## Task 01: Create Packages for Reload Dimension from SA\_\*

* Use Execute Immediate with Bind Parameters

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_dim\_tariffs\_dw

AS

*-- Load Data From Sources table to DataBase*

PROCEDURE load\_tmp\_tariffs

AS

TYPE tariff IS RECORD

(

tariff\_code VARCHAR2 ( 5 )

, tariff\_name VARCHAR2 ( 50 )

, tariff\_payment\_sum NUMBER

, tariff\_type VARCHAR2 ( 50 )

, tariff\_min\_payment NUMBER

, tariff\_max\_payment NUMBER

);

t\_cv SYS\_REFCURSOR;

TYPE tariffs IS TABLE OF tariff;

tar\_ins tariffs;

tar\_up tariffs;

sql\_str CLOB;

cur\_ins NUMBER;

cur\_up NUMBER;

get\_value NUMBER;

BEGIN

sql\_str := 'SELECT DISTINCT

sa\_t.TARIFF\_CODE

, sa\_t.TARIFF\_NAME

, sa\_t.TARIFF\_PAYMENT\_SUM

, sa\_t.TARIFF\_TYPE

, sa\_t.TARIFF\_MIN\_PAYMENT

, sa\_t.TARIFF\_MAX\_PAYMENT

FROM u\_sa\_data.tmp\_tariffs sa\_t

WHERE sa\_t.TARIFF\_CODE NOT IN (SELECT DISTINCT TARIFF\_CODE

FROM u\_dw.dw\_tariffs)';

cur\_ins := ***dbms\_sql.open\_cursor***;

***dbms\_sql.parse*** ( cur\_ins

, sql\_str

, ***dbms\_sql.native*** );

get\_value := ***dbms\_sql.execute*** ( cur\_ins );

t\_cv := ***dbms\_sql.to\_refcursor*** ( cur\_ins );

FETCH t\_cv

BULK COLLECT INTO tar\_ins;

FORALL i IN INDICES OF tar\_ins

INSERT INTO u\_dw.dw\_tariffs ( tariff\_id

, tariff\_code

, tariff\_name

, tariff\_payment\_sum

, tariff\_type

, tariff\_min\_payment

, tariff\_max\_payment

, insert\_dt )

VALUES ( seq\_tariffs.NEXTVAL

, tar\_ins ( i ).tariff\_code

, tar\_ins ( i ).tariff\_name

, tar\_ins ( i ).tariff\_payment\_sum

, tar\_ins ( i ).tariff\_type

, tar\_ins ( i ).tariff\_min\_payment

, tar\_ins ( i ).tariff\_max\_payment

, SYSDATE );

sql\_str := 'SELECT DISTINCT

sa\_t.TARIFF\_CODE

, sa\_t.TARIFF\_NAME

, sa\_t.TARIFF\_PAYMENT\_SUM

, sa\_t.TARIFF\_TYPE

, sa\_t.TARIFF\_MIN\_PAYMENT

, sa\_t.TARIFF\_MAX\_PAYMENT

FROM u\_sa\_data.tmp\_tariffs sa\_t

MINUS

SELECT DISTINCT

tariff\_code

, tariff\_name

, tariff\_payment\_sum

, tariff\_type

, tariff\_min\_payment

, tariff\_max\_payment

FROM u\_dw.dw\_tariffs';

cur\_up := ***dbms\_sql.open\_cursor***;

***dbms\_sql.parse*** ( cur\_up

, sql\_str

, ***dbms\_sql.native*** );

get\_value := ***dbms\_sql.execute*** ( cur\_up );

t\_cv := ***dbms\_sql.to\_refcursor*** ( cur\_up );

FETCH t\_cv

BULK COLLECT INTO tar\_up;

FORALL j IN INDICES OF tar\_up

UPDATE u\_dw.dw\_tariffs

SET tariff\_min\_payment = tar\_up ( j ).tariff\_min\_payment

, tariff\_max\_payment = tar\_up ( j ).tariff\_max\_payment

, tariff\_payment\_sum = tar\_up ( j ).tariff\_payment\_sum

, update\_dt = SYSDATE;

