EPAM Systems, RD Dep.

Advanced Refresh Scenarios

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
1.0	Initial status	Valeryia_Lupanava	02-DEC-2017		

Содержание

1. Производительность DM-слоя

Первоначально соберем статистику по схеме BL_DM, потом по схеме BL_3NF.

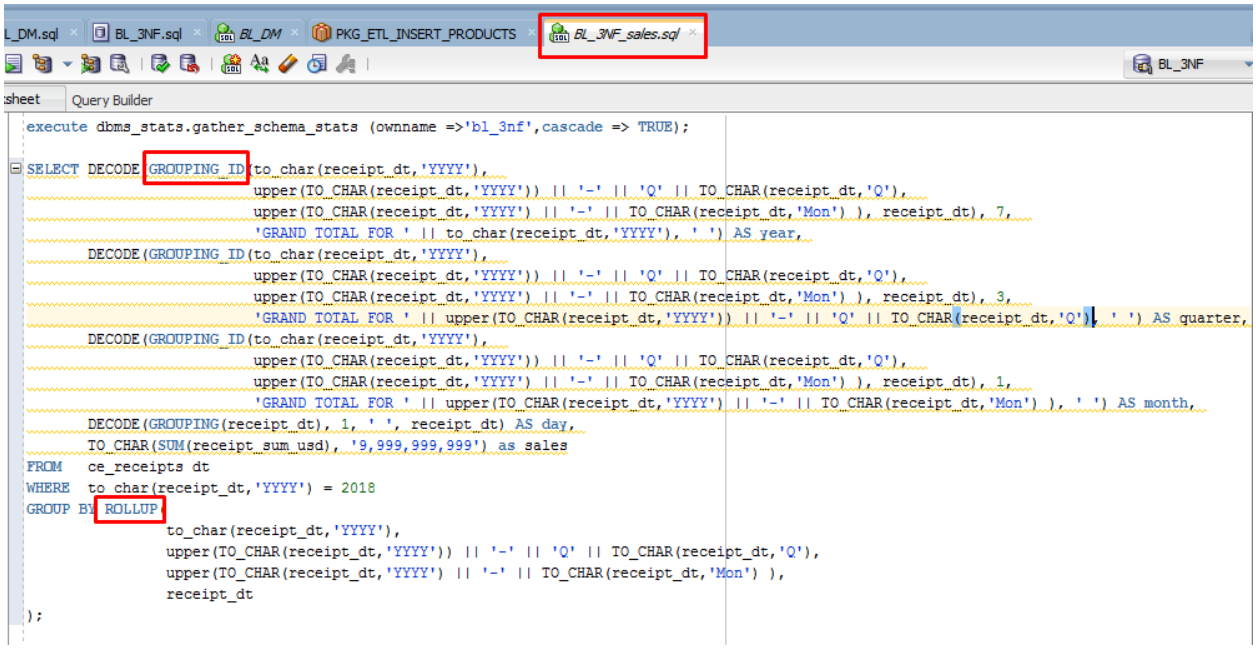
```
execute dbms_stats.gather_schema_stats (ownname =>'bl_dm',cascade => TRUE);

                PL/SQL procedure successfully completed.

execute dbms_stats.gather_schema_stats (ownname =>'bl_3nf',cascade => TRUE);

                PL/SQL procedure successfully completed.
```

1.1. 3NF-слой

- Выполним запрос с применение аналитических функций.
- Проанализируем продажи за 2018 год с группировкой по кварталу и месяцу и д
- Скрипт.

```
SELECT DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' ||
'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' ||
TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 7,
                          'GRAND TOTAL FOR ' ||
to_char(receipt_dt,'YYYY'), ' ') AS year,
       DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' ||
'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' ||
TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 3,
                          'GRAND TOTAL FOR ' ||
upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' ||
TO_CHAR(receipt_dt,'Q'), ' ') AS quarter,
       DECODE(GROUPING_ID(to_char(receipt_dt,'YYYY'),
                          upper(TO_CHAR(receipt_dt,'YYYY')) || '-' ||
'Q' || TO_CHAR(receipt_dt,'Q'),
                          upper(TO_CHAR(receipt_dt,'YYYY') || '-' ||
TO_CHAR(receipt_dt,'Mon') ), receipt_dt), 1,
                          'GRAND TOTAL FOR ' ||
upper(TO_CHAR(receipt_dt,'YYYY') || '-' || TO_CHAR(receipt_dt,'Mon')
), ' ') AS month,
       DECODE(GROUPING(receipt_dt), 1, ' ', receipt_dt) AS day,
       TO_CHAR(SUM(receipt_sum_usd), '9,999,999,999') as sales
FROM   ce_receipts dt
WHERE  to_char(receipt_dt,'YYYY') = 2018
GROUP BY ROLLUP(
               to_char(receipt_dt,'YYYY'),
               upper(TO_CHAR(receipt_dt,'YYYY')) || '-' || 'Q' ||
TO_CHAR(receipt_dt,'Q'),
               upper(TO_CHAR(receipt_dt,'YYYY') || '-' ||
TO_CHAR(receipt_dt,'Mon') ),
```

```
                    receipt_dt
);
```



