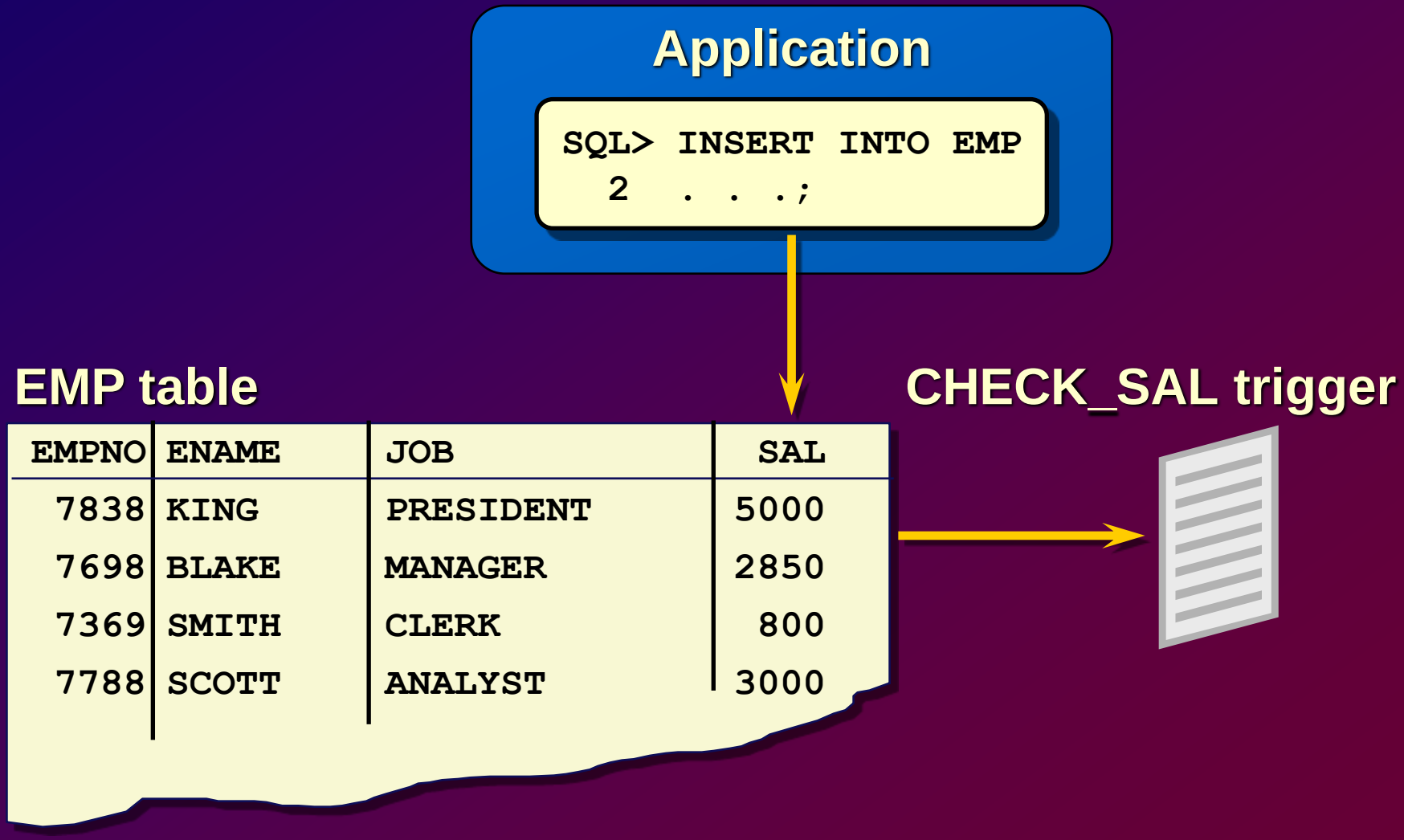
Overview of Triggers

- A trigger is a PL/SQL block that executes implicitly whenever a particular event takes place.
- A trigger can be either a database trigger or an application trigger.

Designing Triggers: Guidelines

- Perform related actions
- Use triggers for global operations
- Do not reinvent the wheel
- Watch out for overkill

Database Trigger: Example



Creating Triggers

- Trigger timing: BEFORE or AFTER
- Triggering event: INSERT or UPDATE or DELETE
- Table name: On table
- Trigger type: Row or statement
- When clause: Restricting condition
- Trigger body: DECLARE
  BEGIN
  END;

Trigger Components

Trigger Timing: When should the trigger fire?

