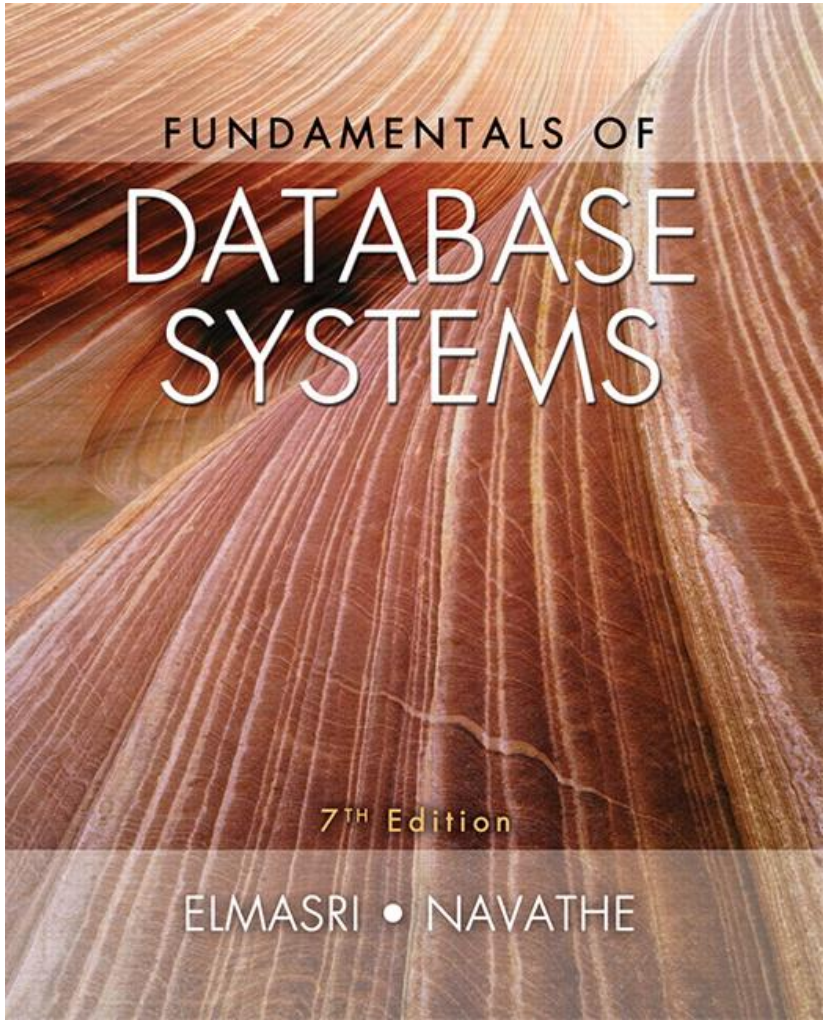

CHAPTER 26

Enhanced Data Models: Introduction to Active, Temporal, Spatial, Multimedia, and Deductive Databases

active rules. These rules can be automatically triggered by events that occur, such as database updates or certain times being reached, and can initiate certain actions that have been specified in the rule declaration to occur if certain conditions are met.

temporal databases, which permit the database system to store a history of changes, and allow users to query both current and past states of the database.

spatial database concepts. We discuss types of spatial data, different kinds of spatial analyses, operations on spatial data, types of spatial queries, spatial data indexing, spatial data mining, and applications of spatial databases.

Multimedia databases provide features that allow users to store and query different types of multimedia information, which includes **images**, **video clips**, **audio clips** and **documents ,** automatic analysis of images, object recognition in images, and semantic tagging of
images,

A **deductive database system** includes capabilities to define **(deductive) rules**, which can deduce or infer additional information from the facts that are stored in a database.

26.1 Active Database Concepts and Triggers

- Database systems implement rules that specify actions automatically triggered by certain events
- Triggers
  - Technique for specifying certain types of active rules
- Commercial relational DBMSs have various versions of triggers available
  - Oracle syntax used to illustrate concepts

Generalized Model for Active Databases and Oracle Triggers

- Event-condition-action (ECA) model
  - Event triggers a rule
    - Usually database update operations
  - Condition determines whether rule action should be completed
    - Optional
    - Action will complete only if condition evaluates to true
  - Action to be taken
    - Sequence of SQL statements, transaction, or external program

Example

- Events that may cause a change in value of Total_sal attribute
  - Inserting new employee
  - Changing salary
  - Reassigning or deleting employees

EMPLOYEE

Name	Ssn	Salary	Dno	Supervisor_ssn

DEPARTMENT

Dname	Dno	Total_sal	Manager_ssn

Figure 26.1 A simplified COMPANY database used for active rule examples

Example (cont'd.)

