

Data Modeling

S1 : ال Datamodel هي طريقة لتنظيم بيهـا ال Elements بـنـاعـهـ الـ data ... و هـنـكـلـمـ عنـ الـ Elements كـمانـ شـويـهـ فـهـنـلـاقـيـ انـ الـ model دـاـ بـيـوـصـفـ كلـ حـاجـهـ مـحـتـاجـيـنـهاـ .. زـيـ الـ Relationships, Entities, Constrains وـ كـمانـ بـيـوـصـفـ اـزـايـ الـ اـبـلـيـكـيـشنـ شـغـالـ اوـ الـ Apisـ الـيـ بـتـحـصـلـ

عـشـانـ كـداـ لـماـ حـدـ Data Engineer بـيـرـوحـ لـمـكـانـ عـمـلـ .. فـ اـولـ حـاجـهـ الـ mـكـانـ بـيـدـهـالـهـ هوـ الـ dataـ مـوـدـلـ الـيـ هـمـاـ عـاـمـلـيـنـهـ Reportingـ عـشـانـ يـفـهـمـ مـنـهـ الـ دـنـيـاـ مـاـشـيـهـ اـزـايـ وـ يـبـقـيـ قـادـرـ يـتـعـامـلـ مـعـ الـ dataـ لـمـاـ يـجـيـ يـشـتـغلـ عـلـيـهـاـ اوـ حـتـيـ يـاخـدـ مـنـهـ لـلـ

What is data model?

The data model

- is An abstract model that organizes elements of data.
- It describes the objects, entities, and data structure properties, semantic, and constraint.
- It formalizes the relationship between entities.
- It describes how the application (report) API data manipulation.
- It describes the conceptual design of a business or an application with its flow, logic, semantic information (rules), and how things are done.
- It refers to a set of concepts used in defining such as entities, attributes, relations, or tables.

الـ Data Model مشـ عـلمـ .. لاـ دـاـ تـجـربـهـ .. وـ مشـ staticـ .. لاـ بـيـخـتـلـفـ مـنـ حـتـهـ لـلـتـانـيـهـ حـسـبـ الـ اـسـتـخـدـامـ وـمشـ نـوـعـ مـنـ الـ دـاـتـاـبـيـزـ وـمشـ اـخـتـرـاعـ لكنـ هوـ مـجـمـوعـهـ مـنـ الـ Conceptsـ الـيـ بـتـطـبـقـهاـ عـشـانـ يـبـقـيـ عـنـدـكـ Designـ مـحـترـمـ لـشـكـلـ الـ dataـ بـتـاعـكـ وـ بـيـخـتـلـفـ باـخـتـلـافـ الـ اـسـتـخـدـامـ .. يـعـنـيـ اـنتـ مـمـكـنـ تـاخـدـ modelـ كـانـ مـعـمـولـ .. وـتـغـيـرـ فـيـهـ حـاجـاتـ تـنـاسـبـ الشـغـلـ بـتـاعـكـ وـداـ الـيـ فيـ الـ اـلـاغـلـ بـيـحـصلـ = مشـعـطـولـ بـنـبـنيـ مـوـدـلـ from scratchـ لوـ عـنـدـنـاـ حـاجـهـ جـاهـزـهـ مـمـكـنـ نـعـدـ عـلـيـهـاـ وـنـسـتـخـدـمـهـاـ

What is data model?

Data model is not

- a science.
- a static design for each organization.
- a type of database.
- a new invention which needs to be done for each project.

Data model is

- a general concept that leads to build full architecture.
- an engineering design practices.
- different based on the use case and the database type.
- customizable, and we can utilize some of the ready built architecture.
- affecting information reporting performance.

ف نقدر نقول ان ال Data Model هو اول حاجه بتتعمل قبل ما تبدأ تعمل .. هتعمل Integrartion ازاي
وانتم مبنيتش ال shema بتاعتك وعرفت مين ال fACT ومدين ال Dim واي هي ال Attributres بتاعه ال Dim دا
كمان هو حلقة الوصل بين اللي ال business عاوزه .. وبين ال technical design نفسه
ولازم يكون ال ليها pattern معين .. في تسميه الجداول – الكولمنز وهكذا

و بالتالي المفروض ال OUTPUT بتاعنا بيقي عباره عن Data Model Design Document
بنفهم منه ال system شغال ازاي

What is data model?

The data model is

- The first part before starting integration with any new source system.
- The connection layer between business requirements and technical design.
- It is also the translation between logical and physical layer.
- It is unified across all systems and has the same patterns and practices.
- It engaged with any source systems integration from the early stages.
- This stage output is a data model design document or mapping sheet.

طب ليه ال Data Model مهمه ... كل الأسباب اللي فاتت كانت كافية ..
لكن كمان ال Data Model بما انها هي اللي بتتعمل ال design بناء على Software .. ف بالتالي هي اللي هتحدد
المهندسين ازاي يفكروا في حل مشكلتهم (عاوزين نطلع ال sal للبرودكت و الموظفين) ف بناء على ال Data model اللي
معمول .. ف شایفين ال Schema و بالتالي هنبقى عارفين ازاي نعمل query مثلًا عشان نجيب الحاجه دي

Why does the data model are important?

- Data models are currently affecting software design.
- It decides how engineers think about the problem they are solving.

أي الفرق بين ال Design و ال Implementation ؟
لو عاوزين نبني عماره ... بنجيب ال user ونسلمه عاوز كام حمام وكام غرفه وهكذا ... ثم نجيب architecture و نقوله
Implementation ع طلبات ال user و يبدأ انه يعمل ... ثم نجيب مقاول و تأتي مرحله ال Design

Data Model Design vs Implementation

- If you need to build a home, so, how do we design this home?
 - Determine if the home is one level or multi-level and decide main bedrooms and bathrooms for each floor. (User needs)
 - Hire an architect to put the architecture in more detailed way
 - ☞ the size for each room, the distribution of the wires, where the plumbing fixtures will be placed, etc. (Architecture phase)
 - Decide the decorations, colors for each room, carpets, etc.
- What do we do for the implementation?
 - Hire a contractor to build (implement the design) the home.
 - This phase implement the design, but it also includes some detail related to the real way to build the tools and the material (Physical Design).

ال Elements DataModel + Dim + Fact table هي ال بتابعه

Elements of Data Model

Facts are the measurements/metrics or facts from the business process

☞ (Telecom industry, measurement would be the count of daily/hourly usage per customer). We could consider facts as the source of reporting for the business.

Dimensions provide the context surrounding a business process event. In simple terms, they give who, what, where the fact,
☞ (Telecom industry, for the fact daily usage, dimensions would be customer_id, location_id).

Attributes are the various characteristics of the dimension. In the previous examples, the attributes can be customer details (from customer_id get the gender, age, nationality, etc.).

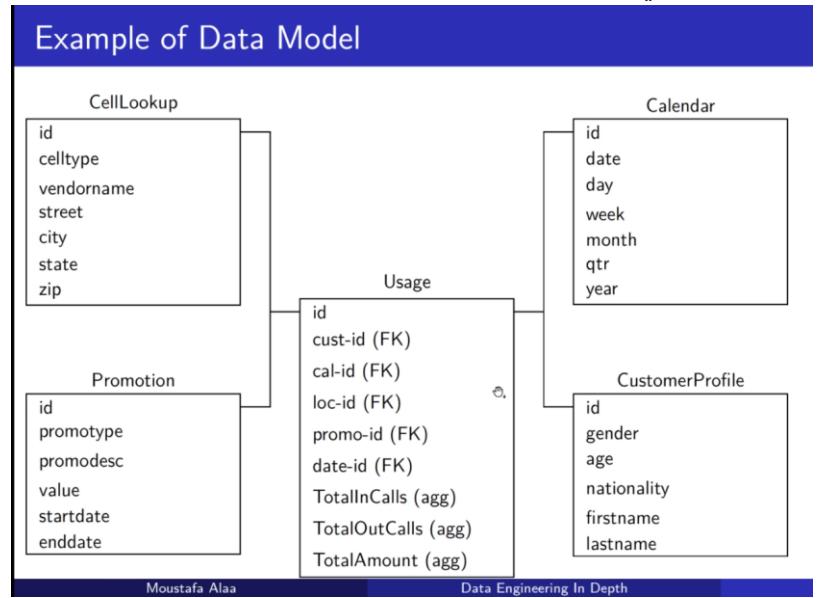
ال Fact table هو ال Primary Table بتابعنا .. وبيقي فيه ال Measures + FK ال Building على ال Fact table اللي عاوز احسب ال Measure اللي في ال Dim table اللي تأثير عليه = حد ليه المودل

Elements of Dimensional Data Model

Fact Table is a primary table in a dimensional model. A Fact Table contains (Measurements/facts and Foreign key to dimension table). It located at the center of a star or snowflake schema and surrounded by dimensions.

Dimension table contains dimensions of a fact and business reference data. They are joined to fact table via a foreign key. Dimension tables are de-normalized tables. It connected to the fact table and located at the edges of the star or snowflake schema.

