تعرفنا سابقاً على Model تمثل بـ Relational Database وهي ERD وتعتمد العلاقات One-To-Many , Many-To-Many سنتعرف في هذا المحاضرة على الـ Cube وأبعاده فهو عبارة عن Multi Dimensions Model وله نوعين :

1. Star Schema : لا تأخذ بعين الاعتبار الـ Normalization وهو الأكثر استخداماً .
2. Snowflake Schema : تأخذ الـ Normalization بعين الاعتبار .

تذكرة : تعني الـ Normalization تنظيم الأعمدة والجدوال وتبسيط التصميم في قاعدة المعطيات .

كلا النوعين السابقين يعتمد بشكل أساسي على الـ Dimension Table والـ Fact Table :

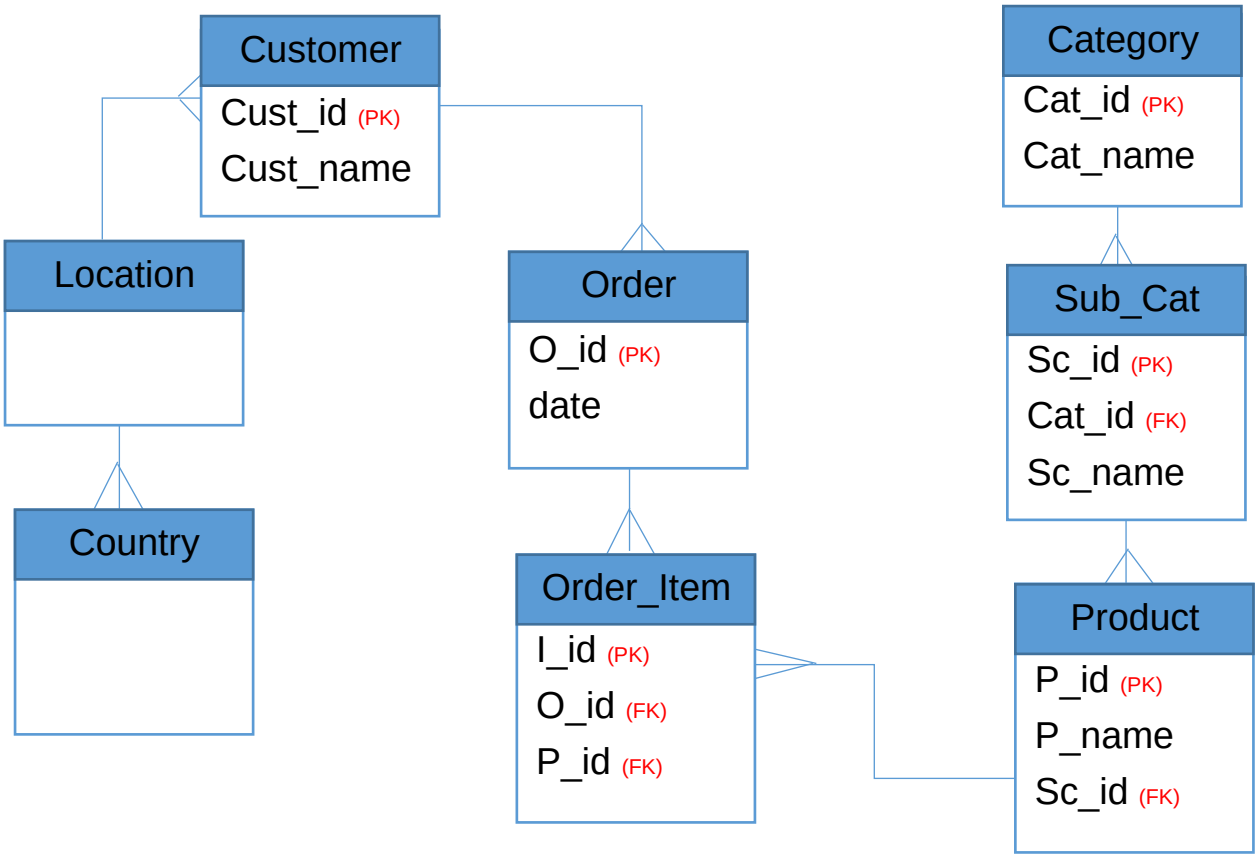
- Dimension Table: يعين الأبعاد ويهم فيه الـ Primary key والـ Hierarchy التي تسمح بعمليتي Break down وRollup .
- Fact Table : يهمني الـ foreign key والـ measure .

