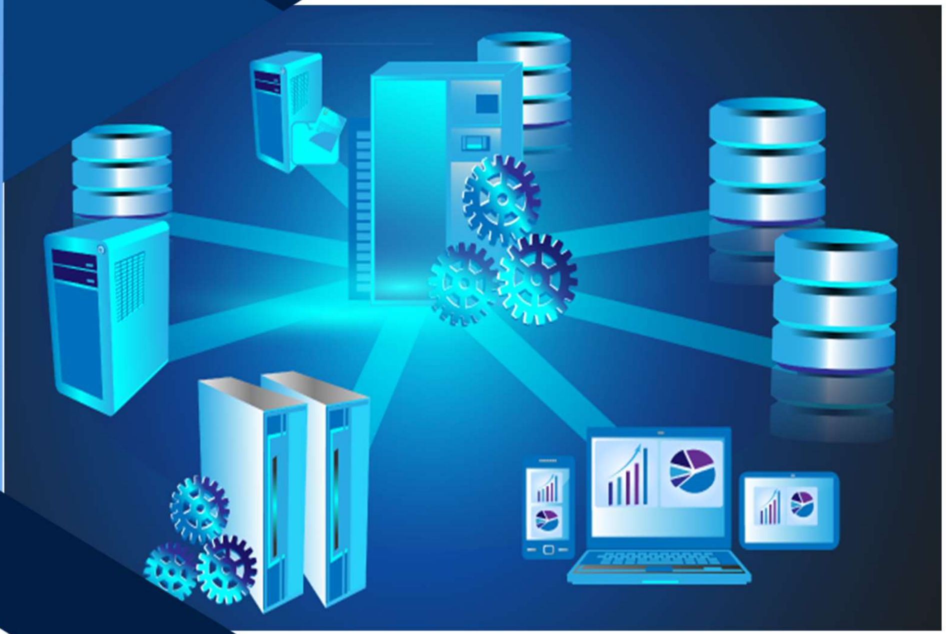
SOUTHERN LEYTE
STATE UNIVERISTY
BONTOC

San Ramon, Bontoc, Southern Leyte, 6604 Philippines
Telefax No.: (053) 382-3121; E-mail: slsubc@yahoo.com
Website: www.slsuonline.edu.ph



ADVANCED DATABASE

LEARNING GUIDE 2020

REXAL S. TOLEDO

SOUTHERN LEYTE
STATE UNIVERSITY-BONTOC

Southern Leyte State University

Vision

A high quality corporate University of Science, Technology and Innovation.

Mission

SLSU will:

a) Develop Science, Technology, and Innovation leaders and professionals;
b) Produce high-impact technologies from research and innovations;
c) Contribute to sustainable development through responsive community engagement programs;
d) Generate revenues to be self-sufficient and financially-viable.

Quality Policy

We at Southern Leyte State University commit enthusiastically to satisfy our stakeholders' needs and expectations by adhering to good governance, relevance and innovations of our instruction, research and development, extension and other support services and to continually improve the effectiveness of our Quality Management System in compliance to ethical standards and applicable statutory, regulatory, industry and stakeholders' requirements.

The management commits to establish, maintain and monitor our quality management system and ensure that adequate resources are available.

COURSE OVERVIEW

Course No.	IT301/IT301L
Course Code	
Descriptive Title	Advanced Database Systems
Credit Units	2 units (Lecture) / 1 unit (Laboratory)
School Year/Term	2020-2021 / 1st semester
Mode of Delivery	
Name of Instructor	Rexal S. Toledo
Course Description	This course covers modern database and information system as well as research issues in the field. It will cover selected topics on NoSQL, object-oriented, active, deductive, spatial, temporal and multimedia databases. The course includes advanced issues of object-oriented database, XML, advanced client server architecture, Information Retrieval and Web Search and distributed database techniques.
Course Outcomes	1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. 2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. 3. Apply software development fundamentals to produce computing-based solutions. 4. Communicate effectively in a variety of professional contexts.

TABLE OF CONTENTS

MODULE 1

Enhanced Data Models

LESSON

PRE-TEST

MODULE 1:

1. What are the differences between row-level and statement-level active rules?
2. What are the differences among immediate, deferred, and detached consideration of active rule conditions?
3. What are the differences among immediate, deferred, and detached execution of active rule actions?
4. Briefly discuss the consistency and termination problems when designing a set of active rules.
5. Discuss some applications of active databases.
6. Discuss how time is represented in temporal databases and compare the different time dimensions.
7. What are the differences among valid time, transaction time, and Bitemporal relations?
8. Describe how the insert, delete, and update commands should be implemented on a valid time relation.
9. Describe how the insert, delete, and update commands should be implemented on a Bitemporal relation.
10. Describe how the insert, delete, and update commands should be implemented on a valid time relation.
11. What are the main differences between tuple versioning and attribute versioning?
12. How do spatial databases differ from regular databases?
13. What are the different types of spatial data?
14. Name the main types of spatial operators and different classes of spatial queries
15. What are the properties of R-trees that act as an index for spatial data?
16. Describe how a spatial join index between spatial objects can be constructed.
17. What are the different types of spatial data mining?
18. State the general form of a spatial association rule. Give an example of a spatial association rule.
19. What are the different types of multimedia sources?
20. How are multimedia sources indexed for content-based retrieval?
21. What important features of images are used to compare them?
22. What are the different approaches to recognizing objects in images?
23. How is semantic tagging of images used?
24. What are the difficulties in analyzing audio sources?
25. What are deductive databases?
26. Define the clausal form of formulas and Horn clauses.
27. What is theorem proving, and what is proof-theoretic interpretation of rules?
28. What is model-theoretic interpretation and how does it differ from proof theoretic interpretation?
29. What are fact-defined predicates and rule-defined predicates?
30. What is a safe rule?