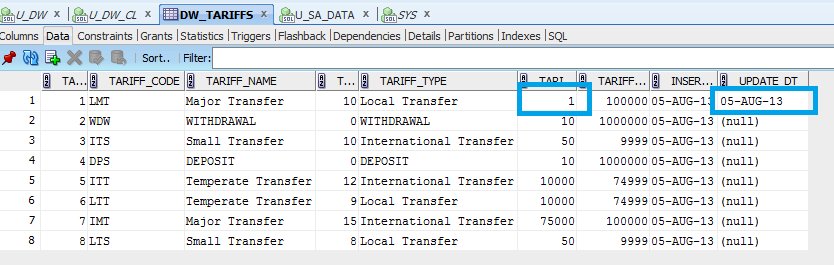
COMMIT;

END load\_tmp\_tariffs;

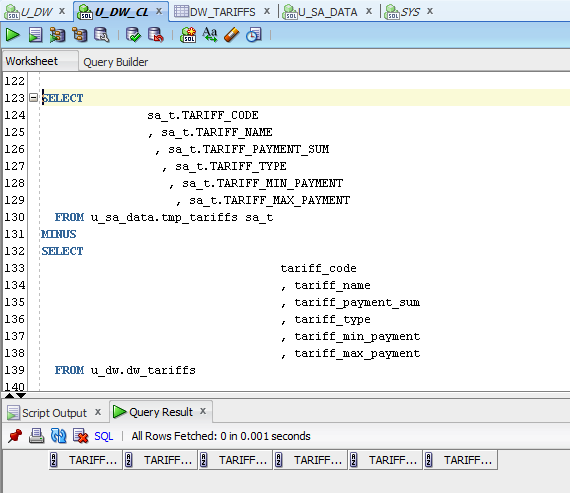
END pkg\_etl\_dim\_tariffs\_dw;

## 

After updating the data in the source table and restart the package data is changed at stage-level.



After repeated run the package the data in the table are unchanged. Test results are shown below:



* Use DBMS\_SQL.TO\_REFCURSOR Function

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_dim\_tar

AS

*-- Load Data From Sources table to DataBase*

PROCEDURE load\_tmp\_tar

AS

CURSOR c1

IS

SELECT DISTINCT dw\_t.tariff\_id

, sa\_t.tariff\_code

, sa\_t.tariff\_name

, sa\_t.tariff\_payment\_sum

, sa\_t.tariff\_type

, sa\_t.tariff\_min\_payment

, sa\_t.tariff\_max\_payment

FROM u\_sa\_data.tmp\_tariffs sa\_t

LEFT OUTER JOIN

u\_dw.dw\_tariffs dw\_t

ON ( dw\_t.tariff\_code = sa\_t.tariff\_code )

WHERE sa\_t.tariff\_code NOT IN (SELECT DISTINCT tariff\_code

FROM u\_dw.dw\_tariffs);

CURSOR c2

IS

SELECT DISTINCT sa\_t.tariff\_code

, sa\_t.tariff\_name

, sa\_t.tariff\_payment\_sum

, sa\_t.tariff\_type

, sa\_t.tariff\_min\_payment

, sa\_t.tariff\_max\_payment

FROM u\_sa\_data.tmp\_tariffs sa\_t

MINUS

SELECT DISTINCT tariff\_code

, tariff\_name

, tariff\_payment\_sum

, tariff\_type

, tariff\_min\_payment

, tariff\_max\_payment

FROM u\_dw.dw\_tariffs;

TYPE t\_ins IS TABLE OF c1%ROWTYPE;

TYPE t\_upd IS TABLE OF c2%ROWTYPE;

nt\_ins t\_ins;

nt\_upd t\_upd;

sql\_ins VARCHAR2 ( 500 );

sql\_upd VARCHAR2 ( 500 );

BEGIN

sql\_ins := 'INSERT INTO u\_dw.dw\_tariffs (tariff\_id

, tariff\_code

, tariff\_name

, tariff\_payment\_sum

, tariff\_type

, tariff\_min\_payment

, tariff\_max\_payment

, insert\_dt)

