Active Database Concepts and Triggers

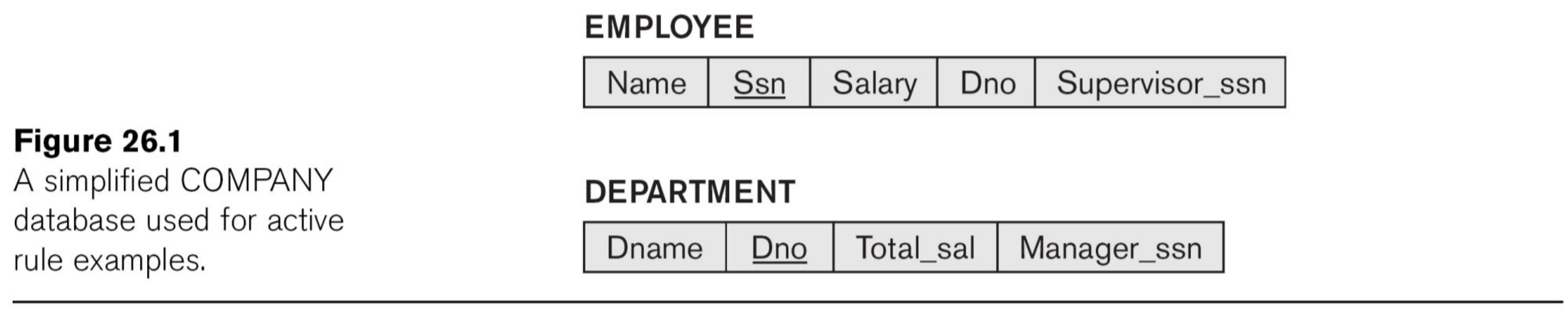
Generalized Model for Active Databases and Oracle **Triggers**

- **Triggers** are executed when a specified condition occurs during insert/delete/update
  - Triggers are action that fire automatically based on these conditions

Event-Condition-Action (ECA) Model

Generalized Model (cont.)

- Triggers follow an Event-condition-action (ECA) model
  - **Event**:
    - Database modification
      - E.g., insert, delete, update),
  - **Condition**:
    - Any true/false expression
      - Optional: If no condition is specified then condition is always true
  - **Action**:
    - Sequence of SQL statements that will be automatically executed



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.

Trigger Example

Generalized Model (cont.)

- When a new employees is added to a department, modify the Total_sal of the Department to include the new employees salary
  - Logically this means that we will Condition TRIGGER, let us call the trigger Total_sal1
    - This trigger will execute AFTER INSERT ON Employee table
    - It will do the following FOR EACH ROW
      - WHEN NEW.Dno is NOT NULL
      - The trigger will UPDATE DEPARTMENT
      - By SETting the new Total_sal to be the sum of
        - old Total_sal and NEW. Salary
        - WHERE the Dno matches the NEW.Dno;

Example: Trigger Definition

Can be CREATE or ALTER

```
CREATE TRIGGER Total_sal1
    AFTER INSERT ON Employee
    FOR EACH ROW
```

Can be FOR, AFTER, INSTEAD OF

Can be INSERT, UPDATE, DELETE

[redacted]

The condition

[redacted]

The action

```
(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;

     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;
```

Figure 26.2
Specifying active rules as triggers in Oracle notation. (a) Triggers for automatically maintaining the consistency of Total_sal of DEPARTMENT. (b) Trigger for comparing an employee's salary with that of his or her supervisor.

```
(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 );
```

CREATE or ALTER TRIGGER

Generalized Model (cont.)

- CREATE TRIGGER <name>
  - Creates a trigger
- ALTER TRIGGER <name>
  - Alters a trigger (assuming one exists)
- CREATE OR ALTER TRIGGER <name>
  - Creates a trigger if one does not exist
  - Alters a trigger if one does exist
    - Works in both cases, whether a trigger exists or not

Conditions

Generalized Model (cont.)

- AFTER
  - Executes after the event
- BEFORE
  - Executes before the event
- INSTEAD OF
  - Executes **instead of** the event
    - Note that event does not execute in this case
      - E.g., used for modifying views

Row-Level versus Statement-level

Generalized Model (cont.)

- Triggers can be
  - **Row-level**
    - FOR EACH ROW specifies a row-level trigger
  - **Statement-level**
    - Default (when FOR EACH ROW is not specified)
- Row level triggers
  - Executed separately for each affected row
- Statement-level triggers
  - Execute once for the SQL statement,

Condition

Generalized Model (cont.)

- Any true/false condition to control whether a trigger is activated or not
  - Absence of condition means that the trigger will always execute for the event
  - Otherwise, condition is evaluated
    - before the event for BEFORE trigger
    - after the event for AFTER trigger

Action

Generalized Model (cont.)

- Action can be
  - One SQL statement
  - A sequence of SQL statements enclosed between a BEGIN and an END
- Action specifies the relevant modifications

Triggers on Views

Generalized Model (cont.)

- INSTEAD OF triggers are used to process view modifications

Active Database Concepts and Triggers

Design and Implementation Issues for Active Databases

- An active database allows users to make the following changes to triggers (rules)
  - Activate
  - Deactivate
  - Drop

Active Database Concepts and Triggers

Design and Implementation Issues for Active Databases

- An event can be considered in 3 ways
  - Immediate consideration
  - Deferred consideration
  - Detached consideration

Active Database Concepts and Triggers

Design and Implementation Issues (cont.)

