



SQL

Seema Sirpal

Delhi University Computer Centre



Views

- In SQL, a VIEW is a virtual table based on the In SQL, result-set of a SELECT statement.
- Views are database objects which contain no data of its own.
- The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from a single table.
- User cannot distinguish between a table and a view.
- Views restricts access to the database.

Views

Syntax:

```
CREATE VIEW view_name  
AS SELECT column_name(s) FROM ) FROM table_name  
WHERE condition
```

Example:

```
CREATE VIEW dept20 AS SELECT * FROM emp WHERE deptno=20;  
  
SELECT * FROM dept20;
```

Views

Example:

```
CREATE OR REPLACE VIEW empview  
(eno, dno, ename,sal,dname)  
AS SELECT empno, emp.deptno, ename, sal, dname  
FROM emp, dept  
WHERE emp.deptno = dept.deptno;
```

Views

Manipulating the Base Table thru View is possible if:

- There are no aggregate functions
- No DISTINCT is used
- No GROUP BY or HAVING

Example : (Read only View)

```
CREATE VIEW empview (empno, tot) AS  
SELECT empno, count(*) FROM incr GROUP BY empno;
```

Views

With Check Options

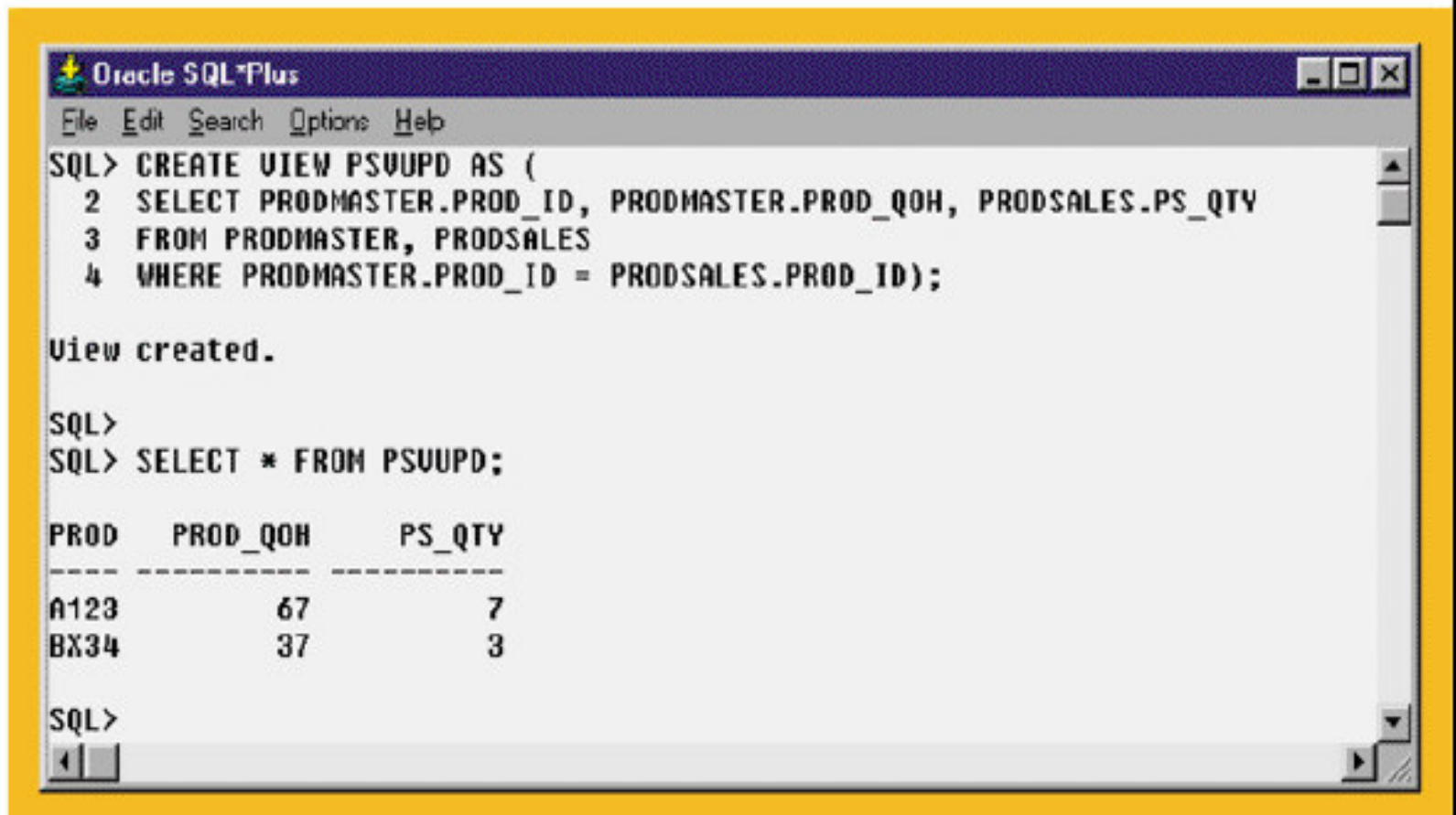
```
CREATE VIEW empdept AS  
SELECT empno, ename, deptno FROM emp WHERE deptno=30  
WITH CHECK OPTION;
```