- Результат.

	YEAR	QUARTER	MONTH	DAY	SALES
363				15-OCT-18	98,632,990
364				16-OCT-18	101,978,967
365				17-OCT-18	99,178,589
366				18-OCT-18	99,693,233
367				19-OCT-18	101,251,898
368				20-OCT-18	99,724,227
369				21-OCT-18	97,660,012
370				22-OCT-18	95,913,192
371				23-OCT-18	100,312,667
372				24-OCT-18	96,909,652
373				25-OCT-18	99,616,695
374				26-OCT-18	101,978,782
375				27-OCT-18	104,302,014
376				28-OCT-18	101,581,387
377				29-OCT-18	103,801,735
378				30-OCT-18	102,354,341
379				31-OCT-18	99,523,147
380			GRAND TOTAL FOR 2018-OCT		3,104,654,916
381		GRAND TOTAL FOR 2018-Q			9,238,938,174
382	GRAND TOTAL FOR 2018				###############
383					###############

All Rows Fetched: 383 in 2.394 seconds

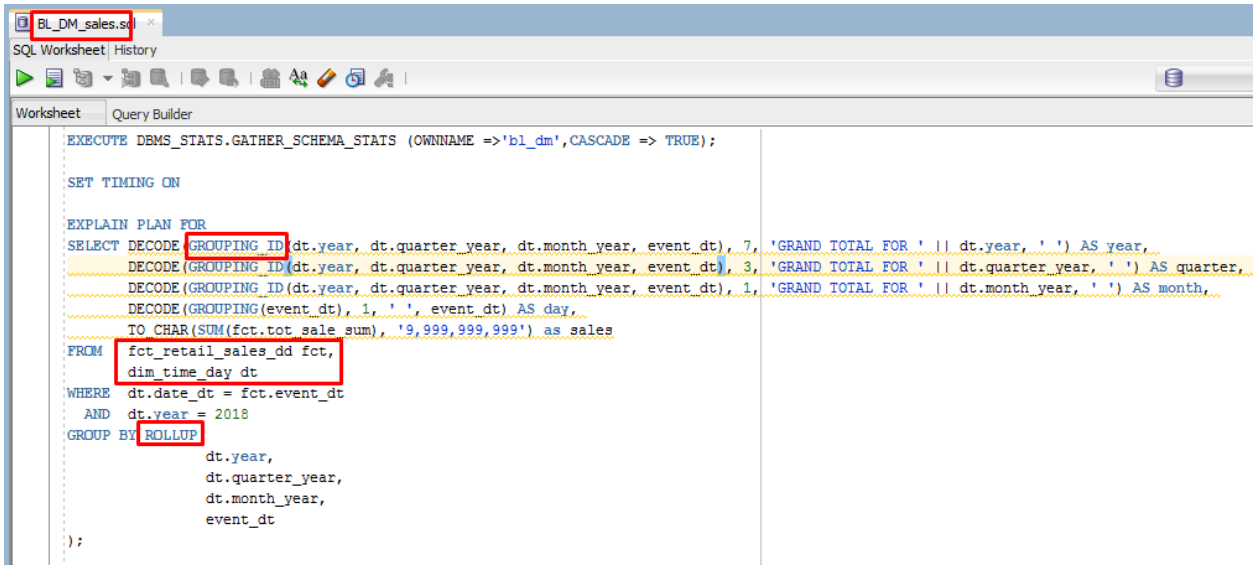
- Explain Plan.

```
PLAN_TABLE_OUTPUT
Plan hash value: 3141232678

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |              |  1001 | 13013 |  4716   (1)| 00:00:01 |
|   1 |  SORT GROUP BY ROLLUP|              |  1001 | 13013 |  4716   (1)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL  | CE_RECEIPTS | 20020 |   254K|  4715   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(TO_NUMBER(TO_CHAR(INTERNAL_FUNCTION("RECEIPT_DT"),'YYYY'))=201
              8)
```

