

File Edit View History Bookmarks Tools Help SMART Ink

More on Oracle keys

www.pgrocer.net/Cis50/morekeys.html

To test this, I am now going to try to add a row to inven with a dept code that does not appear in the department table. The row is rejected because of a violation in the foreign key link to the department table. Specifically the parent was not found in the foreign key table.

**SQL CODE:**

```
SQL> INSERT INTO inven
  2 VALUES ('4567', 'Nancy Drew', 3,4,5,14,16.99,'BO','BK','X100');
INSERT INTO inven
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.DEPT_INVEN_FK) violated - parent key not found
```

In the next code, I am making the custid the primary key of the table invcust. I will then go on and make the salsrep the primary key of the table salsrep so that I can make salsrep in the invcust a foreign key linked to the salsrep primary key (which is also named salsrep). Remember, if I do not establish the primary key in the table salsrep first, I cannot link a foreign key into it. The name after REFERENCES refers to the table not the column/field and is looking for the primary key in that table. Finally, I will show the constraints that were created in the invcust and salsrep tables.

**SQL CODE:**

```
SQL> ALTER TABLE invcust
  2 ADD CONSTRAINT custid_pk PRIMARY KEY(custid);
Table altered.

SQL> DESC invcust;
Name                               Null?    Type
-----
CUSTID                               NOT NULL VARCHAR2(5)
CUSTNAME                             VARCHAR2(20)
STADR                                 VARCHAR2(15)
APT                                   VARCHAR2(5)
CITY                                  VARCHAR2(15)
STATE                                 CHAR(2)
ZIP                                   VARCHAR2(5)
PASTDUE                              NUMBER(6,2)
CURRDUE                              NUMBER(6,2)
CRLIMIT                              NUMBER(6,2)
DATEFST                              DATE
SLSREP                               VARCHAR2(4)
```

SQL> ALTER TABLE salsrep  
2 ADD CONSTRAINT salsrep\_pk PRIMARY KEY(salsrep);  
Table altered.

```
SQL> DESC salsrep;
Name                               Null?    Type
-----
SALSREP                             NOT NULL VARCHAR2(4)
SALSNAME                             VARCHAR2(20)
COMMRATE                             NUMBER(3,2)
```

```
SQL> ALTER TABLE invcust
  2 ADD CONSTRAINT salsrep_fk FOREIGN KEY(salsrep) REFERENCES salsrep;
```

Altering table by adding primary key and foreign key.

Type here to search

9:35 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on Oracle keys x +

www.pgrocer.net/Cis50/morekeys.html

```
SQL> DESC ORDERZ;
Name                Null?    Type
-----
ORDNO                NOT NULL VARCHAR2(6)
CUSTID              NOT NULL VARCHAR2(5)
ORDDATE              DATE

SQL> ALTER TABLE orderz
  2 ADD CONSTRAINT custid_fk FOREIGN KEY(custid) references invcust;

Table altered.

SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
  2 WHERE TABLE_NAME = 'ORDERZ' OR TABLE_NAME = 'INVCUST';

TABLE_NAME          CONSTRAINT_NAME
-----
INVCUST              CUSTID_FK
INVCUST              SLSREP_FK
ORDERZ               ORDNO_FK
ORDERZ               CUSTID_FK
```

The ordline table has a primary key that is composed of the ordno and the itemno. This is done by listing the first followed by a comma and then the second within the parenthesis after primary key. In addition, both ordno and itemno are foreign keys and I want assurance that the ordno exists on the orderz table and the itemno exists on the inven table.

**SQL CODE:**

```
SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT ordno_itemno_pk PRIMARY KEY(ordno,itemno)
  3 ;

Table altered.

SQL> DESC ordline;
Name                Null?    Type
-----
ORDNO                NOT NULL VARCHAR2(6)
ITEMNO              NOT NULL VARCHAR2(4)
NUMORD              NUMBER(3)

SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
  2 WHERE TABLE_NAME = 'ORDLINE';

TABLE_NAME          CONSTRAINT_NAME
-----
ORDLINE              ORDNO_ITEMNO_PK

SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT ordno_fk FOREIGN KEY(ordno) REFERENCES orderz;

Table altered.

SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT itemno_fk FOREIGN KEY(itemno) REFERENCES inven;

Table altered.
```

Setting up a primary key composed of two fields.

Windows taskbar: Type here to search, 9:36 AM, 10/17/2019

File Edit View History Bookmarks Tools Help

More on Oracle keys

www.pgrocer.net/Cis50/morekeys.html

ORDERZ                    ORDNO\_FK  
ORDERZ                    CUSTID\_FK

The ordline table has a primary key that is composed of the ordno and the itemno. This is done by listing the first followed by a comma and then the second within the parenthesis after primary key. In addition, both ordno and itemno are foreign keys and I want assurance that the ordno exists on the orderz table and the itemno exists on the inven table.

**SQL CODE:**

```
SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT ordno_itemno_pk PRIMARY KEY(ordno,itemno)
  3 ;
```

Table altered.

```
SQL> DESC ordline;
Name                    Null?    Type
-----
ORDNO                    NOT NULL VARCHAR2(6)
ITEMNO                    NOT NULL VARCHAR2(4)
NUMORD                                    NUMBER(3)
```

```
SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
  2 WHERE TABLE_NAME = 'ORDLINE';
```

TABLE_NAME	CONSTRAINT_NAME
ORDLINE	ORDNO_ITEMNO_PK

```
SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT ordno_fk FOREIGN KEY(ordno) REFERENCES orderz;
```

Table altered.

```
SQL> ALTER TABLE ordline
  2 ADD CONSTRAINT itemno_fk FOREIGN KEY(itemno) REFERENCES inven;
```

Table altered.

```
SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
  2 WHERE TABLE_NAME = 'ORDLINE';
```

TABLE_NAME	CONSTRAINT_NAME
ORDLINE	ORDNO_ITEMNO_PK
ORDLINE	ORDNO_FK
ORDLINE	ITEMNO_FK

Orderz

ordno ←

custid

ordline

ordno

itemno →

Inven

itemno

ordno data

cust

---

itemno

---



---

In the following example, I have tried to add an order that is not for a valid customer. The order is rejected. Then I add an order with a valid customer and the order is accepted.