LESSON 1

Introduction to Active Database Concepts and Triggers

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Present the general concepts that have been proposed for specifying rules for active databases.
2. Discuss some general design and implementation issues for active databases.
3. Discuss possible applications of active databases and understand how triggers are declared in the SQL-99 standard

Active Databases

- Triggers and rules are developed for data integrity and constraints.
- Triggers make "passive" database "active"
    - Database reacts to certain situations
- Event Condition Action rule :
    - on event insert/update/delete,
    - if condition C is true
    - then do action A

Brief History

- 1975: Idea of "integrity constraints"
- Mid 1980-1990: research in constraints & triggers
    - Languages and algorithms
- SQL-92: constraints
    - Key constraints; referential integrity, domain constraints
    - Declarative spec and procedural interpretation
- SQL-99: triggers/ECA (limited)
- Early acceptance; Widely-varying support in products; "execution semantics" differ, how far to go?

Event-Condition-Action (ECA)

- Event occurs in database
    - addition of new row, deletion of row by DBMS
- Conditions are checked
    - " SQL condition
- Actions are executed if conditions are satisfied
    - SQL + procedures
    - All data actions performed by the trigger execute within the same transaction in which the trigger fires

Triggers

A procedure that runs automatically when a certain event occurs in the DBMS. Implementation of the ECA rules.

The procedure performs some actions, e.g.

- Check certain values
- Fill in some values
- Inserts/deletes/updates other records
- Check that some business constraints are satisfied
- Cancel or roll back forbidden actions

Database Triggers in SQL

Available in most enterprise DBMSs (Oracle, IBM DB2, MS SQL server) and some public domain DBMSs (Postgres).

Some vendor DBMS permit native extensions to SQL for specifying the triggers e.g. PL/SQL in Oracle, Transact SQL in MS SQL Server.

Some DBMS extend the triggers beyond tables for example also to views as in Oracle

Trigger Components

Three components:

- **Event**: When this event happens, the trigger is activated
- **Condition** (optional): If the condition is true, the trigger executes, otherwise skipped
- **Action**: The actions performed by the trigger.

Semantics

When the Event occurs and Condition is true, execute the Action.

Types of SQL Triggers

1. How many times should the trigger body execute when the triggering event takes place?

- **Per statement:** the trigger body executes once for the triggering event. This is the default.
- **For each row:** the trigger body executes once for each row affected by the triggering event.

2. When the trigger can be fired?

- Relative to the execution of an SQL DML statement (before or after or instead of it)
- Exactly in a situation depending on specific system resources (e.g. signal from system clock)

Statement and Row Triggers

Example 1: Monitoring Statement Events

```
  INSERT INTO dept (deptno, dname, loc)
  VALUES (50, 'EDUCATION', 'NEW YORK');
```

Execute only once even if multiple rows affected

Example 2: Monitoring Row Events

```
 UPDATE emp
 SET sal = sal *
1.1
 WHERE deptno = 30;
```

Execute for each row of table affected by event

Granularity of Event

An **UPDATE** or **DELETE** statement may update (or delete) many records at the same time. May insert many records in the same statement as well.

Does the trigger execute for each updated or deleted record, or once for the entire statement? We define such granularity.

That is the timing

That is the event

Create Trigger <*name*>

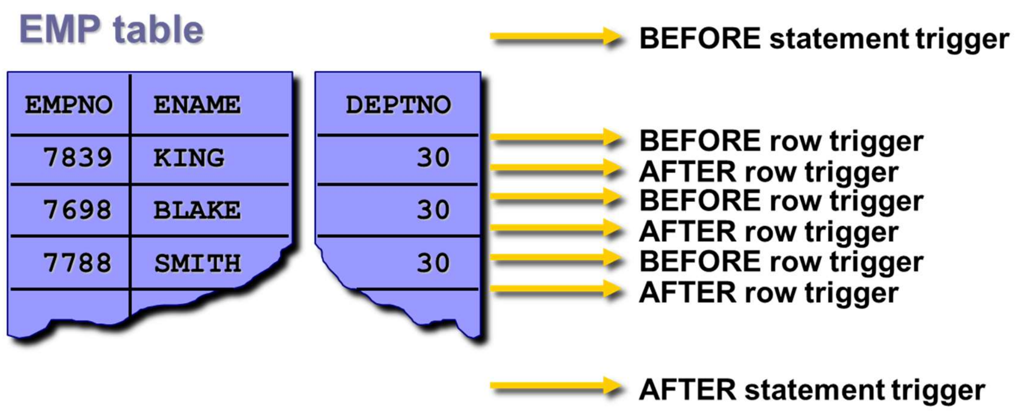
Before| After **Insert| Update| Delete**

For Each Row | For Each Statement

….

That is the granularity

Firing Sequence of Database Triggers on Multiple Rows



Example: Logging Operations

```
CREATE TRIGGER increase_salary_trg
    AFTER UPDATE OF sal
    ON emp
BEGIN
    if :new.sal > :old.sal Then
    INSERT INTO sal_hist(increased, changedOn)
             VALUES ('YES', SYSDATE);
    end;
END;/
```

Trigger name:	`increase_salary_trg`
Timing:	`AFTER` executing the statement
Triggering event:	UPDATE of `sal` column
Target:	`emp` table
Trigger action:	INSERT values INTO `sal_hist table`

Example: Checking Values

If the employee salary increased by more than 10%, make sure the 'comment' field is not empty and its value has changed, otherwise reject the update.

```
Create Trigger EmpSal
Before Update On Employee
Referencing
        OLD ROW AS   oldRec,
        NEW ROW AS  newRec
For Each Row
Begin
     IF (newRec.salary > oldRec.salary * 1.10) Then
        IF (newRec.comment = ‘’ or newRec.comment is null or
            newRec.comment = oldRec.comment)
         RAISE_APPLICATION_ERROR(-20004, ‘Comment field not correct’);
       End IF;
     End IF;
End;
```

Example: Using Temp Variable

If the newly inserted record in employee has null date field, fill it in with the current date.

```
Create Trigger EmpDate
Before Insert On Employee
Referencing
        NEW ROW AS  newRec
For Each Row
Declare
     temp date;
Begin
      Select sysdate into temp from dual;
      IF (newRec.date is null) Then
           newRec.date := temp;
     End IF;
End;
```

Define variables

Oracle system table always has the current

Updating the new value to be inserted

Example: Calculating Derived Columns

```
CREATE OR REPLACE TRIGGER derive_commission_trg
  BEFORE UPDATE OF sal ON emp
  FOR EACH ROW
  WHEN (new.job = 'SALESMAN')
  BEGIN
    :new.comm := :old.comm * (:new.sal/:old.sal);
  END;
/
```

Trigger name:	derive_commission_trg
Timing:	BEFORE executing the statement
Triggering event:	UPDATE of sal column
Filtering condition:	job = 'SALESMAN'
Target:	emp table
Trigger parameters:	old, new
Trigger action:	calculate the new commission to be updated

Controlling Triggers using SQL

Disable/Re-enable database trigger

```
ALTER TRIGGER trigger_name  DISABLE | ENABLE
```

Disable or Re-enable all triggers for table

```
ALTER TABLE table_name   DISABLE | ENABLE  ALL TRIGGERS
```

Removing a trigger from database

```
DROP TRIGGER trigger_name
```

Using Database Triggers

Auditing Table Operations

- Each time a table is updated auditing information is recorded against it.