الـ Dimension Model هو Entity Relationship Model ولكن بـ أداء أفضل عن طريق وجود بعض التغييرات البيسطة في المفهوم .

الانتقال من ERD Model إلى Dimensions:

- تقريباً 70% مما سنقوم بفعله يحدده الزبون , أي المعلومات الإحصائية التي يريد أخذها من الـ DB بالتالي كي نقوم بعملية الانتقال يجب مراعاة متطلباته .
- بعد الانتقال نقوم بعملية ترحيل المعلومات كل ستة أشهر أو سنة .
- سنأخذ مثال لفهم الانتقال.

مثال : لنحول المخطط التالي إلى Dimension Model



نعتمد في عملية التحويل من Relational إلى Dimension بنسبة 30% إلى 40% على أفكار Technical ومن 60% إلى 70% على متطلبات الزبون كتقييس وأبعاد .

مثلاً الزبون يهمه أن يعرف كمية مبيعاته ضمن المنتج ومن هو الـ Customer وإلى أي دولة وما هي الـ Category بالتالي قام بتحديد ما يريده فنقوم بعملية التحويل بمراعات متطلباته.

كي نحقق التحويل يجب أن نفكر ما هي الـ Fact Table مع الـ measures و ما هي الأبعاد الأساسية Dimensions فنستطيع حينها تحديد الجداول ومحتوى كلٍّ منها .

- الـ Fact Table : (هو الـ Root والقلب الأساسي) ينشأ من الجداول التي يجري عليها أكثر Transaction في الـ DB وفي مثالنا هم Order والـ Order_Item ويمكن دمجهما في Fact Table واحد أو اثنين مختلفين ويعود هذا الشيء حسب ما يريده الزبون من متطلبات ودقة بالمعلومة.

السطر في الـ Multi Dimension تقابله n Row في الـ Relational (يقابله سطر أو أكثر من سطر) ومن المستحيل أن يوجد سطر وفي الـ MD ليس له مقابل في الـ Relational .

نستخدم في الـ Fact Table الـ measures وهي aggregation (Count , Avg , Sum) أي قيم تجميعية.

- الأبعاد Dimensions: جداول تنتج عن الـ Tables المرتبطة ارتباط وثيق ومباشر بالـ Transaction tables والتي هي في مثالنا Order و Order_Item فتُشكل الجداول المرتبطة بهما الـ Dimensions ، في هذا المثال تكون الأبعاد هي: Product , Customer وهناك بُعد (موجود دائماً بشكل تقريبي) مهم جداً وهو الـ Date .

مثلاً : أريد معرفة كمية المبيعات كـ عدد و كـ total price (هاتين القيمتين هما measures).

فيكون لدي Row فيه الـ Total Price هو 1,000,000 هل هذا المبلغ تحقق من بيعة واحدة أو أكثر من بيعة ؟ هل من Customer واحد أم أكثر ؟ ؟هذا ما تحدده الأبعاد .

ماهي الـ hierarchy ؟

معلومات تعبر عن هرمية التركيب وتأتي من الجداول المرتبطة بجداول الـ Dimensions .

مثلاً أريد معرفة مبيعات هذا المنتج لكن أريد أن اعرف هذا المنتج من أي Subcategory ومن أي Category

Category → SubCategory → Product

ألبسة رجالي /نسائي جواكيت

تذكرة: الفرق الأساسي بين الـ Star Schema والـ Snowflake Schema هو أن الأولى لا تطبق مبدأ الـ normalization.

Star Schema :

نقوم بتحويل المثال السابق إلى Dimension Model من نوع Star Schema .