**SQL CODE:**

```
SQL> INSERT INTO orderz
  2 VALUES ('000004','22222',sysdate);
INSERT INTO orderz
```

Windows taskbar: Type here to search, 9:40 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on Oracle keys x +

www.pgrocer.net/Cis50/morekeys.html

**SQL CODE:**

```
SQL> ALTER TABLE ordline
2 ADD CONSTRAINT ordno_itemno_pk PRIMARY KEY(ordno,itemno)
3 ;

Table altered.

SQL> DESC ordline;
Name Null? Type
-----
ORDNO NOT NULL VARCHAR2(6)
ITEMNO NOT NULL VARCHAR2(4)
NUMORD NUMBER(3)

SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
2 WHERE TABLE_NAME = 'ORDLINE';

TABLE_NAME CONSTRAINT_NAME
-----
ORDLINE ORDNO_ITEMNO_PK

SQL> ALTER TABLE ordline
2 ADD CONSTRAINT ordno_fk FOREIGN KEY(ordno) REFERENCES orderz;

Table altered.

SQL> ALTER TABLE ordline
2 ADD CONSTRAINT itemno_fk FOREIGN KEY(itemno) REFERENCES inven;

Table altered.

SQL> SELECT TABLE_NAME, CONSTRAINT_NAME FROM USER_CONSTRAINTS
2 WHERE TABLE_NAME = 'ORDLINE';

TABLE_NAME CONSTRAINT_NAME
-----
ORDLINE ORDNO_ITEMNO_PK
ORDLINE ORDNO_FK
ORDLINE ITEMNO_FK
```

In the following example, I have tried to add an order that is not for a valid customer. The order is rejected. Then I add an order with a valid customer and the order is accepted.

**SQL CODE:**

```
SQL> INSERT INTO orderz
2 VALUES ('000004','22222',sysdate);
INSERT INTO orderz
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.CUSTID_FK) violated - parent key not found

SQL> INSERT INTO orderz
2 VALUES('000004','11111',sysdate);
```

Orderz ordline inven  
ordno ← ordno → itemno  
itemno

Type here to search 9:45 AM 10/17/2019

File Edit View History Bookmarks Tools Help

SMART Ink

Introduction to Indexes

www.pgrocer.net/Cis50/oraindex.html

## Indexes in Oracle - An Introduction

If an index has been established and the SQL statement is setup to take advantage of the index, the index will be searched first and the access time will probably be noticeably lower if you are dealing with a large database. Without an index Oracle does a full table search examining every row. Oracle is actively involved in using indexes to satisfy queries and examines the query to determine what indexes it will use.

In Oracle, indexes are grouped with the concept of constraints. There are two major categories of constraints: integrity constraints that refer to the key fields and value constraints that deal with data entered into a column. A constraint is used to protect the validity of data in one or multiple tables and prevent invalid entries. Specifically, constraints enforce certain rules dealing with a table or a column of that table and can be used to prevent the deletion of a table that has children or dependencies. Indexes as constraints are making sure that the primary key field is unique and that the connection through a foreign key is valid.

The constraints that we will look at are shown in the table below.

Constraint	Processing
CHECK	Allows the specification of a condition on the data
FOREIGN KEY	Key used in the relationship of two tables
PRIMARY KEY	Unique key to each row in the table - uniquely identifies row
NOT NULL	Column must not be null
UNIQUE	Column(s) that must be unique for each row in the table

When the programmer is using constraints, they have the option of naming them (then the name can be meaningful) or having the system generate a name with the SYS-Cn format. Constraints can be part of the process to create a table or they can be done as maintenance of the table. Since constraints can be on a column or on a table they can be defined at either level. If you want to see the constraints that have been assigned to a particular table do a SELECT from the USER\_CONSTRAINTS data dictionary table.

To see all tables use:

```
SELECT * FROM USER_CONSTRAINTS;
```

To see a specific table, use:

```
SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME = 'DONOR';
```

In the example below, I have defined one field as a primary key and put a check constraint on another.

**SQL CODE:**

```
1 CREATE TABLE TRYKEY1
2 (idno NUMBER(3) CONSTRAINT idno_pk PRIMARY KEY,
3 name VARCHAR2(20),
4* deptno NUMBER(2) CONSTRAINT valid_dept_ch CHECK (deptno > 0 AND deptno < 20))
SQL> /
```

Table created.

I am now inserting data into the table and making some errors to correspond to the constraints that I put in the table. The first row was inserted with no problems. In the second example, I tried to put in a row with a duplicate of the idno in the first row. An error occurred because of idno\_pk. In the next example, I violated the deptno check and got an error on that constraint entitled valid\_dept\_ch. Notice that when I do a DESC on the table, the idno is described as NOT NULL. This is because a primary key can not contain a null value.

Windows Taskbar: Type here to search, 9:46 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

Notes on index questions x +

www.pgrocer.net/Cis50/keyques.html

You can put the following clauses on the ALTER TABLE command to deal with problematic indexes that you want to drop.

- DROP PRIMARY KEY name of primary key;
- DROP CONSTRAINT name of constraint;
- DROP FOREIGN KEY name of foreign key;

You can disable a constraint by issuing the ALTER TABLE command with the DISABLE CONSTRAINT clause. If you add the CASCADE option you can disable dependent constraints. You can then active them again with the ENABLE CONSTRAINT clause.

You can also use the CASCADE clause with the DROP to drop a primary key and all associated links.

In the examples below, I am creating two tables: pay2x and dept2x. The primary key of pay2x is paycode and the primary key of dept2x is deptno. The deptno field on pay2x is linked to the deptno primary key on dept 2x as a foreign key.

First I created the primary key on pay2x. Notice there are no keys on dept2x as this point.

**SQL CODE:**

```
SQL> alter table pay2x
  add constraint paycode_pk primary key(paycode);

Table altered.

SQL> desc pay2x;
Name                               Null?   Type
-----
PAYCODE                             NOT NULL NUMBER(2)
JOBNAME                             VARCHAR2(15)
DEPTNO                               VARCHAR2(2)
```

```
SQL> select constraint_name, column_name from user_cons_columns where table_name = 'PAY2X';

CONSTRAINT_NAME      COLUMN_NAME
-----
PAYCODE_PK           PAYCODE
```

```
SQL> select constraint_name, column_name from user_cons_columns where table_name = 'DEPT2X';

CONSTRAINT_NAME      COLUMN_NAME
-----
DEPTNO2X_PK         DEPTNO
```

I then went in and dropped the primary key and checked to see that it no longer existed. Then I immediately put it back. I also added a primary key to dept2x (the code for that is not shown) and checked to see that it was there.