It will not allow to insert the following row:

```
INSERT INTO empdept VALUES (1234, 'JAMILA', 20)
```

Updatable Views

FIGURE 7.26 CREATING AN UPDATABLE VIEW IN ORACLE



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE VIEW PSUUPD AS (
  2 SELECT PRODMASTER.PROD_ID, PRODMASTER.PROD_QOH, PRODSALES.PS_QTY
  3 FROM PRODMASTER, PRODSALES
  4 WHERE PRODMASTER.PROD_ID = PRODSALES.PROD_ID);

View created.

SQL>
SQL> SELECT * FROM PSUUPD;

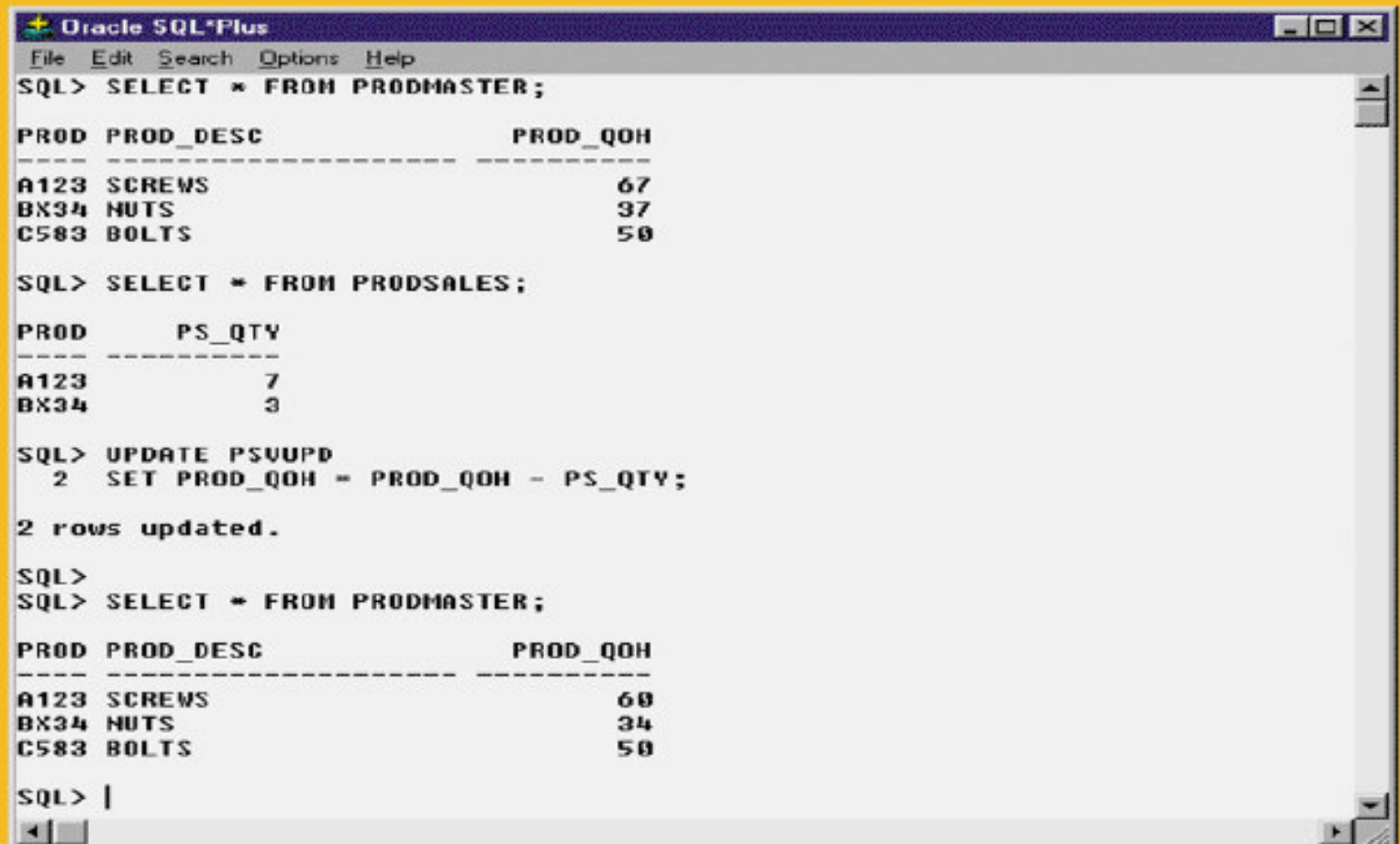
PROD   PROD_QOH   PS_QTY
-----
A123           67         7
BX34           37         3