وهذا مثال على ال Data Model بعد ما اتعملت ... ال Usage TABLE هو ال Fact هنا



عشان تبني ال Elements بتعاه ال Data Model .. لازم تعرف ال Req و بعدين بناء عليها تقول مين ال Dim و ال Fact و كمان تعرف أي هو ال granularity (ال اللي في ال Fact بيبقى per hour مثلاً؟)

Elements of Data Model

Dimensional model life cycle:

- Gathering Requirements (Source Driven, Business/User Driven).
- Identify granularity of the facts
- Identify the dimensions
- Identify the facts

وانتم بتعمل ال Dim هتلaci فيه طرق كتيره لبناء ال Dim زي دول + هنقول فكرتهم كلهم ..بس فيه 5 منهم المهمين

Dimensions Types

- ① Conformed Dimension.
- ② Degenerate Dimension.
- ③ Junk Dimension (Garbage Dimension).
- ④ Role-Playing Dimension.
- ⑤ Outrigger Dimension.
- ⑥ Snowflake Dimension.
- ⑦ Shrunken Rollup Dimension.
- ⑧ Swappable Dimension.
- ⑨ Slowly changing Dimension.
- ⑩ Fast Changing Dimension (Mini Dimension).
- ⑪ Heterogenous Dimensions
- ⑫ Multi-valued dimensions

S2 : زي ما اتفقنا اننا عندنا طرق كتيره لتكوين ال dim .. وممكن في نفس الموديل .. تبني كل Dim بشكل مختلف .. حسب Cases اللي عندك ف مثلا ال Conformed Dimension لو عندنا case عارفين نعمل key معين ليها .. ف هنا وانت بت create ال Key دا لو ال Fact table راح اتعامل مع أي Fact table .. هيفضل معنى ال Key دا ثابت ... وال Key اللي بن create دا ثابته بيبيقي حاجه زي مثلًا مثال ال Date Dim ... زي مثلًا مثال ال Row Dim عاوزين نعمله Key عشان نقدر من خلال ال Key دا نعرف كل الداتا بتابعه ال Row الخاصه بال Fact Table و كمان هيبقى معنى ال Key دا ثابت في أي حته بنسخدمه و مع أي

Conformed Dimensions

Conformed Dimensions the dimension which is *identical* and has the *same meaning* across many fact tables which it relates and used in different areas of the warehouse.

Example

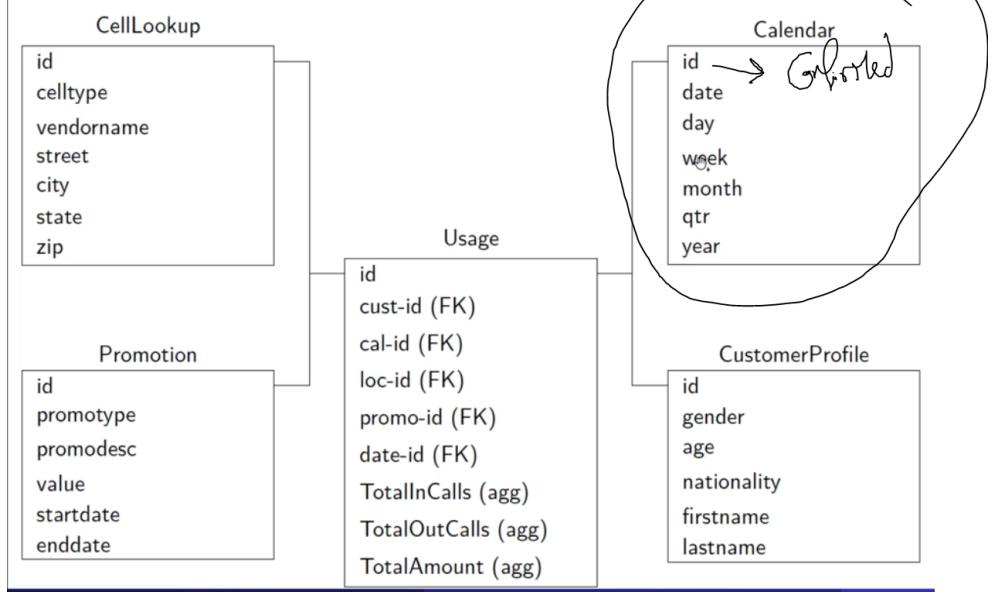
- **(Date as a Key)**: if we have a date column across many facts, we could use the date as key in all tables. So, it should be a unified format.
- **(Product-Id as a Key)**: if we have a product name which could vary between systems
 - ☒ ⊕ (upper/lower) We can create a dimension table for the product details and use product id unified across fact tables.

Moustafa Alaa

Data Engineering In Depth

ال ID هنا في ال Dim Calender هو ال Key بتابعنا و بعدها ال DimConfirmed Dim دا نوعه # بال المناسبه انت لما بتجي تختار نوع ال Dim اللي هتعمله مش بتقول انا هعمل كذا .. هي بتجي بالتجربه + ممكن نعمل أكثر من نوع Dim في نفس ال Data Model

Example of Data Model



Moustafa Alaa

Data Engineering In Depth

S3 : النوع الثاني وهو ال Degenerate .. ودا بيقي عباره عن Key من غير Dim
 يعني أحياناً وانت جاي تعمل ال Dimensions .. بيقي فيه داتا هي مش Dimension بس لازم تتحط في ال Fact Table #
 والنوع دا بنعمله لما نعوز نعمل grouping ل ال cases بتاعه ال business bus معندهاش #
 ف هنا مثلاً قدامنا ال Fact table ... جواه 2fk ... different Dim ... وفيه ال Measures و فيه كولمن كمان لا هو FK ل Dim ولا حتى هو Measure الا هو ال OrderID
 دا ملوش Dim معين منه .. دا محظوظ في ال Fact table عشان يعمل order ل Grouping اللي طالع # بالمناسبة هنا ال OrderDate و ال ProductID Dims .. دول نوعهم conformed #
 لأن بيغير عن ال key بحاجه .. معناها ثابت لما تيجي تستخدمها في أي Fact table

Degenerate Dimension

- Degenerate Dimensions

- Dimension Key without corresponding dimension table.
- Stored in fact table.
- It used to provide a grouping for business cases.

OrderDetail				
OrderID	OrderDate (FK)	ProductID (FK)	Quantity	Amount
123	123456789	111	2	120.45
123	123456789	222	5	10.45
431	98765122	333	1	15.45
431	98765122	555	6	4.45

OrderID	OrderDate	ProductID	Quantity	Amount
123	123456789	111	2	120.45
123	123456789	222	5	10.45
431	98765122	333	1	15.45
431	98765122	555	6	4.45

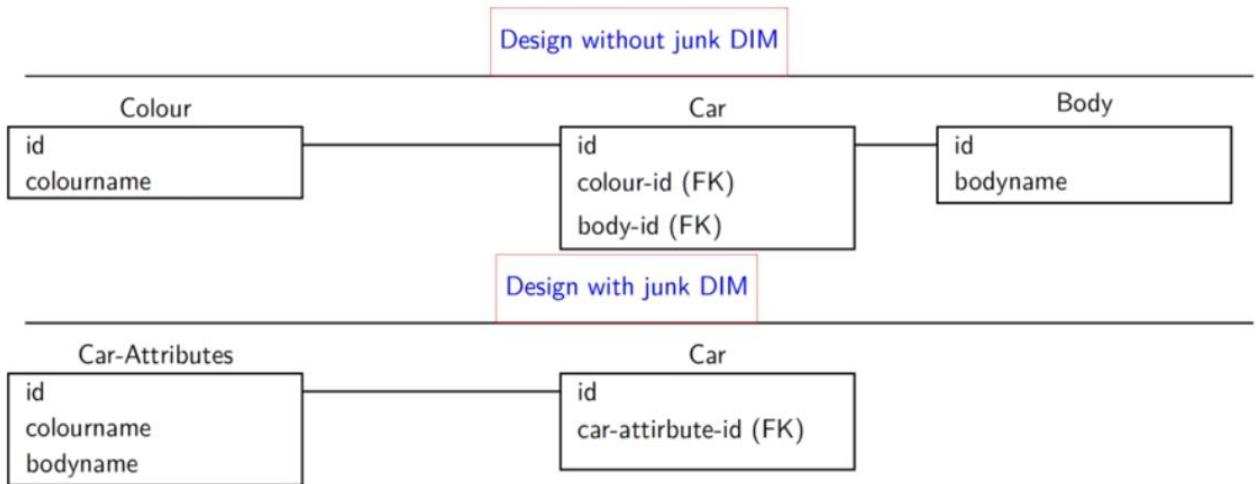
S4 : النوع الرابع وهو ال JUNK او ال Garbage ... بتسخدمه امتى ؟
 أوقات بيقي عندك Dims كتيره جداً مرتبطين بعض ... ف الأفضل بدل ما تكتر عدد ال Joins ف وبالتالي عدد ال Dims يبقى كتير .. ف احنا هنقل عدد ال Dims عشان الدنيا تبقي اسهل .. عن طريق اننا هنضم كل ال Dims دي وتبقى Dim واحد # ف كذا هنا عدد ال Dims هيفيل .. وبالتالي عدد ال Joins هيفيل .. وبالتالي عدد ال Columns هيفيل

Junk Dimension = Garbage Dim

- It used to reduce the number of dimensions (low-cardinality columns) in the dimensional model and reduce the number of columns in the fact table. It is a collection of random transnational codes, flags, or text attributes.
- It optimizes space as fact tables should not include low-cardinality or text fields. It mainly includes measures, foreign keys, and degenerate dimension keys.