1- Fact Table : نسميه Fact Order وينشأ من الجداول Order , Order_Item

والـ measures هنا ممكن أن يكون quantity والـ total price حيث نريد قياس كمية المبيعات والسعر الإجمالي.

2- Dimension Table : Product , Customer , Date وتكون علاقة الـ Dimensions مع الـ Fact Table علاقة One-To-Many بحيث يوجد في الـ Fact Table مجموعة foreign key تعبر عن الـ Dimensions .

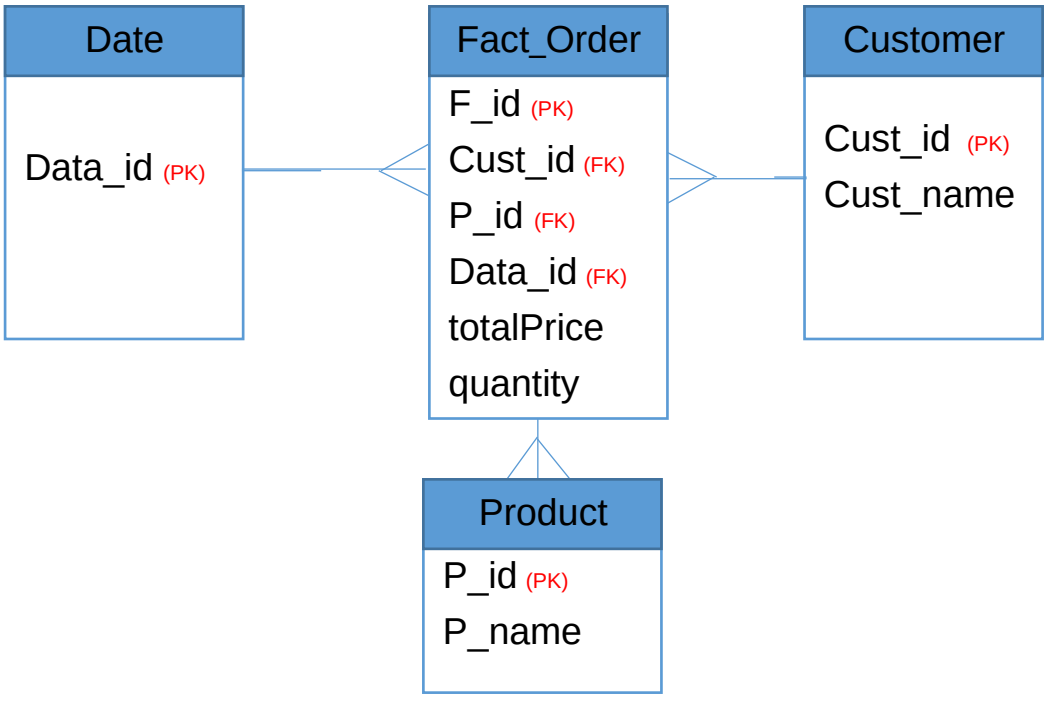
الـ row في الـ Customer

1	أحمد
2	سمير

وفي الـ Product

1	p1
2	p2

وفي الـ date لدي أيام.



ويكون مثلاً شكل السطر في الـ Fact_Order :

Cust_id	P_id	Date	Total price	Quantity
1	5	2017-1-1	100	50

هذا يعني أن الزبون 1 اشترى المنتج 5 في اليوم 2017-1-1 و كان الـtotal price 100 والكمية 50
وهذا الـ row ممكن أن يكون قد تشكل نتيجة 50 row في الـ ERD الأساسي أي هناك 50 Order أو مثلاً نتيجة 3 Order وكل Order فيه quantity مختلفة .
أصبح هذا الـ row جزء من الـ Cube بحيث لدي بُعد للـ Product والـ Date والـ Customer ما يعني أن أي نقطة من هذا الفضاء هي عبارة عن Row ولهذه النقطة عدة أبعاد في مثالنا .
من المستحيل أن يكون هناك record في الـ Fact_Order ليس لديه أب في ERD لأن عملية ترحيل الـ Data من الـ ERD إلى الـ MD يقوم بها ETL (Extract Transformation Load) وهي أداة جاهزة .
لا ننصح بزيادة الـ Dimensions لكي لا نصعب الأمور فإذا كانت الـ Dimensions كثيرة نبني مكعبين يحوي كل منهما 3 – 4 Dimensions وندمج بينهما بالـ Tool .
عند التحدث عن Cube لا تكون الـ query مثلاً select from table وإنما select from cube وهي لغة query أخرى تسمى MDX (Multidimensional Expressions) لن نتعلمها في منهاجنا بل سنتعلم الـ structure فقط .
نلاحظ أن شكل الجدوال وترابطها يشبه شكل النجمة.