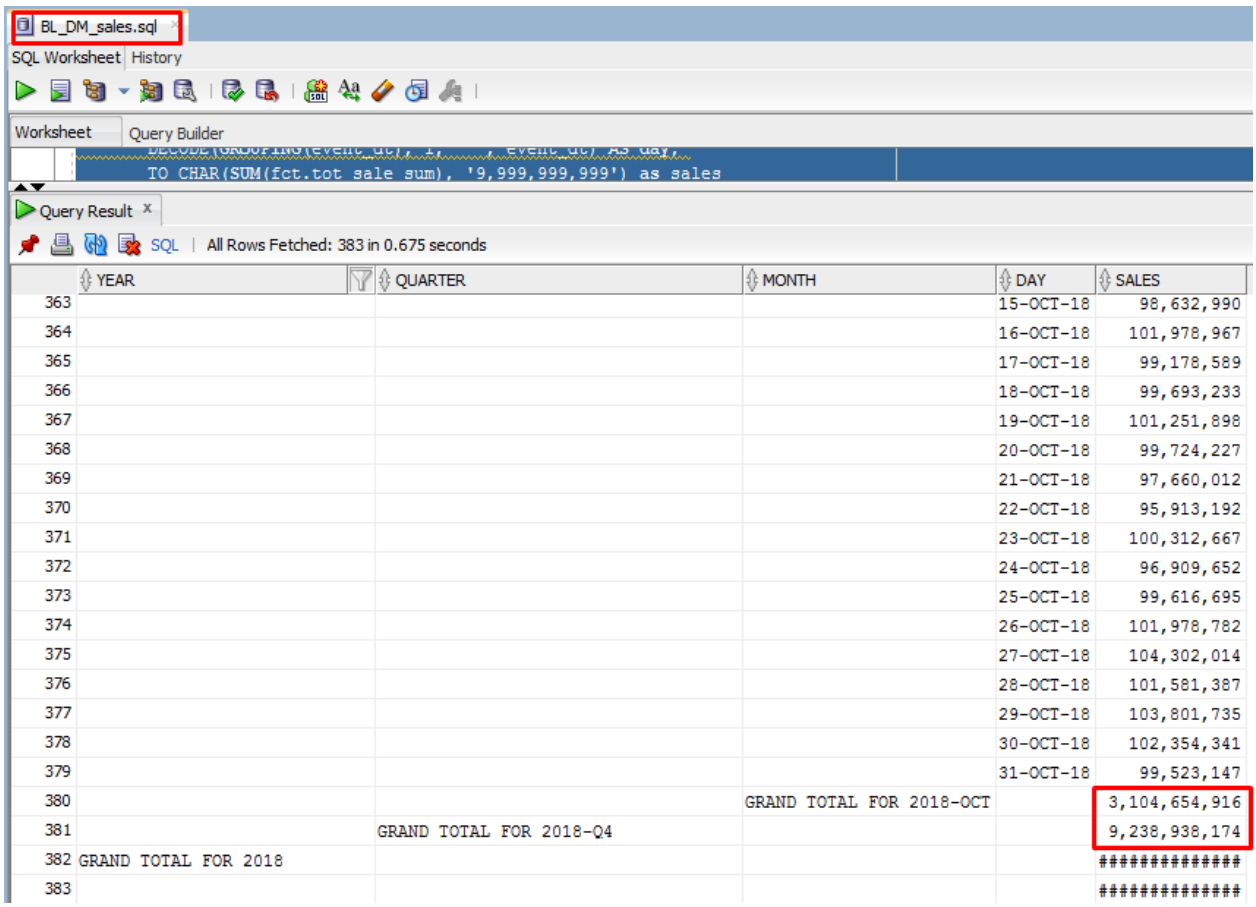
1.2. DM-слой

- Выполним аналогичный запрос на DM-слое.
- Скрипт.

