من أجل الـ Star schema يوجد أمثلة من Oracle من الممكن تضمينها أثناء التنصيب عن طريق خيار "with example" من الأمثلة الـ sales history وهو يشبه المثال الذي تحدثنا عنه الدرس الماضي .

في حال كان الخيار مُفعلاً ندخل كـ administration Sys ← Other Schema ← Sh وهو عبارة عن Stars schema

حيث يوجد فيه: مبيعات و Product و Chanel و Coast ... وهي عبارة عن Dimensions والـ Fact table لديه هو الـ Sales لأنه يحوي عدة foreign keys تتعلق بالـ Dimensions و 2 measures هما الـ amount والـ quantity .

وهو مثال مفيد جداً للاطلاع عليه قبل حل وظيفة star schema المطلوبة فهي فقط 3 Dimensions مع التاريخ يعني يتبقى فقط 2 Dimensions وتحتاج إلى بعض التفكير لمعرفة الـ Dimensions .

Partitioning

لماذا نحتاج لـ Partition في الـ warehouse ؟

عندما نقول warehouse هذا يعني أنه لدينا historical data أي حجم Data كبير جداً , وكما رأينا عند الانتقال من Relational إلى Dimensional فالـ Dimension Tables لن تحوي عدد كبير من الـ Records فهي جداول ترميز مثل اسماء الموظفين والـ data الخاصة بالموظفين والخاصة بالمنتجات فلن يكون هناك مشكلة مع الـ Data الكبيرة بالنسبة لـ Dimension Table بل ستنتج المشكلة في الـ Fact Table حيث أنه سيحوي عدد كبير جداً من الـ records بغض النظر عن نوع الـ Dimensions (Star , Snowflake) وهي نتيجة ترحيل السنوات السابقة.
فإذا أردنا إجراء query على table يحوي مئات ملايين الـ rows سيكون هناك معالجة كبيرة ، من هنا جاءت حاجة تجزيء الـ Fact Table فيصبح لدينا n Fact Table في جدول واحد وهذه نمذجة جديدة تسمى الـ Partition .

ماهو الـ Partition ؟

Physically يبقى الـ Fact Table عبارة عن table واحد لكنه يكون مقطع Logically إلى Partitions ويعلم كل record مكانه ضمن هذا الـ Partitions.

أنواع الـ Partition :

1. Range 2. Hash 3. List

- **Range Partition :**

وهو الأكثر أستخداماً ، يعتمد على حقل الـDate أي نحدد الـ key Partition بقيمة تاريخ ضمن هذا الجدول ويكون تقطيع الـ Fact Table على أساس قيم تاريخية.

مثلاً لدي table فيه مبيعات لـ 50 سنة و أريد التقطيع وفق السنة فيكون لدي 50 Partition .

إذاً عندما نريد تقطيع Table وفق Date أستخدم الـ Range

- **Hash :**

وهو الأقل استخداماً ، لا يعتمد على مفتاح صريح للتقطيع كما في الـ Range وإنما يجري التقطيع بقيم متساوية **فمثلاً** نريد 10 Partition فنجد بعد التقطيع أنهم متوازنين وهذا غير موجود في الـ Range لأن مبيعات سنة 2020 مثلاً قد تكون أضعاف مضاعفة لمبيعات 2010 فنقوم بالـ Range بوضع Data تخص السنة الواحدة في كل Partition فلا يكون هناك توازن بينهم.

إذاً عند الحاجة للتوازن نستخدم الـHash فهو الوحيد الذي يضمن هذه الصفة

- **List:**

يضمن تقسيم الـ Fact Table وفق المناطق ففي هذه الحالة يوجد لدينا قيمة نُقسِم وفقها

مثلاً تقسيم الجدول حسب المحافظات فتصبح دمشق ممثلة بجزء وحلب بجزء ....

❖ Composite Partitioning:

يمكن ان يبقى كل partition بحد ذاته ذو حجم كبير جداً فنقوم بعملية تجزئة له

أشهر أنواع الـcomposite partitioning

1- Range-hash: نقسم وفق ranges ثم كل range نقسمه وفق hash

2- Range-list: نقسم وفق ranges ثم كل range نقسمه وفق list

➢ مثال على التقسيم بطريقة الـRange:

```
CREATE TABLE sales_range (
sales_id                NUMBER(5) CONSTRAINT sales_id_pk PRIMARY KEY,
salesman_id             NUMBER(5),
sales_amount            NUMBER(10),
sales_date              DATE)
PARTITION BY RANGE (sales_date)
(PARTITION sales_jan2011 VALUES LESS THAN(TO_DATE('01/02/2011','DD/MM/YYYY')),
 PARTITION sales_feb2011 VALUES LESS THAN(TO_DATE('01/03/2011','DD/MM/YYYY')),
 PARTITION sales_mar2011 VALUES LESS THAN(TO_DATE('01/04/2011','DD/MM/YYYY')),
 PARTITION sales_apr2011  VALUES LESS THAN(TO_DATE('01/05/2011','DD/MM/YYYY'))
);
```

قمنا بتقسيم الجدول السابق بالطريقة الأولى Range عن طريق partition key هو الـ sales_date
ثم قمنا بتحديد كل قسم (تحديد اسمه وقيمته) فحددنا في المثال 4 أقسام .