Star Schema and Hierarchy :

نلاحظ أن الـ Product له أب وهو Sub category والذي له أب Category .
نفس الأمر بالنسبة للـ Customer له أب Location وله أب Country .
هنا نطرح سؤال هل نستخدم الـ Snowflake أم الـ Star ؟
في الـ Star Schema في الـ table نفسه نضع الـ Cat_id و الـ Cat_name و الـ Sc_id والـ Sc_name وبهذه الخطوة ناقضنا مبدأ normalization لأنه أصبح لدينا في كل منتج من هو أباه ومن هو جده وهكذا... بالتالي يوجد هناك redundant (زيادة) على مستوى الـ row .
فيصبح شكل الـ product

Product
P_id
P_name
Cat_id
Cat_name
Sc_id
Sc_name

فمثلاً : لدينا محل يبيع منتجات متنوعة بأقسام متنوعة لمدة 20 سنة وهنالك قسم يحوي أدوات منزلية وفيه بعض الأدوات الكهربائية لكنها تندرج تحت Sub_Cat "منزلية" يكون الـ rows في product يحمل القيم:

P_id	P_name	Sc_id	Sc_name	Cat_id	Cat_name
111	خلاط	11	منزلية	1	أدوات
112	مكسر	11	منزلية	1	أدوات

نلاحظ أنه :

في نموذج ERD كانت Sub_Cat , Category كلٍّ منها في جدول لوحده (ونفس الأمر بالنسبة لـ customer) لذلك لكل منهم record لوحده أما في الـ Dimensional يصبحون عبارة عن حقول في جدول product.

قبل ترحيل الـ data قام الزبون بإضافة Sub_Cat أبوها "أدوات" واسمها "كهربائية" فيصبح لدينا new record في جدول Sub_Category هو <12 , كهربائية> ثم قام بجعل المنتج "خلاط" ينتمي إلى Sub_Cat "كهربائية" .
نفترض أن هذا التغيير طرأ في 2010 فإن مبيعات منتج " خلاط " قبل 2010 مدرج في Sub_Cat " منزلية " وبعد 2010 أصبحت مدرجة ضمن " كهربائية" .

كيف سينعكس هذا الكلام عند نقله الى Dimensional في حال القيام بتجميع المنتجات وفق P_id لينتج لدينا total_price والـ quantity ونلاحظ أن P_id نفسه في النموذجين ؟ ؟