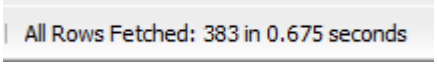
```
SELECT DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year,
event_dt), 7, 'GRAND TOTAL FOR ' || dt.year, ' ') AS year,
       DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year,
event_dt), 3, 'GRAND TOTAL FOR ' || dt.quarter_year, ' ') AS quarter,
       DECODE(GROUPING_ID(dt.year, dt.quarter_year, dt.month_year,
event_dt), 1, 'GRAND TOTAL FOR ' || dt.month_year, ' ') AS month,
       DECODE(GROUPING(event_dt), 1, ' ', event_dt) AS day,
       TO_CHAR(SUM(fct.tot_sale_sum), '9,999,999,999') as sales
FROM   fct_retail_sales_dd fct,
       dim_time_day dt
WHERE  dt.date_dt = fct.event_dt
  AND  dt.year = 2018
GROUP BY ROLLUP(
                dt.year,
                dt.quarter_year,
                dt.month_year,
                event_dt
);
```



- Результат.



	YEAR	QUARTER	MONTH	DAY	SALES
363				15-OCT-18	98,632,990
364				16-OCT-18	101,978,967
365				17-OCT-18	99,178,589
366				18-OCT-18	99,693,233
367				19-OCT-18	101,251,898
368				20-OCT-18	99,724,227
369				21-OCT-18	97,660,012
370				22-OCT-18	95,913,192
371				23-OCT-18	100,312,667
372				24-OCT-18	96,909,652
373				25-OCT-18	99,616,695
374				26-OCT-18	101,978,782
375				27-OCT-18	104,302,014
376				28-OCT-18	101,581,387
377				29-OCT-18	103,801,735
378				30-OCT-18	102,354,341
379				31-OCT-18	99,523,147
380			GRAND TOTAL FOR 2018-OCT		3,104,654,916
381		GRAND TOTAL FOR 2018-Q4			9,238,938,174
382	GRAND TOTAL FOR 2018				###############
383					###############