SQL>
```

PROD	PROD_QOH	PS_QTY
A123	67	7
BX34	37	3

Updatable Views

FIGURE 7.27 PRODMASTER TABLE UPDATE, USING AN UPDATABLE VIEW



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT * FROM PRODMASTER;

PROD PROD_DESC          PROD_QOH
-----
A123  SCREWS                 67
BX34  NUTS                   37
C583  BOLTS                  50

SQL> SELECT * FROM PRODSALES;

PROD    PS_QTY
-----
A123         7
BX34         3

SQL> UPDATE PSUUPD
      2 SET PROD_QOH = PROD_QOH - PS_QTY;

2 rows updated.

SQL>
SQL> SELECT * FROM PRODMASTER;

PROD PROD_DESC          PROD_QOH
-----
A123  SCREWS                 60
BX34  NUTS                   34
C583  BOLTS                  50

SQL> |
```


GRANT

Granting Privileges on Columns

INSERT, UPDATE or REFERENCES privileges can be granted on individual columns

Example :

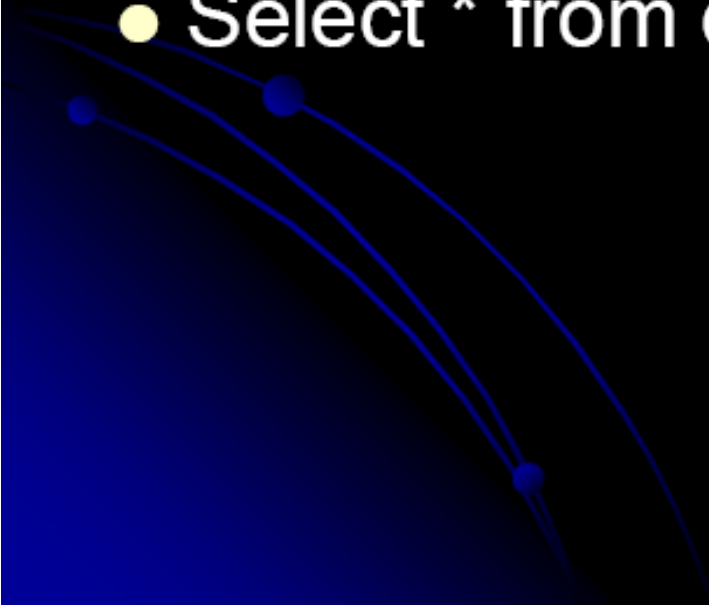
```
GRANT UPDATE (acct_no) ON accts TO user1;
```

Data dictionary

Contains info about:

- users and their privileges,
- tables, table columns and their data types, integrity constraints, indexes,
- statistics about tables and indexes used by the optimizer,
- privileges granted on database objects,
- storage structures of the database.

Data Dictionary

- **select * from DICT** – selects all tables / views of data dictionary accessible to the user
 - **Select * from tab - ??**
 - **Select * from col - ??**
- 

Standard Views

- The views provided by the data dictionary are divided into three groups: USER, ALL, and DBA.
- The group name builds the prefix for each view name.

● These are explained below:



Standard Views

- **USER** : Tuples in the USER views contain information about objects owned by the account performing the SQL query (current user)
- **USER_TABLES** all tables with their name, number of columns, storage information, statistical information etc. (TABS)
- **USER_CATALOG** tables, views, and synonyms (CAT)
- **USER_COL_COMMENTS** comments on columns
- **USER_CONSTRAINTS** constraint definitions for tables
- **USER_INDEXES** all information about indexes created for tables (IND)
- **USER_OBJECTS** all database objects owned by the user (OBJ)
- **USER_TAB_COLUMNS** columns of the tables and views owned by the user (COLS)
- **USER_TAB_COMMENTS** comments on tables and views
- **USER_TRIGGERS** triggers defined by the user
- **USER_USERS** information about the current user
- **USER_VIEWS** views defined by the user

Standard Views

- ALL : Rows in the ALL views include rows of the USER views and all information about objects that are accessible to the current user. The structure of these views is analogous to the structure of the USER views.
- ALL_CATALOG owner, name and type of all accessible tables, views, and synonyms
- ALL_TABLES owner and name of all accessible tables
- ALL_OBJECTS owner, type, and name of accessible database objects
- ALL_TRIGGERS
- ALL_USERS
- ALL_VIEWS

Standard Views

- DBA : The DBA views encompass information about all database objects, regardless of the owner. Only users with DBA privileges can access these views.
- DBA_TABLES tables of all users in the database
- DBA_CATALOG tables, views, and synonyms defined in the database
- DBA_OBJECTS object of all users
- DBA_DATA_FILES information about data files
- DBA_USERS information about all users known in the database

ROWID

Every tuple in a database has a pseudo-column ROWID that is used to identify tuples.

