**DBMS\_CHANGE\_NOTIFICATION**

The DBMS\_CHANGE\_NOTIFICATION package is part of the database change notification feature that provides the functionality to create registration on queries designated by a client application and so to receive notifications in response to DML or DDL changes on the objects associated with the queries. The notifications are published by the database when the DML or DDL transaction commits.

The DBMS\_CHANGE\_NOTIFICATION package provides PL/SQL based registration interfaces. A client can use this interface to create registrations on queries based on objects of interest and specify a PL/SQL call back handler to receive notifications.

When a registration is created through the PL/SQL interface, a unique registration ID is assigned to the registration by the RDBMS. The client application can use the registration ID to keep track of registrations created by it. When a notification is published by the RDBMS, the registration ID will be part of the notification.

The DBMS\_CHANGE\_NOTIFICATION package requires that the user have the CHANGE NOTIFICATION system privilege to receive notifications and be granted EXECUTE privilege on the DBMS\_CHANGE\_NOTIFICATION package.

**The DBMS\_CHANGE\_NOTIFICATION package uses the following OBJECT types**

SYS.CHNF$\_DESC Object Type

**Syntax**

TYPE SYS.CHNF$\_DESC IS OBJECT(

registration\_id NUMBER,

transaction\_id RAW(8),

dbname VARCHAR2(30),

event\_type NUMBER,

numtables NUMBER,

table\_desc\_array CHNF$\_TDESC\_ARRAY)

|  |  |
| --- | --- |
| Qosflags | Table level operation flags. This is a flag field (bit-vector) which describes the operations that occurred on the table. It can be an OR of the following bit fields - INSERTOP, UPDATEOP, DELETEOP, DROPOP, ALTEROP, ALL\_ROWS. If the ALL\_ROWS (0x1) bit is set it means that either the entire table is modified (for example, DELETE \* FROM t) or row level granularity of information is not requested or not available in the notification and the receiver must conservatively assume that the entire table has been invalidated. |

**QOS\_RELIABLE** - Reliable or persistent notification. Also implies that the notifications will be inserted into the persistent storage atomically with the committing transaction that results in an object change.

create or replace procedure tables\_Changed\_chnt(ntfnds IN sys.chnf$\_desc)

is

l\_regid NUMBER;

l\_table\_name VARCHAR2(60);

l\_Even\_type NUMBER;

l\_numtables NUMBER;

l\_operation\_types NUMBER;

l\_numrows NUMBER;

l\_row\_id VARCHAR2(20);

l\_operation VARCHAR2(20);

l\_message VARCHAR2(4000);

begin

l\_regid:=ntfnds.registration\_id;

l\_numtables:=ntfnds.numtables;

l\_Even\_type:=ntfnds.Event\_type;

if l\_Even\_type=dbms\_change\_notification.event\_objchange then

for i in 1 .. l\_numtables loop

l\_table\_name:=ntfnds.table\_desc\_array(i).table\_name;

l\_operation\_types:=ntfnds.table\_desc\_array(i).Opflags;

If (bitand(l\_operation\_types,dbms\_change\_notification.all\_rows)=0) then

l\_numrows:=ntfnds.table\_desc\_array(i).numrows;

else

l\_numrows:=0;

end if;

l\_row\_id := ntfnds.table\_desc\_array(i).row\_desc\_array(i).row\_id;

case when BITAND(l\_operation\_types,dbms\_change\_notification.INSERTOP)!=0 then

l\_operation:='records inserted';

when BITAND(l\_operation\_types,dbms\_change\_notification.updateop)!=0 then

l\_operation:='records updated';

when BITAND(l\_operation\_types,dbms\_change\_notification.deleteop)!=0 then

l\_operation:='records deleted';

when BITAND(l\_operation\_types,dbms\_change\_notification.alterop)!=0 then

l\_operation:='table altered';

when BITAND(l\_operation\_types,dbms\_change\_notification.dropop)!=0 then

l\_operation:='table droped';

when BITAND(l\_operation\_types,dbms\_change\_notification.insertop)!=0 then

l\_operation:='unkown operation';

else

l\_operation:='?';

end case;

l\_message:='Table('||l\_table\_name||')-'||l\_operation||'.Rows='||l\_numrows;

insert into notifications(id,message,notification\_date)

values(notification\_seq.nextval,l\_message,sysdate);

commit;

end loop;

end if;

end;

/

DECLARE

l\_regds SYS.CHNF$\_REG\_INFO;

l\_regid NUMBER;

l\_qosflags NUMBER;

l\_id emp.empno%TYPE;

begin

l\_qosflags:=dbms\_change\_notification.QOS\_RELIABLE+dbms\_change\_notification.QOS\_ROWIDS;

l\_regds:=SYS.CHNF$\_REG\_INFO('tables\_changed\_chnt',l\_qosflags,0,0,0);

l\_regid:=DBMS\_CHANGE\_NOTIFICATION.new\_reg\_Start(l\_regds);

select empno into l\_id from emp where rownum=1;

dbms\_change\_notification.reg\_end;

end;

/

**NEW\_REG\_START** Function

This procedure begins a new registration block. Any objects referenced by queries executed within the registration block are considered interesting objects and added to the registration. The registration block ends upon calling the REG\_END procedure.

**Syntax**

DBMS\_CHANGE\_NOTIFICATION.NEW\_REG\_START (

regds IN sys.chnf$\_reg\_info)

RETURN NUMBER;

### REG\_END Procedure

This procedure marks the end of the registration block. No newly executed queries are tracked.

**Syntax**

DBMS\_CHANGE\_NOTIFICATION.REG\_END;

select \* from notifications;

select \* from user\_change\_notification\_regs;

table created for storing information

Below operations are performed

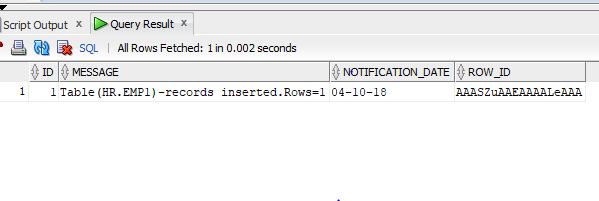
alter table emp modify (ename varchar2(20));

select \* from emp;

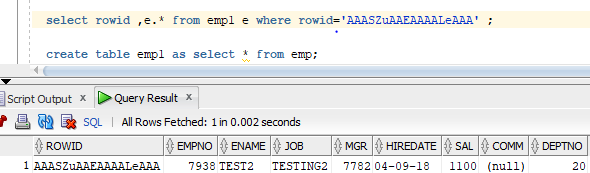
insert into emp1 values(7938,'TEST2','TESTING2',7782,sysdate-30,1100,null,20);

commit;

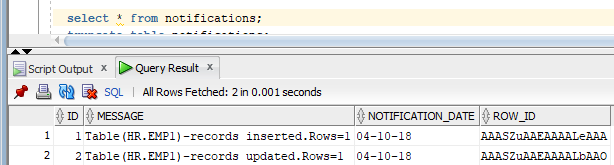
select \* from notifications;

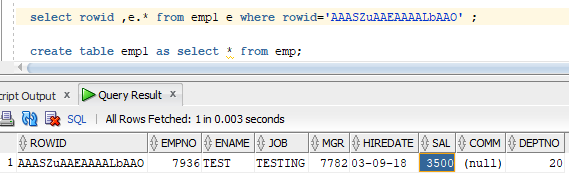


We can see same rowed inserted into table;



update emp1 set sal=3500 where empno=7936;





delete from emp where empno=7938;