عندما نقوم بترحيل الداتا سيظهر لدينا ضمن ETL خطأ بإن هذا الـ record قيمته

111	خلاط	11	منزلية	1	أدوات

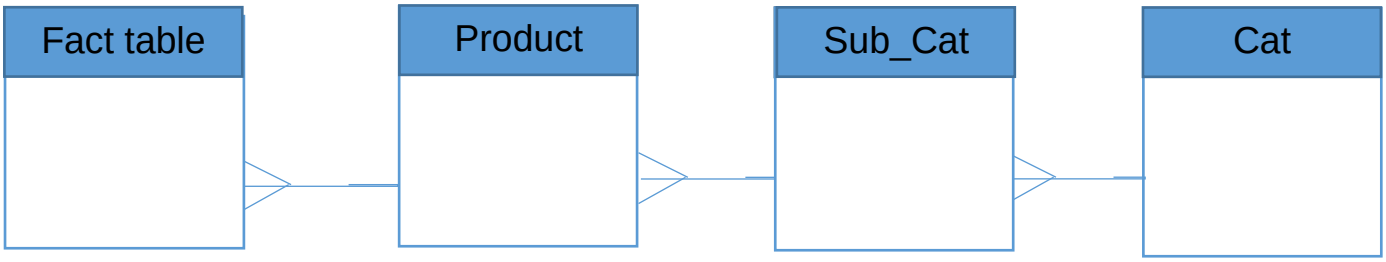
سيتضارب مع الـ record الجديد

111	خلاط	12	كهربائية	1	أدوات

يكون الحل بجهل P_id هو عبارة عن info وتوليد PK عند كل insert في Dimensional ونربطه مع Fact_table فإذا أراد معرفة المبيعات من الخلاط ستختلف قبل 2010 وبعدها
ضمنياً نحن نعلم أن كلا الـ records السابقين هما لنفس المنتج ولكن كما قلنا سابقاً نحن نقوم بتجميع المنتجات أي اننا سنراكم فوق هذا الـ record ، وعندما نصل لمرحلة مشابهة للسابقة أي أننا لا نريد أن نراكم فوق الـ record القديم فنضيف عمود يدل ان الـ record القديم لم يعد active وأصبح الجديد هو الفعال ويمكن ان أضيف قيمة أخرى تدل على التاريخ الذي أصبح فيه هذا الـ record inactive.
ونقوم بهذا الأمر عن طريق tools للـ ETL في Microsoft و Oracle (وذلك لصعوبة تنفيذ الأمر) وهنا خالفنا مبدأ الـ normalization وحافظنا على الـ Star Schema .
من الصعب إعادة عملية الترحيل لسنوات سابقة في قاعدة معطيات ضخمة لأنه مكلف زمنياً للترحيل الواحد (والذي نقوم به كل سنة تقريباً) فمن الممكن أن يحتاج لـ 3 – 4 أيام .

Snowflake Schema :

نمثل الآباء بجداول منفصلة فيكون لدينا عدة مستويات :



الشكل يصبح مثل قطعة ثلج متراكمة فوق بعضها البعض أي انه لم يعد مثل الـ Star بويوجد core وهو الـ fact table وعدة أبعاد بل أصبح كل بعد لديه ارتباطات أخرى (نفس الشيء بالنسبة للـ customer).

ملاحظة :

معظم المستخدمين تميل للـ Star Schema من أجل الـ performance بالرغم من سلبية وجود redundant في الـ data ويعود ذلك إلى أن حجم التخزين الكبير جداً في الـ Relational سيتحول لحجم ضئيل في الـ Star وفي هذه الحالة لن يؤثر الـ Redundant على سير العمل فلن نصل لحجم كبير في الـ Star. بينما في حال استخدام Snowflake فهذا يعني وجود عدد من الـ Levels مما يجعل الأمر أكثر تعقيد وصعوبة (يوجد الكثير من عمليات join فتكون الداتا موزعة على الجداول) .

* في المثال السابق كان لدينا ثلاث records في الـ Relational (product, customer, date) وعند الترحيل تجمعت في record واحد في الـ Star ضمن الـ Fact Table .

في الـ MD عند بناء الـ DB ننشئ component نسميه hierarchy ونحدد الجدول الذي يعتمد عليه و قد يكون للجدول الواحد n hierarchy فمثلاً في الـ date لدينا attribute التالية (quarter, year, month, day, week, ...) فممكن أن نبني hierarchy 1 (year , month , day) او hierarchy 2 (quarter, month)

وإمكانية تعدد الـ hierarchy هذه احصل عليها كلما زدت الـ attribute في الـ Dimension.

مثلا: أريد مبيعاتي من البنطلونات الجينز السوداء ف لدي attribute: color.

واذا احتجت معرفةالمبيعات من بنطلونات الجينز ذات القياس M فإنني أقوم بالتفصيل أكثر أي أدخل level أعمق

ويمكن ان أصعد 1 level ايضاً باستخدام هذه الـ Attribute .

ويمكن ان يتشارك اكثر من cube بنفس الـ Dimension وذلك ببناء أكثر من Fact Table .

انتهت المحاضرة

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