**SQL CODE:**

```
SQL> alter table pay2x
  2 drop primary key;

Table altered.

SQL> select constraint_name, column_name from user_cons_columns where table_name = 'PAY2X';

no rows selected
```

Windows search bar: Type here to search

Taskbar: Edge, File Explorer, Mail, Firefox, VS Code

System tray: 9:47 AM, 10/17/2019

File Edit View History Bookmarks Tools Help

Notes on index questions

www.pgrocer.net/Cis50/keyques.html

```
SQL> select constraint_name, column_name from user_cons_columns where table_name = 'PAY2X';
no rows selected

SQL> alter table pay2x
  2 add constraint paycode2x_pk primary key(paycode);

Table altered.

SQL> select constraint_name, column_name from user_cons_columns where table_name = 'DEPT2X';
```

CONSTRAINT_NAME	COLUMN_NAME
DEPTNO2X_PK	DEPTNO

**SQL CODE:**

In this example, I am adding the foreign key to go from the deptno in pay2x to the deptno in dept2x.

**SQL CODE:**

```
SQL> alter table pay2x
  2 add constraint deptno2x_fk foreign key (deptno) references dept2x;

Table altered.

SQL> select constraint_name, column_name from user_cons_columns where table_name = 'PAY2X';
```

CONSTRAINT_NAME	COLUMN_NAME
DEPTNO2X_FK	DEPTNO
PAYCODE2X_PK	PAYCODE

```
SQL> select constraint_name, column_name from user_cons_columns where table_name = 'DEPT2X';
```

CONSTRAINT_NAME	COLUMN_NAME
DEPTNO2X_PK	DEPTNO

Finally, I dropped the primary key in dept2x which is deptno and did it with the cascade clause which means that the foreign key will be dropped to. I checked it and it worked!

**SQL CODE:**

```
SQL> ALTER TABLE DEPT2X
  2 DROP PRIMARY KEY CASCADE;
```

Table altered.

```
SQL> select constraint_name, column_name from user_cons_columns where table_name = 'DEPT2X';
no rows selected

SQL> select constraint_name, column_name from user_cons_columns where table_name = 'PAY2X';
```

CONSTRAINT_NAME	COLUMN_NAME
PAYCODE2X_PK	PAYCODE

SMART Ink

Type here to search

9:47 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on views x +

www.pgrocer.net/Cis50/moreview.html

View created.

```
SQL> DESC disord1;
Name                Null?   Type
-----
ITEM_NO              VARCHAR2(4)
ITEM_NAME             VARCHAR2(15)
NUMBER_ORDERED       NUMBER(3)
```

```
SQL> SELECT * FROM disord1;
ITEM ITEM_NAME          NUMBER_ORDERED
-----
1111 Good Night Moon      3
1212 Heidi                1
2121 Teddy Bear           1
2345 Doll House           1
3333 Basketball           1
3333 Basketball           2
3456 Net/Hoop             1
```

7 rows selected.

If you want to modify a view, you use the CREATE OR REPLACE clause which allows the view to be created as a new view or an old version of the view to be replaced. This means you can change the view without first deleting it, but it also means you have to be careful or you may loose a view that you wanted to retain.

**SQL CODE:**

```
SQL> CREATE OR REPLACE VIEW disord1
2 AS
3 SELECT l.itemno ITEM_NO, itemname ITEM_NAME, price ITEM_PRICE, numord NUMBER_ORDERED
4 FROM ordline l, inven i
5 WHERE l.itemno = i.itemno;
```

View created.

```
SQL> DESC disord1;
Name                Null?   Type
-----
ITEM_NO              VARCHAR2(4)
ITEM_NAME             VARCHAR2(15)
ITEM_PRICE            NUMBER(6,2)
NUMBER_ORDERED       NUMBER(3)
```

```
SQL> SELECT * FROM disord1;
ITEM ITEM_NAME          ITEM_PRICE NUMBER_ORDERED
-----
1111 Good Night Moon      12.99      3
1212 Heidi                14.99      1
2121 Teddy Bear           19.95      1
2345 Doll House           55.98      1
3333 Basketball           17.99      1
3333 Basketball           17.99      2
3456 Net/Hoop             27.95      1
```

7 rows selected.

Windows search bar: Type here to search

Taskbar: File Explorer, Edge, Mail, Store, Firefox, Chrome

System tray: 9:48 AM, 10/17/2019



File Edit View History Bookmarks Tools Help SMART Ink

More on views x +

www.pgrocer.net/Cis50/moreview.html

View created.

```
SQL> DESC disord1;
Name                               Null?   Type
-----
ITEM_NO                             VARCHAR2(4)
ITEM_NAME                           VARCHAR2(15)
ITEM_PRICE                          NUMBER(6,2)
NUMBER_ORDERED                      NUMBER(3)
```

```
SQL> SELECT * FROM disord1;

ITEM ITEM_NAME          ITEM_PRICE NUMBER_ORDERED
-----
1111 Good Night Moon    12.99          3
1212 Heidi              14.99          1
2121 Teddy Bear        19.95          1
2345 Doll House        55.98          1
3333 Basketball        17.99          1
3333 Basketball        17.99          2
3456 Net/Hoop           27.95          1

7 rows selected.
```

The example below shows another way to assign names to the view that are different from the names in the table(s). In this example, I am only using one table and taking selected fields from that table and giving them a new name. Note that the alias list is in the same order as the column list in the subquery. The where cost > 15 means that only rows where the cost is > 15 will appear in the new view.

**SQL CODE:**

```
1 CREATE VIEW disord2
2   (itmno, itmnam, itmcost, itmprice)
3 AS
4   SELECT itemno, itemname, cost, price
5   FROM inven
6  WHERE COST > 15
SQL> /
```

View created.

```
SQL> DESC disord2;
Name                               Null?   Type
-----
ITMNO                               VARCHAR2(4)
ITMNAM                              VARCHAR2(15)
ITMCOST                             NUMBER(6,2)
ITMPRICE                             NUMBER(6,2)
```

```
SQL> SELECT * FROM disord2;

ITMN ITMNAM          ITMCOST ITMPRICE
-----
2222 Building Blocks    23      27.98
2345 Doll House        45      55.98
3456 Net/Hoop          25      27.95
```

You can update data on the table and in the view through the view. In this case I am updating the view and changing the cost of item 2222. Notice that the change takes place in both the view and the base table behind the view. In the second example, I update the table. Notice that the change takes place in both the table and the view.

Type here to search

9:50 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on views

www.pgrocer.net/Cis50/moreview.html

You can update data on the table and in the view through the view. In this case I am updating the view and changing the cost of item 2222. Notice that the change takes place in both the view and the base table behind the view. In the second example, I update the table. Notice that the changes take place in both the table and the view.