Example:

To select a particular row (2nd) –

```
Select * from emp a where 2=(select count(rowid) from emp b where a.rowid>=b.rowid);
```

To select a range of rows (91 to 100) –

```
Select * from (select ename,rownum rn from emp where rownum<101) where rn between 91 and 100;
```


Rowid

- Every tuple in a database has a pseudo-column ROWID that is used to identify tuples.

Example: Assume we want to add an integrity constraint to our table EMP which requires

that each manager must earn more than 4000:

```
alter table EMP add constraint manager_sal  
check(JOB = 'MANAGER' or SAL >= 4000)  
exceptions into EXCEPTIONS;
```

If the table EMP already contains tuples that violate the constraint, the constraint cannot be activated and information about violating tuples is automatically inserted into the table EXCEPTIONS.

Finding violating tuples

- Detailed information about the violating tuples can be obtained by joining the tables EMP and EXCEPTIONS, based on the join attribute ROWID:


```
select EMP.*, CONSTRAINT from EMP,  
EXCEPTIONS where EMP.ROWID =  
EXCEPTIONS.ROW ID;
```

- Before this table can be used, it must be created using the SQL script utlexcept.sql which can be found in the directory \$ORACLE_HOME/rdbms/admin.

Dropping tables with enabled constraints

- If a table is used as a reference of a foreign key, this table can only be dropped using the command **drop table <table> cascade constraints;**
- Information about integrity constraints, their status (enabled, disabled) etc. is stored in the data dictionary, more precisely, in the tables **USER_CONSTRAINTS** and **USER_CONS_CONSTRAINTS**.

Procedures

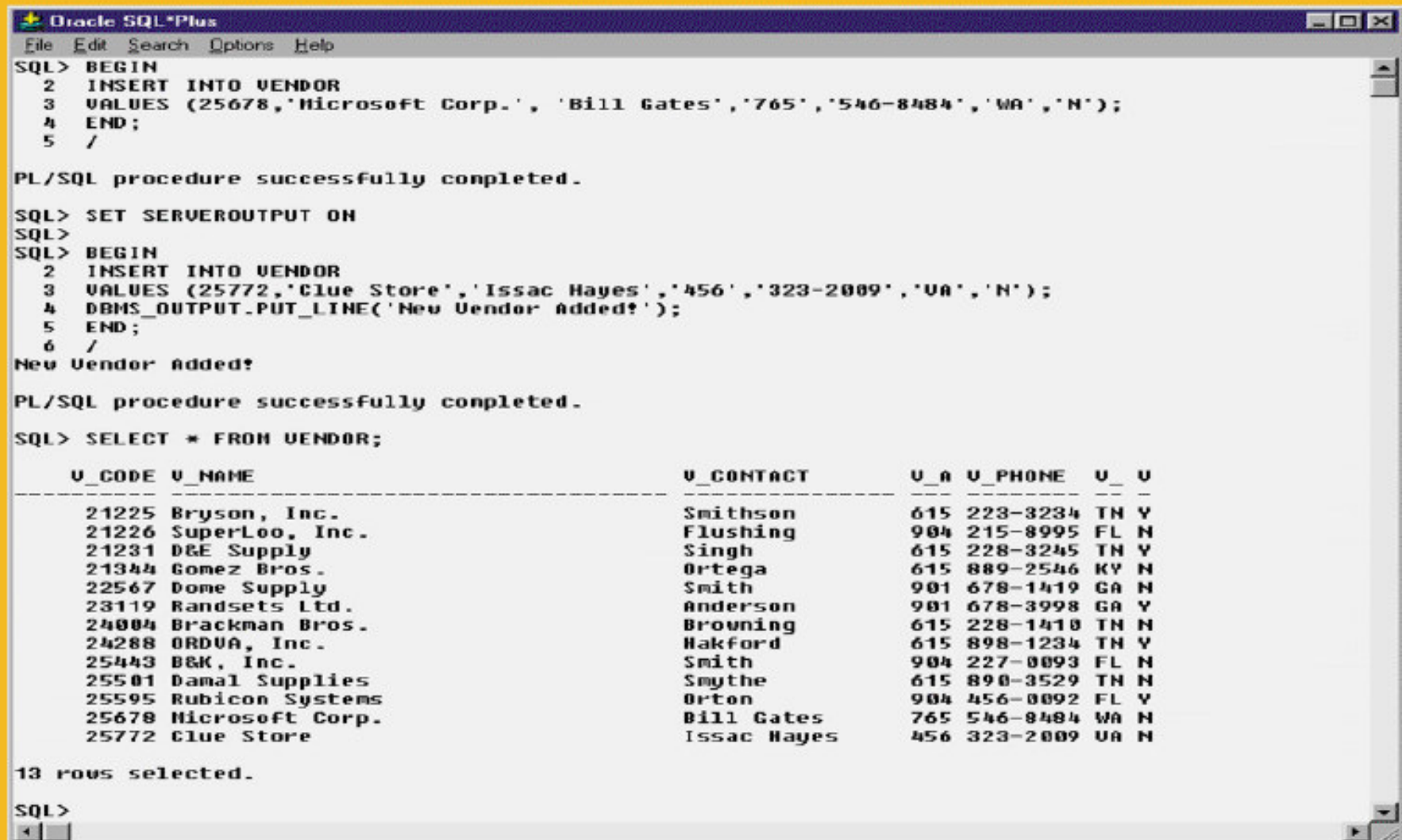
- Extension to standard SQL
 - Basic construct – Block
 - Can declare constants and variables
 - Variables can store query results
 - Can process query results one row at a time using **Cursors**
- 