ننفذ تعليمات insert التالية لنرى كيف تتم عملية التقسيم :

```
INSERT INTO sales_range VALUES (1,1, 10, TO_DATE('02/01/2011', 'DD/MM/YYYY'));
INSERT INTO sales_range VALUES (2,2, 10, TO_DATE('20/01/2011', 'DD/MM/YYYY'));
INSERT INTO sales_range VALUES (3,1, 10, TO_DATE('12/12/2010', 'DD/MM/YYYY'));
INSERT INTO sales_range VALUES (4,2, 20, TO_DATE('01/02/2011', 'DD/MM/YYYY'));
```

نلاحظ حسب التاريخ فإنه اول ثلاث قيم ستكون ضمن الـ partition الأول والقيمة الأخيرة ستكون ضمن الثاني.

حيث أن كل السنين السابقة لـ 2011 ستكون موجودة ضمن الجزء الأول بسبب وجود less than ويمكننا بعد فترة أن نقوم بعمل drop للـ partition وإنشاء partition جديد دون ان تتأثر الـ data مطلقاً.

عند الإظهار لمساعدة الـ optimizer في البحث نحدد له criteria بـ where متبوعة بقيمة محددة للـ sales_date

```
SELECT * FROM sales_range WHERE sales_date = to_date('02-01-2001', 'DD-MM-YYYY');
```

90% من الـ warehouses تتضمن date لذلك أفضل طريقة أن يكون هو الـ partition key.

يمكننا تنفيذ query على partition محدد ولكنها غير مستخدمة :

```
SELECT * FROM sales_range PARTITION (sales_jan2011);
```

سنلاحظ أنه ظهر لدينا الـ 3 rows الموجودة ضمن الـ partition الأول.
إذا قمنا بإدخال data خارج المجال الذي قسمناه سيظهر لدينا error ، حيث أننا أنشأنا 4 partition فإذا قمنا بعملية insert يكون فيها التاريخ 01/08/2011 مثلاً هنا سيظهر خطأ ولتفادي هذا الخطأ نقوم بإضافة partition جديد بحيث أن أصغر قيمة يحتويها هي أعلى قيمة للـ partition السابق و قيمته الكبرى هي max value :

```
ALTER TABLE sales_range
ADD PARTITION sales_after5_2011 VALUES LESS THAN (MAXVALUE);
```

بعد التنفيذ يصبح لدينا partition الأول مفتوح لكل القيم الأصغر والأخير مفتوح لكل القيم الأكبر وقد يصبح huge مع مرور الوقت لأن كل البيانات الحديثة تضاف الى هذا الجزء وهذا أمر خاطئ لذلك عند وجود بيانات كبيرة تصعب المعالجة فيكون الحل هو إعادة التجزئة كل فترة من الزمن (أي أقوم ب partitioning من جديد أي إعادة ترتيب) .

الـ partitioning لا يقوم بعمل ترتيب فيزيائي للـ data بل هو فقط يقوم بتحزيمها بأن يقول هذا الـ row موجود في هذا المكان دون أن يغير مكانه.

بعد أن قمنا بإضافة الـ partition الجديد يمكننا القيام بعملية insert كالتالي دون ان تفشل:

```
INSERT INTO sales_range VALUES (6, 60, TO_DATE('02/06/2011', 'DD/MM/YYYY'));
INSERT INTO sales_range VALUES (8, 80, TO_DATE('08/08/2011', 'DD/MM/YYYY'));
```

ملاحظة : لإضافة partition جديد تتم الإضافة حصراً بعد آخر partition أي لا يمكن الإضافة ضمن مجال مستخدم.

في التعليمة التالية نقوم بتعديل الـ partition key فنجد أن هذا التعديل سوف يفشل لأنه partition key لذلك في حال أردنا تعديله فالحل أن نقوم بعملية حذف للسطر بأكمله وإعادة اضافته من جديد بالقيمة الجديدة :

```
UPDATE sales_range
SET sales_date = TO_DATE('02/01/2011', 'DD/MM/YYYY')
WHERE sales_date = TO_DATE('08/08/2011', 'DD/MM/YYYY');
```

لإظهار execution plan وذلك لمعرفة فيما إذا استطاع الـoptimizer بإحضار الـdata من الـ partition أم لا :

```
EXPLAIN PLAN FOR
SELECT * FROM sales_range
WHERE sales_date = to_date('02-01-2001', 'DD-MM-YYYY');
SELECT * FROM TABLE(dbms_xplan.display);
```

يظهر لنا partition range وحقلين pstar =1، pstop=1 فإن لم تكن أحدهما 1 فهو لم يقم باستخدام الـ partition

➢ مثال عن الـhash:

```
CREATE TABLE sales_hash (
salesman_id          NUMBER(5),
salesman_name        VARCHAR2(30),
sales_amount         NUMBER(10),
week_no              NUMBER(2))
PARTITION BY HASH(salesman_id)
PARTITIONS 4;
```

بسبب عدم وجود date كانت أفضل طريقة هي الـhash حسب salesman_id وبأربع اقسام ولا نستطيع تسيمتهم.