- Condition to be evaluated
  - Check that value of Dno attribute is not NULL
- Action to be taken
  - Automatically update the value of Total_sal

(a)

```
R1: CREATE TRIGGER Total_sal1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN ( NEW.Dno IS NOT NULL )
        UPDATE DEPARTMENT
        SET Total_sal = Total_sal + NEW.Salary
        WHERE Dno = NEW.Dno;

R2: CREATE TRIGGER Total_sal2
    AFTER UPDATE OF Salary ON EMPLOYEE
    FOR EACH ROW
    WHEN ( NEW.Dno IS NOT NULL )
        UPDATE DEPARTMENT
        SET Total_sal = Total_sal + NEW.Salary – OLD.Salary
        WHERE Dno = NEW.Dno;

R3: CREATE TRIGGER Total_sal3
    AFTER UPDATE OF Dno ON EMPLOYEE
    FOR EACH ROW
        BEGIN
        UPDATE DEPARTMENT
        SET Total_sal = Total_sal + NEW.Salary
        WHERE Dno = NEW.Dno;
        UPDATE DEPARTMENT
        SET Total_sal = Total_sal – OLD.Salary
        WHERE Dno = OLD.Dno;
        END;
```

Fig
au

```
R4:  CREATE TRIGGER Total_sal4
     AFTER DELETE ON EMPLOYEE
     FOR EACH ROW
     WHEN ( OLD.Dno IS NOT NULL)
          UPDATE DEPARTMENT
          SET Total_sal = Total_sal – OLD.Salary
          WHERE Dno = OLD.Dno;

(b)  R5:  CREATE TRIGGER Inform_supervisor1
     BEFORE INSERT OR UPDATE OF Salary, Supervisor_ssn
          ON EMPLOYEE
     FOR EACH ROW
     WHEN ( NEW.Salary > ( SELECT Salary FROM EMPLOYEE
                           WHERE Ssn = NEW.Supervisor_ssn ) )
          inform_supervisor(NEW.Supervisor_ssn, NEW.Ssn );
```

Figure 26.2 (cont'd.) Specifying active rules as triggers in Oracle notation (b) Trigger for comparing an employee's salary with that of his or her supervisor

Figure 26.3
A syntax summary for specifying triggers in the Oracle system (main options only).

```
<trigger>            ::= CREATE TRIGGER <trigger name>
                         ( AFTER | BEFORE ) <triggering events> ON <table name>
                         [ FOR EACH ROW ]
                         [ WHEN <condition> ]
                         <trigger actions> ;
<triggering events>  ::= <trigger event> {OR <trigger event> }
<trigger event>      ::= INSERT | DELETE | UPDATE [ OF <column name> { , <column name> } ]
<trigger action>     ::= <PL/SQL block>
```

Design and Implementation Issues for Active Databases

- Deactivated rule- Will not be triggered by the triggering event - This feature allows users to selectively deactivate rules for certain periods of time when they are not needed.
- Activate command - Makes the rule active again
- Drop command - Deletes the rule from the system : Approach: group rules into rule sets

  - Entire rule set can be activated, deactivated, or dropped

Design and Implementation Issues for Active Databases (cont'd.)

- Timing of action
  - Before trigger executes trigger before executing event that caused the trigger
  - After trigger executes trigger after executing the event
  - Instead of trigger executes trigger instead of executing the event
- Action can be considered separate transaction
  - Or part of same transaction that triggered the rule

Design and Implementation Issues for Active Databases (cont'd.)

- Rule consideration
  - Immediate consideration
    - Condition evaluated as part of same transaction
    - Evaluate condition either before, after, or instead of executing the triggering event
  - Deferred consideration
    - Condition evaluated at the end of the transaction
  - Detached consideration
    - Condition evaluated as a separate transaction

Design and Implementation Issues for Active Databases (cont'd.)