Tracking Record Value Changes

- Each time a record value is changed the previous value is recorded

Maintenance of Semantic Integrity

- e.g. when the factory is closed, all employees should become unemployed

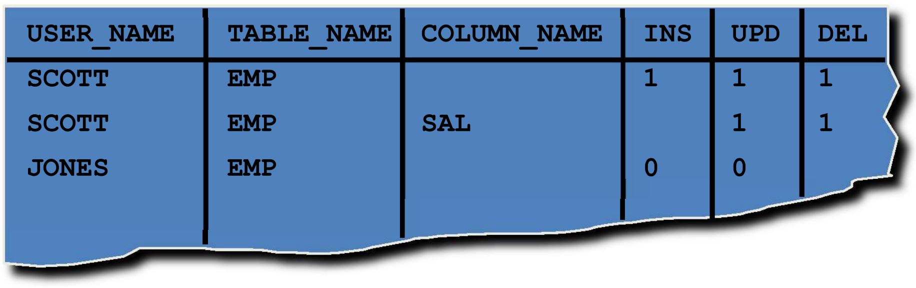
Storing Derived Data

- e.g. the number of items in the trolley should correspond to the current session selection

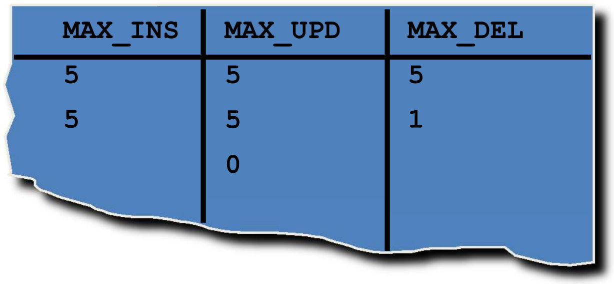
Security Access Control

- e.g. checking user privileges when accessing sensitive information

Auditing Table Operations



USER_NAME	TABLE_NAME	COLUMN_NAME	INS	UPD	DEL
SCOTT	EMP		1	1	1
SCOTT	EMP	SAL		1	1
JONES	EMP		0	0	



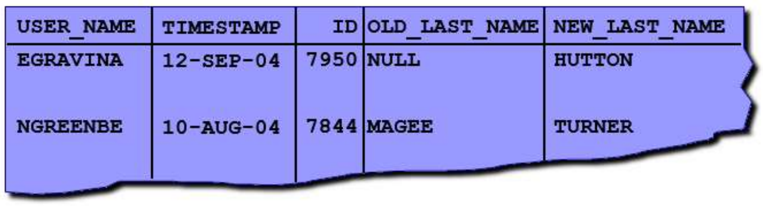
MAX_INS	MAX_UPD	MAX_DEL
5	5	5
5	5	1
	0	

Example: Counting Statement Execution

```
CREATE OR REPLACE TRIGGER audit_emp
AFTER DELETE ON emp
FOR EACH ROW
BEGIN
     UPDATE audit_table SET del = del + 1
     WHERE user_name = USER
     AND table_name = 'EMP';
END;   /
```

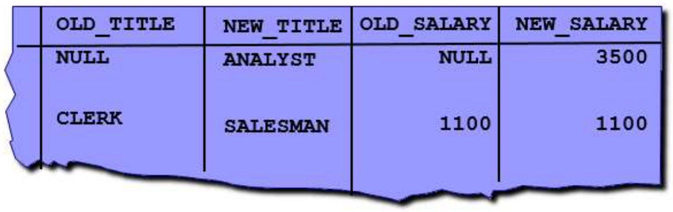
Whenever an employee record is deleted from database, counter in an audit table registering the number of deleted rows for current user in system variable USER is incremented.

Example: Tracing Record Value Changes



USER_NAME	TIMESTAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME
EGRAVINA	12-SEP-04	7950	NULL	HUTTON
NGREENBE	10-AUG-04	7844	MAGEE	TURNER

… continuation



OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
NULL	ANALYST	NULL	3500
CLERK	SALESMAN	1100	1100

Example: Recording Changes

```
CREATE OR REPLACE TRIGGER audit_emp_values
   AFTER UPDATE ON emp
   FOR EACH ROW
   BEGIN
     INSERT INTO audit_emp_values (user_name,
      timestamp, id, old_last_name, new_last_name,
      old_title, new_title, old_salary, new_salary)
     VALUES (USER, SYSDATE, :old.empno, :old.ename,
      :new.ename, :old.job, :new.job,
      :old.sal, :new.sal);
  END;/
```

Restrictions for Database Triggers

1. **Problem**: impossible to determine certain values during execution of a sequence of operations belonging to one and the same transaction

2. **Mutating tables:** contain rows which change their values after certain operation and which are used again before the current transaction commits

Example: Mutating Table

```
CREATE OR REPLACE TRIGGER emp_count
    AFTER DELETE ON emp
    FOR EACH ROW
    DECLARE
        num INTEGER;
    BEGIN
        SELECT COUNT(*) INTO num FROM emp;
        DBMS_OUTPUT.PUT_LINE(' There are now ' ||
              num || ' employees.');
    END;
/
```

```
 DELETE FROM emp
 WHERE  deptno = 30;
ERROR at line 1:
ORA-04091: table CGMA2.EMP is mutating, trigger/
function may not see it
```

Example: Mutating Table (fixed)

```
  CREATE OR REPLACE TRIGGER emp_count
    AFTER DELETE ON emp
    -- FOR EACH ROW
    DECLARE
        num INTEGER;
    BEGIN
        SELECT COUNT(*) INTO num FROM emp;
        DBMS_OUTPUT.PUT_LINE(' There are now ' ||
              num || ' employees.');
    END;
   /
```

```
SQL> DELETE FROM emp WHERE deptno = 30;
There are now 8 employees.
6 rows deleted.
```

Classifications & Scalability

Types of Triggers

1. **Generated:** based on some higher-level specification.
    - Foreign keys, primary keys, unique constraints, etc.
2. **Handcrafted**: usually specific to some application
    - Capture the application semantics

Why "Generated" Triggers?