➢ مثال عن الـlist:

```
CREATE TABLE sales_list (
salesman_id     NUMBER(5),
sales_state     VARCHAR2(20),
sales_amount    NUMBER(10),
sales_date      DATE)
PARTITION BY LIST(sales_state)
(PARTITION sales_west VALUES('California', 'Hawaii'),
 PARTITION sales_east VALUES('New York', 'Virginia', 'Florida'),
 PARTITION sales_central VALUES('Texas', 'Illinois'),
 PARTITION sales_other VALUES(DEFAULT) );
```

قسمنا الجدول على أساس مناطق sales_state وحددنا جميع الـ Partitions وأيضاً لدينا default partition نضع فيه القيم التي من خارج مجال التقطيع كي لا يظهر لدينا error .
ففي حال الاضافة وكان الـ 'California'=salse_state تتم الإضافة على أول partition أما في حال sales_state='soso' فسيتم الإضافة إلى آخر partition.

➢ مثال عن الـrange-hash:

```
CREATE TABLE sales_range_hash(
s_productid     NUMBER(8),
s_saledate      DATE,
s_custid        NUMBER(4),
s_totalprice    NUMBER(6) )
PARTITION BY RANGE (s_saledate)
SUBPARTITION BY HASH (s_productid)
SUBPARTITION TEMPLATE(
SUBPARTITION sp1 TABLESPACE tbs1,
SUBPARTITION sp2 TABLESPACE tbs2,
SUBPARTITION sp3 TABLESPACE tbs3,
SUBPARTITION sp4 TABLESPACE tbs4,
SUBPARTITION sp5 TABLESPACE tbs5,
SUBPARTITION sp6 TABLESPACE tbs6,
SUBPARTITION sp7 TABLESPACE tbs7,
SUBPARTITION sp8 TABLESPACE tbs8)
(PARTITION sal11q1 VALUES LESS THAN (TO_DATE('01-APR-2011', 'DD-MON-YYYY')),
 PARTITION sal11q2 VALUES LESS THAN (TO_DATE('01-JUL-2011', 'DD-MON-YYYY')),
 PARTITION sal11q3 VALUES LESS THAN (TO_DATE('01-OCT-2011', 'DD-MON-YYYY')),
 PARTITION sal11q4 VALUES LESS THAN (TO_DATE('01-JAN-2011', 'DD-MON-YYYY'))
);
```

لدينا اولاً partition by range على أساس s_saledate والـ partitions الخاصة به و هم الـ 4 في الأسفل وكل واحد منهم مقسم by hash على أساس الـ s_proudctid الى 8 اقسام هذا يعني ان الـ data الموجودة في partition محدد يمكن ان تكون مقسمة في 8 tablespace لأن كل واحد من الـ sub partitions مخزن في table space.

التخزين يتم physically على tablespace واحد لكن logically يكون مقسم الى 4 اقسام جزئية
كل قسم جزئي مقسم إلى 8 اقسام كل قسم جزئي منهم في table space .

➢ مثال عن الrange-list:

```
CREATE TABLE quarterly_regional_sales  (
deptno          NUMBER(4),
item_no         VARCHAR2(20),
txn_date        DATE,
txn_amount      NUMBER(6,2),
state           VARCHAR2(2)  )
PARTITION BY RANGE (txn_date)
SUBPARTITION BY LIST (state)
SUBPARTITION TEMPLATE(
SUBPARTITION northwest VALUES ('OR', 'WA') TABLESPACE ts1,
SUBPARTITION southwest VALUES ('AZ', 'UT', 'NM') TABLESPACE ts2,
SUBPARTITION northeast VALUES ('NY', 'VM', 'NJ') TABLESPACE ts3,
SUBPARTITION southeast VALUES ('FL', 'GA') TABLESPACE ts4,
SUBPARTITION northcentral VALUES ('SD', 'WI') TABLESPACE ts5,
SUBPARTITION southcentral VALUES ('NM', 'TX') TABLESPACE ts6)
(
PARTITION q1_2011 VALUES LESS THAN(TO_DATE('1-APR-2011','DD-MON-YYYY')),
PARTITION q2_2011 VALUES LESS THAN(TO_DATE('1-JUL-2011','DD-MON-YYYY')),
PARTITION q3_2011 VALUES LESS THAN(TO_DATE('1-OCT-2011','DD-MON-YYYY')),
PARTITION q4_2011 VALUES LESS THAN(TO_DATE('1-JAN-2011','DD-MON-YYYY'))
);
```

انتهت المحاضرة

Written by :
Aisha Awaty
Wordpress and preparation :
Anas Alazmeh
Reviewed by :
Mouayyad Taja