- Row-level rule
  - Rule considered separately for each row
- Statement-level rule
  - Rule considered once for entire statement
- Difficult to guarantee consistency and termination of rules

Examples of Statement-Level Active Rules in STARBURST

```
R1S: CREATE RULE Total_sal1 ON EMPLOYEE
     WHEN   INSERTED
     IF     EXISTS    ( SELECT * FROM INSERTED WHERE Dno IS NOT NULL)
     THEN   UPDATE      DEPARTMENT AS D
            SET         D.Total_sal = D.Total_sal +
                      ( SELECT SUM (I.Salary) FROM INSERTED AS I WHERE D.Dno = I.Dno )
            WHERE     D.Dno IN ( SELECT Dno FROM INSERTED );

R2S: CREATE RULE Total_sal2 ON EMPLOYEE
     WHEN   UPDATED   ( Salary )
     IF     EXISTS    ( SELECT * FROM NEW-UPDATED WHERE Dno IS NOT NULL)
            OR EXISTS ( SELECT * FROM OLD-UPDATED WHERE Dno IS NOT NULL)
     THEN   UPDATE      DEPARTMENT AS D
            SET         D.Total_sal = D.Total_sal +
                      ( SELECT SUM (N.Salary) FROM NEW-UPDATED AS N
                        WHERE D.Dno = N.Dno ) -
                      ( SELECT SUM (O.Salary) FROM OLD-UPDATED AS O
                        WHERE D.Dno = O.Dno )
            WHERE     D.Dno IN ( SELECT Dno FROM NEW-UPDATED ) OR
                      D.Dno IN ( SELECT Dno FROM OLD-UPDATED);
```

Figure 26.5 (continues) Active rules using statement-level semantics in STARBURST notation

Examples of Statement-Level Active Rules in STARBURST (cont'd.)

```
R3S:  CREATE RULE Total_sal3 ON EMPLOYEE
      WHEN   UPDATED   ( Dno )
      THEN   UPDATE      DEPARTMENT AS D
             SET         D.Total_sal = D.Total_sal +
                       ( SELECT SUM (N.Salary) FROM NEW-UPDATED AS N
                         WHERE D.Dno = N.Dno )
             WHERE       D.Dno IN ( SELECT Dno FROM NEW-UPDATED );
             UPDATE      DEPARTMENT AS D
             SET         D.Total_sal = Total_sal –
                       ( SELECT SUM (O.Salary) FROM OLD-UPDATED AS O
                         WHERE D.Dno = O.Dno )
             WHERE       D.Dno IN ( SELECT Dno FROM OLD-UPDATED );
```

Figure 26.5 (cont'd.) Active rules using statement-level semantics in STARBURST notation

Potential Applications for Active Databases

- Allow notification of certain conditions that occur
- Enforce integrity constraints
- Automatically maintain derived data
- Maintain consistency of materialized views
- Enable consistency of replicated tables

Triggers in SQL-99

```
T1: CREATE TRIGGER Total_sal1
    AFTER UPDATE OF Salary ON EMPLOYEE
    REFERENCING OLD ROW AS O, NEW ROW AS N
    FOR EACH ROW
    WHEN  ( N.Dno IS NOT NULL )
    UPDATE DEPARTMENT
    SET Total_sal = Total_sal + N.salary – O.salary
    WHERE Dno = N.Dno;

T2: CREATE TRIGGER Total_sal2
    AFTER UPDATE OF Salary ON EMPLOYEE
    REFERENCING OLD TABLE AS O, NEW TABLE AS N
    FOR EACH STATEMENT
    WHEN  EXISTS ( SELECT *FROM N WHERE N.Dno IS NOT NULL ) OR
          EXISTS ( SELECT * FROM O WHERE O.Dno IS NOT NULL )
    UPDATE DEPARTMENT AS D
    SET D.Total_sal = D.Total_sal
    + ( SELECT SUM (N.Salary) FROM N WHERE D.Dno=N.Dno )
    – ( SELECT SUM (O.Salary) FROM O WHERE D.Dno=O.Dno )
    WHERE Dno IN ( ( SELECT Dno FROM N ) UNION ( SELECT Dno FROM O ) );
```

Figure 26.6 Trigger T1 illustrating the syntax for defining triggers in SQL-99

26.2 Temporal Database Concepts

- Temporal databases require some aspect of time when organizing information
- Healthcare- where patient histories need to be maintained
- Insurance-where claims and accident histories are required as well as information about the times when insurance policies are in effect
- Reservation systems- hotel, airline, car rental, train, and so on), where information on the dates and times when reservations are in effect are required;
- Scientific databases- where data collected from experiments includes the time when each data is measured and so on
- Time considered as ordered sequence of points
  - Granularity determined by the application

Temporal Database Concepts (cont'd.)