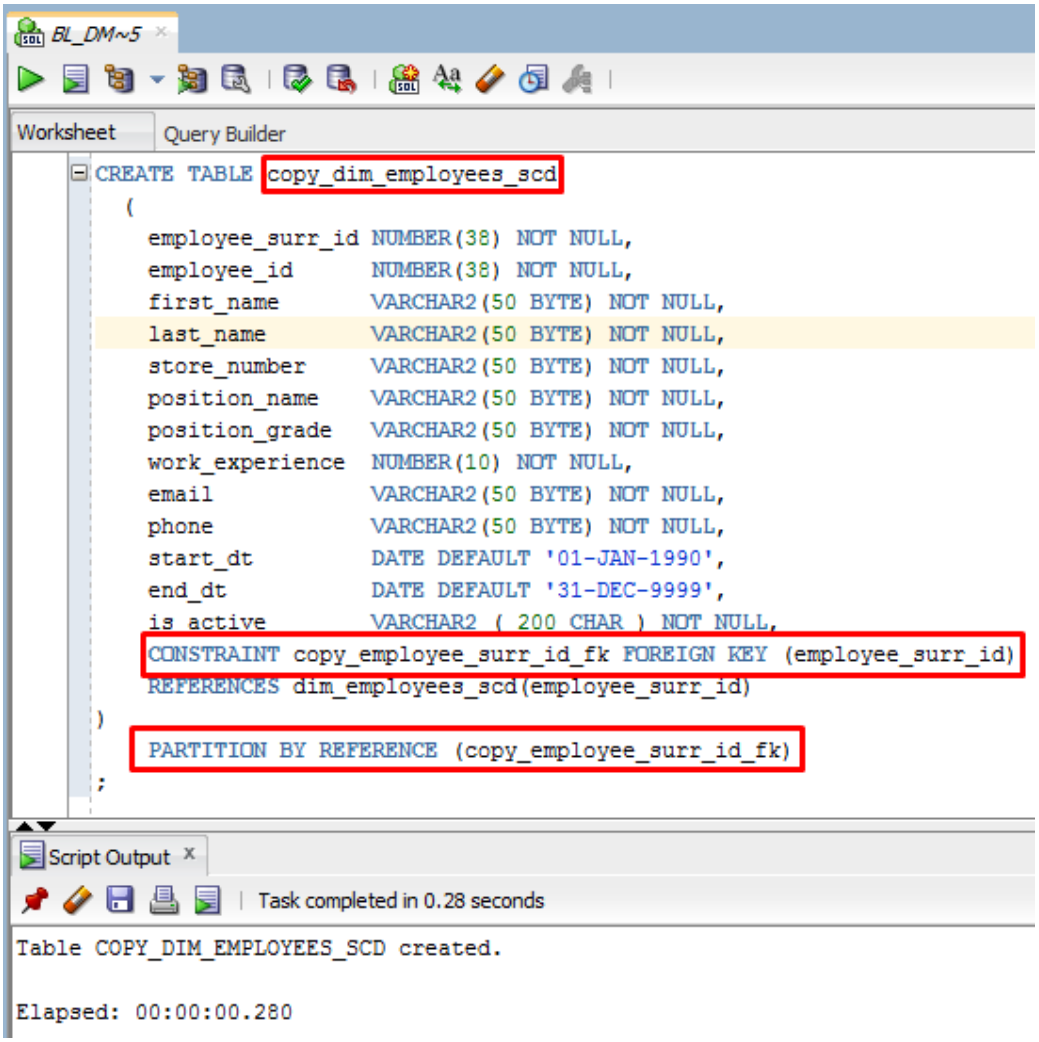
- Explain Plan.

```
PLAN_TABLE_OUTPUT
Plan hash value: 3413384702

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                     |   589K|    24M|       | 12849   (1)| 00:00:01 |       |       |
|   1 |  SORT GROUP BY ROLLUP         |                     |   589K|    24M|    29M| 12849   (1)| 00:00:01 |       |       |
|*  2 |   HASH JOIN                   |                     |   589K|    24M|       |  6378   (1)| 00:00:01 |       |       |
|   3 |    PART JOIN FILTER CREATE    | :BF0000             |   589K|    24M|       |  6378   (1)| 00:00:01 |       |       |
|*  4 |     TABLE ACCESS FULL         | DIM_TIME_DAY        |   365 | 10950 |       |    68   (0)| 00:00:01 |       |       |
|   5 |     PARTITION RANGE JOIN-FILTER|                    |  2001K|    24M|       |  6305   (1)| 00:00:01 |:BF0000|:BF0000|
|   6 |      TABLE ACCESS FULL        | FCT_RETAIL_SALES_DD |  2001K|    24M|       |  6305   (1)| 00:00:01 |:BF0000|:BF0000|
-----------------------------------------------------------------------------------------------------------------------------------------
```

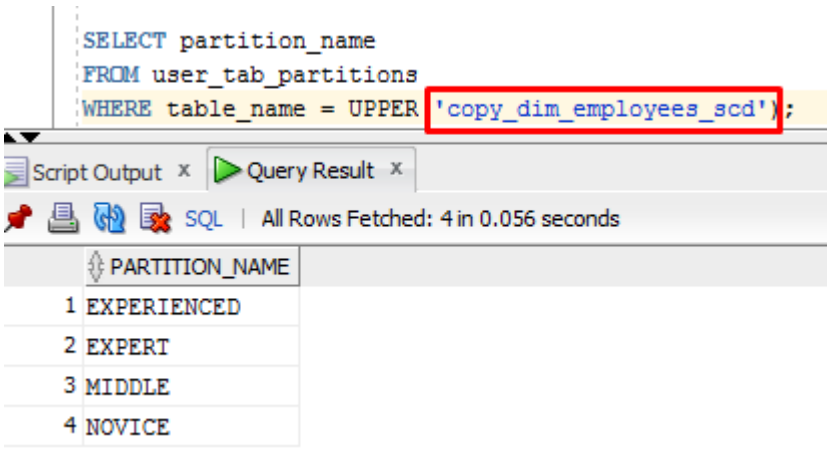
- Вывод.
- Несмотря на то, что параметр COST в 3NF гораздно ниже, запрос в 3NF выполнился за 2.39 секунды, а в DWH – за 0.68, что демонстрирует эффективность работы DWH.

2. Advanced loading

- В качестве родительской таблицы рассмотрим DIM_EMPLOYEES_SCD. Создадим дочернюю таблицу с внешним ключом на родительскую таблицу.