- **BEFORE: The code in the trigger body will execute before the triggering DML event.**
- **AFTER: The code in the trigger body will execute after the triggering DML event.**

Trigger Components

Trigger Timing: When should the trigger fire?

- **INSTEAD OF: The code in the trigger body will execute instead of the the triggering statement. Used for VIEWS that are not otherwise modifiable.**

Trigger Components

Triggering Event:

What DML operation will cause the trigger to execute?

- **INSERT**
- **UPDATE**
- **DELETE**
- **Any combination of the above**

Trigger Components

Trigger Type:

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

- **Statement: The trigger body executes once for the triggering event. This is the default.**
- **Row: The trigger body executes once for each row affected by the triggering event.**

Trigger Components

Trigger Body:

What action should the trigger perform?

- **The trigger body is defined with an anonymous PL/SQL block.**

```
[DECLARE]
BEGIN
[EXCEPTION]
END;
```

Firing Sequence of Database Triggers on a Single Row

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

BEFORE statement trigger

BEFORE row trigger

AFTER row trigger

AFTER statement trigger

Statement and Row Triggers

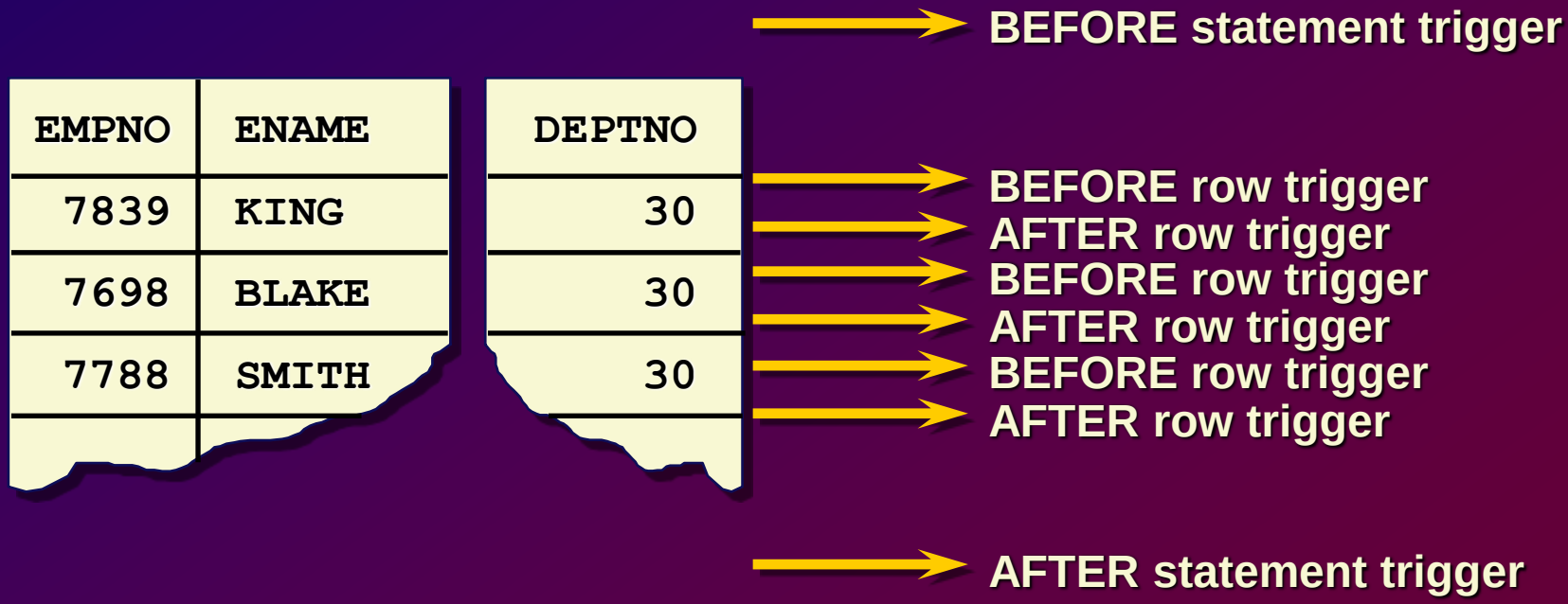
Example 1

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

Example 2

```
SQL> UPDATE emp
  2  SET sal = sal * 1.1
  3  WHERE deptno = 30;
```