Triggers (active rules) are difficult to write correctly

Idea:

- Trigger application specified at higher level (declarative)
- Automatic generation of actual triggers
- Guaranteed Correctness

Classification of Usage

Generated Triggers

- Kernel DBMS**:** hard coded into kernel
- DBMS services**:** enhances database functionality
- External applications**:** creating triggers specific to application

Handcrafted Triggers

- External applications: creating triggers specific to application

"Generated" Triggers/ DBMS Kernel

Referential integrity

- If foreign key in a table is deleted or updated, it causes an action usually specified by user: set null/cascade
- Primary keys, unique columns, etc.…

Materialized views

- Set of triggers that keep data consistent
    - DBMS services: enhances database functionality
    - External applications: creating triggers specific to application

"Generated" Triggers/ DBMS Services

Alerter

When data changes, message can be sent to user

Replication

If a table is copied, a trigger will observe updates to that original table and will change copied table.

Audit Trails

Monitoring any changes over a given table

"Generated" Triggers/ External Applications

- Workflow management

- External tools with support for generation of “Process Rules/Models”

“Handcrafted” Triggers

External Applications

- Straightforward use of triggers
- Application specific
    - Additional forms of “data integrity”
    - Could be used to compute derived columns
    - Or, enforce arbitrarily complex application-specific semantics

Examples:
Business rules, supply chain management, web applications, etc.

LESSON 2

Temporal Database Concepts

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Learn some of the concepts that have been developed to deal with the complexity of temporal database applications.
2. Understand how time is represented in databases, the different types of temporal information, and some of the different dimensions of time that may be needed.
3. Discuss how time can be incorporated into relational databases.

Temporal databases require some aspect of time when organizing information.

1. Healthcare
2. Insurance
3. Reservation systems
4. Scientific databases

Time considered as ordered sequence of points. Granularity determined by the application.

Chronon – Term used to describe minimal granularity of a particular application.

Reference point for measuring specific time events.

- Various calendars

SQL2 temporal data types

- DATE
- TIME
- TIMESTAMP
- INTERVAL
- PERIOD

Point events or facts

- Typically associated with a single time point
- Time series data

Duration events or facts

- Associated with specific time period
- Time period represented by start and end points

Valid time

- True in the real world

Transaction time

- Value of the system clock when information is valid in the system.

