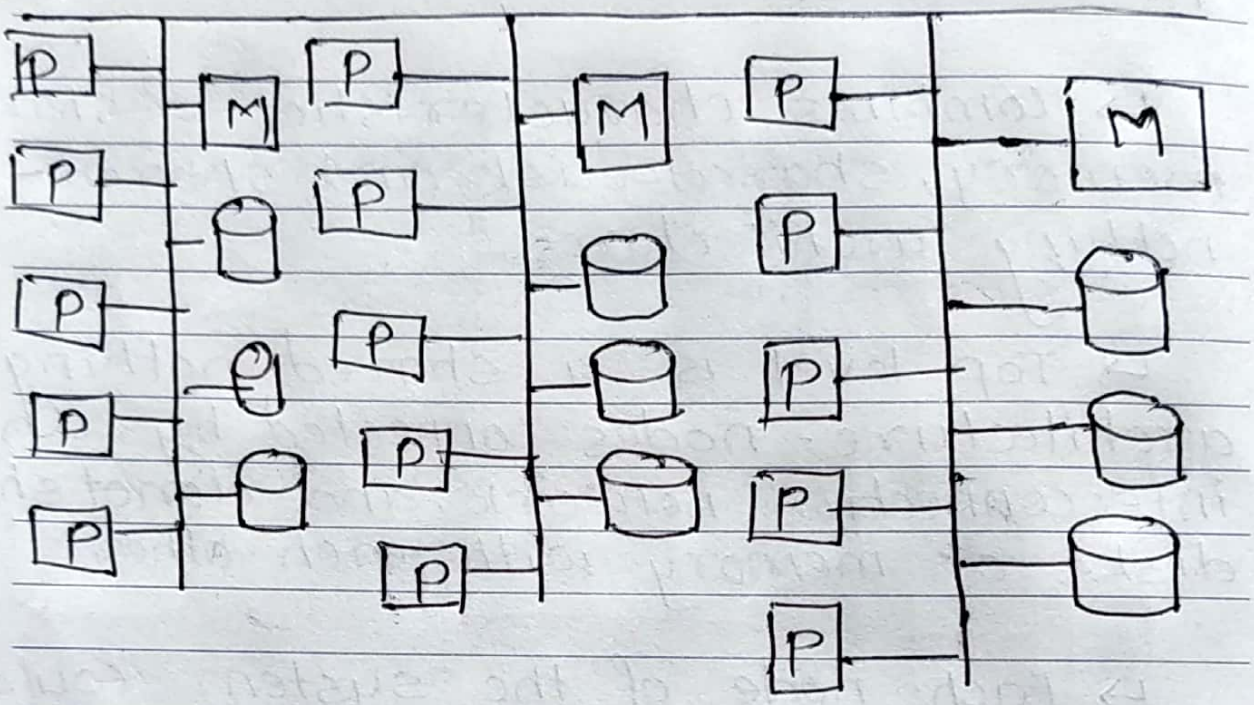
↳ Each node of the system could be shared-memory system with a few processors.

↳ Alternatively, each node could be shared-disk system, and each of the systems sharing a set of disks could be a shared memory system.

↳ Reduce the complexity of programming such systems by distributed virtual memory architectures.

↳ Also called non-uniform memory architecture (NUMA)

Notes



Data Warehouse.

↳ A data warehouse (DW) is a collection of corporate information & data derived from operational systems and ~~data~~ external data sources.

↳ A data warehouse is designed to support business decisions by allowing data consolidated, analysis and repoting at different aggregate levels.

↳ Data is populated into the DW through the processes of extraction, transformation & loading.

↳ A single, complete and consistent store of data obtained from a variety of different sources made available to end users in what they can understand and use in a business context is called warehouse.

↳ A data warehouse is subject-oriented, integrated, time-varient, non volatile collection of data in support of management's decision making process

Subject-oriented.

↳ A data wasehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations.

↳ These subjects can be product, customers, suppliers, sales, revenue, etc.

↳ A data wasehouse doesnot focus on the ongoing operations, rather it focuses on modeling and analysis of data for decision making.

↳ Excludes data not useful in decision support process.

Notes

Integrated:

↳ A data warehouse is constructed by integrating data from heterogenous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.

↳ Data preprocessing are applied to ensure consistency.

↳ The data warehouse is a centralized, consolidated database that integrates data from entire organization.

Multiple sources
Diverse sources
Diverse formats.

Time Vasiant.

↳ The data collected in a data warehouse is identified with a particular time period.

↳ The data in a data warehouse provides information from the historical point of view.

↳ The data warehouse represents the flow of data through time.

↳ Every key structure contains either implicitly or explicitly an element of time.

Nonvolatile:

↳ Once data is entered it is never removed.