VALUES (SEQ\_TARIFFS.NEXTVAL, :code, :name, :sum, :type, :min, :max, SYSDATE)';

sql\_upd := 'UPDATE u\_dw.dw\_tariffs tar SET tar.tariff\_payment\_sum=:sum,

tar.tariff\_min\_payment=:min, tar.tariff\_max\_payment=:max,

update\_dt=SYSDATE

WHERE tar.tariff\_code=:code';

OPEN c1;

FETCH c1

BULK COLLECT INTO nt\_ins;

FORALL i IN INDICES OF nt\_ins

EXECUTE IMMEDIATE sql\_ins

USING nt\_ins ( i ).tariff\_code

, nt\_ins ( i ).tariff\_name

, nt\_ins ( i ).tariff\_payment\_sum

, nt\_ins ( i ).tariff\_type

, nt\_ins ( i ).tariff\_min\_payment

, nt\_ins ( i ).tariff\_max\_payment;

OPEN c2;

FETCH c2

BULK COLLECT INTO nt\_upd;

FORALL j IN INDICES OF nt\_upd

EXECUTE IMMEDIATE sql\_upd

USING nt\_upd ( j ).tariff\_payment\_sum

, nt\_upd ( j ).tariff\_min\_payment

, nt\_upd ( j ).tariff\_max\_payment

, nt\_upd ( j ).tariff\_code;

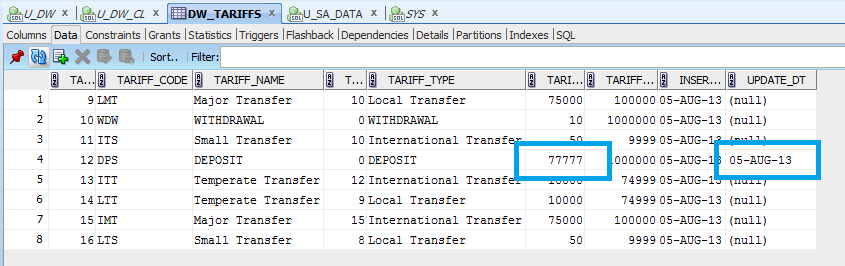
COMMIT;

END load\_tmp\_tar;

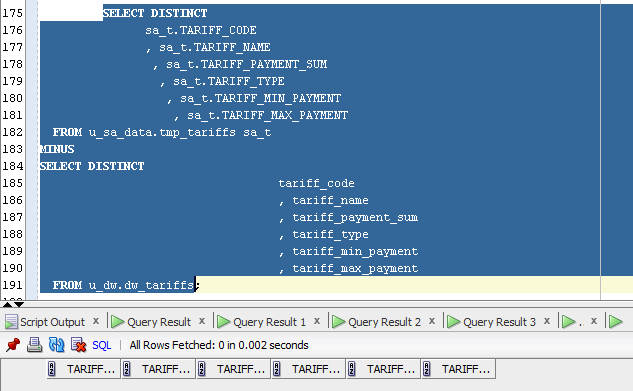
END pkg\_etl\_dim\_tar;

## 

After updating the data in the source table and restart the package data is changed at stage-level.



After repeated run the package the data in the table are unchanged. Test results are shown below:



* Use DBMS\_SQL.TO\_CURSOR\_NUMBER Function

CREATE OR REPLACE PACKAGE BODY pkg\_etl\_dim\_currensy\_dw

AS

*-- Load Data From Sources table to DataBase*

PROCEDURE load\_tmp\_currensy

AS

cur\_id NUMBER;

column\_var NUMBER;

describe\_table ***dbms\_sql.desc\_tab***;

numvar NUMBER;

sql\_str VARCHAR2 ( 10000 );

c1 SYS\_REFCURSOR;

c\_id NUMBER;

c\_t\_id NUMBER;

c\_t\_dol NUMBER;

c\_name VARCHAR2 ( 5 BYTE );

c\_code NUMBER;

c\_desc VARCHAR2 ( 50 BYTE );

old\_val NUMBER;