Structure of a procedure

```
[<Block header>]
[declare
<Constants>
<Variables>
<Cursors>
<User defined exceptions>]
begin
<PL/SQL statements>
[exception
<Exception handling>]
end;
```


Example

FIGURE 7.28 ANONYMOUS PL/SQL BLOCK EXAMPLES



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> BEGIN
  2 INSERT INTO VENDOR
  3 VALUES (25678,'Microsoft Corp.', 'Bill Gates','765','546-8484','WA','N');
  4 END;
  5 /

PL/SQL procedure successfully completed.

SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2 INSERT INTO VENDOR
  3 VALUES (25772,'Clue Store','Issac Hayes','456','323-2009','UA','N');
  4 DBMS_OUTPUT.PUT_LINE('New Vendor Added!');
  5 END;
  6 /
New Vendor Added!

PL/SQL procedure successfully completed.

SQL> SELECT * FROM VENDOR;

  U_CODE U_NAME                                U_CONTACT          U_A U_PHONE  U_  U
-----
  21225 Bryson, Inc.                            Smithson           615 223-3234 TN Y
  21226 SuperLoe, Inc.                          Flushing          904 215-8995 FL N
  21231 D&E Supply                               Singh              615 228-3245 TN Y
  21344 Gomez Bros.                             Ortega             615 889-2546 KY N
  22567 Dome Supply                             Smith              901 678-1419 GA N
  23119 Randsets Ltd.                           Anderson           901 678-3998 GA Y
  24004 Brackman Bros.                           Browning           615 228-1410 TN N
  24288 ORDUA, Inc.                             Hakford            615 898-1234 TN Y
  25443 B&K, Inc.                               Smith              904 227-0093 FL N
  25501 Damal Supplies                          Smythe             615 890-3529 TN N
  25595 Rubicon Systems                         Orton              904 456-0092 FL Y
  25678 Microsoft Corp.                         Bill Gates         765 546-8484 WA N
  25772 Clue Store                              Issac Hayes        456 323-2009 UA N

13 rows selected.