مثال : هنا عندنا dims 3 واحد للعربيه والثاني لنوع العربيه و الثالث لوصف ال Body بتابع العربيه
 طب ما دي كدا joins كتيره .. و الـ dims related كلهم في Dim واحد او حتى اتنين
 ف مثلا تعمل Dim للعربيه .. و Dim للصفات بتابعتها

Junk Dimension



بس قالك خلي بالك دايما وانت بتعمل الـ Junk Dimension عشان الـ Complexity بتابعتك متكبرش
 لأنك يعني انك بتقلل عدد ال Dims ف دا معناه ان حجم ال Dim نفسه بيزيديد (عدد هم بيقيل بس حجم ال Dim نفسه زاد)
 لانه بقى فيه كذا معلومه (اللي ضميتهم ع بعض)

ف كدا ال Complexity ممكن تزيد .. ودي مش حاجه حلوه
 ع سبيل المثل .. لو الل Body فيه منه 3 أنواع ... وال Color فيه منه 3 أنواع
 ف انا كدا اكني عملت Dim حجم الداتا فيه هتبقي $3 * 3 = 9$

Junk Dimension

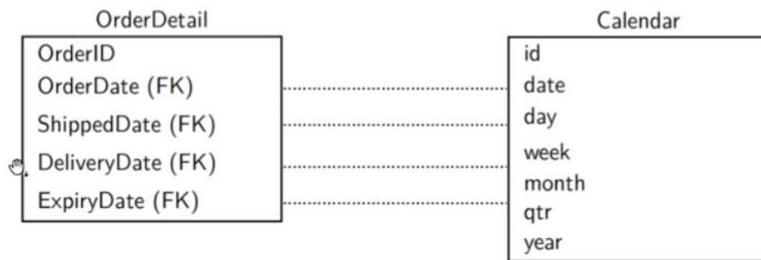
Junk Dimension Table Size

- We must split the Junk dimension into more dimensions in case the size grows by the time.
- It is easy to calculate the expected number of rows as it is the total number of combinations between the low-cardinality attributes;
☞ 3 columns each have 3 values total = $3 * 3 = 9$.

S5: النوع الرابع وهو ال Role- Playing و هنا معناها ان ال Key اللي بعمله في ال Dimenseion بيلعب كذا دور مختلف لما باجي استخدمه في ال Fact Table هنا مثلا ال Date مره بيقي Order و مره Shipped وهذا

Role-Playing Dimension

Role-Playing Dimensions (Re-usable Dimension) A single physical dimension helps to reference multiple times in a fact table as each reference linking to a logically distinct role for the dimension.



و بالتالي لازم نفرق بين ال Conformed, RolePlaying .. ال Key ليه نفس المعنى في ال Conformed .. ال Fact Table .. انما ال Role-Playing .. ال Key بيبقى ليه اكتر من استخدام = اكتر من معنى في نفس ال Fact Table

Conformed vs Role-Playing Dimension

Conformed vs Role-Playing

- **Conformed** is the same dimension used in different facts and has *the same meaning*
  CustomerID.
 - **Role-Playing** is the same dimension which used multiple times within the same fact but *with different meanings*
  Date.

S6 : النوع الخامس وهو ال OUTtrigger ... ودا معناه اني عندي Dim مربوط ب ال Fact table يعني ال Dim مستخدم ك refrence في ال Fact table ... ودا الطبيعي وف نفس الوقت بقى هو مستخدم ك refrence في Dim اخرر

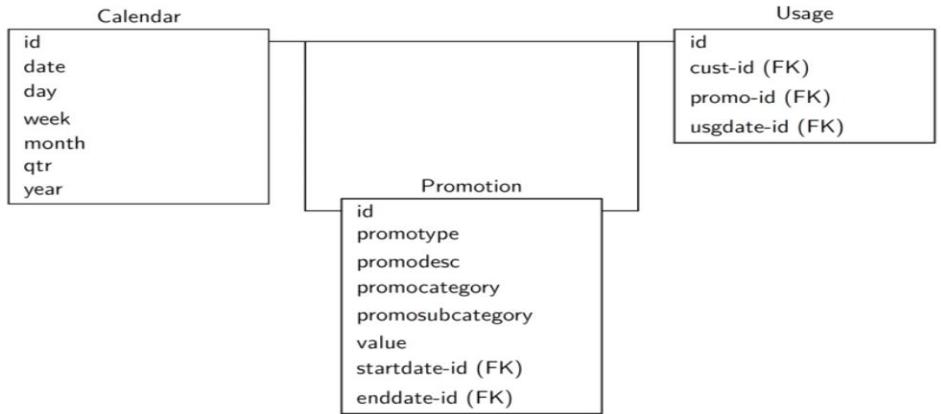
Outrigger Dimensions

- A dimension which has a reference to another dimension table. The secondary dimension called outrigger dimension.
 - ~~This dimension design should be used carefully with limited cases.~~

وخلی بالک النوع دا لازم تستخدمه بحرص ... ويفضل عدم استخدامه لانه بيعمل مشاكل قدام .. اثناء ال Report # هنا مثلا ال Dim اللي اسمه Promotion دا Reference في ال Fact table .. ودا الطبيعي وفي نفس الوقت هو reference في Dim آخر وهو ال Calender

برحل مشاكل في Report

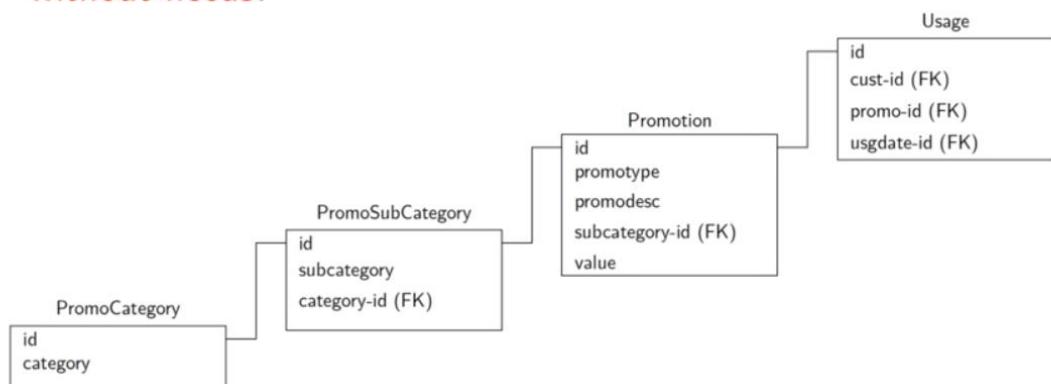
Outrigger Dimensions



S7 : النوع السادس وهو ال Snowflake و هنا دا نوع Dim وليس نوع Schema . يعني بتتكلم عنه من مفهوم كونه Dim ال Dim دا فكرته انه ببقي عباره عن شويه Dims ليهم علاقه بعض متصلين بشكل Hierarchy يعني مثلا عندنا ال Promotion Dim مرتبط Dim تاني بـ Hierarchy وهذا في اللي بعده لكنه كان عنده Promotion Dim كبير وهو عمله Normalize وخلال كذا حاجه Hierarchy و لا ينصح تماما انك تستخدم النوع دا من ال Dim و دا لانه بيعمل Complexity عاليه جدا ... شبه ال Junk Dimension مربوطه بعض ولكنها مش مربوطه بشكل Hierarchy .. بس في ال Junk Dimension شويه .. بس في ال

Snowflake Dimensions Not Snowflake Schema

- Snowflake Dimension is a dimension that has a hierarchy of attributes. This attribute is normalized, and each dimension has a relationship with another hierarchy dimension table.
- **This dimension design not recommended as it has much complexity to the model and query performance. Also, it complicates the ETL process and makes too many dimensions without needs.**



58 : النوع السابع وهو ال Slowly Changing Dimension ... وهو ببساطه من اسمه ..اني أوقات بيبقى عندي
 attribute change و لكن بيطرى و بشكل عام ..لأزم تبقي عارف ان انت في رحلتك في ال DWH انت بتحاول ت manage ال
 علشان نال Slowly change اللي بيحصل ف عندها 4 حلول ممكن تستخدم أي واحد فيهم (حسب كل case)
 انا نقول انا حفظت اول قيمه لل attribute Da ..ومليش دعوه لو اتغير
 انا بنقول انا حفظت اخر قيمه لل attribute .. لكن القديم مش هسيفه عندي ك history
 انا بنقول لا انا هسجل كل التغيرات في نفس ال table و هعملها 2times واحد للقديم واحد للجديد وهذا
 دي عباره عن combination من 1 و 2 وهذا انا حفظت history بس ب اخر فيتين ..الجديد والي قبلها
 دي بقى بنقسم الداتا لجدولين ..جدول للقيمه الحاليه والتاني لكا ال History type 4

Slowly changing Dimensions

- It the dimension which changes over time. So, for a specific date we have different value.
- It has different types as following
 - Type 0 (Fixed Dimension): We don't change the current even the source changes.
 - Type 1 (No History): No history is maintained only the latest replace the current.
 - Type 2 (History): Series of history of records are maintained.
 - Type 3 (Hybrid): Only the last Change and the Current new change is stored
 - Type 4 : We split the data into two tables, first the current record and second is the historical (most common usage).

و خلي بالك ..ممكن وانت بتسييرش تلاقي types manage تانية بن فيها ال SCD
 بس أي حاجه هتلافقها هتلافقها فكرتها شبه ال TYPE3 الا وهي انها Combination بين 1 , type2

Slowly changing Dimensions

Note

*There are some other types which is a combination between the above similar than type 3 combined between 1 & 2.
 You can check the chapter resources for more information about the other types.*

مثال 0 type 0 : هنا عندنا في اليسار ال record القديم .. ولما غيرت ال city . بقى عندنا ال record الجديد اللي ع اليمين في ال 0 type 0 بيقولك انا بحتفظ ب اول قيمه فقط و بنقض للتغيرات ... ف ال Dimension هبيقي فيه القيمه القديمه لل city

Slowly changing Dimensions

- Type 0.

CustomerID	Name	City	CustomerID	Name	City
123456789	Ronaldo	Madrid	123456789	Ronaldo	Turin

Table: Source System Old vs New

ID	CustomerID	Name	City
1	123456789	Ronaldo	Madrid

Table: Customer Profile Dimension

مثال 1 type 1 : لو نفس ال case بس المرادي هطبق ال type 1 هبيقي فيه القيمه الجديدة ومش
بحتفظ ب أي تغيرات حصلت .. يعني هعرف اخر قيمه للتغيير اللي حصل فقط

Slowly changing Dimensions

- Type 1.