User-defined time

Bitemporal database

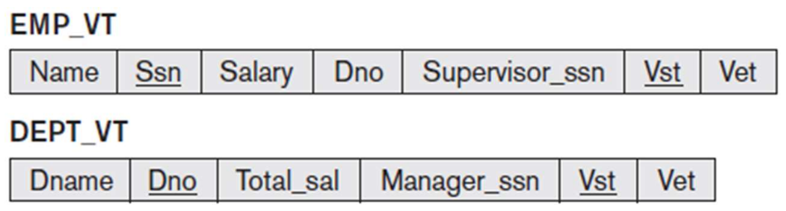
- Uses valid time and transaction time

Valid time relations

- Used to represent history of changes

Different types of temporal relational databases

1. Valid time database schema



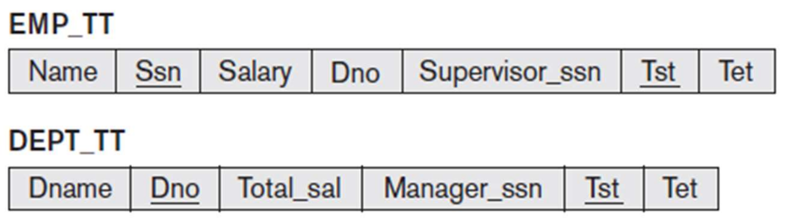
EMP_VT

Name	Ssn	Salary	Dno	Supervisor_ssn	Vst	Vet

DEPT_VT

Dname	Dno	Total_sal	Manager_ssn	Vst	Vet

2. Transaction time database schema



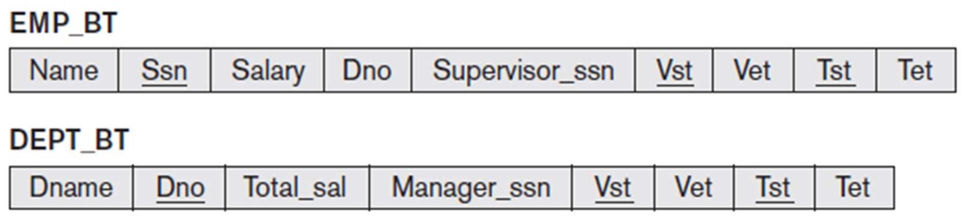
EMP_TT

Name	Ssn	Salary	Dno	Supervisor_ssn	Tst	Tet

DEPT_TT

Dname	Dno	Total_sal	Manager_ssn	Tst	Tet

3. Bitemporal database schema



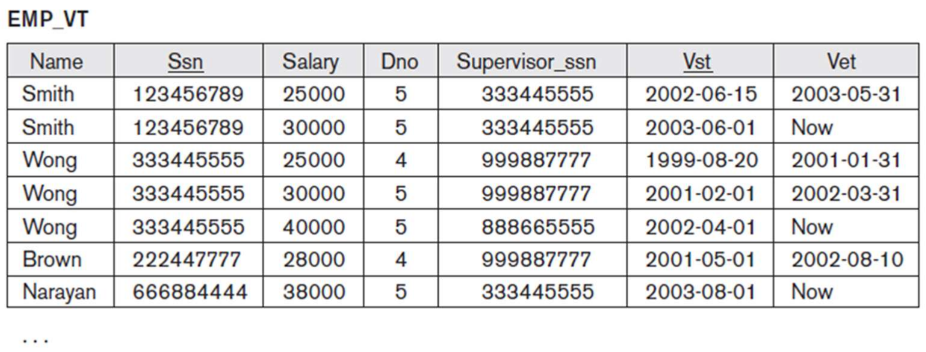
EMP_BT

Name	Ssn	Salary	Dno	Supervisor_ssn	Vst	Vet	Tst	Tet

DEPT_BT

Dname	Dno	Total_sal	Manager_ssn	Vst	Vet	Tst	Tet

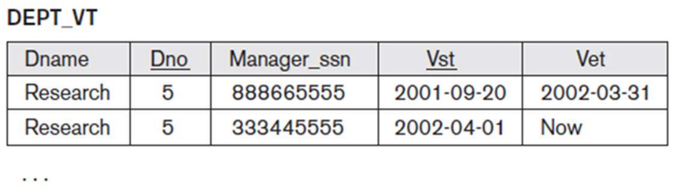
Some tuple versions in the valid time relations EMP_VT and DEPT_VT



EMP_VT

Name	Ssn	Salary	Dno	Supervisor_ssn	Vst	Vet
Smith	123456789	25000	5	333445555	2002-06-15	2003-05-31
Smith	123456789	30000	5	333445555	2003-06-01	Now
Wong	333445555	25000	4	999887777	1999-08-20	2001-01-31
Wong	333445555	30000	5	999887777	2001-02-01	2002-03-31
Wong	333445555	40000	5	888665555	2002-04-01	Now
Brown	222447777	28000	4	999887777	2001-05-01	2002-08-10
Narayan	666884444	38000	5	333445555	2003-08-01	Now

. . .



DEPT_VT

Dname	Dno	Manager_ssn	Vst	Vet
Research	5	888665555	2001-09-20	2002-03-31
Research	5	333445555	2002-04-01	Now

. . .

Types of updates

- Proactive
- Retroactive
- Simultaneous

Timestamp recorded whenever change is applied to database.

Bitemporal relations

- Application requires both valid time and transaction time

Implementation considerations

- Store all tuples in the same table
- Implementation considerations
- Create two tables: one for currently valid information and one for the rest
- Vertically partition temporal relation attributes into separate relations
  - New tuple created whenever any attribute updated

Append-only database – Keeps complete record of changes and corrections.

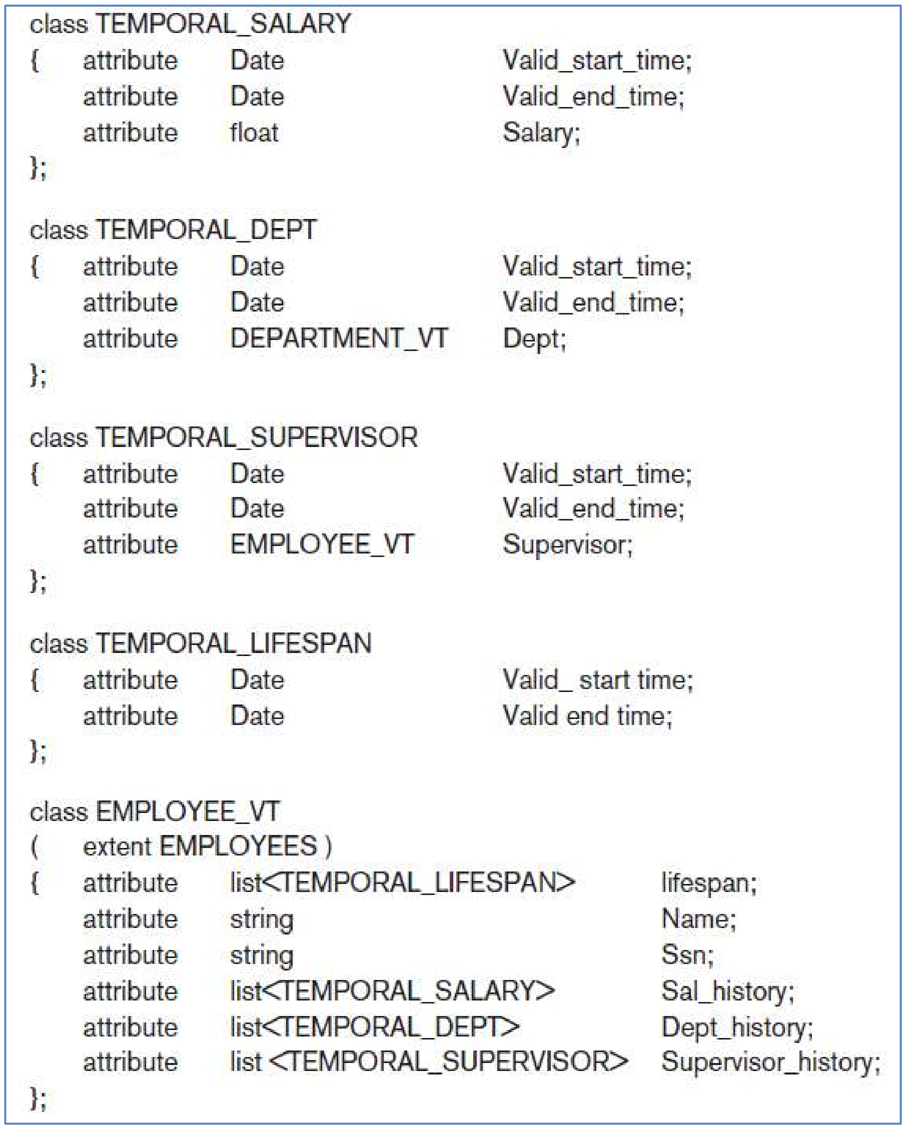
Attribute versioning

Simple complex object used to store all temporal changes of the object.