↳ A data wasehouse is kept sepasate from the operational database and therefore frequent changes in operational database is not reflected in the data wasehouse.

Note: A data warehouse doesnot requse transaction processing, recovery and conurreny controls, because it is physically stored and separate from the operational database.

↳ Represents the company's entire history

Near term history is continually added to it.

Always growing

Must support tesabyte databases

and multiprocessors.

↳ Read only database for data analysis & query processing.

↳ Data warehouse requires two operations in data accessing

Initial loading of data
Access of data.

Rules of a Data Warehouse.

↳ Data warehouse and Operational Environments are separated.

↳ Data is integrated.

↳ contains historical data over a long period of time.

↳ Data is a snapshot data captured at a given point in time.

↳ Data is subject oriented.

↳ Mainly read-only with periodic batch updates.

↳ Development Life cycle has a

data driven approach versus the traditional process-driven approach.

↳ Data contains several levels of detail. current, old, Lightly summarized, Highly summarized.

↳ Environment is characterized by Read only transactions to very large data sets.

↳ System that traces data sources, transformations, and storage.

↳ Metadata is a critical component. source, transformation, integration, storage, relationships, history etc.

↳ Contains chargeback mechanism for resource usage that enforces optimal use of data by end users.

Phases of Data Mining.

Data preparation.

Identify the main data sets to be used by the data mining operation.

Notes

Data Analysis and Classification.

Study the data to identify common data characteristics or patterns.

* Data grouping, classification, clusters, sequences.

* Data dependencies, links, or relationships.

* Data patterns, trends, deviation.

Knowledge Acquisition.

Uses the Results of Data Analysis & classification phase

* Decision Trees
* Rules Induction.
* Genetic algorithms.
* Memory-Based Reasoning.

Prognosis.

* Predict future Behaviour
* Forecast buisness outcomes.
* 65 %. of customers who didnot use a particulas credit card in

the last 6 months are 88% likely to cancel the account

Need for Data Warehousing.

↳ Industry has huge amount of data. knowledge worker wants this data into useful information. This information is used by them to support strategic decision making.

↳ It is a platform for consolidated historical data for analysis.

↳ It stores data for good quality so that knowledge worker can make correct decisions.

↳ From business perspective
- It is latest marketing weapon.
- Helps to keep customers by learning more about theis needs.
- Valuable tool in today's competitive fast evolving world.

Application Area of Data Wasehouse.

OLAP (Online Analytical Processing) is a term used to describe the analysis of complex data from data warehouse.

Notes

DSS (Decision Support Systems) also known as EIS (Executive Information Systems) supports organization's leading decision makers for making complex and important decisions.

↳ Data mining is used for knowledge discovery, the process of seaching data for unanticipated new knowledge.

Characteristics of a Data Wasehouse.

↳ Subject oriented
Organized based on use.

↳ Integrated
Inconsistencies removed.

↳ Non volatile
stored in Read only format.

↳ Time variant
'data are normally time series

↳ Summarized
in decision-usable format

↳ Large volume
data sets are quite large

↳ Non normalized
often redundant.

↳ Metadata
data about data are stored.

↳ Data sources
comes from non integrated sources.

Data Warehouse Architecture.

Data warehouse Server.

Almost always a relational DBMS, rarely flat files.

OnLine Analysis Processing (OLAP) servers.

To support and operate on multi-dimensional data structures.

Clients.

Query and reporting tools.
Analysis tools
Data mining tools.

Spatial Database System.

↳ A spatial database is a database that is optimized to store and query data related to objects in space, including points, lines and polygons. While typical databases can understand various numeric and character types of data, additional functionality needs to be added for database to process spatial data types. These are typically called geometry or feature.

↳ Spatial databases provide structures for storage and analysis of spatial data and spatial data is comprised of objects in multi-dimensional space.

↳ Storing spatial data in standard database would require excessive amounts of space.

↳ Queries to retrieve and analyze spatial data from a standard database would be long and cumberscome leaving a lot room for error.

↳ Spatial databases provide much more efficient storage, retrieval, and analysis of spatial

Notes

Types of Data stored in Spatial Databases

Two dimensional data examples

- Geographical
- Cartesian co-ordinates (2D)
- Networks
- Direction.

Three-dimensional data examples.

- Weather
- Castesian co-ordinates (3-D)
- Topological
- satellite images.

Spatial Database Management System (SDBMS)

↳ SDBMS provides the capabilities of a traditional DBMS while allowing special storage and handling of spatial data.