CustomerID	Name	City	CustomerID	Name	City
123456789	Ronaldo	Madrid	123456789	Ronaldo	Turin

Table: Source System Old vs New

ID	CustomerID	Name	City
1	123456789	Ronaldo	Turin

Table: Customer Profile Dimension

مثال 2 type 2 : هنا بقى في type 2 بحتفظ بكل التغيرات .. كل ما يحصل تغيير جديد في ال city .. بضيف row جديد
طب مثلا لو عاوز اعرف من ال Dim اي هي ال city الحاليه user = اخر city هو فيها حاليا .. ف هقوله where ان
ال null terminationDate =null او ممكن أقوله where ان ال iscurrent true .. لكن تكرار نفس ال row في
ال table بيخلي ال performance مش احسن حاجه في بعض الحالات .. ف خلينا نشوف فكره ال type 2 اللي بعده
=ال null في ال TermiDate .. ممكن نحط مكانها Date = infinite Date كبيير اوكي علامه لينا(أي طريقه براحتكم)

Slowly changing Dimensions

- Type 2.

CustomerID	Name	City	UpdatedDt
123456789	Ronaldo	Madrid	2018-12-12
123456789	Ronaldo	Turin	2019-06-12
123456789	Ronaldo	London	2019-08-12
123456789	Ronaldo	Porto	2019-12-12

Table: Source System Old vs New

ID	CustomerID	Name	City	effectiveDt	TerminationDt	isCurrent
1	123456789	Ronaldo	Madrid	2018-12-12	2019-06-12	false
2	123456789	Ronaldo	Madrid	2019-06-12	2019-08-12	false
3	123456789	Ronaldo	London	2019-08-12	2019-12-12	false
4	123456789	Ronaldo	Porto	2019-12-12	null	true

Table: Customer Profile Dimension We can replace null with infinite date (9999-12-31)
but it needs to be consistent

infinite

: هنا كنا بنقول انها comibation بين typ1,2 لأننا بنحتفظ بقيمتين في ال Dim بيقي فيه اخر تعديل و اللي قبله

Slowly changing Dimensions

- Type 3.

CustomerID	Name	City	UpdatedDt
123456789	Ronaldo	Madrid	2018-12-12
123456789	Ronaldo	Turin	2019-06-12
123456789	Ronaldo	London	2019-08-12
123456789	Ronaldo	Porto	2019-12-12

Table: Source System Old vs New

ID	CustomerID	Name	City	UpdatedDate	previousCity
1	123456789	Ronaldo	Porto	2019-12-12	London

Table: Customer Profile Dimension

: يعتبر كsnapshot .. بيقي عندنا ال history Dim فيه كل التغييرات و ال DIM تاني فيه اخر تعديل فقط

Slowly changing Dimensions

- Type 4 (Split current and Historical).

ID	CustomerID	Name	City	effectiveDt	TerminationDt
1	123456789	Ronaldo	Madrid	2018-12-12	2019-06-12
2	123456789	Ronaldo	Madrid	2019-06-12	2019-08-12
3	123456789	Ronaldo	London	2019-08-12	2019-12-12
4	123456789	Ronaldo	Porto	2019-12-12	null

Table: Customer Profile Dimension Hist

ID	CustomerID	Name	City	UpdatedDate
1	123456789	Ronaldo	Porto	2019-12-12

Table: Customer Profile Dimension

طب ازاي ال SCD بت join مع ال Fact table .. هنا لو عاوز اجيب اخر مجموع لمكالمات ال user .. ف لازم أقوله
ال id بنطاع ال customer وكمان ال date اللي عاوزه

Slowly changing Dimensions

- How does the Facts join SCD? We have two scenarios as following:
 - Getting the current customer information (Join with the latest).
 - Getting the historical customer information (Join with the historical table based on **cust id & date**).

ID	CustomerID	TotalCalls	CallIDate
1	123456789	30	2018-12-12
2	123456789	30	2019-12-12

Table: Customer Usage

ودا في الغالب بيبقى شكلها الكودي .. لو احنا بنتكلم عن ال type 4 .. لو حبيت اجيب بيانات العميل
 لو من ال snapshot = اللي بيبقى فيه اخر قيمه للتغير .. هنعمل الكود الاولاني .. يكفي فقط أقوله مين ال user
 لو من ال history Dim = اللي بيبقى فيه كل الداتا بال time .. بيبقى الكود الثاني .. لازم أقوله انا عاوز ال user الفلاني
 عند ال time الفلاني

Slowly changing Dimensions

--Get latest customer details from customer profile **snapshot** تيپ ٤

```
select * from cust_usage_dly a
inner join cust_profl b
on a.CustomerID = b.CustomerID;
```

--Get historical customer details from customer profile hist

```
select * from cust_usage_dly a
inner join cust_profl_hist b
on a.CustomerID = b.CustomerID
and CallDate between effectiveDt and TerminationDt
```

Listing 1: Example to show how to use SCD

و زي ما قولت علي حسب ال Case Study بتاعك و حسب رغبه العميل .. بنشوف هنسخدم اني type في ال 5

Fast Changing Dimension (SCD Type 2)

لما بيكون عندنا case ان فيه attribute بيتغير كتير جدا وبسرعه .. زي مثلا انه بيتغير كل أسبوع
 ف الحل اني افصل ال attribute دا او لو كانوا اكتر من ... attribute ... افصلهم عن ال Dim واعمل بيهم Dim جديد
 واربط بين ال 2 Dims عن طريق حاجه اسمها mini Dimension ... و دا ببساطه Dim بيربط بينهم فقط لا غير

Fast Changing Dimension (Mini Dimension)

- When we have a dimension with one or more of its attributes changing very fast.
- It causes a performance issue if we tried to handle this case similar SCD Type 2 because of the rapidly changing in this dimension and the table will includes a lot of rows for this dimension.**
- We solve this case by separation the attributes into one or more dimensions. This technique also called **mini-dimensions**.

فمثلا لو عندنا الداتا دي .. هنا ال Weight وال Presure كل أسبوع (مريض بيكتش كل أسبوع) ف الحل اني افصل ال weight وال presure واعملهم ك Dim جديد ... واربط بين ال 2 دول ب Dim تالت Mini Dimension يبقى كدا ال Fast Changing Dim حلها هو ال mini Dimension

Fast Changing Dimension (Mini Dimension)

- How to implement FCD (Mini Dimension)?

Hint: Search for the mini-dimension relation table.

Patient_id	Name	Gender	BirthDate	Weight	B_Presure	UpdateDt
123	Anna	F	1968-01-12	50	110.0	2019-01-01
123	Anna	F	1968-01-12	55	130.0	2019-01-07
123	Anna	F	1968-01-12	59	115.0	2019-01-14
123	Anna	F	1968-01-12	65	120.0	2019-01-21

Table: Patient Profile Dimension

Patient_Key	Weight	B_Presure
1	50	110.0
2	55	130.0
3	59	115.0
4	65	120.0

Patient_id	Name	Gender	BirthDate
123	Anna	F	1968-01-12

Table: Patient Profile Dimension After Removing FCD and Split it into Junk-Dimension table

وشكلها هيبقى كدا

ولو تلاحظ : احنا كدا الي حد ما بقينا شبه ال Junk Dim ... يعني ال Fact Changing .. حلها هو ال mini Dim و دا بيخلينا نبقي شبه ال Junk Dim

Fast Changing Dimension (Mini Dimension)

Patient_id	Patient_Key	Start_Date	End_Date
123	1	2019-01-01	2019-01-07
123	2	2019-01-07	2019-01-14
123	3	2019-01-14	2019-01-14
123	4	2019-01-21	null

Table: Patient Mini Dimension



S10 : النوع التاسع وهو ال Shrunken Dim
 لو عندنا Case فيها مثلاً ال sal بتابع مبيعات كل يوم .. وانا عاوز اجيب مبيعات الشهر .. ف بما ان ال Data بتابعتي
 معموله ع Low Level ف اقدر اجيب ال High Level منها .. لأننا يعتبر هنعمل Aggregate لـ low level وبيس

Shrunken Rollup Dimensions

- Shrunken Rollup dimension is used for developing aggregate (higher level of summary) fact tables.
- It required that the data model has a lower level of granularity.

Example

- We have a daily usage fact table, and we need to have a higher level of monthly usage. So, we use the monthly dimension to get a summary of the daily.
- We have a daily usage fact table aggregated on area-id, and we need to create another summary table aggregated based on city id. So, the new grain level here is the new dimension for the city.

: مثال

لو عندنا Dim فيه ال orders لكل Area .. وعندنا Dim ثاني فيه city بتابع كل City
 ف ممكن استخدم ال 2 Dims aggregate عشان اجيب عدد ال orders لكل Area

Shrunken Rollup Dimensions Cont.

OrderDate	AreaID	TotalOrders
123456789	123	20
123456789	123	30
123456789	678	10
123456789	678	12

AreaID	AreaName	CityID
123	Al-Matareya	1
678	Ain shams	1

CityID	CityName
1	Cairo

OrderDate	CityID	TotalOrders
123456789	1	72

S11 : النوع العاشر وهو ال Multivalued

وهو ببساطه .. لو عندنا Dimension فيه arrtribute فيه (مش بيترر .. لا) هو ممكن يكون اكتر من حاجه بس محدوده M to M بقت Fact table .. اكن العلاقة بين ال Dim و ال Fact بقت العلاقات بين ال Dim ل Fact يا تكون 1 to M يا تكون 1 to 1 # مثال لو عندي Dim ل Patient و ال Patient ليه اكتر من major # مثال لو عندي Dim ل Student و ال Student ليه اكتر من account # مثال لو عندي Dim ل Customer و ال Customer ليه اكتر من rows ف في الحاله دي ببساطه حلها انك هتعمل ما يسمى بمفهوم ال Bridge Table عشان نحذف ال Bridge Dim منفصل و نربط بين Dim Multivalued وبين ال Fact table بعشان ال Fact table وقتها هيكون جواه rows بتترر

Multi-valued dimensions

- When the relationships between the dimension member and the fact are many to many which means the dimension members are lower granularity than the facts.
- Fact table should contains one-to-one relationship with the dimension. So, we introduce the **Bridge table** when we need to related multiple dimensions values with one record.

