Basic Fact Table Techniques:

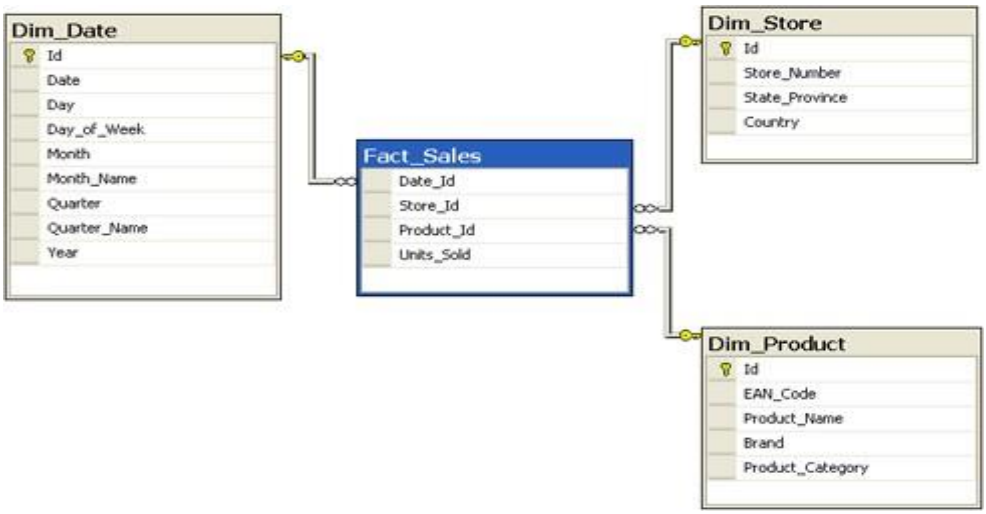
Fact Table Definition

- A Fact Table is a central table in a star schema of a data warehouse. It is an important concept required for ***Data Warehousing and BI Certification***.
- A fact table stores quantitative information for analysis and is often denormalized.
- A fact table works with dimension tables and it holds the data to be analyzed and a dimension table stores data about the ways in which the data can be analyzed.
- A fact table consists of two types of columns. The foreign keys column allows to join with dimension tables and the measure columns contain the data that is being analyzed.

An example of a fact table

In the schema below, we have a fact table FACT_SALES that has a grain that gives us the number of units sold by date, by store, and product.

All other tables such as DIM_DATE, DIM_STORE and DIM_PRODUCT are dimensions tables. This schema is known as the star schema.



Measure types

A fact table can store different types of measures such as additive, non-additive, semi-additive.

Addictive Type:

Additive facts can be used with any aggregation function like Sum(), Avg() etc.

Measurements in a fact table that can be summed up across all dimensions

Consider the following retail fact table:

STORE-KEY	PRODUCT-KEY	TRANSACTION - DATE	REVENUE
S1	P1	27 AUG 2010	100
S2	P1	27 AUG 2010	150
S3	P1	27 AUG 2010	300
S1	P2	28 AUG 2010	600
S2	P2	28 AUG 2010	300
S1	P1	29 AUG 2010	200
S2	P2	29 AUG 2010	100
S3	P3	29 AUG 2010	400

Store wise sales

S1	900
S2	550
S3	700

Product-wise

sales

P1	750
P2	1000
P3	400

Daily Sales

27 Aug 2010	550
28 Aug 2010	900
29 Aug 2010	700

Semi Addictive Fact:

Measurements in a fact table that can be summed up across only a few dimensions keys

Semi-additive facts are those where only a few of aggregation function can be applied

Following table is used to record current balance and profit margin for each id at a particular instance of time (Day end)

Acct_Id	Trans_Dt	Curr_Bal	Profit margin
21653	27Aug09	80000	0.06
21654	27Aug09	120000	0.08
21653	28Aug09	22000	0.08
21654	28Aug09	48000	012

In the above table, we cannot sum up current balance across Acct Id

If we ask balance for Id 21653 we will say that 22000, not 22000+8000

Non-addictive fact:
Facts that cannot be summed up across any dimension key

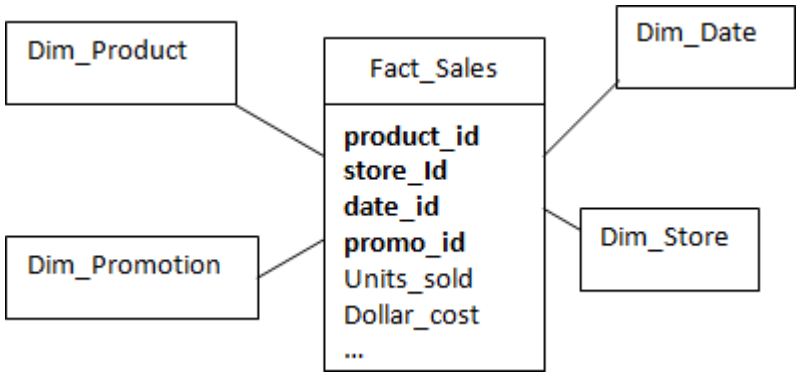
You cannot use numeric aggregation functions such as Sum(), Avg() etc on Non-additive facts

Note: % and ratio columns are non-addictive facts

Factless fact table:

A fact table without any measures is called the factless fact table

- It contains only keys
- It acts as a bridge between dimension keys



Fact Table Types
There are main three types of fact tables:

- Transaction.
- Periodic.
- Accumulated Snapshot.