- **Time-varying attribute**
  - Values versioned over time by adding temporal periods to the attribute
- **Non-time-varying attribute**

- Values do not change over time

Possible ODL schema for a temporal valid time EMPLOYEE_VT object class using attribute versioning



```
class TEMPORAL_SALARY
{     attribute     Date                    Valid_start_time;
      attribute     Date                    Valid_end_time;
      attribute     float                   Salary;
};

class TEMPORAL_DEPT
{     attribute     Date                    Valid_start_time;
      attribute     Date                    Valid_end_time;
      attribute     DEPARTMENT_VT           Dept;
};

class TEMPORAL_SUPERVISOR
{     attribute     Date                    Valid_start_time;
      attribute     Date                    Valid_end_time;
      attribute     EMPLOYEE_VT             Supervisor;
};

class TEMPORAL_LIFESPAN
{     attribute     Date                    Valid_ start time;
      attribute     Date                    Valid end time;
};

class EMPLOYEE_VT
(     extent EMPLOYEES )
{     attribute     list<TEMPORAL_LIFESPAN>        lifespan;
      attribute     string                         Name;
      attribute     string                         Ssn;
      attribute     list<TEMPORAL_SALARY>          Sal_history;
      attribute     list<TEMPORAL_DEPT>            Dept_history;
      attribute     list <TEMPORAL_SUPERVISOR>     Supervisor_history;
};
```

TSQL2 language

Extends SQL for querying valid time and transaction time tables. Used to specify whether a relation is temporal or non-temporal.

Temporal database query conditions may involve time and attributes. Pure time condition involves only time. Attribute and time conditions.

CREATE TABLE statement

- Extended with optional AS clause

- Allows users to declare different temporal options

Examples:

AS VALID STATE<GRANULARITY> (valid time relation with valid time period)

AS TRANSACTION (transaction time relation with transaction time period)

STATE and EVENT Keywords - Specify whether a time period or point is associated with valid time dimension

Time series data

- Often used in financial, sales, and economics applications
- Special type of valid event data
- Event's time points predetermined according to fixed calendar
- Managed using specialized time series management systems
- Supported by some commercial DBMS packages

LESSON 3

Spatial Database Concepts

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Learn Spatial database and its common Types of Analysis for Spatial Data
2. Define spatial data types
3. Understand Spatial data mining techniques

Spatial databases

Spatial databases support information about objects in multi-dimensional space

Examples:

- cartographic databases
- geographic information systems
- weather information databases

A **spatial database** is optimized to store and query data related to objects in space, including points, lines and polygons.

Satellite images are a prominent example of **spatial data**.

Measurement operations

- Used to measure global properties of single objects

Spatial analysis operations

- Uncover spatial relationships within and among mapped data layers.

Flow analysis operations

- Help determine shortest path between two points

Location analysis

- Determine whether given set of points and lines lie within a given polygon

Digital terrain analysis

- Used to build three-dimensional models

Common Types of Analysis for Spatial Data

Analysis Type	Type of Operations and Measurements
Measurements	Distance, perimeter, shape, adjacency, and direction
Spatial analysis/statistics	Pattern, autocorrelation, and indexes of similarity and topology using spatial and nonspatial data
Flow analysis	Connectivity and shortest path
Location analysis	Analysis of points and lines within a polygon
Terrain analysis	Slope/aspect, catchment area, drainage network
Search	Thematic search, search by region

Spatial data types

Map data – Geographic or spatial features of objects in a map.
Attribute data – Descriptive data associated with map features.
Image data – Satellite images

Models of spatial information

Field models – Used to model spatial data that is continuous in nature, such as terrain elevation, temperature data, and soil variation characteristics.

Object models – Used for applications such as transportation networks, land parcels, buildings, and other objects that possess both spatial and non-spatial attributes.

Spatial Operators and Spatial Queries

Spatial operators are used to capture all the relevant geometric properties of objects embedded in the physical space and the relations between them, as well as to perform spatial analysis.

OPERATORS ARE CLASSIFIED INTO THREE BROAD CATEGORIES

1. **Topological operators** – Properties do not change when topological transformations applied.
2. **Projective operators** – Express concavity/convexity of objects.
3. **Metric operators –** Specifically describe object's geometry.

Dynamic Spatial Operators. Dynamic operations alter the objects upon which the operations act.

Three fundamental dynamic operations:
- Create
- Destroy
- Update

Spatial queries – Spatial queries are requests for spatial data that require the use of spatial operations.

Three typical types of spatial queries:
1. Range queries
   Example: find all hospitals with the Metropolitan Atlanta city area
2. Nearest neighbor queries
   Example: find police car nearest location of a crime
3. Spatial joins or overlays
   Example: find all homes within two miles of a lake

Spatial data indexing

A spatial index is used to organize objects into a set of buckets (which correspond to pages of secondary memory), so that objects in a particular spatial region can be easily located.

Some of the spatial indexing techniques:
1. Grid Files
2. R-Trees
3. Spatial Join Index

Spatial data mining techniques

1. **Spatial classification** – The task of classification is to assign an object to a class from a given set of classes based on the attribute values of this object. In spatial classification the attribute values of neighboring objects are also considered.

   The classification algorithm works as follows: The relevant attributes are extracted by comparing the attribute values of the target objects with the attribute values of their nearest neighbors. The determination of relevant attributes is based on the concepts of the nearest hit (the nearest neighbor belonging to the same class) and the nearest miss (the nearest neighbor belonging to a different class). In the construction of the decision tree, the neighbors of target objects are not considered individually. Instead, so-called buffers are created around the target objects and the non-spatial attribute values are aggregated over all objects contained in the buffer.

For instance, in the case of shopping malls a buffer may represent the area where its customers live or work. The size of the buffer yielding the maximum information gain is chosen and this size is applied to compute the aggregates for all relevant attributes.

2. **Spatial Association Rule Discovery Techniques –** Spatial association rule is a rule indicating certain association relationship among a set of spatial and possibly some non-spatial predicates. A strong rule indicates that the patterns in the rule have relatively frequent occurrences in the database and strong implication relationships. Additional data organization and retrieval tools can only handle the storage and retrieval of explicitly stored data. The extraction and comprehension of the knowledge implied by the huge amount of spatial data, though highly desirable, pose great challenges to currently available spatial database technologies.