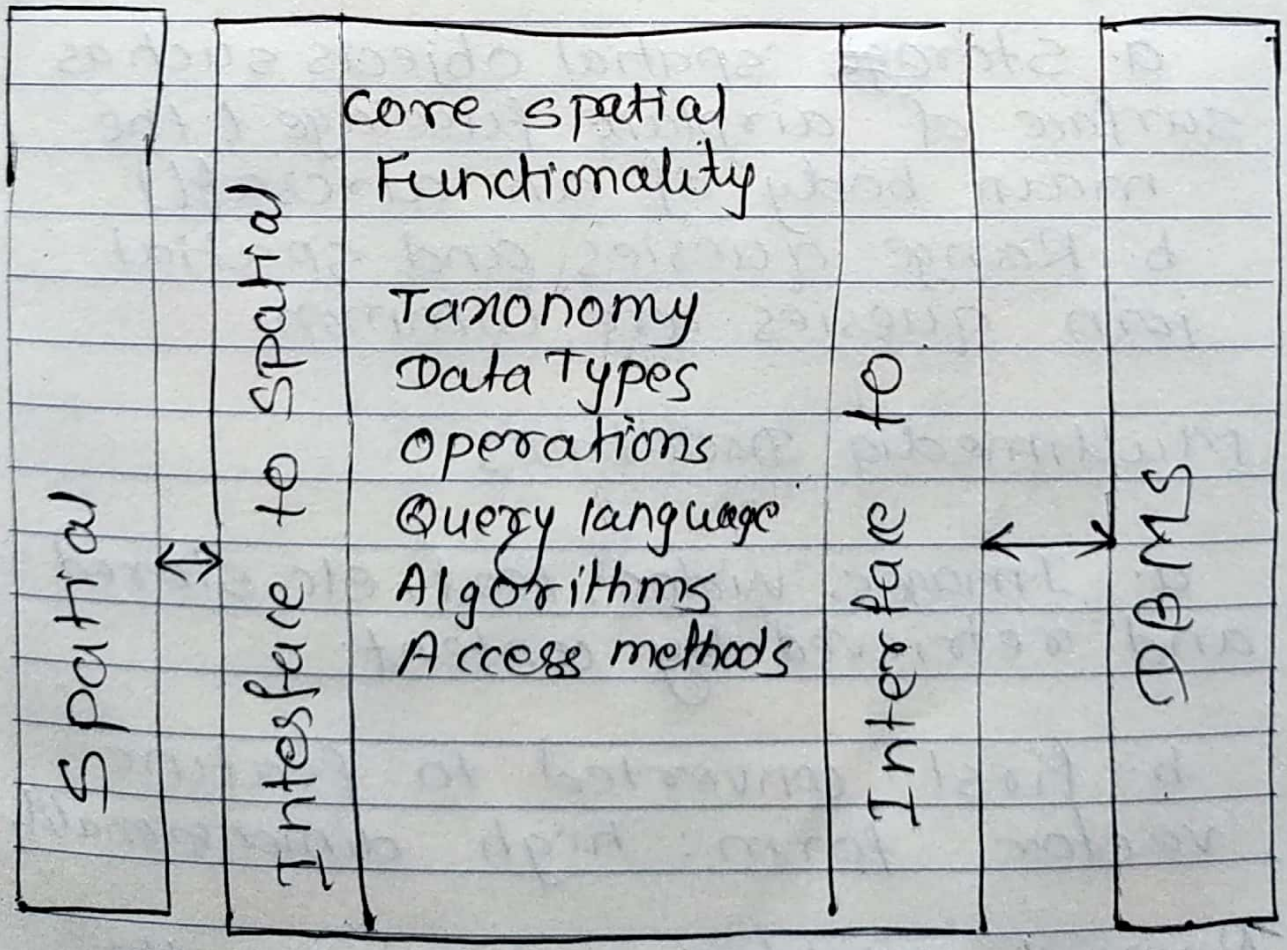
↳ SDBMS works with an underlying DBMS, allows spatial data models & types, supports querying language specific to spatial data types &

provides handling of spatial data & operations.

↳ SDBMS works with a spatial application at the front end and a DBMS at the back end.

SDBMS has three layers:

Interface to spatial application
core spatial functionality
Interface to DBMS



Notes

Application of spatial data.

Geographic Information System (GIS)

a. E.g, ESRI's ArcInfo, OpenGIS consortium.

b. Geospatial information.

c. All classes of spatial queries & data are common.

Computer-Aided Design/Manufacturing

a. Storage spatial objects such as surface of airplane fuselage. (the main body of an aircraft)

b. Range queries and spatial join queries are common.

Multimedia Databases

a. Images, video, text, etc. stored and retrieved by content.

b. first converted to feature vector form: high dimensionality

Nearest-neighbor queries are the most common.

Rename operation:

↳ Allow us to name, and therefore to refer to, the results of relational-algebra expressions.

↳ Allow us to refer to a relation by more than one name.