BEGIN

sql\_str := 'SELECT DISTINCT DW\_T.CURRENCY\_ID

, SA\_T.CURRENCY\_TYPE\_ID

, SA\_T.CURRENCY\_TO\_DOLLAR

, SA\_T.CURRENCY\_NAME

, SA\_T.CURRENCY\_CODE

, SA\_T.CURRENCY\_DESC

, DW\_T.CURRENCY\_TO\_DOLLAR old\_value

FROM u\_sa\_data.TMP\_CURRENCY SA\_T

LEFT JOIN u\_dw.DW\_CURRENCY DW\_T

ON ( SA\_T.CURRENCY\_CODE = DW\_T.CURRENCY\_CODE )';

OPEN c1 FOR sql\_str;

cur\_id := ***dbms\_sql.to\_cursor\_number*** ( c1 );

***dbms\_sql.describe\_columns*** ( cur\_id

, column\_var

, describe\_table );

***dbms\_sql.define\_column*** ( cur\_id

, 1

, c\_id );

***dbms\_sql.define\_column*** ( cur\_id

, 2

, c\_t\_id );

***dbms\_sql.define\_column*** ( cur\_id

, 3

, c\_t\_dol );

***dbms\_sql.define\_column*** ( cur\_id

, 4

, c\_name

, 5 );

***dbms\_sql.define\_column*** ( cur\_id

, 5

, c\_code );

***dbms\_sql.define\_column*** ( cur\_id

, 6

, c\_desc

, 50 );

***dbms\_sql.define\_column*** ( cur\_id

, 7

, old\_val );

WHILE ***dbms\_sql.fetch\_rows*** ( cur\_id ) > 0 LOOP

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 1

, c\_id );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 2

, c\_t\_id );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 3

, c\_t\_dol );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 4

, c\_name );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 5

, c\_code );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 6

, c\_desc );

***dbms\_sql.COLUMN\_VALUE*** ( cur\_id

, 7

, old\_val );

IF c\_id IS NULL THEN

INSERT INTO u\_dw.dw\_currency cu ( cu.currency\_id

, cu.currency\_type\_id

, cu.currency\_to\_dollar

, cu.currency\_name

, cu.currency\_code

, cu.currency\_desc )

VALUES ( seq\_currensy.NEXTVAL

, c\_t\_id

, c\_t\_dol

, c\_name

, c\_code

, c\_desc );

ELSIF c\_t\_dol != old\_val THEN

UPDATE u\_dw.dw\_currency

SET currency\_to\_dollar = c\_t\_dol;

INSERT INTO u\_dw.dw\_currency\_actions ca ( ca.currency\_action\_id

, ca.currency\_id

, ca.action\_date

, ca.currency\_action\_type\_id

, ca.value\_old

, ca.value\_new )

VALUES ( seq\_currensy\_action.NEXTVAL

, c\_id

, SYSDATE

, 1

, old\_val

, c\_t\_dol );

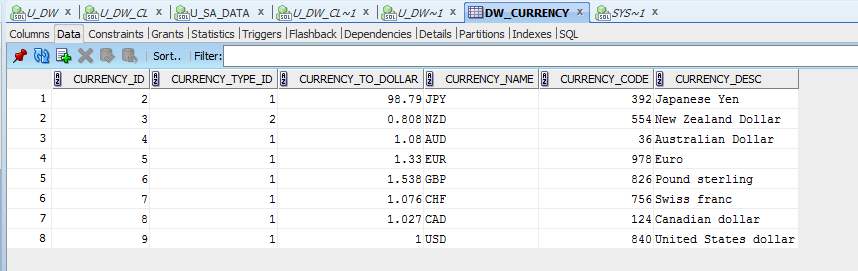
END IF;

END LOOP;