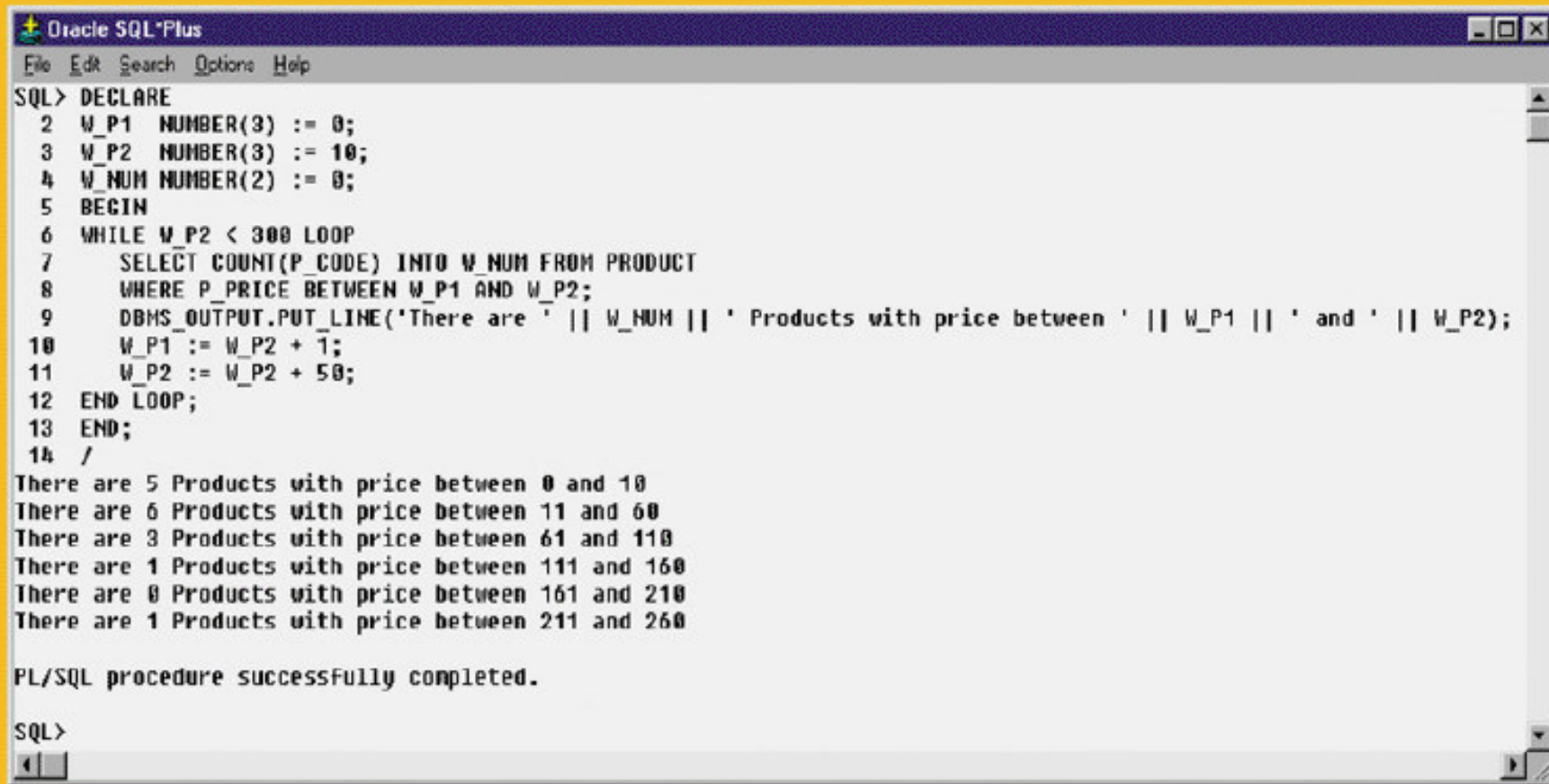
SQL>
```

Show Errors

- Can help diagnose errors found in PL/SQL blocks
- Yields additional debugging information whenever an error is generated after an PL/SQL block is created or executed

Using variables

FIGURE 7.29 ANONYMOUS PL/SQL BLOCK WITH VARIABLES AND LOOPS



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> DECLARE
2  W_P1  NUMBER(3) := 0;
3  W_P2  NUMBER(3) := 10;
4  W_NUM NUMBER(2) := 0;
5  BEGIN
6  WHILE W_P2 < 300 LOOP
7      SELECT COUNT(P_CODE) INTO W_NUM FROM PRODUCT
8      WHERE P_PRICE BETWEEN W_P1 AND W_P2;
9      DBMS_OUTPUT.PUT_LINE('There are ' || W_NUM || ' Products with price between ' || W_P1 || ' and ' || W_P2);
10     W_P1 := W_P2 + 1;
11     W_P2 := W_P2 + 50;
12 END LOOP;
13 END;
14 /
There are 5 Products with price between 0 and 10
There are 6 Products with price between 11 and 60
There are 3 Products with price between 61 and 110
There are 1 Products with price between 111 and 160
There are 0 Products with price between 161 and 210
There are 1 Products with price between 211 and 260

PL/SQL procedure successfully completed.

SQL>
```


Cursors

cursor <cursor name> [(<list of parameters>)] is <select statement>;

Example: We want to retrieve the following attribute values from the table EMP in a tuple oriented way: the job title and name of those employees who have been hired after a given date, and who have a manager working in a given department.

cursor employee_cur (start_date date, dno number) is
select JOB, ENAME **from** EMP E **where** HIREDATE > start_date **and exists** (**select** * **from** EMP **where** E.MGR = EMPNO **and** DEPTNO = dno);

Triggers

- *Triggers* provide a procedural technique to specify and maintain integrity constraints.
- Essentially a PL/SQL procedure
- Such a procedure is associated with a table and is automatically called by the database system whenever a certain modification (*event*) occurs on that table.
- Modifications on a table may include **insert**, **update**, and **delete** operations

Trigger Definitions

A trigger definition consists of the following (optional) components:

- *trigger name*

create [or replace] trigger <trigger name>

- *trigger time point*

before | after

- *triggering event(s)*

insert or update [of <column(s)>] or delete on <table>

- *trigger type* (optional)

for each row

- *trigger restriction* (only for for each row triggers !)

when (<condition>)