- Chronon
  - Term used by researchers to describe minimal granularity of a particular application
- Reference point for measuring specific time events
- Various calendars organizes time into different time units for convenience.
- SQL2 temporal data types
- DATE, TIME, TIMESTAMP, INTERVAL- (a relative time duration, such as 10 days or 250 minutes),, PERIOD -(an *anchored* time duration with a fixed starting point, such as the 10-day period from January 1, 2009, to January 10, 2009, inclusive)

Temporal Database Concepts (cont'd.)

- 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 and Transaction Time Dimensions
- True in the real world - the time that the event occurred, or the period during which the fact was considered to be true *in the real world.*
- A temporal database using this interpretation is called a **valid time database**.

Temporal Database Concepts (cont'd.)

- Transaction time
  - Value of the system clock when information is valid in the system called as time dimensions
- User-defined time-other interpretations are intended for time, the user can define the semantics and program the applications appropriately, and it is called a **user-defined time**.
- Bitemporal database
  - Uses valid time and transaction time
- Valid time relations
  - Used to represent history of changes

Temporal Database Concepts (cont'd.)

Figure 26.7 Different types of temporal relational databases (a) Valid time database schema (b) Transaction time database schema (c) Bitemporal database schema

Temporal Database Concepts (cont'd.)

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

. . .

Figure 26.8 Some tuple versions in the valid time relations EMP_VT and DEPT_VT

Temporal Database Concepts (cont'd.)

- Types of updates
- Proactive - it is applied to the database *before* it becomes effective in the real world -For example, the salary update of Smith may have been entered in the database on May 15, 2003, at 8:52:12 A.M., say, even though the salary
- change in the real world is effective on June 1, 2003.
- Retroactive - If the update is applied to the database *after* it becomes effective in the real world,
- Simultaneous - An update that is applied at the same time as it becomes effective.
- Timestamp recorded whenever change is applied to database – is **Transaction Time Relations.**
- Bitemporal relations -Application requires both valid time and transaction time

EMP_BT

Name	Ssn	Salary	Dno	Supervisor_ssn	Vst	Vet	Tst	Tet
Smith	123456789	25000	5	333445555	2002-06-15	Now	2002-06-08, 13:05:58	2003-06-04,08:56:12
Smith	123456789	25000	5	333445555	2002-06-15	2003-05-31	2003-06-04, 08:56:12	uc
Smith	123456789	30000	5	333445555	2003-06-01	Now	2003-06-04, 08:56:12	uc
Wong	333445555	25000	4	999887777	1999-08-20	Now	1999-08-20, 11:18:23	2001-01-07,14:33:02
Wong	333445555	25000	4	999887777	1999-08-20	2001-01-31	2001-01-07, 14:33:02	uc
Wong	333445555	30000	5	999887777	2001-02-01	Now	2001-01-07, 14:33:02	2002-03-28,09:23:57
Wong	333445555	30000	5	999887777	2001-02-01	2002-03-31	2002-03-28, 09:23:57	uc
Wong	333445555	40000	5	888667777	2002-04-01	Now	2002-03-28, 09:23:57	uc
Brown	222447777	28000	4	999887777	2001-05-01	Now	2001-04-27, 16:22:05	2002-08-12,10:11:07
Brown	222447777	28000	4	999887777	2001-05-01	2002-08-10	2002-08-12, 10:11:07	uc
Narayan	666884444	38000	5	333445555	2003-08-01	Now	2003-07-28, 09:25:37	uc

. . .

DEPT_VT

Dname	Dno	Manager_ssn	Vst	Vet	Tst	Tet
Research	5	888665555	2001-09-20	Now	2001-09-15,14:52:12	2001-03-28,09:23:57
Research	5	888665555	2001-09-20	1997-03-31	2002-03-28,09:23:57	uc
Research	5	333445555	2002-04-01	Now	2002-03-28,09:23:57	uc

Figure 26.9
Some tuple versions in the bitemporal relations EMP_BT and DEPT_BT.

Temporal Database Concepts (cont'd.)

- Implementation considerations
  - Store all tuples in the same table
  - 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

Temporal Database Concepts (cont'd.)