Firing Sequence of Database Triggers on Multiple Rows



Syntax for Creating Statement Triggers

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing event1 [OR event2 OR event3]
ON table_name
PL/SQL block;
```

Before Statement Trigger: Example

```
SQL> CREATE OR REPLACE TRIGGER secure_emp
  2  BEFORE INSERT ON emp
  3  BEGIN
  4   IF (TO_CHAR (sysdate,'DY') IN ('SAT','SUN'))
  5     OR (TO_CHAR(sysdate,'HH24')NOT BETWEEN
  6    '08' AND '18'
  7     THEN RAISE_APPLICATION_ERROR (-20500,
  8    'You may only insert into EMP during normal
  9     hours.');
 10   END IF;
 11  END;
 12  /
```

Example

```
SQL> INSERT INTO emp (empno, ename, deptno)
  2  VALUES           (7777, 'BAUWENS', 40);
INSERT INTO emp (empno, ename, deptno)
            *
ERROR at line 1:
ORA-20500: You may only insert into EMP during
normal hours.
ORA-06512: at "SCOTT.SECURE_EMP", line 4
ORA-04088: error during execution of trigger
'SCOTT.SECURE_EMP'
```

Using Conditional Predicates

```
SQL>CREATE OR REPLACE TRIGGER secure_emp
  2 BEFORE INSERT OR UPDATE OR DELETE ON emp
  3 BEGIN
  4  IF (TO_CHAR (sysdate,'DY') IN ('SAT','SUN')) OR
  5  (TO_CHAR (sysdate, 'HH24') NOT BETWEEN '08' AND '18') THEN
  6   IF DELETING THEN
  7     RAISE_APPLICATION_ERROR (-20502,
  8     'You may only delete from EMP during normal hours.');
  9     ELSIF INSERTING THEN
 10      RAISE_APPLICATION_ERROR (-20500,
 11      'You may only insert into EMP during normal hours.');
 12     ELSIF UPDATING ('SAL') THEN
 13      RAISE_APPLICATION_ERROR (-20503,
 14      'You may only update SAL during normal hours.');
 15     ELSE
 16       RAISE_APPLICATION_ERROR (-20504,
 17       'You may only update EMP during normal hours.');
 18    END IF;
 19  END IF;
 20 END;
 21 /
```

After Statement Trigger: Example

```
SQL>CREATE OR REPLACE TRIGGER check_salary_count
  2 AFTER UPDATE OF sal ON emp
  3 DECLARE
  4   v_salary_changes NUMBER;
  5   v_max_changes    NUMBER;
  6 BEGIN
  7   SELECT upd, max_upd
  8   INTO   v_salary_changes, v_max_changes
  9   FROM   audit_table
 10   WHERE  user_name = user
 11   AND    table_name = 'EMP'
 12   AND    column_name = 'SAL';
 13   IF v_salary_changes > v_max_changes THEN
 14     RAISE_APPLICATION_ERROR (-20501,
 15     'You may only make a maximum of '||
 16     TO_CHAR (v_max_changes) ||
 17     ' changes to the SAL column');
 18   END IF;
 19 END;
 20 /
```

User Audit Table

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

Continuation

MAX_INS	MAX_UPD	MAX_DEL
5	5	5
	5	
5	0	0

Creating a Row Trigger

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing event1 [OR event2 OR event3]
ON table_name
[REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW
[WHEN condition]
PL/SQL block;
```

After Row Trigger: Example

```
SQL>CREATE OR REPLACE TRIGGER audit_emp
  2 AFTER DELETE OR INSERT OR UPDATE ON emp
  3 FOR EACH ROW
  4 BEGIN
  5  IF DELETING THEN
  6       UPDATE audit_table SET del = del + 1
  7       WHERE  user_name = user AND table_name = 'EMP'
  8       AND    column_name IS NULL;
  9  ELSIF INSERTING THEN
 10       UPDATE audit_table SET ins = ins + 1
 11       WHERE  user_name = user AND table_name = 'EMP'
 12       AND    column_name IS NULL;
 13  ELSIF UPDATING ('SAL') THEN
 14       UPDATE audit_table SET upd = upd + 1
 15       WHERE  user_name = user AND table_name = 'EMP'
 16       AND    column_name = 'SAL';
 17  ELSE /* The data manipulation operation is a general UPDATE. */
 18       UPDATE audit_table SET upd = upd + 1
 19       WHERE  user_name = user AND table_name = 'EMP'
 20       AND    column_name IS NULL;
 21  END IF;
 22 END;
 23 /
```