**SQL CODE:**

```
SQL> SELECT * FROM disord2;
```

ITMN	ITMNAM	ITMCOST	ITMPRICE
2222	Building Blocks	23	27.98
2345	Doll House	45	55.98
3456	Net/Hoop	25	27.95

```
SQL> UPDATE disord2
  2 SET itmcost = 48
  3 WHERE itmno = '2222';

1 row updated.
```

```
SQL> SELECT * FROM disord2;
```

ITMN	ITMNAM	ITMCOST	ITMPRICE
2222	Building Blocks	48	27.98
2345	Doll House	45	55.98
3456	Net/Hoop	25	27.95

```
SQL> SELECT * FROM inven;
```

ITEM	ITEMNAME	ONHAND	ONORDER	REORDPT	COST	PRICE	DE	IT	LOCA
1111	Good Night Moon	24	30	40	8	12.99	BK	BY	X100
1212	Heidi	12	25	25	10	14.99	BK	CH	X112
1234	Adven Reddy Fox	5	0	10	9	14.75	BK	CH	X100
2121	Teddy Bear	5	20	40	15	19.95	TY	CH	X115
2222	Building Blocks	4	0	15	48	27.98	TY	CH	Z200
2345	Doll House	2	5	10	45	55.98	TY	CH	Z212
3333	Basketball	24	25	50	14	17.99	SP	BK	Y200
3456	Net/Hoop	12	0	25	25	27.95	SP	BK	Y200

```
8 rows selected.
```

```
SQL> UPDATE inven
  2 SET price = 51.99
  3 WHERE itemno = '2222';

1 row updated.
```

```
SQL> SELECT * FROM inven;
```

ITEM	ITEMNAME	ONHAND	ONORDER	REORDPT	COST	PRICE	DE	IT	LOCA
1111	Good Night Moon	24	30	40	8	12.99	BK	BY	X100
1212	Heidi	12	25	25	10	14.99	BK	CH	X112
1234	Adven Reddy Fox	5	0	10	9	14.75	BK	CH	X100
2121	Teddy Bear	5	20	40	15	19.95	TY	CH	X115

Type here to search

9:51 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on views x +

www.pgrocer.net/Cis50/moreview.html

```
SQL> SELECT * FROM inven;
ITEM  ITEMNAME          ONHAND  ONORDER  REORDPT   COST   PRICE DE IT LOCA
-----
1111  Good Night Moon    24      30       40        8     12.99 BK BY X100
1212  Heidi              12      25       25        10    14.99 BK CH X112
1234  Adven Reddy Fox    5        0       10        9     14.75 BK CH X100
2121  Teddy Bear         5        20      40        15    19.95 TY CH X115
2222  Building Blocks    4        0       15        48    51.99 TY CH Z200
2345  Doll House         2        5       10        45    55.98 TY CH Z212
3333  Basketball         24      25       50        14    17.99 SP BK Y200
3456  Net/Hoop           12        0       25        25    27.95 SP BK Y200

8 rows selected.

SQL> SELECT * from disord2;

ITMN  ITMNAM          ITMCOST  ITMPRICE
-----
2222  Building Blocks    48      51.99
2345  Doll House         45      55.98
3456  Net/Hoop           25      27.95
```

In the example below, I am putting a constraint on the view which means that through the view I cannot change any cost so that it falls below the criteria for the view which is that itmcost > 15. After setting the constraint, first I inserted a new row with a valid cost. Then, I tried to change the cost of the new record in the view to fall below 15, note the response. Next, I update the inventory file and changed cost. This time when I showed the view, this record was not selected because it did not meet the criteria.

**SQL CODE:**

```
1 CREATE VIEW disord3
2 AS
3 SELECT *
4 FROM disord2
5 WHERE itmcost > 15
6* WITH CHECK OPTION CONSTRAINT disord2_cost_ck
SQL> /


View created.

SQL> SELECT * FROM disord3;

ITMN  ITMNAM          ITMCOST  ITMPRICE
-----
2222  Building Blocks    48      51.99
2345  Doll House         45      55.98
3456  Net/Hoop           25      27.95

SQL> DESC disord3;
Name                               Null?   Type
-----
ITMNO                               YES     VARCHAR2(4)
ITMNAM                              YES     VARCHAR2(15)
ITMCOST                             YES     NUMBER(6,2)
ITMPRICE                             YES     NUMBER(6,2)

SQL> INSERT into disord3
2 VALUES ('6789', 'BAT/BALL', 18, 21.99);
```



File Edit View History Bookmarks Tools Help SMART Ink

More on views x +

www.pgrocer.net/Cis50/moreview.html

2345 Doll House	45	55.98
3456 Net/Hoop	25	27.95

In the example below, I am putting a constraint on the view which means that through the view I cannot change any cost so that it falls below the criteria for the view which is that  $itmcost > 15$ . After setting the constraint, first I inserted a new row with a valid cost. Then, I tried to change the cost of the new record in the view to fall below 15, note the response. Next, I update the inventory file and changed cost. This time when I showed the view, this record was not selected because it did not meet the criteria.

**SQL CODE:**

```

1 CREATE VIEW disord3
2 AS
3 SELECT *
4 FROM disord2
5 WHERE itmcost > 15
6* WITH CHECK OPTION CONSTRAINT disord2_cost_ck
SQL> /

View created.

SQL> SELECT * FROM disord3;

ITMN ITMNAM          ITMCOST  ITMPRICE
-----
2222 Building Blocks      48      51.99
2345 Doll House          45      55.98
3456 Net/Hoop            25      27.95

SQL> DESC disord3;
Name                               Null?    Type
-----
ITMNO                               NUMBER(6,2)
ITMNAM                              VARCHAR2(15)
ITMCOST                              NUMBER(6,2)
ITMPRICE                             NUMBER(6,2)

SQL> INSERT into disord3
2 VALUES ('6789', 'BAT/BALL', 18, 21.99);

1 row created.

SQL> SELECT * FROM disord3;

ITMN ITMNAM          ITMCOST  ITMPRICE
-----
2222 Building Blocks      48      51.99
2345 Doll House          45      55.98
3456 Net/Hoop            25      27.95
6789 BAT/BALL            18      21.99

SQL> UPDATE DISORD3
2 SET itmcost = 14.99
3 WHERE itmno = '6789';
UPDATE DISORD3
*
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation

```

Type here to search

9:52 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

More on views x +

www.pgrocer.net/Cis50/moreview.html

```

1 row updated.
SQL> SELECT * from inven;
ITEM  ITEMNAME          ONHAND  ONORDER  REORDPT    COST    PRICE DE IT LOCA
-----
1111  Good Night Moon     24      30       40         8      12.99 BK BY X100
1212  Heidi               12      25       25        10     14.99 BK CH X112
1234  Adven Reddy Fox     5        0        10         9     14.75 BK CH X100
2121  Teddy Bear          5        20       40        15     19.95 TY CH X115
2222  Building Blocks     4        0        15         48     51.99 TY CH Z200
2345  Doll House          2        5        10         45     55.98 TY CH Z212
3333  Basketball          24      25       50         14     17.99 SP BK Y200
3456  Net/Hoop            12       0        25         25     27.95 SP BK Y200
6789  BAT/BALL            14.99   21.99
-----
9 rows selected.
SQL> SELECT * FROM disord3;
ITMN  ITMNAM          ITMCOST  ITMPRICE
-----
2222  Building Blocks     48      51.99
2345  Doll House          45      55.98
3456  Net/Hoop            25      27.95
-----

```

In this example, I created a view that was labeled as READ ONLY, this means that no data base maintenance operations can be performed through this view.