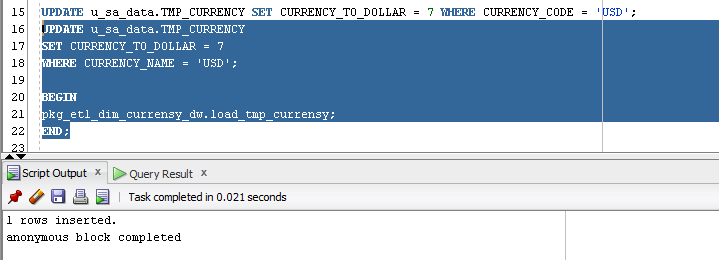
COMMIT;

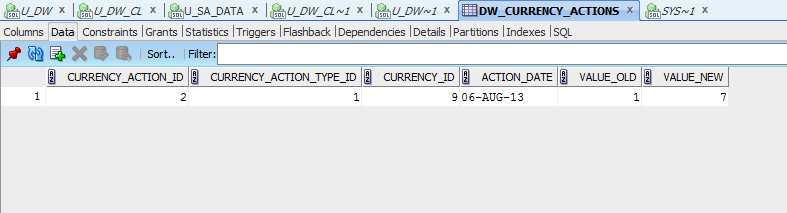
END load\_tmp\_currensy;

END pkg\_etl\_dim\_currensy\_dw;

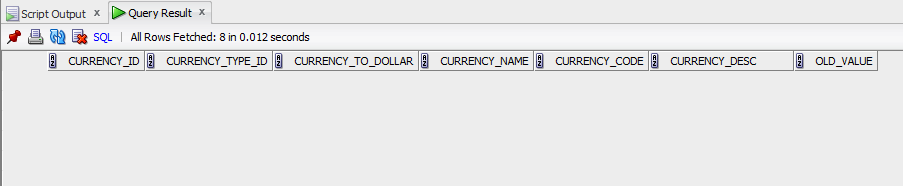


After updating the data in the source table and restart the package data is changed at stage-level.





After repeated run the package the data in the table are unchanged. Test results are shown below:



## Task 02: CREATE Monthly Reports Layouts

SELECT event\_dt

, tariff\_type

, tariff\_name

, SUM ( am\_sum )

, SUM ( count\_tr )

, SUM ( profit )

, SUM ( prg )

FROM ( SELECT event\_dt

, tariff\_type

, tariff\_name

, am\_sum

, count\_tr

, profit

, prg

FROM ( SELECT TRUNC ( TO\_DATE ( event\_dt

, 'dd-mon-yy' )

, 'month' )

event\_dt

, tariff\_type

, tariff\_name

, tariff\_payment\_sum

, SUM ( SUM ( payment\_sum / currency\_to\_dollar ) ) OVER (PARTITION BY tariff\_type) par\_sum

, ROUND ( SUM ( payment\_sum / currency\_to\_dollar )

, 2 )

am\_sum

, COUNT ( payment\_sum / currency\_to\_dollar ) count\_tr

FROM u\_dw\_ext\_references.tmp\_transactions\_info

WHERE tariff\_type IN ('Local Transfer', 'International Transfer')

AND tariff\_name IN ('Small Transfer', 'Temperate Transfer')

AND TO\_CHAR ( TRUNC ( event\_dt

, 'YYYY' )

, 'YYYY' ) IN ('2012')

GROUP BY (tariff\_type, tariff\_name, TRUNC ( TO\_DATE ( event\_dt

, 'dd-mon-yy' )

, 'month' ), tariff\_payment\_sum))

MODEL RETURN UPDATED ROWS

PARTITION BY ( tariff\_type, tariff\_name )

DIMENSION BY ( event\_dt )

MEASURES ( 0 prg, 0 profit, am\_sum, par\_sum, count\_tr, tariff\_payment\_sum )

RULES AUTOMATIC ORDER

( prg [event\_dt] = ROUND ( ( 100 \* am\_sum[CV ( event\_dt )] / par\_sum[CV ( event\_dt )] )

, 5 ),

profit [event\_dt] = ROUND ( ( count\_tr[CV ( event\_dt )] \* tariff\_payment\_sum[CV ( event\_dt )] / 100 )

, 2 ) )

ORDER BY 2

, 1

, 3)

GROUP BY ROLLUP ( ( event\_dt, tariff\_name ) )

, tariff\_type;