Using Old and New Qualifiers

```
SQL>CREATE OR REPLACE TRIGGER audit_emp_values
  2 AFTER DELETE OR INSERT OR UPDATE ON emp
  3 FOR EACH ROW
  4 BEGIN
  5   INSERT INTO audit_emp_values (user_name,
  6    timestamp, id, old_last_name, new_last_name,
  7    old_title, new_title, old_salary, new_salary)
  8   VALUES (USER, SYSDATE, :old.empno, :old.ename,
  9    :new.ename, :old.job, :new.job,
 10    :old.sal, :new.sal);
 11 END;
 12 /
```

User Audit_Emp_Values Table

USER_NAME	TIMESTAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME
EGRAVINA	12-NOV-97	7950	NULL	HUTTON
NGREENBE	10-DEC-97	7844	MAGEE	TURNER

Continuation

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

Restricting a Row Trigger

```
SQL>CREATE OR REPLACE TRIGGER derive_commission_pct
  2 BEFORE INSERT OR UPDATE OF sal ON emp
  3 FOR EACH ROW
  4 WHEN (new.job = 'SALESMAN')
  5 BEGIN
  6  IF INSERTING THEN  :new.comm := 0;
  7   ELSE          /* UPDATE of salary */
  8    IF :old.comm IS NULL THEN
  9      :new.comm :=0;
 10    ELSE
 11      :new.comm := :old.comm * (:new.sal/:old.sal);
 12    END IF;
 13  END IF;
 14 END;
 15 /
```

Differentiating Between Triggers and Stored Procedures

Triggers	Procedure
Use CREATE TRIGGER	Use CREATE PROCEDURE
Data dictionary contains source and p-code	Data dictionary contains source and p-code
Implicitly invoked	Explicitly invoked
COMMIT, SAVEPOINT, ROLLBACK not allowed	COMMIT, SAVEPOINT, ROLLBACK allowed

Managing Triggers

Disable or Re-enable a database trigger

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

Disable or Re-enable all triggers for a table

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

Recompile a trigger for a table

```
ALTER TRIGGER trigger_name COMPILE
```

Removing Triggers

To remove a trigger from the database, use the DROP TRIGGER syntax:

```
DROP TRIGGER trigger_name
```

Trigger Test Cases

- Test each of the triggering data operations, as well as non-triggering data operations.
- Test each case of the WHEN clause.
- Cause the trigger to fire directly from a basic data operation, as well as indirectly from a procedure.
- Test the effect of the trigger upon other triggers.
- Test the effect of other triggers upon the trigger.

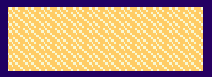
Trigger Execution Model and Constraint Checking

1. Execute all BEFORE STATEMENT triggers
2. Loop for each row affected
   a. Execute all BEFORE ROW triggers
   b. Execute the DML statement and perform integrity constraint checking
   c. Execute all AFTER ROW triggers
3. Complete integrity constraint checking
4. Execute all AFTER STATEMENT triggers

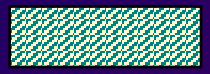
Rules Governing Triggers

- **Rule 1: Do *not* change data in the primary key, foreign key, or unique key columns of a constraining table.**
- **Rule 2: Do *not* read data from a mutating table.**

Changing Data in a Constraining Table



Triggering event



Trigger action

```
SQL> UPDATE dept
  2    SET deptno = 1
  3    WHERE deptno = 30;
```

EMP table

EMPNO	ENAME	DEPTNO
7698	BLAKE	30
7654	MARTIN	30
7499	ALLEN	30

Referential integrity

DEPT table

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

Constraining table

Failure

AFTER UPDATE row

Triggered table

xxxxxxxxxxxxxxxxxxxxxxxxxxxx
vvvvvvvvvvvvvvvvvvvvvvvvvvvv
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
vvvvvvvvvvvvvvvvvvvvvvvvvvvv
xxxxxxxxxxxxxxxxxxxxxxxxxxxx