**SQL CODE:**

```

SQL> CREATE VIEW disord4 AS
  2 SELECT * FROM disord3
  3 WITH READ ONLY;
View created.
SQL> SELECT * FROM disord4;
ITMN  ITMNAM          ITMCOST  ITMPRICE
-----
2222  Building Blocks     48      51.99
2345  Doll House          45      55.98
3456  Net/Hoop            25      27.95
-----
SQL> INSERT INTO disord4
  2 VALUES ('7890', 'Mother Goose', 15.25, 16.99);
INSERT INTO disord4
*
ERROR at line 1:
ORA-01733: virtual column not allowed here

```

To drop a view from the data dictionary use the command DROP VIEW followed by the name of the view.

**SQL CODE:**

Type here to search

9:53 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

Database objects x +

www.pgrocer.net/Cis50/dataobj.html

## Database Objects in Oracle:

There are five data base objects covered in this course: table, view, index, sequence, and synonym. As you know, the table is the basic unit of storage that contains records called rows and fields called columns. A view is a logical entity that contains data from one or more tables. An index is generated for a column to improve the search and retrieval of data or to prevent duplication of key values. A sequence is the automatic generating of primary key values. A synonym is used to give an object an alternative name.

### SEQUENCE:

The sequence generator can be used to generate data for columns that are unique numbers in sequence. An obvious application for the generator is the creation of primary keys. The sequence generator allows you to set a value to increment by, a value to start with, a maximum and minimum value and to set the option to cycle or nocycle and to cache or nocache. These options will be discussed more as we look at examples. In the first example, I created a sequence called id\_no. This sequence will start with 1000, and increment by 1. The default minimum value will be 1 and the maximum value will be 1999. The NOCACHE means that oracle will not keep a certain number of values in memory during processing and the nocycle default option means that when the maxvalue is reached, additional values will not be generated. Nocycle is recommended when you are creating primary keys because of purging old rows. Notice in the listings below, that last\_number shows the next available number. Since the sequence has not been used, the starting value is the next available number.

### SQL CODE:

```
SQL> CREATE SEQUENCE id_no
2 INCREMENT BY 1
3 START WITH 1000
4 MAXVALUE 1999
5 NOCACHE
6 NOCYCLE;
```

Sequence created.

```
SQL> DESC USER_SEQUENCES;
Name                               Null?    Type
-----
SEQUENCE_NAME                       NOT NULL VARCHAR2 (30)
MIN_VALUE                            NUMBER
MAX_VALUE                            NUMBER
INCREMENT_BY                         NOT NULL NUMBER
CYCLE_FLAG                           VARCHAR2 (1)
ORDER_FLAG                           VARCHAR2 (1)
CACHE_SIZE                           NOT NULL NUMBER
LAST_NUMBER                          NOT NULL NUMBER
```

```
SQL> SELECT * FROM USER_SEQUENCES;
SEQUENCE_NAME      MIN_VALUE  MAX_VALUE  INCREMENT_BY  C O CACHE_SIZE  LAST_NUMBER
-----
ID_NO              1          1999       1 N N         0          1000
```

Now that the sequence has been created, it can be used in tables. NEXTVAL can be used to retrieve the next available sequence value and CURRVAL can be used to retrieve the current sequence value. The way this works is when you use nextval a new sequence number is generated and the current one is then placed in curval. From this explanation you can see that curval does not contain a number before nextval is used.

In the example below, I first created a table called TESTSEQ to use in testing the sequence object. Then I inserted data into the table. Note that when I inserted into ID, I said id\_no which is the name of the sequence followed by .NEXTVAL. At this point nextval has been used so curval now contains a number - as you can see, they are both 1001.

Windows Taskbar: Type here to search, 9:54 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

Database objects x +

www.pgrocer.net/Cis50/dataobj.html

### SQL CODE:

```
SQL> CREATE SEQUENCE dept_no
2 INCREMENT BY 2
3 START WITH 2
4 MAXVALUE 12;

Sequence created.

SQL> INSERT INTO testseq1
2 VALUES(dept_no.NEXTVAL, 'CIS' , 15000);

1 row created.

SQL> INSERT INTO testseq1
2 VALUES(dept_no.NEXTVAL, 'Payroll', 10000);

1 row created.

SQL> SELECT * FROM testseq1;
```

DEPT	DEPTNAME	BUDGET
2	CIS	15000
4	Payroll	10000

Now I am adding some more records to the table and then showing the results below. As you can see, when I tried to add a record that would take me past the maximum value, I ran into problems. I now need to alter the table to change the maxvalue.

### SQL CODE:

```
SQL> SELECT * FROM testseq1;
```

DEPT	DEPTNAME	BUDGET
2	CIS	15000
4	Payroll	10000
6	Acct Pay	11000
8	Acct Pay	9000
10	Inventory	20000
12	Training	15000

```
6 rows selected.

SQL> INSERT INTO testseq1
2 VALUES(dept_no.NEXTVAL, 'Personnel', 6000);
INSERT INTO testseq1
*
ERROR at line 1:
ORA-08004: sequence DEPT_NO.NEXTVAL exceeds MAXVALUE and cannot be instantiated

SQL> ALTER SEQUENCE dept_no
2 MAXVALUE 100;

Sequence altered.

SQL> INSERT INTO testseq1
```

Type here to search

9:58 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

Database objects x +

www.pgrocer.net/Cis50/dataobj.html

### SQL CODE:

```
SQL> SELECT * FROM testseq1;
```

DEPT	DEPTNAME	BUDGET
2	CIS	15000
4	Payroll	10000
6	Acct Pay	11000
8	Acct Pay	9000
10	Inventory	20000
12	Training	15000

6 rows selected.

```
SQL> INSERT INTO testseq1
  2 VALUES(dept_no.NEXTVAL, 'Personnel', 6000);
INSERT INTO testseq1
*
ERROR at line 1:
ORA-08004: sequence DEPT_NO.NEXTVAL exceeds MAXVALUE and cannot be instantiated
```

```
SQL> ALTER SEQUENCE dept_no
  2 MAXVALUE 100;
```

Sequence altered.

```
SQL> INSERT INTO testseq1
  2 VALUES(dept_no.NEXTVAL, 'Personnel', 6000);
```

1 row created.

```
SQL> SELECT * FROM testseq1;
```

DEPT	DEPTNAME	BUDGET
2	CIS	15000
4	Payroll	10000
6	Acct Pay	11000
8	Acct Pay	9000
10	Inventory	20000
12	Training	15000
14	Personnel	6000

7 rows selected.