- *trigger body*

<PL/SQL block>

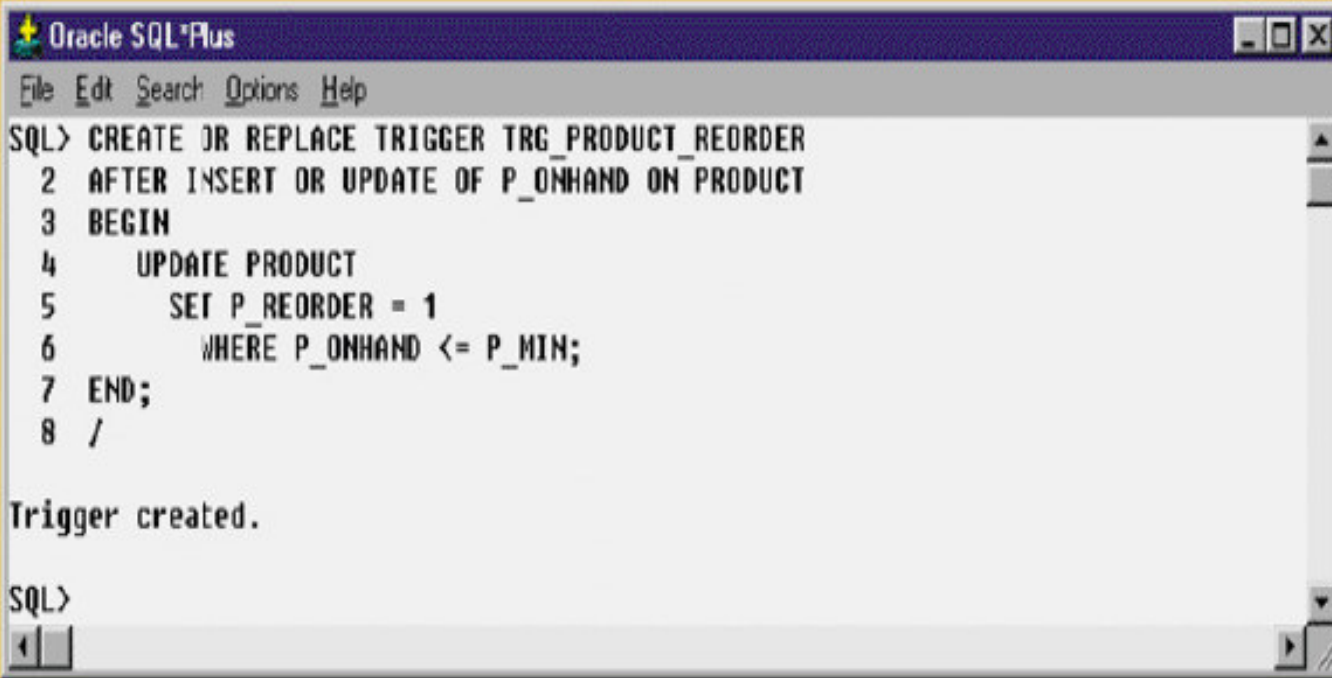
Row and Statement triggers

- A row trigger executes once for each row after (before) the event.
- A statement trigger is executed once after (before) the event, independent of how many rows are affected by the event
- 12 possible combinations

event	trigger time point		trigger type	
	before	after	statement	row
insert	X	X	X	X
update	X	X	X	X
delete	X	X	X	X

