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

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
Figure 26.2 Specifying active rules as triggers in Oracle notation (a) Triggers for automatically maintaining the consistency of Total_sal of DEPARTMENT
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

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 );
Deactivated rule
- Will not be triggered by the triggering event

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
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
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 STARBUSTRST (cont’d.)

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

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 );
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
  - Insurance
  - Reservation systems
  - Scientific databases

- Time considered as ordered sequence of points
  - Granularity determined by the application
Temporal Database Concepts (cont’d.)

- 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
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
  - True in the real world
Temporal Database Concepts (cont’d.)

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

- Timestamp recorded whenever change is applied to database

- Bitemporal relations
  - Application requires both valid time and transaction time
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.)

- 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
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
Table 26.1 Common types of analysis for spatial data

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

(b) 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)?
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
Figure 26.12 Proving a new fact

1. SUPERIOR(X, Y) :- SUPervise(X, Y).
2. SUPERIOR(X, Y) :- SUPervise(X, Z), SUPERIOR(Z, Y).
3. SUPervise(jennifer, ahmad).
4. SUPervise(james, jennifer).
5. SUPERIOR(jennifer, ahmad).
6. SUPERIOR(james, ahmad).

(rule 1)
(rule 2)
(ground axiom, given)
(ground axiom, given)
(apply rule 1 on 3)
(apply rule 2 on 4 and 5)
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) :-
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