   A spatial characteristic rule is a general description of a set of spatial-related data. For example, the description of the general weather patterns in a set of geographic regions is a spatial characteristic rule. A spatial discriminant rule is the general description of the contrasting or discriminating features of a class of spatial-related data from other class(es). For example, the comparison of the weather patterns in two geographic regions is a spatial discriminant rule. A spatial association rule is a rule which describes the implication of one or a set of features by another set of features in spatial databases. For example, a rule like most big cities in Canada are close to the Canada-U.S. border" is a spatial association rule. A strong rule indicates that the patterns in the rule have relatively frequent occurrences in the database and strong implication relationships.

3. **Spatial clustering** – Spatial clustering is a process of grouping a set of spatial objects into clusters so that objects within a cluster have high similarity in comparison to one another, but are dissimilar to objects in other clusters. For example, clustering is used to determine the "hot spots" in crime analysis and disease tracking. Hot spot analysis is the process of finding

unusually dense event clusters across time and space. Many criminal justice agencies are exploring the benefits provided by computer technologies to identify crime hot spots in order to take preventive strategies such as deploying saturation patrols in hot spot areas.

Spatial clustering can be applied to group similar spatial objects together; the implicit assumption is that patterns in space tend to be grouped rather than randomly located.

However, the statistical significance of spatial clusters should be measured by testing the assumption in the data. The test is critical before proceeding with any serious clustering analyses.

LESSON 4

Multimedia and Deductive Database Concepts

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Learn Multimedia and Deductive Database

Multimedia database is the collection of interrelated multimedia data that includes text, graphics (sketches, drawings), images, animations, video, audio etc. and have vast amounts of multisource multimedia data. The framework that manages different types of multimedia data which can be stored, delivered and utilized in different ways is known as multimedia database management system. There are three classes of the multimedia database which includes static media, dynamic media and dimensional media.

Content of Multimedia Database management system:

1. **Media data –** The actual data representing an object.
2. **Media format data –** Information such as sampling rate, resolution, encoding scheme etc. about the format of the media data after it goes through the acquisition, processing and encoding phase.
3. **Media keyword data –** Keywords description relating to the generation of data. It is also known as content descriptive data. Example: date, time and place of recording.
4. **Media feature data –** Content dependent data such as the distribution of colors, kinds of texture and different shapes present in data.

Types of multimedia applications based on data management characteristic are:

1. **Repository applications –** A Large amount of multimedia data as well as meta-data (Media format date, Media keyword data, and Media feature data) that is stored for retrieval purpose, e.g., Repository of satellite images, engineering drawings, and radiology scanned pictures.
2. **Presentation applications –** They involve delivery of multimedia data subject to temporal constraint. Optimal viewing or listening requires DBMS to deliver data at certain rate offering the quality of service above a certain

threshold. Here data is processed as it is delivered. Example: Annotating of video and audio data, real-time editing analysis.

3. **Collaborative work using multimedia information –** It involves executing a complex task by merging drawings, changing notifications. Example: Intelligent healthcare network.

There are still many challenges to multimedia databases, some of which are:

1. **Modelling –** Working in this area can improve database versus information retrieval techniques thus, documents constitute a specialized area and deserve special consideration.
2. **Design –** The conceptual, logical and physical design of multimedia databases has not yet been addressed fully as performance and tuning issues at each level are far more complex as they consist of a variety of formats like JPEG, GIF, PNG, MPEG which is not easy to convert from one form to another.
3. **Storage –** Storage of multimedia database on any standard disk presents the problem of representation, compression, mapping to device hierarchies, archiving and buffering during input-output operation. In DBMS, a "BLOB" (Binary Large Object) facility allows untyped bitmaps to be stored and retrieved.
4. **Performance –** For an application involving video playback or audio-video synchronization, physical limitations dominate. The use of parallel processing may alleviate some problems but such techniques are not yet fully developed. Apart from this multimedia database consume a lot of processing time as well as bandwidth.
5. **Queries and retrieval –**For multimedia data like images, video, audio accessing data through query opens up many issues like efficient query formulation, query execution and optimization which need to be worked upon.

Areas where multimedia database is applied are:

- **Documents and record management:** Industries and businesses that keep detailed records and variety of documents. Example: Insurance claim record.
- **Knowledge dissemination:** Multimedia database is a very effective tool for knowledge dissemination in terms of providing several resources. Example: Electronic books.
- **Education and training:** Computer-aided learning materials can be designed using multimedia sources which are nowadays very popular sources of learning. Example: Digital libraries.
- Marketing, advertising, retailing, entertainment and travel. Example: a virtual tour of cities.
- **Real-time control and monitoring:** Coupled with active database technology, multimedia presentation of information can be very effective means for monitoring and controlling complex tasks Example: Manufacturing operation control.

Deductive Databases

- Deductive database uses facts and rules
    - Inference engine can deduce new facts using rules
- Prolog/Datalog notation
    - Based on providing predicates with unique names
    - Predicate has an implicit meaning and a fixed number of arguments
        - If arguments are all constant values, predicate states that a certain fact is true
        - If arguments are variables, considered as a query or part of a rule or constraint

Prolog Notation and the Supervisory Tree

(a)

Facts

```
SUPERVISE(franklin, john).
SUPERVISE(franklin, ramesh).
SUPERVISE(franklin, joyce).
SUPERVISE(jennifer, alicia).
SUPERVISE(jennifer, ahmad).
SUPERVISE(james, franklin).
SUPERVISE(james, jennifer).
. . .
```

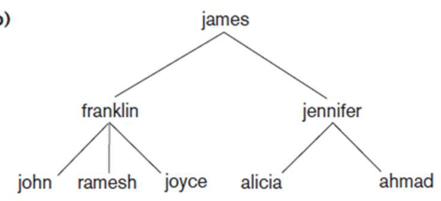
Rules

```
SUPERIOR(X, Y) :– SUPERVISE(X, Y).
SUPERIOR(X, Y) :– SUPERVISE(X, Z), SUPERIOR(Z, Y).
SUBORDINATE(X, Y) :– SUPERIOR(Y, X).
```