CASCADE_UPDATES trigger

Constraining Table: Example

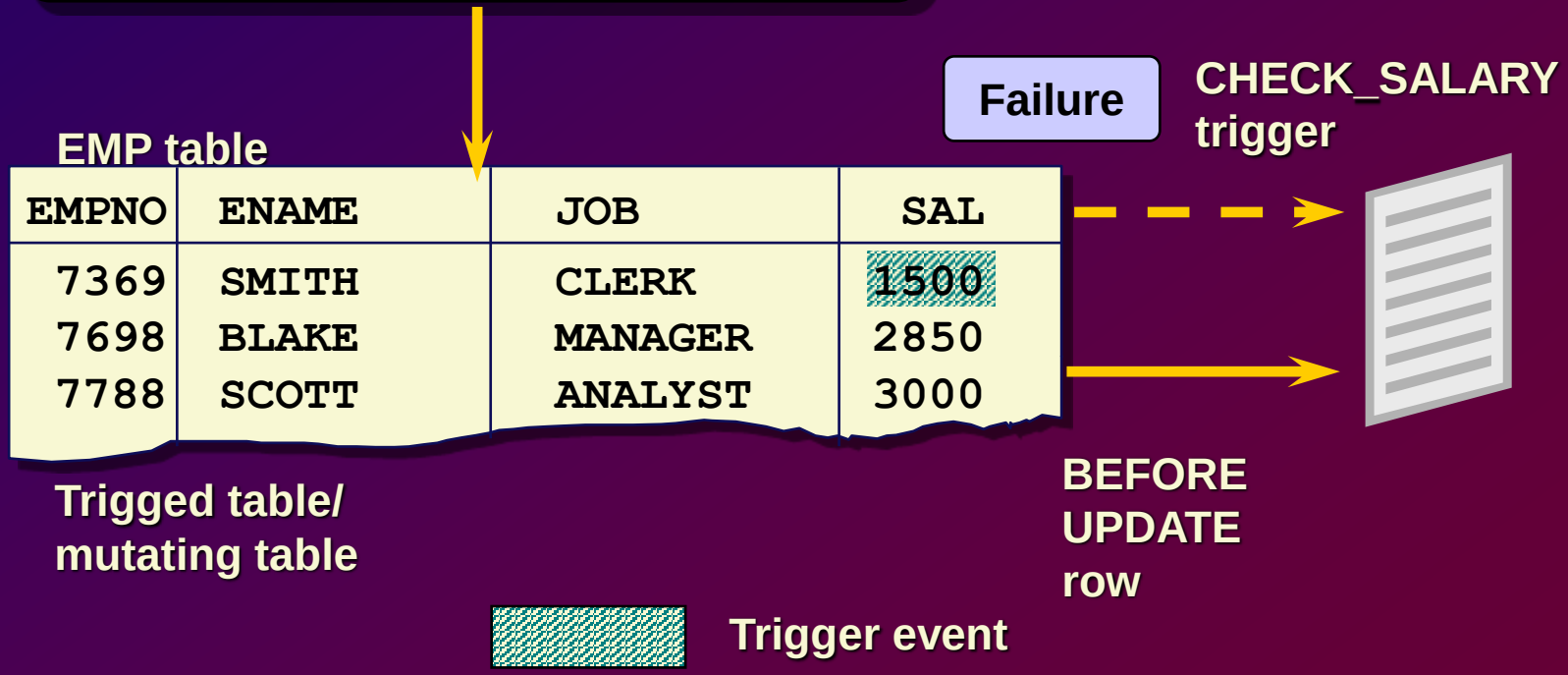
```
SQL>CREATE OR REPLACE TRIGGER cascade_updates
  2 AFTER UPDATE OF deptno on DEPT
  3 FOR EACH ROW
  4 BEGIN
  5       UPDATE emp
  6       SET    emp.deptno = :new.deptno
  7       WHERE  emp.deptno = :old.deptno;
  8 END;
  9 /
```

Constraining Table: Example

```
SQL> UPDATE dept
  2  SET    deptno = 1
  3  WHERE  deptno = 30;
*
ERROR at line 1:
ORA-04091: table DEPT is mutating, trigger/function
 may not see it
```

Reading Data from a Mutating Table

```
SQL> UPDATE emp
  2  SET sal = 1500
  3  WHERE ename = 'SMITH';
```