Example

- Patients can have multiple diagnoses.
- Students can have multiple majors.
- customers can have multiple account.

مثال : لو عندنا case عاوزين نعمل مبيعات ال article و مع العلم ان كل article ممكن يكون ليها واحد او اكتر وعاوز اعرف نسبة مشاركه كل author في ال article كام عمل ازاي بقى ك author Dimension واربطهم بال article # هنا فيه مشكلتين .. الاولى ان ال article الواحد ممكن يكون ليها اكتر من author (Wight factor) ... الثانيه اني معرفش نسبة مشاركه كل author كام في ال article

Multi-valued dimensions

Example (Sales of Articles)

- Assume we need to report the sales of article and we have some articles has more than one author.
- Each author has weighting factor for each article.
- According to the report we need to check each author and associate with the articles they have authored. How can we model this case?
- Assume the first article has only one author *Moustafa*, and the second article has two authors *Ahmed & Amr*.

ID	Name	Email	Bio
123	Moustafa	abc@gability.com	S-Engineer
234	Ahmed	def@gability.com	L-Engineer
345	Amr	geh@gability.com	S-Manager

ID	Title	Journal	Price
11	50	IEEE	110.0
22	55	ACM	130.0

Table: author and articles sample data.

ف زی ما قولنا .. لو عملنا ال Article Dim و ال Author Dim ربطناهم عطول بال Fact هیسببو المشکلتین

Multi-valued dimensions (Implementation-1)

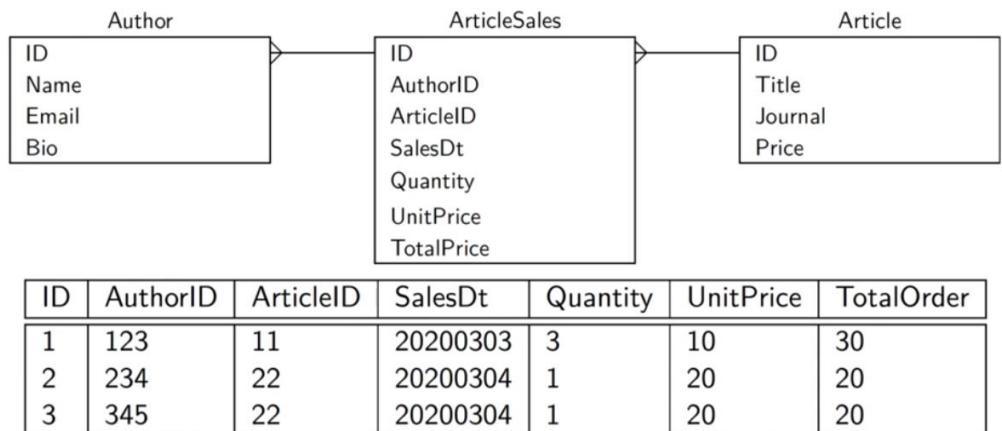


Table: Output of wrong implementation of ArticleSales

What are the problems in this implementation?

- We can't get the weighting factor for each author.
- Duplicated rows in sales.

ف تعال نفكر في الحل ... تعالى أولاً نحل مشكله ال Weighting Factor = نسبة مشاركه كل مؤلف في ال Article # لازم اعمل Dim جديد واربطه ب Dim ال Author عشان احط ال Weight Factor ... ومعلمتش ال Weight Factor في Author Dim نفسه .. لأن كدا هيحصل Duplication في Author Dim

بس ثواني .. احنا كدا معندناش Duplication في Author Dim ولا حتى Article Dim
بس عندنا Duplication في Article Sales في Fact Table ... ومشكله ال Duplication في Author Dim

Multi-valued dimensions (Implementation-2)

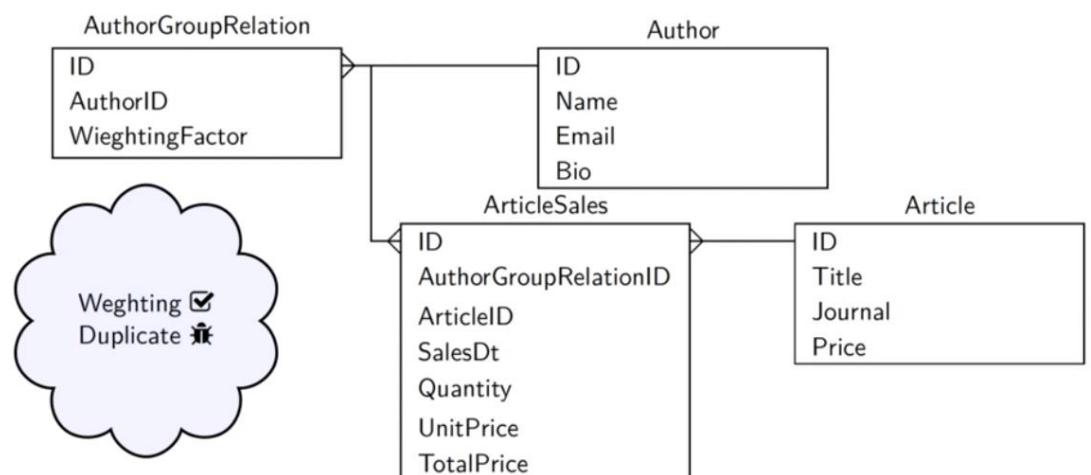


Table: Output of wrong implementation of ArticleSales

ف الحل بقى عشان نحذف ال Bridge Table اللي في ال Fact Table اتنا نعمل ال Duplication وفكته ببساطه اتنا بدل ما نفضل نكرر ال rows في ال Fact table.. نعمل Dim جديد واكنه بيشار على ال Dim المكرر

Multi-valued dimensions (Final Implementation)

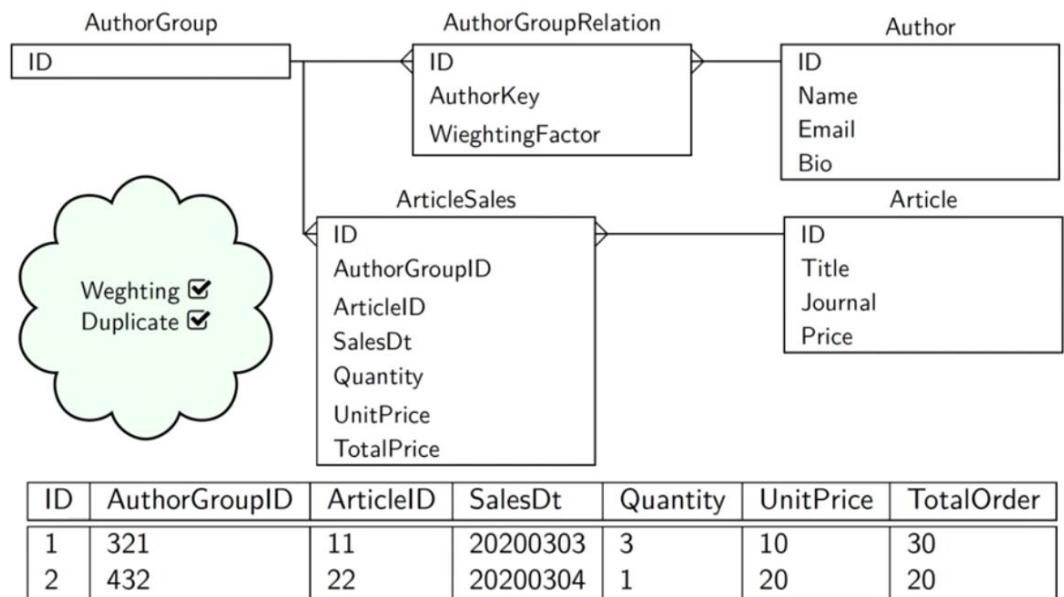


Table: Expected output of ArticleSales

S12 : النوع الحادي عشر وهو ال Swappable

مما لو عندنا case .. شركه اتصال و عندها قسم اسمه party دا فيه كل الناس اللي بتعامل مع الشركه .. فقد يكون مثلاً وقد يكون seller ... وانا عاوز اعمل Dimension للموضوع دا الكلام دا معناه ان عندنا ال party مره يكون agent معينه .. ومره يكون seller ب اخري كما عندنا اكتر من version لنفس ال Dimension (معاني مختلفه – structure مختلف) ف ماما فعل ؟

Swappable Dimensions

- A dimension that has multiple alternate versions of itself that can be **swapped at query time**.
- Each version of the hot-swappable dimension (sub-types)
 - It has a different meaning
 - It has a different structure.
 - It has fewer data compared to the primary dimension (fewer rows and columns).
 - It has a different output based on the input version and its alternatives.
 - Multi versions could be used together in the same fact with different types.
 - It can act as the primary dimension and join to the same fact table.
 - It has different target users and sometimes we restrict the users to access the primary dimension and only access the swapped version to restrict the data without needs to show the whole primary attributes.