Example

FIGURE 7.31 CREATING THE TRG_PRODUCT_REORDER TRIGGER



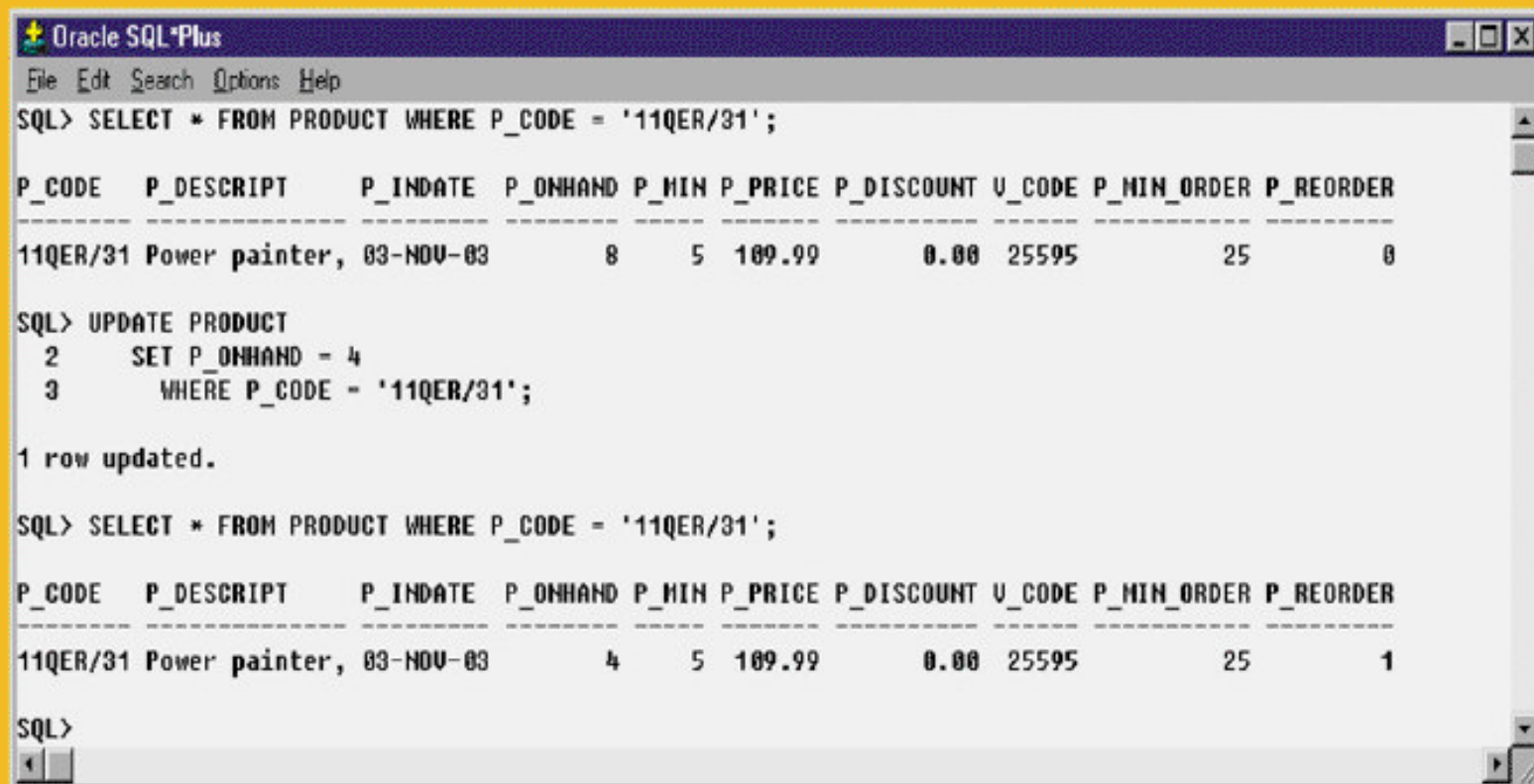
```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE OR REPLACE TRIGGER TRG_PRODUCT_REORDER
2 AFTER INSERT OR UPDATE OF P_ONHAND ON PRODUCT
3 BEGIN
4   UPDATE PRODUCT
5     SET P_REORDER = 1
6     WHERE P_ONHAND <= P_MIN;
7 END;
8 /

Trigger created.

SQL>
```

Example cont.

FIGURE 7.32 VERIFYING THE TRG_PRODUCT_REORDER TRIGGER EXECUTION



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT * FROM PRODUCT WHERE P_CODE = '11QER/31';

P_CODE  P_DESCRIPT  P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT U_CODE P_MIN_ORDER P_REORDER
-----
11QER/31 Power painter, 03-NOV-03      8     5 109.99      0.00 25595          25         0

SQL> UPDATE PRODUCT
  2   SET P_ONHAND = 4
  3   WHERE P_CODE = '11QER/31';

1 row updated.

SQL> SELECT * FROM PRODUCT WHERE P_CODE = '11QER/31';

P_CODE  P_DESCRIPT  P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT U_CODE P_MIN_ORDER P_REORDER
-----
11QER/31 Power painter, 03-NOV-03      4     5 109.99      0.00 25595          25         1

SQL>
```


Example

```
create or replace trigger emp_check
after insert or delete or update on EMP
for each row
begin
if inserting then
<PL/SQL block>
end if ;
if updating then
<PL/SQL block>
end if ;
if deleting then
<PL/SQL block>
end if ;
end;
```

Advantages of PL/SQL

- Substantially reduce network traffic and increase performance
- No transmission of individual SQL statements over network
- Help reduce code duplication by means of code isolation and code sharing
- Minimize chance of errors and cost of application development and maintenance

EXAMPLE

To create a trigger for the emp table, which makes the entry in ename column in UPPERCASE

```
CREATE OR REPLACE TRIGGER upper_trig
BEFORE INSERT OR UPDATE OF ename ON emp
FOR EACH ROW
BEGIN
:new.ename := UPPER (:new.ename);
END;
```

:new – keyword refer to new value of the column

:old – keyword refer to old value of the column



EXAMPLE



Write a trigger `total_salary` to maintain a derived column `totsal` that stores total salary of all the members in a department.



DROPPING A TRIGGER



```
DROP TRIGGER t1;
```