Mutating Table: Example

```
SQL>CREATE OR REPLACE TRIGGER check_salary
  2 BEFORE INSERT OR UPDATE OF sal, job ON emp
  3 FOR EACH ROW
  4 WHEN (new.job <> 'PRESIDENT')
  5 DECLARE
  6       v_minsalary emp.sal%TYPE;
  7       v_maxsalary emp.sal%TYPE;
```

Mutating Table: Example

```
 8 BEGIN
 9       SELECT MIN(sal), MAX(sal)
10       INTO   v_minsalary, v_maxsalary
11       FROM   emp
12       WHERE  job = :new.job;
13       IF :new.sal < v_minsalary OR
14          :new.sal > v_maxsalary THEN
15           RAISE_APPLICATION_ERROR(-20505,
16                                   'Out of range');
17       END IF;
18 END;
19 /
```

Mutating Table: Example

```
SQL> UPDATE emp
  2  SET sal = 1500
  3  WHERE ename = 'SMITH';
*
ERROR at line 2
ORA_4091 : Table EMP is mutating, trigger/function
may not see it
ORA_06512: at line 4
ORA_04088: error during execution of trigger
'check_salary'
```

Implementation of Triggers

- Security
- Auditing
- Data integrity
- Referential integrity
- Table replication
- Derived data
- Event logging

Controlling Security Within the Server

```
SQL> GRANT SELECT, INSERT, UPDATE, DELETE
  2  ON     emp
  3  TO     CLERK;    -- database role
SQL> GRANT CLERK TO SCOTT;
```

Controlling Security with a Database Trigger

```
SQL>CREATE OR REPLACE TRIGGER secure_emp
  2 BEFORE INSERT OR UPDATE OR DELETE ON emp
  3 DECLARE
  4   v_dummy VARCHAR2(1);
  5 BEGIN
  6   IF TO_CHAR (sysdate, 'DY' IN ('SAT','SUN'))
  7   THEN RAISE_APPLICATION_ERROR (-20506,
  8     'You may only change data during normal business
  9      hours.');
 10   END IF;
 11   SELECT COUNT(*) INTO v_dummy FROM holiday
 12   WHERE holiday_date = TRUNC (sysdate);
 13   IF v_dummy > 0 THEN RAISE_APPLICATION_ERROR (-20507,
 14     'You may not change data on a holiday.');
 15   END IF;
 16 END;
 17 /
```

Auditing Using the Server Facility

```
SQL> AUDIT INSERT, UPDATE, DELETE
  2  ON     emp
  3  BY ACCESS
  4  WHENEVER SUCCESSFUL;
```

Auditing Using a Trigger

```
SQL>CREATE OR REPLACE TRIGGER audit_emp_values
  2 AFTER DELETE OR INSERT OR UPDATE ON emp
  3 FOR EACH ROW
  4 BEGIN
  5   IF audit_emp_package.g_reason IS NULL THEN
  6     RAISE_APPLICATION_ERROR (-20059, 'Specify a reason
  7     for the data operation with the procedure
  8     SET_REASON before proceeding.');
  9   ELSE
 10     INSERT INTO audit_emp_values (user_name, timestamp, id,
 11      old_last_name, new_last_name, old_title, new_title,
 12      old_salary, new_salary, comments)
 13     VALUES (user, sysdate, :old.empno,:old.ename,
 14      :new.ename, :old.job, :new.job, :old.sal,
 15      :new.sal, :audit_emp_package.g_reason);
 16  END IF;
 17 END;
 18 /
```

```
SQL>CREATE TRIGGER cleanup_audit_emp
  2 AFTER INSERT OR UPDATE OR DELETE ON emp
  3 BEGIN
  4   audit_emp_package.g_reason := NULL;
  5 END;
  6 /
```

Enforce Data Integrity Within the Server

```
SQL> ALTER TABLE emp ADD
  2  CONSTRAINT ck_salary CHECK (sal >= 500);
```

Protect Data Integrity with a Trigger

```
SQL>CREATE OR REPLACE TRIGGER check_salary
  2 BEFORE UPDATE OF sal ON emp
  3 FOR EACH ROW
  4 WHEN (new.sal < old.sal) OR
  5      (new.sal > old.sal * 1.1)
  6 BEGIN
  7   RAISE_APPLICATION_ERROR (-20508,
  8   'Do not decrease salary nor increase by
  9    more than 10%.');
 10 END;
 11 /
```