الفكره دي نفس فكره ال Super type وال Sub type في موضوع ال EERD ف فاكر الحوارات بتاعتها ؟

ممكن ال SUB كله يبقى عباره عن داتا ال SUPER يكون عندهم attributes مختلفه وهكذا وهكذا Subs type

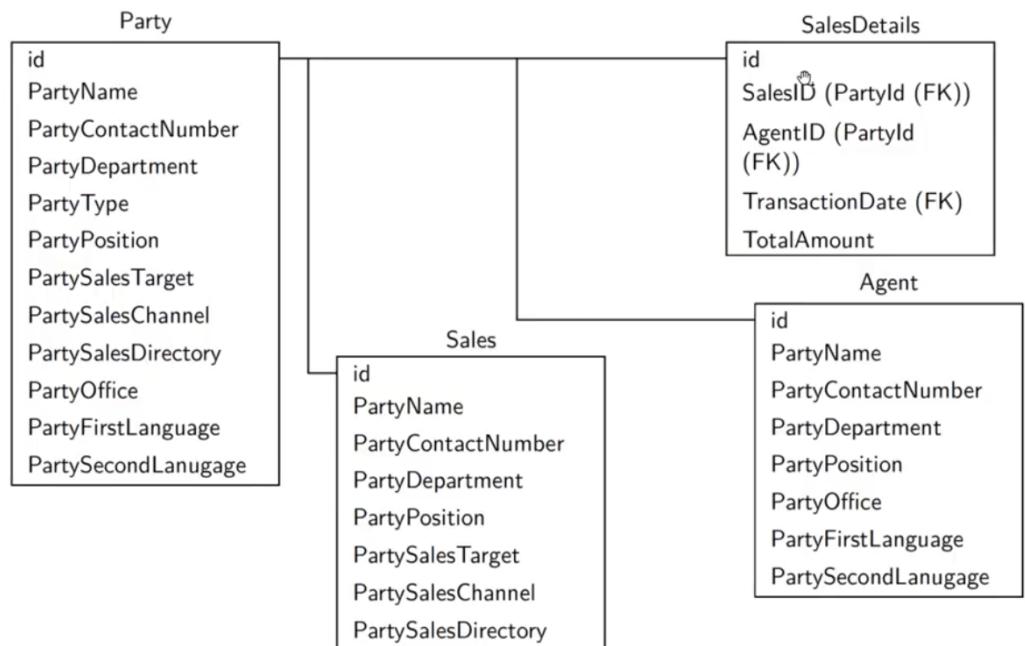
ف ال case دي ليها حلين .. يا امااني اعمل في ال party type حاجه اسمها party واربطها ب fact table عطول بس كدا لو انا بدخل row خاص ب agent ف كدا ال columns بتاعه ال sales هتبقي NULL و العكس صحيح ودا كدا هيختلي حجم ال Dim كبيير جدا ف دي حاجه مش كويسه

والحل الثاني هو اني اعتبر ان ال Agent دا و ال Seller دا و ال Views عباره عن بناء ع ال party type دا سهل بس ال performance بتاعه مش احسن حاجه (و في الغالب دا اللي بيستخدم)

والحل الاخير هو اللي في الصوره اني اعملها فكره ال sub و ال super و عندي نفس ال column بتاع ال party type بالاضافه اني عندي ال 2 بتوع ال Dim Agent و ال Seller (و دا هيزود عدد ال Dim والداتا هتبقي مكرر هلكن هيختلي الدنيا اظرف)

بالمناسبه ال SalesDetails هنا هو ال Fact table

Swappable Dimensions



Swappable Dimensions

Implementation

- Direct join between Fact and Dimension with filter based on PartyType (run-time). In this case Party includes some empty columns based on the type.
- Logical views each view has its own number of columns and rows based on the type details.
 - Pros: Easy for (managing, implementation) with consistent views.
 - Cons: Performance and manage the authorization per view.
- Physical tables (Types & Sub-types).
 - Pros: Performance, better design.
 - Cons: Data redundancy, key could be duplicated (when join with fact), increase in data size, and ETL headache.

خلينا بقى نفرق بين ال Conformed و بين ال RolePlaying وبين ال Swappable
 ال Conformed هو Dim فيه Key .. ال Key دا ليه نفس المعنى و نفس القيمه في أي حته هستخدمو
 ال Fact table هو Dim فيه Key ليه نفس القيمه و لكن كل مره ليه معنی مختلف جوه ال
 ال Swappable هو Dim فيه منه اكتر من نسخه .. مره وهو كذا .. ومره وهو كذا بـ attributes مختلفه

Attention: Conformed vs Role-Playing Dimension vs Swappable

- **Conformed** is the same dimension which used in different facts and has the same meaning and value
 CustomerID 123 can be represented into the whole model using the same value and same meaning.
- **Role-Playing** is the same dimension which used multiple times within the same fact but with different meanings and same value
 Date Dimension 20191012 can be used for different purpose order delivery date, expire date but different meanings.
- **Swappable** different version from the primary dimension each version has its own attributes and meanings based on the use case (different meaning based on the category).
 party id dimension has different version sales, agent, employee and all of them sub-types of the party but used for different purpose in different facts.



S13 : النوع الثاني عشر والأخير من أنواع ال Dim .. وهو ال Heterogeneous و هو ببساطه من اسمه .. لو عندي case اني عاوز احط اكتر من حاجه في نفس ال Dim .. وال حاجات دي غير متجانسه = أي ملهمش علاقه ببعض # خلينا في الأول نتفق .. ان ال Dim دا فكره ال IMplemenation بتاعته مش ثابتة و هي أفكار نظرية

فهنا مثلاً عندنا اكتر من بروduct بيتبع لنفس ال customer .. وكمان كل product ليه attributes مختلفه # هام خلي بالك لو كانت ال case ان عندنا اكتر من بروduct بناء ع ال customer (يعني اكتر من customer) وكمان كل product ليه attributes مختلفه (ف دي كدا ال Swappable Dim)

أشهر مثال لـ heterogenous ان شركه التامين عندها نوعين من ال product car و ال car health و لو ليك خصائص معينه و لو health ف ليه خصائص اخري (لنفسس ال customer)

Heterogeneous Dimensions

- This type works when we have a case that a company selling different product to the same base of customer. Every product has its different attributes.
- One famous example of this type assume an insurance company has two types of products like health and car. In this case Car insurance has different attributes than the health insurance.
- If we tried to model these two different products this type name Heterogeneous dimensions.

ف عندنا 3 طرق نظرية .. ف لو هنتكلم عن المثال السابق

اولاً: اننا نقسم كل product منهم ل Dim

ثانياً: نحطهم كلهم في واحد و كذا Null Dim كتيره (لو بنحط داتا car ف داتا health لنفس ال row هتبقي بـ null)

ثالثاً: هنعمل Dim فيه Common attributes

Heterogeneous Dimensions

- There are different scenarios to implement this type Separate Dimensions Split each one in separate dimensions and facts. It will be less data and business will do this analysis from two separate facts.

Merge Attributes We will merge all the attributes in one single table and we will add the common attributes and null for unrelated attributes. Implementing this scenario when we have less different attributes. However, this implementation is not recommended because of the table size, performance, and maintenance.

Generic Design In this approach we will create a single fact table and a single dimension with the common attributes. The problem of this design is we will report or care about the common attributes only.

S14 : هنتكلم عن أنواع ال Fact table Type ... بس خلينا في الأول نفكّر ... قولنا ان ال fact table دا هو الأساس في ال DWH ... وكمان بيقي مرتبط Dims ... وهو عباره عن شويه fk+measures

Fact Table Recap

What is the fact table?

- It is the foundation of the data warehouse.
- It consists of facts and measurements of a particular business aspect and processes ex: daily revenue for a product.
- It is the target of queries in most of DWH analysis and reports.
- It contains measurements/facts and foreign keys to *dimensions table*.
- It located at the center of the schema and surrounded by dimension tables.

و عمّا بيقول انك متعملش fact table لو ال business مش طالب كدا .. لآن ال DWH عشان تبنيها بتبقى مكلفه

Fact Table Recap

“*There is no point in hoisting fact tables up the flagpole unless they have been chosen to reflect urgent business priorities*”

Ralph Kimball, kimballgroup.com

عشان تبدأ ت design ال Fact table أي هي ال Process بقاعدته business
بعدين تحدد ال Grainularity
ثم تحدد ال Dims و ال Fact table بتوعك

How to design a fact table?

- Choose the business process.
- Identify the grain.
- Identify the dimensions.
- Identify the fact.

طب يعني أي grain ؟
التعريف الحرفي ليها .. هو ما يمثله ال row

يعني لو ال row بيكلم مثلا عن Product يبقى ال Grain هنا هو Per Product
انما لو ال row بيكلم مثلا عن items بتوع ال product يبقى ال Grain هنا هو Per items
فيه نقطه مهمه هنا : دايما يفضل انك تبقي ال Fact بتاعك على ال Low Level Grain
يعني هنا مثلا .. تخلي ال Row بتاعك يعبر عن ال items .. عشان لو حبيت تروح ل High level في المستقبل و تحسب حاجه ليها علاقه بالproduct كل .. ف يبقى سهل #قدر تجيء معلومه من Low level Grain لو انت كنت High level ... والعكس لا

Fact Granularity

- The grain is the definition of what a single row in the fact table will represent or contains.
- The grain describes the physical event which needs to be measured.
- Grain controls the dimensions which are available in fact.
- Grain represents the level of information we need to represent. It is not always time; it could be the physical business measurement level.
- Design from the lowest possible grain.

طب أي هي أنواع ال Fact table بقى ؟ عندنا 3 أنواع
1- per transaction: ال Row فيه بيبقى عباره عن transaction جديده حصل ... يعني ال grain هي Transaction
هنا لما بيحصل update ل row بيتحط أك row جديد
2- Periodic: ال row فيه بيبقى عباره عن مجموعة من ال transactioin في periodic معين
يعني ال grain هي per periodic
3- Accumulated: ال row فيه بيبقى عباره عن مجموعة من ال Stages .. في أيا كان الوقت اللي هتاخده

Fact Types

There are three types of fact tables:

- Transaction.
- Periodic.
- Accumulated Snapshot.

ال grain هنا بيفي transaction واحد per single transaction و ال row واحد بيفي عباره

Fact Types: Transaction Fact Table

- Fact grain set at a single transaction
- It has one row per transaction.
- For each transaction, we add a new single record.
- The transaction fact table is known to grow very fast as the number of transactions increases.