Example: $\rho_x(E)$

returns the expression E under the name X.

↳ If a relational-algebra Expression E has arity n, then

$$\rho_{x(A_1, A_2, A_3, \ldots A_n)}(E)$$

returns the result of expression E under the name X, and with attributes renamed to $A_1, A_2, A_3, \ldots A_n$

Example Query:

Find the largest salary in the university.

Step 1: Find instructor salaries that are less than some other instructor salary

(i.e. not maximum)

- using a copy of instructor under a new name d.

$\Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

Step 2: Find the largest salary.

$\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

Aggregate Functions and Operations.

Aggregate operation in relational algebra

$$G_1, G_2, \ldots G_n \mathcal{G} F_1(A_1), F_2(A_2), \ldots F_n(A_n) \ (E)$$

E is any relational-algebra expression.

- $G_1, G_2, \ldots, G_n$ is a list of attributes on which to group.
- Each Fi is an aggregate function.

- Each Ai is an attribute name.

Example:

Relation r

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$\mathcal{G}_{sum(c)}(r)$

sum(c)
27

Find the average salary in each department.

ID	name	dept-name	salary

$dept\text{-}name\,\mathcal{G}_{avg(salary)}(instructor)$

$dept\text{-}name\,\mathcal{G}_{avg(salary)\ as\ avg\_sal}(instructor)$

Notes

select A_1, sum(A_3)
from $r_1, r_2, \ldots, r_m$
where P
group by A_1, A_2

$\Pi_{A_1, \text{sum}A_3}(A_1, A_2 \mathcal{G}_{\text{sum}(A_3)} \text{ as } \text{sum}A_3(\sigma_P(r_1 \times r_2 \times \ldots \times r_m)))$

completed_times.

Person_name	Puzzle_name	Seconds

How many puzzles has each person completed?

person_name $\mathcal{G}$ count-distinct(puzzle_name) (completed_times)

How many people have completed each puzzle?

puzzle_name $\mathcal{G}$ count(person_name) (completed_times)

Inserting New Tuples

Inserting tuples simply involves a union.

$r \leftarrow r \cup E$

E has to have correct arity.

completed ← completed ∪ {("Bob", "altekruse"), ("carl", "clutch box")}

How to construct new tuples with name "Dave" and each of Bob's puzzles?

could use a cartesian product:

$\{(\text{"Dave"})\} \times \Pi_{\text{puzzle\_name}}(\sigma_{\text{person\_name}=\text{"Bob"}}(\text{completed}))$

or, use generalized projection with a constant:

$\Pi_{\text{"Dave" as person\_name, puzzle\_name}}(\sigma_{\text{person\_name}=\text{"Bob"}}(\text{completed}))$

Add new tuples to completed relation.

completed ← completed ∪ ·

Notes

$\Pi_{\text{"Dave" as person\_name}}(\sigma_{\text{person\_name = "Bob"}}(\text{completed}))$

Deleting Tuples.

Deleting tuples uses the - operation.

$r \leftarrow r - E$

Example:

Get rid of the "soma cube" puzzle.

person_name	puzzle_name
Alex	altekruse
Alex	soma cube
Bob	puzzle box
carl	altekruse
Bob	soma cube
carl	puzzle box
Alex	puzzle box
carl	somacube

completed.

puzzle_name
altekruse
soma cube
puzzle box

puzzle_list.

completed ← completed - $\sigma_{puzzle\_name = "soma\ cube"}$ (completed)

puzzle_list ← puzzle_list - $\sigma_{puzzle\_name = "soma\ cube"}$ (puzzle_list)

OR

completed ← $\sigma_{puzzle\_name \neq "soma\ cube"}$ (completed)

Updating Tuples

acc_id	branch_name	balance
A-301	New York	350
A-307	Seattle	275
A-318	Los Angeles	550
A-319	New York	80
A-322	Los Angeles	275

account.

Example:

"Add 5% interest to all bank account balances."

(account)

account ← $\Pi_{acc\_id, branch\_name, balance * 1.05}$

Notes

Note: Must include unchanged attributes too.

"Add 5% interest to accounts with a balance less than $10,000."

account ← Π acc-id, branch-name, balance * 1.05 ($\sigma_{balance < 10000}$ (account)) $\cup$ $\sigma_{balance \geq 10000}$ (account)

"Add 5% interest to accounts with a balance less than $10,000, and 6% interest to accounts with a balance of $10,000 or more."

account ← Π acc-id, branch-name, balance * 1.05 ($\sigma_{balance < 10000}$ (account)) $\cup$ Π acc-id, branch-name, balance * 1.06 ($\sigma_{balance \geq 10000}$ (account))

Don't forget to include any non-updated tuples in your update operations!