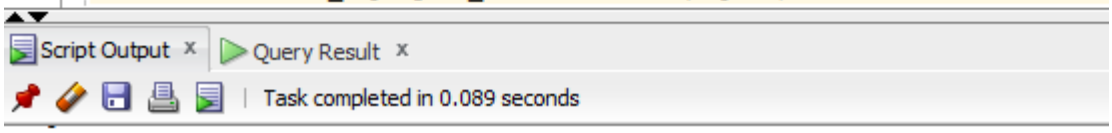


- Проверим, что партиции действительно создались.



- Создадим еще одну таблицу для PARTITION EXCHANGE с COPY_ DIM_EMPLOYEES_SCD, чтобы не менять данные в дименшене.

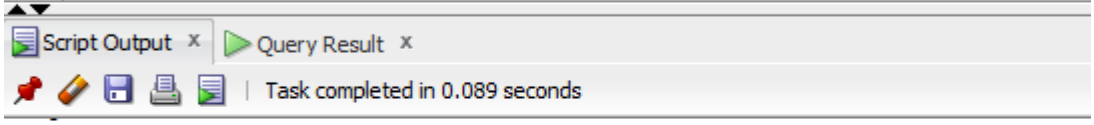
```
CREATE TABLE copy_dim_employees_scd_exch
  (
    employee_surr_id NUMBER(38) NOT NULL,
    employee_id      NUMBER(38) NOT NULL,
    first_name       VARCHAR2(50 BYTE) NOT NULL,
    last_name        VARCHAR2(50 BYTE) NOT NULL,
    store_number     VARCHAR2(50 BYTE) NOT NULL,
    position_name    VARCHAR2(50 BYTE) NOT NULL,
    position_grade   VARCHAR2(50 BYTE) NOT NULL,
    work_experience  NUMBER(10) NOT NULL,
    email            VARCHAR2(50 BYTE) NOT NULL,
    phone            VARCHAR2(50 BYTE) NOT NULL,
    start_dt         DATE DEFAULT '01-JAN-1990',
    end_dt           DATE DEFAULT '31-DEC-9999',
    is_active        VARCHAR2 ( 200 CHAR ) NOT NULL
);

--------------------------------------------------------------------------------
INSERT INTO copy_dim_employees_scd
SELECT * FROM dim_employees_scd PARTITION (expert);
```




```
Table COPY_DIM_EMPLOYEES_SCD_EXCH created.
```

- В первую созданную таблицу COPY_ DIM_EMPLOYEES_SCD с PARTITION BY REFERENCE вставим данные партиции EXPERT из дименшена.

```
--------------------------------------------------------------------------------
INSERT INTO copy_dim_employees_scd
SELECT * FROM dim_employees_scd PARTITION (expert);
```




```
Table COPY_DIM_EMPLOYEES_SCD_EXCH created.

Elapsed: 00:00:00.056
4,710 rows inserted.

Elapsed: 00:00:00.089
```

- Во вторую созданную таблицу COPY_DIM_EMPLOYEES_SCD_EXCH вставим данные партиции NOVICE из дименшена.

```
INSERT INTO copy_dim_employees_scd_exch
SELECT * FROM dim_employees_scd PARTITION (novice);
```

Script Output | Query Result | Task completed in 0.035 seconds

```
Elapsed: 00:00:00.091
Table COPY_DIM_EMPLOYEES_SCD altered.

Elapsed: 00:00:00.085
4 742 rows inserted
```

- Теперь осуществим PARTITION EXCHANGE между двумя вышеупомянутыми таблицами. Вполне естественно, требуется внешний ключ (обязательно в статусе ENABLED VALIDATED, аналогичный ключу в секционированной таблице), иначе поддержка ссылочного секционирования невозможна.

```
ALTER TABLE copy_dim_employees_scd EXCHANGE PARTITION expert WITH TABLE copy_dim_employees_scd_exch;
```

Script Output | Query Result | Task completed in 0.034 seconds

```
          ALTER TABLE EXCHANGE PARTITION statement have different FOREIGN KEY
          constraints.
*Action:  Ensure that the two tables do not have FOREIGN KEY constraints
          defined on any column or disable all FOREIGN KEY constraints
          on both tables. Then retry the operation.
Elapsed: 00:00:00.034
```