- Immediate consideration
  - Part of the same transaction and can be one of the following depending on the situation
    - Before
    - After
    - Instead of
- Deferred consideration
  - Condition is evaluated at the end of the transaction
- Detached consideration
  - Condition is evaluated in a separate transaction

Active Database Concepts and Triggers

Potential Applications for Active Databases

- Notification
  - Automatic notification when certain condition occurs
- Enforcing integrity constraints
  - Triggers are smarter and more powerful than constraints
- Maintenance of derived data
  - Automatically update derived data and avoid anomalies due to redundancy
    - E.g., trigger to update the Total_sal in the earlier example

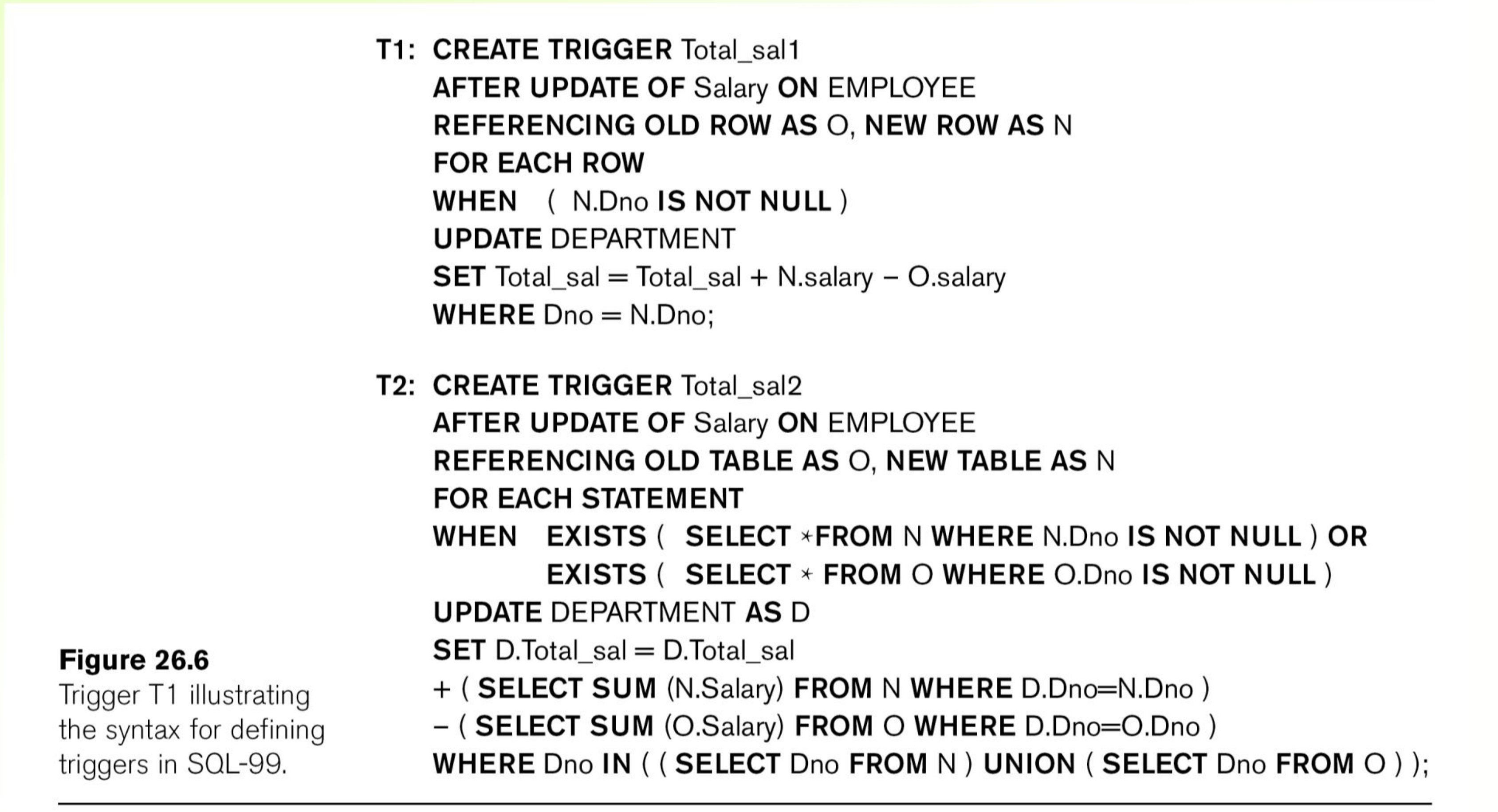
Active Database Concepts and Triggers

Triggers in SQL-99

- Can alias variables inside the REFERENCING clause

Active Database Concepts and Triggers

- Trigger examples



```
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.

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

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

```
R1: CREATE TRIGGER T1
    AFTER INSERT ON TABLE1
    FOR EACH ROW
        UPDATE TABLE2
        SET Attribute1 = ... ;

R2: CREATE TRIGGER T2
    AFTER UPDATE OF Attribute1 ON TABLE2
    FOR EACH ROW
        INSERT INTO TABLE1 VALUES ( ... );
```

Figure 26.4
An example to illustrate the termination problem for active rules.

```
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.