### INDEXES:

Remember that when you define a primary key or a unique key a unique index is automatically created to meet this definition. Non unique indexes are created using CREATE INDEX and there purpose is to improve processing efficiency.

Indexes should be created on columns that are frequently referred to in the WHERE clause or in a relational join condition, columns that have a lot of different values or that contain a lot of null values, or where tables are very large and most queries will retrieve less than 2-4% of the rows. A counter indication for creating indexes is when the table is frequently updated.

You can create an index on one column or multiple columns. Below I created two indexes, one on department name alone and one on budget and deptname. I then showed the two indexes from user\_indexes and the columns used in the indexes from user\_ind\_columns. Finally, I dropped the budget\_deptname\_idx index.

Windows taskbar: Type here to search, 10:00 AM 10/17/2019



File Edit View History Bookmarks Tools Help SMART Ink

Database objects x +

www.pgrocer.net/Cis50/dataobj.html

```
SQL> CREATE INDEX deptname_idx
  2 ON testseq1(deptname);

Index created.

SQL> CREATE INDEX budget_deptname_idx
  2 ON testseq1(budget, deptname);

Index created.

SQL> SELECT INDEX_NAME, UNIQUENESS FROM USER_INDEXES WHERE TABLE_NAME = 'TESTSEQ1';

INDEX_NAME                UNIQUENES
-----
BUDGET_DEPTNAME_IDX       NONUNIQUE
DEPTNAME_IDX              NONUNIQUE

SQL> SELECT INDEX_NAME, COLUMN_NAME FROM USER_IND_COLUMNS
  2 WHERE TABLE_NAME = 'TESTSEQ1';

INDEX_NAME                COLUMN_NAME
-----
BUDGET_DEPTNAME_IDX       BUDGET
BUDGET_DEPTNAME_IDX       DEPTNAME
DEPTNAME_IDX              DEPTNAME

SQL> DROP INDEX BUDGET_DEPTNAME_IDX;

Index dropped.
```

**Index made up of multiple fields.**

### SYNONYMS:

Synonyms create another name for an object. If you want access to a table owned by another user, you have to put the name of the user followed by a period followed by the table name. If you use a synonym you can simply refer to that name. When creating a synonym, the database administrator can use the DBA privileges to make the synonym public so it is available to all users. As you can see, I am working as scott/tiger and do not have these privileges.

### SQL CODE:

```
SQL> CREATE SYNONYM ts1
  2 FOR testseq1;

Synonym created.

SQL> DROP SYNONYM ts1;

Synonym dropped.

SQL> CREATE PUBLIC SYNONYM ts1
  2 FOR testseq1;
CREATE PUBLIC SYNONYM ts1
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Windows Taskbar: Type here to search, 10:01 AM 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

Script for a form x +

www.pgrocer.net/Cis50/scriptin.html

```
SQL> SELECT * FROM DONATION;

IDNO  DRI  CONTDATE  CONTAMT
-----
11111 100  07-JAN-99    25
12121 200  23-FEB-99    40
23456 100  03-MAR-99    20
33333 300  10-MAR-99    10
22222 100  14-MAR-99    10
12121 100  04-JUN-99    50

6 rows selected.

SQL> DESC DONATION;
Name                               Null?    Type
-----
IDNO                               VARCHAR2(5)
DRIVENO                            VARCHAR2(3)
CONTDATE                            DATE
CONTAMT                             NUMBER(6,2)

SQL> EDIT DEFINEFORM
```

The script that I create is shown below. The accept will take the data in and the INSERT will put the information into the fields on the form. Notice that I am taking the data into fields I define as in\_ followed by the column name. In fact I could call them anything. Remember the & is used with temporary variables. In the INSERT, the varchar2 and the date fields are inserted in quotes and the number field is left without the quotes. This means that the user does not have to enter quotes.

**SQL CODE:**

```
ACCEPT in_idno PROMPT 'Please enter the idno: ';
ACCEPT in_driveno PROMPT 'Please enter the driveno: ';
ACCEPT in_contdate PROMPT 'Please enter the contribution date: ';
ACCEPT in_contamt PROMPT 'Please enter the contribution amount: ';
INSERT INTO donation
VALUES('&in_idno', '&in_driveno', '&in_contdate', &in_contamt);
```

After the script has been written it, I save and go back to the SQL> prompt where I issue the command @ defineform, this executes the script. The accept prompts are shown, the data is entered. Again remember because there are quotes in the INSERT, the user does not have to enter quotes. To avoid the old/new you can set verify off, I decided to leave it on to show what was happening. As you can see, the insert works and the row is created. Another execution creates another row in the table.

**SQL CODE:**

```
SQL> @ defineform
Please enter the idno: 11111
Please enter the driveno: 200
Please enter the contribution date: 12-JUN-99
Please enter the contribution amount: 35
old 2: VALUES('&in_idno', '&in_driveno', '&in_contdate', &in_contamt)
new 2: VALUES('11111', '200', '12-JUN-99', 35)

1 row created.