: مثال

Fact Types: Transaction Example

customer_id	trns_date	trns_time	call_type	duration
1234	2020-01-01	12:22:45.9	Incoming	29
1234	2020-01-01	12:22:45.9	Incoming	3134
1234	2020-01-02	15:22:45.0	Outgoing	890
1234	2020-01-02	15:22:45.0	International	119
1234	2020-01-03	23:22:45.0	Incoming	145
1234	2020-01-03	23:22:45.0	Outgoing	124
1234	2020-01-03	23:22:45.0	Outgoing	1200

Table: Transaction fact example of telecom calls data.

ال grain هنا بيمثل مجموعة من transactions في خلال مده معينه و ال row هنا بيفي transaction واحد per periodic

Fact Types: Periodic Fact Table

- A periodic fact table contains one row for a *group* of transactions over a period.
- It must be from lower granularity to higher granularity hourly, daily, monthly, and quarterly, then yearly.

مثال : هنا مثلا اخذت ال transaction بتابع cust اللي جدوله فوق .. لكن خلال شهر كام ... ف عملناها ف row واحد

Fact Types: Periodic Fact Table Example

cust_id	month_id	incoming	outgoing	international
1234	20200131	3308	2124	119

Table: Periodic fact example of telecom calls data.

ثالث نوع هو ال Accumulated زي ما قولنا ال row ببقي عباره عن Process كامله و ال grain مش خلال وقت معين ... لا ببقي خلال process معين

Fact Types: Accumulated Snapshot Fact Table

- An accumulating fact table stores one row for the entire process.
- It does not accumulate time it accumulates business process.
- A row in an accumulating snapshot fact table summarizes the measurement events occurring at predictable steps between the beginning and the end of a process
- Accumulating Fact tables are used to show the activity of progress through a well-defined process and are most often used to research the time between milestones.
- These fact tables are updated as the business process unfolds, and each milestone is completed.

و بعض الامثله على هذا ... زي مثلاً أي حاجه فيها Life Cycle .. زي مثلاً ال بتاع الاوردرات مرحله التجهيز ... مرحله التعبئه ... مرحله الشحن ... مرحله الاستلام ... بعد ما كل المراحل دي تخلص.. هنعمل ال row

Fact Types: Accumulated Snapshot Fact Table Example

- Accumulated Snapshot use cases are engaged when we need to report the entire process life-cycle. Fact Types: Accumulated Snapshot Use Cases.
- It also uses to measure the process performance life-cycle.
 - Order life-cycle.
 - Insurance processing.
 - Hiring process.

فمثلاً هنا لو مثال ال insurance ليها LifeCycle معين .. حد كان عامل تامين و عاوزين نصلحه حاجه مثلاً ف بتتشي ب Process معينه ومن ثم بعد ما ال Process دي بتخلص بنحطها لك One Row بس خلي بالك .. احنا هنا مهتمين اننا نحسب كل Stage من ال Process دي ابتدئي امتى .. من وقت بدايه ال Request يعني مثلاً عاوز نعرف اخذت وقت اد اي من بدايه ال Request لحد ما أوصل ل ال Review و وقت اد اي من بدايه ال Request لحد ما أوصل ل ال Decision ... وفيه ناس بتحط الوقت بين كل stage عادي

Fact Types: Accumulated Snapshot Fact Table Example

Example of Accumulated Snapshot: An insurance company

- Its fact table named: fact_claim_processing.
- This fact represents the claim life-cycle inside the company.
- It contains detail related to claim.
- This fact updates after each stage finished.
- The requirement is to report the number of days (lag) between stages (milestone) and the claim data (starting).



Figure: Claim Life-Cycle

طب ازاي بن Implement ال Accumulated Snapshot ... عندنا طریقتین الاولی : ال Slowly Changing Dimension و عشان اسحب ال time بين كل stage و اول Stage ف هستخدمن Sub Quieris الثانيه : ان ال Table نفسه يكون فيه الوقت اللي بيستهلك بين كل stage والثانيه ... اضيف كولمنز عادي

Fact Types:Accumulated Snapshot Example

- One solution to implement the requirement is to use SCD.
- In this case, we will have stages and dates, and we will calculate the difference between stages and dates using complex sub-query.
- Another solution is to implement an accumulated snapshot fact.

FACT_CLAIM_PROCESSING
CLAIM_KEY
CUSTOMER_KEY
POLICY_KEY
CLAIM_DATE
INVESTIGATION_DATE
REVIEW_DATE
DECISION_DATE
PAYMENT_DATE

ف هتبقى كدا

Fact Types:Accumulated Snapshot Example

FACT_CLAIM_PROCESSING	FACT_CLAIM_PROCESSING_ACCUM
CLAIM_KEY CUSTOMER_KEY POLICY_KEY CLAIM_DATE INVESTIGATION_DATE REVIEW_DATE DECISION_DATE PAYMENT_DATE	CLAIM_KEY CUSTOMER_KEY POLICY_KEY CLAIM_DATE INVESTIGATION_DATE DAY_TO_INVESTIGATE REVIEW_DATE DAY_TO REVIEW DECISION_DATE DAY_TO_DECISION PAYMENT_DATE DAY_TO_PAYMENT

وبالتالي دا هيقي شكل الداتا ... العدد 2 و 6 و 7 و 10 دا الوقت اللي بين ما ال stage دي هتببدأ وبين اول # Stage اخر Column مخطوط لك flag وقيمه يا 0 يا 1 عشان اعرف بيها ال process كلها تمت ولا لا ... ال 10 دي غلط

Fact Types: Accumulated Snapshot Table Example

column_name	column_value
claim_key	123
customer_key	5235326
policy_key	23632623
claim_date	2020-01-01
investigation_date	2020-01-03
day_to_investigate	2
review_date	2020-01-07
day_to_review	6
decision_date	2020-01-08
day_to_decision	7
payment_date	2020-01-11
day_to_payment	10
process_completed_flag	10

Table: Accumulated Snapshot Fact Example on Claim Process Data.

Fact Tables Types ال 3 نخص

Low Grain .. قولنا ان ال Date Dimension# transaction كل row بيسجل.. ف كدا

اخر حاجه اللي هي Update معاناها بعد ما يحصل تعديل .. ف ال transaction بتحطه ك new row

و كذلك ال ... انما ال Accumulated بتحطه نفس ال row ... لانها بستني ال Process كلها تخلص

Fact Table Types: Comparison

Feature	Transaction	Periodic	Accumulating
Grain	1 row/transaction	1 row/time-period	1 row/entire event stages
Date Dimension	Lowest granularity	End-of-period granularity	Multiple date
Facts	Transaction activities	Periodic activities	Defined lifetime activities
Size	Largest	Medium	Smallest
Update	No	No	Yes, after stage finished

Table: Fact tables types comparison.

أنواع ال Fact بقى هما 3 : ال Fact هو أي Column في ال Fact Table غير ال FKs

Fact types

6

Each fact table includes facts and it has different types:

- Additive facts.
- Semi-additive facts.
- Non-additive facts.
- Derived facts.
- Textual facts.
- Factless fact.

أولاً : ال Additive column هو لما تستخدمه مع أي FKs في ال Fact Table هيديك معني

Additive facts

- It is the most flexible and useful facts.
- Its measures can be summed across any of the dimensions associated with the fact table.

مثال : هنا ال Sale-Amout نوعه Additive ... لأننا لما نستخدم ال sales-amount مع ال Date او Store او Product هيديك معني (حجم المبيعات خلال مده Dim Date) او تعرف حجم مبيعات الاستور او حجم مبيعات المنتجات

Additive facts

- It is the most flexible and useful facts.
- It can be summed across any of the dimensions associated with the fact table.

Sales

Date
Store
Product
Sales_Amount

ثانياً : ال Semi Additive : وهو Column لو استخدمته من بعض ال FKs في ال Fact table اللي في ال زى هنا مثلاً هنا ال Current_balance لو استخدمته مع ال Date او ال Account هيديك معني ... انما لو استخدمته مع ال Profit مش هيديك معني او اي (Column) لو استخدمته مع بعض الكولمنز مش شكلها = هيديك معني)

Semi-additive facts

- It can be added across some dimensions but not all also known as (partially-additive).

account_details

Date
Account
Current_Balance
Profit_Margin

- what's the total current balance for all accounts in the bank?
- What's the current balances for a given account for each day of the month does not give us any useful information?

ثلاثاً : ال Non – Additive

هو ال Column اللي لو استخدمته من أي Fks ف ال Fact Table مش هيضيف أي معنى للDims زوي ال Profit_Margin هو كولمن بيعبر عن نسبة مئوية .. غالباً ال NonAdditive

Non-additive facts

- It can't be added for any of the dimensions.
- Non-additive facts are usually the result of ratios (percentage) or other mathematical calculations.
- **Profit_Margin** is an example non-additive.

account_details

Date
Account
Current_Balance
Profit_Margin

رابعاً : ال Derived هو كولمن بيتحسب في ال Run Time

Derived facts

- Derived facts are created by performing a mathematical calculation on a number of other facts, and are sometimes referred to as calculated facts. Derived facts may or may not be stored inside the fact table.
- $\text{Total_sales} = \text{Qty_Sold} * (\text{Unit_price} - \text{Discount})$

Order_Details

Order_id
Item_id
Order_date
Qty_Sold
Unit_price
Discount
Total_sales

خامساً : ال Textual هو كولمن بيعبر عن text او رموز او أي حاجه غير ارقام .. واحنا بنحاول دايماً ميكونش عندنا كولمنز بتعبّر عن text .. يفضل كلها تبقي ارقام

Textual facts

- A textual fact consists of one or more characters such as flags and indicators.
- **It should be avoided in the fact table.**

سادساً : ال Factless : وهو ال Fact table اللي مفيهوش غير FKs

Factless fact

- A fact table with only foreign keys and no facts is called a factless fact table.