- Создадим внешний ключ в таблице COPY_ DIM_EMPLOYEES_SCD_EXCH по полю с SURR_ID на дименшен таблицу.

```
ALTER TABLE copy_dim_employees_scd_exch
ADD CONSTRAINT copy_dim_employees_scd_exch_fk FOREIGN KEY (employee_surr_id)
REFERENCES dim_employees_scd(employee_surr_id)
ENABLE VALIDATE PARALLEL 32;
```

Script Output | Query Result | Task completed in 0.091 seconds

```
          defined on any column or disable all FOREIGN KEY constraints
          on both tables. Then retry the operation.
Elapsed: 00:00:00.034
Table COPY_DIM_EMPLOYEES_SCD_EXCH altered.

Elapsed: 00:00:00.091
```

- Теперь осуществим PARTITION EXCHANGE между таблицами.

```
ALTER TABLE copy_dim_employees_scd EXCHANGE PARTITION novice WITH TABLE copy_dim_employees_scd_exch;
```

Script Output | Query Result | Task completed in 0.035 seconds

```
invalid rows and delete them.
Elapsed: 00:00:00.025
Table COPY_DIM_EMPLOYEES_SCD altered.

Elapsed: 00:00:00.035
```

- Проверим результат перемещения.

```
SELECT employee_surr_id, position_grade FROM copy_dim_employees_scd;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.006 seconds

	EMPLOYEE_SURR_ID	POSITION_GRADE
1	35510	novice
2	35511	novice
3	35513	novice
4	35516	novice

- Можно наблюдать, что таблица, которая изначально была заполнена значениями из партиции EXPERT, теперь содеожит значения для партиции NOVICE. Таблица, которую заполняли значениями партиции NOVICE и их которой перемещали данные, теперь пустая.

```
SELECT employee_surr_id, position_grade FROM copy_dim_employees_scd_exch;
```

ipt Output | Query Result | SQL | All Rows Fetched: 0 in 0.003 seconds

EMPLOYE...	POSITION...

- Проверим еще одно перемещение. Вставим в нашу пустую таблицу данные по партиции EXPERIENCED.

```
INSERT INTO copy_dim_employees_scd_exch
SELECT * FROM dim_employees_scd PARTITION (experienced);
```

Script Output | Query Result | Task completed in 0.045 seconds

```
*Action:   Retry the command with correct syntax.
Elapsed: 00:00:00.017
4,772 rows inserted.

Elapsed: 00:00:00.045
```

```
SELECT employee_surr_id, position_grade FROM copy_dim_employees_scd_exch;
```

Script Output | Query Result | SQL | Fetched 50 rows in 0.006 seconds

	EMPLOYEE_SURR_ID	POSITION_GRADE
1	29492	experienced
2	29493	experienced
3	29495	experienced
4	29499	experienced
5	29500	experienced
6	29501	experienced
7	29502	experienced
8	29506	experienced

- Теперь осуществим PARTITION EXCHANGE между таблицами.

```
ALTER TABLE copy_dim_employees_scd EXCHANGE PARTITION experienced WITH TABLE copy_dim_employees_scd_exch;
```

Script Output | Query Result | Task completed in 0.027 seconds

```
*Cause:    Table or index is not partitioned.  Invalid syntax.
*Action:   Retry the command with correct syntax.
Elapsed: 00:00:00.017
4,772 rows inserted.

Elapsed: 00:00:00.045
Table COPY_DIM_EMPLOYEES_SCD altered.

Elapsed: 00:00:00.027
```

- Результат PARTITION EXCHANGE.
- Наблюдаем, что таблица снова пустая, поскольку данные переместились.

```
SELECT employee_surr_id, position_grade FROM copy_dim_employees_scd_exch;
```

cript Output | Query Result

SQL | All Rows Fetched: 0 in 0.004 seconds

EMPLOYE...	POSITION...

- Вторая таблица дописалась новыми данными.

```
SELECT employee_surr_id, position_grade FROM copy_dim_employees_scd
WHERE position_grade = 'experienced';
```

cript Output | Query Result

SQL | Fetched 50 rows in 0.035 seconds

	EMPLOYEE_SURR_ID	POSITION_GRADE
1	29492	experienced
2	29493	experienced
3	29495	experienced
4	29499	experienced
5	29500	experienced
6	29501	experienced
7	29502	experienced