Enforce Referential Integrity Within the Server

```
SQL> ALTER TABLE emp
  2  ADD CONSTRAINT emp_deptno_fk
  3  FOREIGN KEY (deptno) REFERENCES dept(deptno)
  4  ON DELETE CASCADE;
```

Protect Referential Integrity with a Trigger

```
SQL>CREATE OR REPLACE TRIGGER cascade_updates
  2 AFTER UPDATE OF deptno ON dept
  3 FOR EACH ROW
  4 BEGIN
  5   UPDATE emp
  6   SET    emp.deptno = :new.deptno
  7   WHERE  emp.deptno = :old.deptno;
  8 END;
  9 /
```

Replicate a Table Within the Server

```
SQL> CREATE SNAPSHOT emp_copy AS
  2  SELECT * FROM emp@ny;
```

Replicate a Table with a Trigger

```
SQL>CREATE OR REPLACE TRIGGER emp_replica
  2 BEFORE INSERT OR UPDATE ON emp
  3 FOR EACH ROW
  4 BEGIN /*Only proceed if user init. data operation,
  5          NOT the casc. trigger.*/
  6   IF INSERTING THEN
  7      IF :new.flag IS NULL THEN
  8        INSERT INTO emp@sf VALUES (:new.empno,
  9        :new.ename,...,'B');
 10        :new.flag = 'A';
 11      ELSE   /* Updating. */
 12      IF :new.flag = :old.flag THEN
 13        UPDATE emp@sf SET ename = :new.ename, ...,
 14                          FLAG = :new.flag
 15        WHERE empno = :new.empno;
 16      END IF;
 17     IF :old.flag = 'A' THEN :new.flag := 'B';
 18     ELSE :new.flag := 'A';
 19     END IF;
 20  END IF;
 21 END;
 22 /
```

Compute Derived Data Within the Server

```
SQL> UPDATE dept
  2> SET total_sal = (SELECT SUM(salary)
  3>                  FROM emp
  4>                  WHERE emp.deptno = dept.deptno);
```

Compute Derived Values with a Trigger

```
SQL>CREATE OR REPLACE PROCEDURE increment_salary
  2  (v_id      IN dept.deptno%TYPE,
  3   v_salary IN dept.total_salary%TYPE)
  4 IS
  5 BEGIN
  6     UPDATE dept
  7     SET    total_sal = NVL (total_sal,0)+ v_salary
  8     WHERE  deptno = v_id;
  9 END increment_salary;
 10 /
```

```
SQL>CREATE OR REPLACE TRIGGER compute_salary
  2 AFTER INSERT OR UPDATE OF sal OR DELETE ON emp
  3 FOR EACH ROW
  4 BEGIN
  5 IF DELETING THEN increment_salary(:old.deptno, -1 * :old.sal);
  6 ELSIF UPDATING THEN increment_salary(:new.deptno,
  7                                      :new.sal-:old.sal);
  8 ELSE /*inserting*/ increment_salary(:new.deptno, :new.sal);
  9 END IF;
 10 END;
 11 /
```

Log Events with a Trigger

```
SQL>CREATE OR REPLACE TRIGGER notify_reorder_rep
  2 AFTER UPDATE OF amount_in_stock, reorder_point ON inventory
  3 FOR EACH ROW
  4 WHEN new.amount_in_stock <= new.reorder_point
  5 DECLARE
  6      v_descrip         product.descrip%TYPE;
  7      v_msg_text        VARCHAR2(2000);
  8 BEGIN
  9 SELECT descrip INTO  v_descrip
 10 FROM   PRODUCT WHERE prodid = :new.product_id;
 11  v_msg_text := 'It has come to my personal attention that,
 12         due to recent '
 13          CHR(10) || 'transactions, our inventory for product # '||
 14         TO_CHAR(:new.product_id)||'--'
 15        || v_name ||'-- has fallen' || CHR(10) || CHR(10) ||
 16        'Yours,' ||CHR(10) ||user || '.';
 17       dbms_mail.send ('Inventory', user,null,null,'Low
 18   Inventory',null,v_msg_text);
 19 END;
 20 /
```

Benefits of Database Triggers

Improved data security

- Provide value-based security checks
- Provide value-based auditing

Improved data integrity

- Enforce dynamic data integrity constraints
- Enforce complex referential integrity constraints
- Ensure related operations are performed together implicitly