S17 : هنكلم عن أنواع ال Schema اللي بنربط بيهم ال Facts بال Fact table بتوعه ... مع ال Dims هنكلم عن أشهر اتنين .. ال Snowflake Schema و ال Star Schema

Schema Types

- Star Schema.
 - Snowflake Schema.
- أولاً : ال Star Schema

هي الأبسط وأقل ك performance لان بيحصل عندي Redundancy نتيجة ان ال Dims ليست Normalized بالإضافة ان بيحصل عليها joins قليلاً نتيجة ان كل ال Dims مرتبطة مباشرة ب Fact table فيه Tools كتيره بتدعها عن طريق انك بتقدر من خلالها تعمل Implementation ودا لان ال Star Schema بقى لها بسيط

Star Schema Characteristics

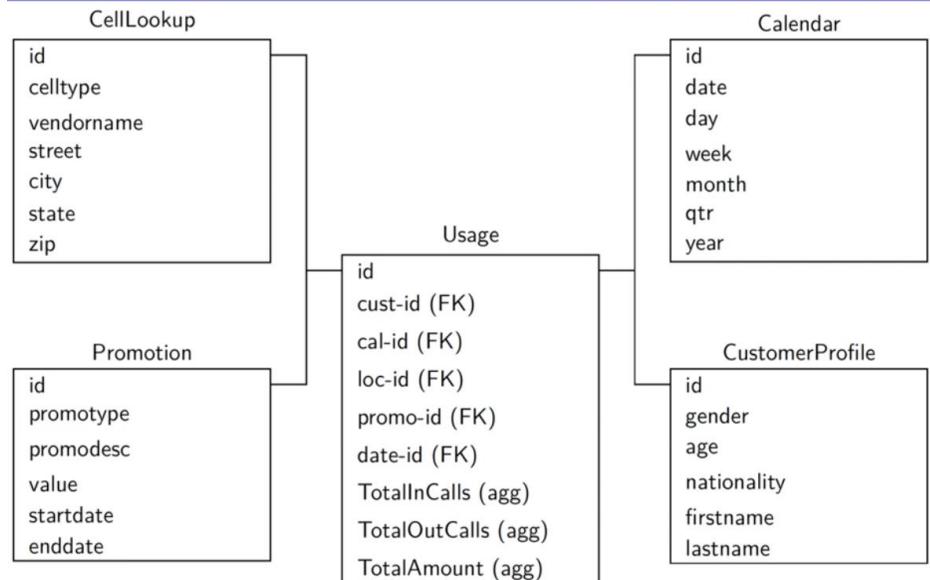
- **Simplicity:** It is the simplest type of DWH schemas.
- **Query effectiveness:** Because of simplicity, It needs less join to query the data (It is optimized to query large dataset).
- **Data Redundancy and Large Table Size:** Due to de-normalization, it has a data redundancy, and the table size is huge.
- **Most used and widely supported.**

و اهم حاجه بما انها مش Data Integrity ف دا معناه ان ال Data Integrity فيها مش قويه لأننا قولنا ال Data Integrity = Non Duplication = Normalized

Star Schema Characteristics

- Dimensions represented by one one-dimension table.
- The dimension table are not joined to each other
- The fact table would contain key and measure.
- Data integrity is not enforced due to the de-normalized structure.

Schema Types: Star Schema Example



ثانياً : ال SnowFlake Schema

وعشان دايماً تفتكـرـها .. خلينـاـ نقولـ انـهـ شـبـهـ الـ Snowـ الثـلـجـ الليـ بـيـطـلـعـ منهـ تـفـرـعـاتـ كـتـيرـهـ .. وـديـ صـورـهـ مـبـسـطـهـ

What is Snowflake?



Figure: Snowflake Simple Design

لكنـ هوـ فيـ الحـقـيقـهـ بـيـقـيـ اـكـبـرـ منـ كـدـاـ بـشـوـيـهـ .. الـ Dimsـ بـتـوـعـ الـ DWHـ كـتـيرـهـ حـبـتـينـ .. حـسـبـ المـشـرـوـعـ طـبـعاـ

What is Snowflake?

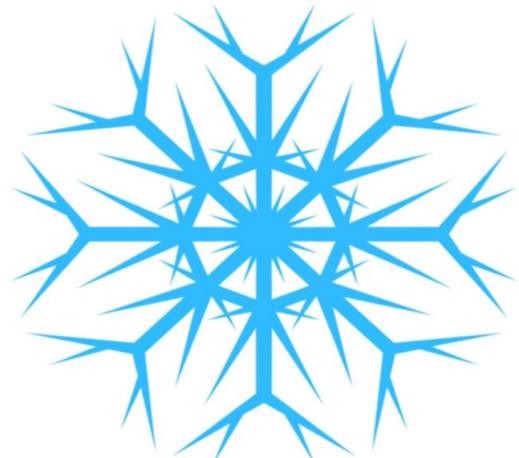


Figure: Snowflake Final Design

الـ SnowFlakeـ بـقـيـ عـبـارـهـ عـنـ Fact~tableـ وـاحـدـهـ وـمـرـبـوطـ بـيـهـ Dimsـ بـسـ مـشـ Levelـ وـاحـدـ منـ الـ Dimsـ .. لـاـ اـكـترـ
مـنـ levelـ ... وـحـطـ فـيـ دـمـاـغـكـ كـدـاـ .. اوـلـ ماـ تـسـمـعـ كـلـمـهـ SnowFlakeـ .. اـرـبـطـهـ بـحـاجـهـ Hierarchyـ

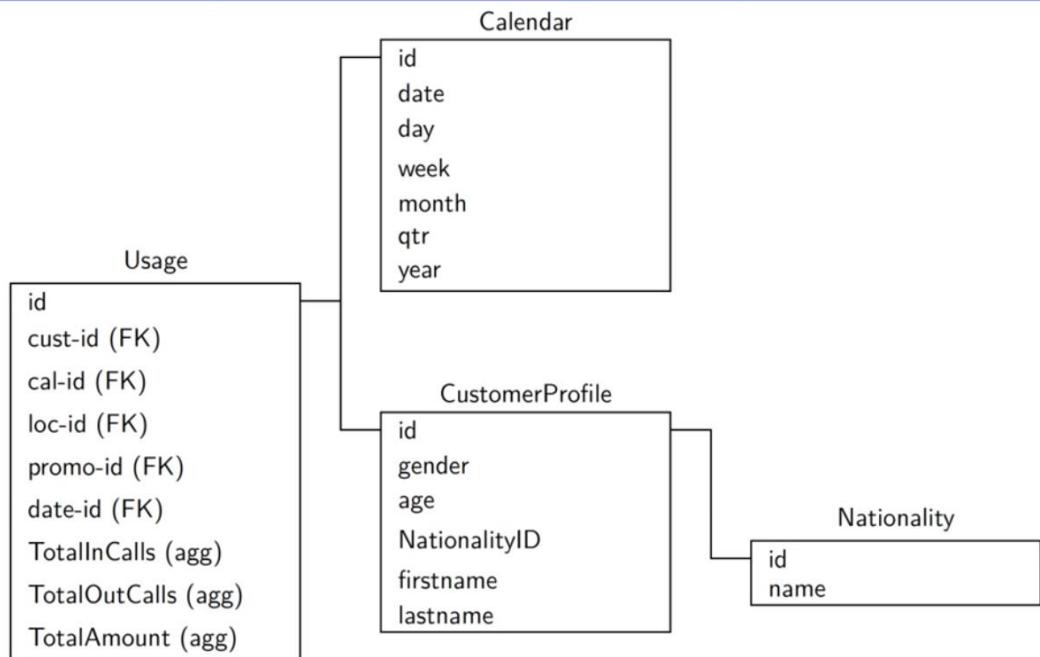
زـيـ الـ SnowFlake Dimـ بـرـضـواـ كـانـتـ Hierarchyـ (SnowFlake = Hierarchy)ـ Hierarchyـ #
كمـانـ هيـ Normalizedـ فـ بالـتـالـيـ مـفـيـشـ Redundancyـ وـ Data~Integrityـ #
use less desk Spaceـ فـ بالـتـالـيـ فيـهـ Joinsـ كـتـيرـهـ .. فـ بـتـقـيـ #
وـ بماـ انـهـ بـرـضـواـ Normalizedـ .. فـ عـنـدـنـاـ اـكـترـ منـ Dimـ فـ بالـتـالـيـ عـنـدـنـاـ Complicatedـ

Snowflake Schema Characteristics

- **Extension:** Snowflake is an extension of the Star Schema.
- **Normalized:** Dimension tables are normalized; this means every dimension may expand into additional tables.
- **Disk Space Efficiency:** Due to its normalization methodology, it uses less desk space, which enhances the query as we scan less data size.
- **Complicated:** Due to the normalization query needs to join more table in some cases to get the data which reduces the performance.

هنا شال ال Nationality اللي هو Multivalued Dim ثاني .. عشان يمنع ال Redundancy الناتجه عن ان ال user ممكن يكون عنده اكتر من Nationality وكدا بقت

Schema Types: Star schema (Example)



وأخيرا هذه مقارنه بينهم ... واحنا اكتر من 90 في الميه بنستخدم ال SnowFlake في شغلنا

Star Vs. Snowflake Schema

Star	Snowflake
Dimension represented by one-table	Dimension tables are expanded into multi-tables
Fact table surrounded by dimension tables	Fact table surrounded by Hierarchy of dimension tables
Less join	Requires many joins
Simple Design	Very Complex Design
De-normalized Data structure	Normalized Data Structure
High level of Data redundancy	Very low-level data redundancy
Maintenance is difficult	Maintenance is easier
Good for datamarts with simple relationships (1:1 or 1:many)	Good for core to simplify (many:many)