Input truncated to 1 characters
```

Type here to search

10:03 AM 10/17/2019

The screenshot shows a web browser window with the following elements:

- Browser Tabs:** "CIS150/50 Course Page" and "Indexes and keys etc".
- Address Bar:** "www.pgrocer.net/CIS150/assignments/KeyindexF18.html".
- Page Title:** "Indexes and keys etc".
- Problem #1:** "Set up a patient table with a patient id, name, address, code for their primary care doctor. Set up a doctor table that has the code for the doctor, their name, type of doctor. The patient table should have a primary key and the code should be a foreign into the doctor table. The patients name should be set up as an index."
- Problem #2:** "Design and develop a set of tables for a payroll system using the information below. Set up the necessary tables for this database and populate them with data. You need to create the appropriate primary keys and foreign keys. Your must design a relational database following the rules to achieve third normal form. If you want me to look at your design before you implement, please send it to me with a comment in the subject that tells me what it is. You need to use a check constraints, a unique constraint and a null constraint somewhere in this development. You should also create an index. We will look at a very simple version of payroll. Your payroll has to carry the following information:"
- Payroll System Requirements:**
  - Employee Id
  - Employee Name
  - Salary
  - Tax percent withheld
  - Medical withheld
  - Other deductions withheld
  - Projects the person is worked on (note there can be more than one project)
  - Name of the project (each project should have a name)
  - Manager of the project (each project is assigned a manager)
  - Hours the employee worked on the project (each time they worked on it)

File Edit View History Bookmarks Tools Help

CIS150/50 Course Page x Indexes and keys etc x CIS150/50 Course Page x +

www.pgrocer.net/CIS150/assignments/KeyindexF18.html

## Indexes and keys etc

Problem #1: Set up a patient table with a patient id, name, address, code for their primary care doctor.  
 Set up a doctor table that has the code for the doctor, their name, type of doctor.  
 The patient table should have a primary key and the code should be a foreign into the doctor table. The patients name should be set up as an index.

Problem #2: Design and develop a set of tables for a payroll system using the information below. Set up the necessary tables for this database and populate them with data. You need to create the appropriate primary keys and foreign keys. You must design a relational database following the rules to achieve third normal form. If you want me to look at your design before you implement, please send it to me with a comment in the subject that tells me what it is.  
 You need to use a check constraints, a unique constraint and a null constraint somewhere in this development. You should also create an index.  
 We will look at a very simple version of payroll. Your payroll has to carry the following information:

- Employee Id
- Employee Name
- Salary
- Tax percent withheld
- Medical withheld
- Other deductions withheld
- Projects the person is worked on (note there can be more than one project)
- Name of the project (each project should have a name)
- Manager of the project (each project is assigned a manager)
- Hours the employee worked on the project (each time they worked on it)

PK EMP

Emp id

Emp name

Salary

Tax

Med

Other

Emp Proj

Emp id

Proj id

Date

Hrs

Project PK

Proj id

Proj name

Manag

Windows taskbar: Type here to search, 10:32 AM, 10/17/2019

File Edit View History Bookmarks Tools Help

Introduction to PL/SQL x Indexes and keys etc x CIS150/50 Course Page x +

www.pgrocer.net/Cis50/introplshtml

## Introduction to PL/SQL

PL/SQL is a procedural language that can use a step through records approach to handle processing. It implements IF statements and loops, variables and types, procedures and functions while still allowing for traditional query and maintenance type processing through SQL. The procedural constructions available in PL/SQL add tremendous power to the processing capabilities available with the Oracle relational database.

PL/SQL is written in a block structure where each block performs a logical task. Blocks can be nested within each other. The structure of a block is:

```
DECLARE (optional)
    Declarative section - allows for variables, types, cursors, user defined-exceptions etc
BEGIN (required)
    Executable section - PL/SQL procedural statements and SQL statements
EXCEPTION (optional)
    Exception section - error and exception handling
END; (required)
```

### Example:

In this example, I declared a message called v\_msg in the declarative section. Then I assigned a literal to v\_msg. The assign is done using the assignment code :=. This means that the data to the right of the assignment sign will be assigned to the dataname on the left of the assignment sign. This ended the PL/SQL block.

Looking at the code below, the edit afst\_intro takes me into the editor where I coded the PL/SQL code. When I had completed the code and saved it, I exited the editor and ran the code. This can be done with START followed by the name of the code or @ followed by the name of the code. In this case it would be START afst\_intro or @ afst\_intro.

#### SQL CODE:

```
SQL> EDIT afst_intro
```

#### PL/SQL CODE:

```
DECLARE
  v_msg VARCHAR2(25);
BEGIN
  v_msg := 'This PL/SQL block works!';
END;
/
```

#### SQL CODE:

```
SQL> @ afst_intro
```

PL/SQL procedure successfully completed.

If I want to be able to see the message and assure myself that it works, I can use the following code to show the output. If you set serveroutput on in SQL\*PLUS (notice you also set it off) then you can display output on the screen using dbms\_output.put\_line and enclosing the information you want to see within quotes. In the example below, I have a literal message followed by the contents of a PL/SQL defined variable.

#### SQL CODE:

Windows taskbar: Type here to search, 10:33 AM, 10/17/2019

File Edit View History Bookmarks Tools Help SMART Ink

PL/SQL IF statements x Indexes and keys etc x CIS150/50 Course Page x +

www.pgrocet.net/Cis50/plsqlif1.html

### PL/SQL CODE:

```
SET VERIFY OFF
DECLARE
  v_idno   VARCHAR2(5) := &input_idno;
  v_yrgoal NUMBER(7,2);
BEGIN
  SELECT yrgoal INTO v_yrgoal
  FROM donornew
  WHERE idno = v_idno;
  IF v_yrgoal > 250 THEN
    v_yrgoal := v_yrgoal * 1.1;
  END IF;
  UPDATE donornew
  SET yrgoal = v_yrgoal
  WHERE idno = v_idno;
  COMMIT;
END;
/
SET VERIFY ON
```

To execute the code, I can either enter START donor0 or @ donor0. When prompted for the idno, I entered 12121. The processing was done and the successful message was returned. To verify the processing I then did a select to show the file and indeed record 12121 was increased from 400 to 440.

### SQL CODE:

```
SQL> @ donor0
Enter value for input_idno: 12121

PL/SQL procedure successfully completed.

Input truncated to 13 characters

SQL> SELECT * FROM donornew;
```

IDNO	NAME	STADR	CITY	ST	ZIP	DATEFST	YRGOAL	CONTACT
11111	Stephen Daniels	123 Elm St	Seekonk	MA	02345	03-JUL-98	500	John Smith
12121	Jennifer Ames	24 Benefit St	Providence	RI	02045	24-MAY-97	440	Susan Jones
22222	Carl Hersey	24 Benefit St	Providence	RI	02045	03-JAN-98		Susan Jones
23456	Susan Ash	21 Main St	Fall River	MA	02720	04-MAR-92	120	Amy Costa
33333	Nancy Taylor	26 Oak St	Fall River	MA	02720	04-MAR-92	50	John Adams
34567	Robert Brooks	36 Pine St	Fall River	MA	02720	04-APR-98	50	Amy Costa

6 rows selected.

I then reran the PL/SQL code and input an id where the goal was not > 250. As you can see, no change was made to the table.

### SQL CODE:

```
SQL> @ donor0
Enter value for input_idno: 34567

PL/SQL procedure successfully completed.
```

Windows taskbar: Type here to search, 10:41 AM 10/17/2019

File Edit View History Bookmarks Tools Help

PL/SQL IF statements | Indexes and keys etc | CIS150/50 Course Page

www.pgrocer.net/Cis50/plsqlif1.html

Input truncated to 13 characters  
SQL> select \* from donornew;

IDNO	NAME	STADR	CITY	ST	ZIP	DATEFST	YRGOAL	CONTACT
11111	Stephen Daniels	123 Elm St	Seekonk	MA	02345	03-JUL-98	500	John Smith
12121	Jennifer Ames	24 Benefit St	Providence	RI	02045	24-MAY-97	440	Susan Jones
22222	Carl Hersey	24 Benefit St	Providence	RI	02045	03-JAN-98		Susan Jones
23456	Susan Ash	21 Main St	Fall River	MA	02720	04-MAR-92	120	Amy Costa
33333	Nancy Taylor	26 Oak St	Fall River	MA	02720	04-MAR-92	50	John Adams
34567	Robert Brooks	36 Pine St	Fall River	MA	02720	04-APR-98	50	Amy Costa

6 rows selected.

I now when in and modified the code to put an ELSE clause in. When the v\_yrgoal is greater than 250 I will increase the goal by 10% otherwise I will increase the goal by 5%. In looking at the code, notice that the processing that takes place in the IF is followed by a ; and the processing that takes place in the ELSE is followed by a ;. Also remember the structure which has the IF condition THEN and the terminating END IF;

**SQL CODE:**

```
SQL> edit donor0a
```

**PL/SQL CODE:**

```
SET VERIFY OFF
DECLARE
  v_idno  VARCHAR2(5) := &input_idno;
  v_yrgoal NUMBER(7,2);
BEGIN
  SELECT yrgoal INTO v_yrgoal
  FROM donornew
  WHERE idno = v_idno;
  IF v_yrgoal > 250 THEN
    v_yrgoal := v_yrgoal * 1.1;
  ELSE
    v_yrgoal := v_yrgoal * 1.05;
  END IF;
  UPDATE donornew
  SET yrgoal = v_yrgoal
  WHERE idno = v_idno;
  COMMIT;
END;
/
SET VERIFY ON
```

I then verified the change by entering the idno for a person that had a goal less than >250 and the goal was upped by 5%.

**SQL CODE:**

```
SQL> @ donor0a
Enter value for input_idno: 34567

PL/SQL procedure successfully completed.

Input truncated to 13 characters
```

Windows taskbar: Type here to search, 10:41 AM 10/17/2019

PL/SQL IF statements

www.pgrocer.net/Cis50/plsqlif1.html

by the next IF. At the end, all of the IF statements are closed down with the END IF.

**SQL CODE:**

```
SQL> edit donor0b
```

**PL/SQL CODE:**

```
SET VERIFY OFF
DECLARE
  v_idno  VARCHAR2(5) := &input_idno;
  v_yrgoal NUMBER(7,2);
BEGIN
  SELECT yrgoal INTO v_yrgoal
  FROM donornew
  WHERE idno = v_idno;
  IF v_yrgoal > 300 THEN
    v_yrgoal := v_yrgoal * 1.3;
  ELSE
    IF v_yrgoal > 200 THEN
      v_yrgoal := v_yrgoal * 1.2;
    ELSE
      IF v_yrgoal > 100 THEN
        v_yrgoal := v_yrgoal * 1.1;
      ELSE
        v_yrgoal := v_yrgoal * 1.05;
      END IF;
    END IF;
  END IF;
  UPDATE donornew
  SET yrgoal = v_yrgoal
  WHERE idno = v_idno;
  COMMIT;
END;
/
SET VERIFY ON
```

To test this code I first put in idno 23456 which had a yrgoal that was > 100 and it was increased by 10%. Then I put in idno 11111 which had a yrgoal greater than 300 and it was increased by 30%.

**SQL CODE:**

```
SQL> @ donor0b
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;
```

IDNO	NAME	STADR	CITY	ST	ZIP	DATEFST	YRGOAL	CONTACT
11111	Stephen Daniels	123 Elm St	Seekonk	MA	02345	03-JUL-98	500	John Smith
12121	Jennifer Ames	24 Benefit St	Providence	RI	02045	24-MAY-97	440	Susan Jones
22222	Carl Hersey	24 Benefit St	Providence	RI	02045	03-JAN-98		Susan Jones
23456	Susan Ash	21 Main St	Fall River	MA	02720	04-MAR-92	132	Rmy Costa



File Edit View History Bookmarks Tools Help SMART Ink

More IF statements with PL/SQL | Indexes and keys etc | CIS150/50 Course Page

www.pgrocer.net/Cis50/ormoreif.html

```

11111 Stephen Daniels 123 Elm St Seekonk MA 02345 03-JUL-98 845 John Smith
12121 Jennifer Ames 24 Benefit St Providence RI 02045 24-MAY-97 550 Susan Jones
22222 Carl Hersey 24 Benefit St Providence RI 02045 03-JAN-98 Susan Jones
23456 Susan Ash 21 Main St Fall River MA 02720 04-MAR-92 158.4 Amy Costa
33333 Nancy Taylor 26 Oak St Fall River MA 02720 04-MAR-92 50 John Adams
34567 Robert Brooks 36 Pine St Fall River MA 02720 04-APR-98 63.53 Amy Costa

```

6 rows selected.

The IF statement in the code below uses an embedded AND.

**SQL CODE:**

```
SQL> edit donor3
```

**PL/SQL CODE:**

```

SET VERIFY OFF
DECLARE
  v_idno   VARCHAR2(5) :=&input_idno;
  v_yrgoal NUMBER(7,2);
  v_newgoal NUMBER(7,2);
  v_state  VARCHAR2(2);
BEGIN
  SELECT yrgoal, state INTO v_yrgoal, v_state
  FROM donornew
  WHERE idno = v_idno;
  IF v_yrgoal > 250 AND v_state = 'MA' THEN
    v_newgoal := v_yrgoal * 1.1 ;
  ELSE
    v_newgoal := v_yrgoal * 1.2;
  END IF;
  UPDATE donornew
  SET yrgoal = v_newgoal
  WHERE idno = v_idno;
END;
/
SET VERIFY ON

```

**SQL CODE:**

```
SQL> @ donor3
Enter value for input_idno: 23456

PL/SQL procedure successfully completed.

Input truncated to 13 characters
SQL> SELECT * FROM donornew;
```

IDNO	NAME	STADR	CITY	ST	ZIP	DATEFST	YRGOAL	CONTACT
11111	Stephen Daniels	123 Elm St	Seekonk	MA	02345	03-JUL-98	845	John Smith
12121	Jennifer Ames	24 Benefit St	Providence	RI	02045	24-MAY-97	550	Susan Jones
22222	Carl Hersey	24 Benefit St	Providence	RI	02045	03-JAN-98		Susan Jones
23456	Susan Ash	21 Main St	Fall River	MA	02720	04-MAR-92	190.08	Amy Costa
33333	Nancy Taylor	26 Oak St	Fall River	MA	02720	04-MAR-92	50	John Adams

Windows Taskbar: Type here to search | 10:43 AM 10/17/2019