Queries

```
SUPERIOR(james, Y)?
SUPERIOR(james, joyce)?
```

(b)



- **Datalog notation**
    - **Program built from basic objects called atomic formulas**
    - **Literals of the form $p(a_1,a_2,\ldots a_n)$**
        - **p is the predicate name**
        - **n is the number of arguments for predicate p**
- **Interpretations of rules**
    - **Proof-theoretic versus model-theoretic**
    - **Deductive axioms**
    - **Ground axioms**

Proving a new fact

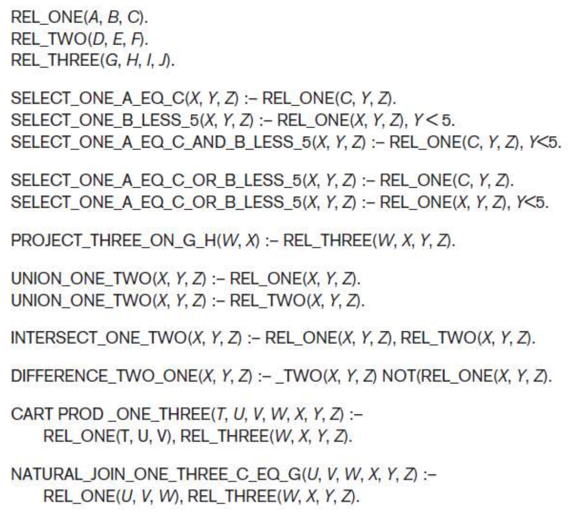
```
1. SUPERIOR(X, Y) :– SUPERVISE(X, Y).                     (rule 1)
2. SUPERIOR(X, Y) :– SUPERVISE(X, Z), SUPERIOR(Z, Y).     (rule 2)
3. SUPERVISE(jennifer, ahmad).                            (ground axiom, given)
4. SUPERVISE(james, jennifer).                            (ground axiom, given)
5. SUPERIOR(jennifer, ahmad).                             (apply rule 1 on 3)
6. SUPERIOR(james, ahmad).                                (apply rule 2 on 4 and 5)
```

- **Safe program or rule**
    - Generates a finite set of facts

- **Nonrecursive query**
    - Includes only nonrecursive predicates

Use of Relational Operations

Predicates for illustrating relational operations



```
REL_ONE(A, B, C).
REL_TWO(D, E, F).
REL_THREE(G, H, I, J).

SELECT_ONE_A_EQ_C(X, Y, Z) :– REL_ONE(C, Y, Z).
SELECT_ONE_B_LESS_5(X, Y, Z) :– REL_ONE(X, Y, Z), Y < 5.
SELECT_ONE_A_EQ_C_AND_B_LESS_5(X, Y, Z) :– REL_ONE(C, Y, Z), Y<5.

SELECT_ONE_A_EQ_C_OR_B_LESS_5(X, Y, Z) :– REL_ONE(C, Y, Z).
SELECT_ONE_A_EQ_C_OR_B_LESS_5(X, Y, Z) :– REL_ONE(X, Y, Z), Y<5.

PROJECT_THREE_ON_G_H(W, X) :– REL_THREE(W, X, Y, Z).

UNION_ONE_TWO(X, Y, Z) :– REL_ONE(X, Y, Z).
UNION_ONE_TWO(X, Y, Z) :– REL_TWO(X, Y, Z).

INTERSECT_ONE_TWO(X, Y, Z) :– REL_ONE(X, Y, Z), REL_TWO(X, Y, Z).

DIFFERENCE_TWO_ONE(X, Y, Z) :– _TWO(X, Y, Z) NOT(REL_ONE(X, Y, Z).

CART PROD _ONE_THREE(T, U, V, W, X, Y, Z) :–
    REL_ONE(T, U, V), REL_THREE(W, X, Y, Z).

NATURAL_JOIN_ONE_THREE_C_EQ_G(U, V, W, X, Y, Z) :–
    REL_ONE(U, V, W), REL_THREE(W, X, Y, Z).
```

POST-TEST

MODULE 1:

1. What are the differences between row-level and statement-level active rules?
2. What are the differences among immediate, deferred, and detached consideration of active rule conditions?
3. What are the differences among immediate, deferred, and detached execution of active rule actions?
4. Briefly discuss the consistency and termination problems when designing a set of active rules.
5. Discuss some applications of active databases.
6. Discuss how time is represented in temporal databases and compare the different time dimensions.
7. What are the differences among valid time, transaction time, and Bitemporal relations?
8. Describe how the insert, delete, and update commands should be implemented on a valid time relation.
9. Describe how the insert, delete, and update commands should be implemented on a Bitemporal relation.
10. Describe how the insert, delete, and update commands should be implemented on a valid time relation.
11. What are the main differences between tuple versioning and attribute versioning?
12. How do spatial databases differ from regular databases?
13. What are the different types of spatial data?
14. Name the main types of spatial operators and different classes of spatial queries
15. What are the properties of R-trees that act as an index for spatial data?
16. Describe how a spatial join index between spatial objects can be constructed.
17. What are the different types of spatial data mining?
18. State the general form of a spatial association rule. Give an example of a spatial association rule.
19. What are the different types of multimedia sources?
20. How are multimedia sources indexed for content-based retrieval?
21. What important features of images are used to compare them?
22. What are the different approaches to recognizing objects in images?
23. How is semantic tagging of images used?
24. What are the difficulties in analyzing audio sources?
25. What are deductive databases?
26. Define the clausal form of formulas and Horn clauses.
27. What is theorem proving, and what is proof-theoretic interpretation of rules?
28. What is model-theoretic interpretation and how does it differ from proof theoretic interpretation?
29. What are fact-defined predicates and rule-defined predicates?
30. What is a safe rule?

REFERE NCES

1. Database Systems: Design, Implementation, and Management
2. 4th Edition, Peter Rob & Carlos Coronel
3. Database Systems: Design, Implementation, and Management, Fifth Edition, Rob and Coronel
4. Database System Concepts Seventh Edition, Avi Silberschatz ET. Al.
5. Fundamentals of Database Systems Seventh Edition, by Ramez Elmasri and Shamkant B. Navathe.

Internet:

6. https://www.geeksforgeeks.org/