Incorporating Time in Object-Oriented Databases Using Attribute Versioning

- 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

```
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;
};
```

Figure 26.10 Possible ODL schema for a temporal valid time EMPLOYEE_VT object class using attribute versioning

Temporal Database Concepts (cont'd.)

- TSQL2 language
  - Extends SQL for querying valid time and transaction time tables
  - Used to specify whether a relation is temporal or nontemporal
- Temporal database query conditions may involve time and attributes
  - Pure time condition involves only time
  - Attribute and time conditions

Temporal Database Concepts (cont'd.)

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

Temporal Database Concepts (cont'd.)

- 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

26.3 Spatial Database Concepts

- Spatial databases support information about objects in multidimensional space
  - Examples: cartographic databases, geographic information systems, weather information databases
- Spatial relationships among the objects are important
- Optimized to query data such as points, lines, and polygons
  - Spatial queries

Spatial Database Concepts (cont'd.)

- 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

Spatial Database Concepts (cont'd.)

- Location analysis
  - Determine whether given set of points and lines lie within a given polygon
- Digital terrain analysis
  - Used to build three-dimensional models

Spatial Database Concepts (cont'd.)

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

Table 26.1 Common types of analysis for spatial data

Spatial Database Concepts (cont'd.)

- 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
  - Object models

Spatial Database Concepts (cont'd.)

- Spatial operator categories
  - Topological operators
    - Properties do not change when topological transformations applied
  - Projective operators
    - Express concavity/convexity of objects
  - Metric operators
    - Specifically describe object's geometry
  - Dynamic spatial operators
    - Create, destroy, and update

Spatial Database Concepts (cont'd.)

- Spatial queries
  - Range queries
    - Example: find all hospitals with the Metropolitan Atlanta city area
  - Nearest neighbor queries
    - Example: find police car nearest location of a crime
  - Spatial joins or overlays
    - Example: find all homes within two miles of a lake

Spatial Database Concepts (cont'd.)

- Spatial data indexing
  - Grid files
  - R-trees
  - Spatial join index
- Spatial data mining techniques
  - Spatial classification
  - Spatial association
  - Spatial clustering

26.4 Multimedia Database Concepts

- Multimedia databases allow users to store and query images, video, audio, and documents
- Content-based retrieval
  - Automatic analysis
  - Manual identification
  - Color often used in content-based image retrieval
  - Texture and shape
- Object recognition
  - Scale-invariant feature transform (SIFT) approach

Multimedia Database Concepts (cont'd.)

- Semantic tagging of images
  - User-supplied tags
  - Automated generation of image tags
  - Web Ontology Language (OWL) provides concept hierarchy
- Analysis of audio data sources
  - Text-based indexing
  - Content-based indexing

26.5 Introduction to 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)

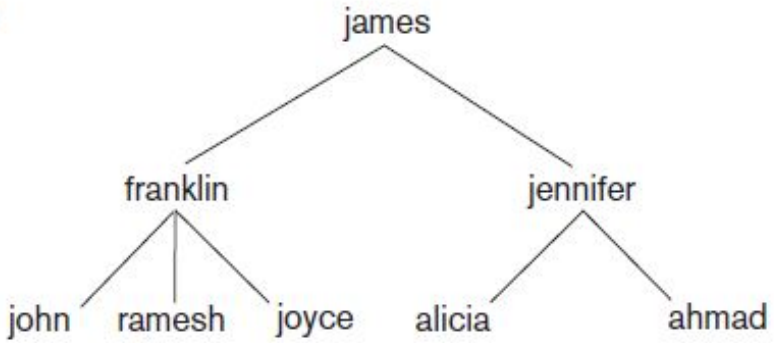

Figure 26.11 (a) Prolog notation (b) The supervisory tree

Introduction to Deductive Databases (cont'd.)

- 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

Introduction to Deductive Databases (cont'd.)

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)

Figure 26.12 Proving a new fact

Introduction to Deductive Databases (cont'd.)

- Safe program or rule
  - Generates a finite set of facts
- Nonrecursive query
  - Includes only nonrecursive predicates

Use of Relational Operations

Figure 26.16
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).
```

26.6 Summary

- Active databases
  - Specify active rules
- Temporal databases
  - Involve time concepts
- Spatial databases
  - Involve spatial characteristics
- Multimedia databases
  - Store images, audio, video, documents, and more
- Deductive databases
  - Prolog and Datalog notation