Transaction Fact Table:

- The transaction fact table is a basic approach to operate the businesses.
- These fact tables represent an event that occurs at the primary point.

- A line exists in the fact table for the customer or product when the transaction occurs.
- Many rows in a fact table connect to a customer or product because they are involved in multiple transactions. Transaction data is often structured quickly in a one-dimensional framework.
- The lowest-level data is the rawest dimensional data that cannot be done by summarized data.

An example of a Transaction Fact Table is shown below.

Transaction id	Customer id	Ordered amount	Order type	Ordered date
101	1234	1000	Purchase	10-Aug-2000
102	1234	5000	Purchase	11-Aug-2000
103	1234	1000	Refund	14-Aug-2000
104	1234	15000	Purchase	15-Aug-2000
105	1234	500	Refund	17-Aug-2000

Periodic Snapshot Fact Table

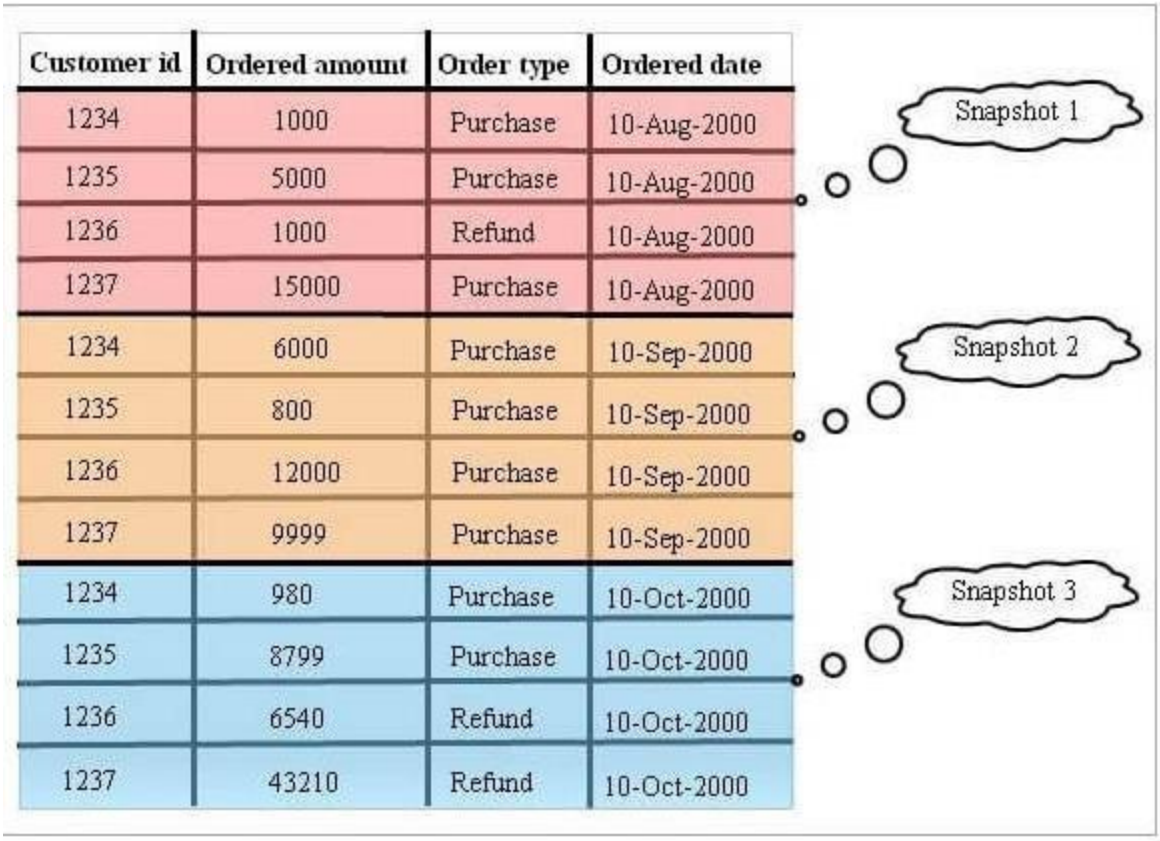
- The snapshot fact table describes the state of things at a particular time and contains many **semi-additive** and **non-additive** facts.

Example: The daily equilibrium fact is expressed by the customer dimension but not by the time dimension.

- Periodic snapshots require the performance of the business at regular and estimated time intervals.
- Unlike a transaction fact table where we load a row for each event, with periodic snapshots, we take a picture of the activity at the end of the day, week, or month, and then another picture at the end of the next period.

Example: Performance summary of a salesman during the previous month

An example of a Periodic Snapshot Fact Table is shown below.

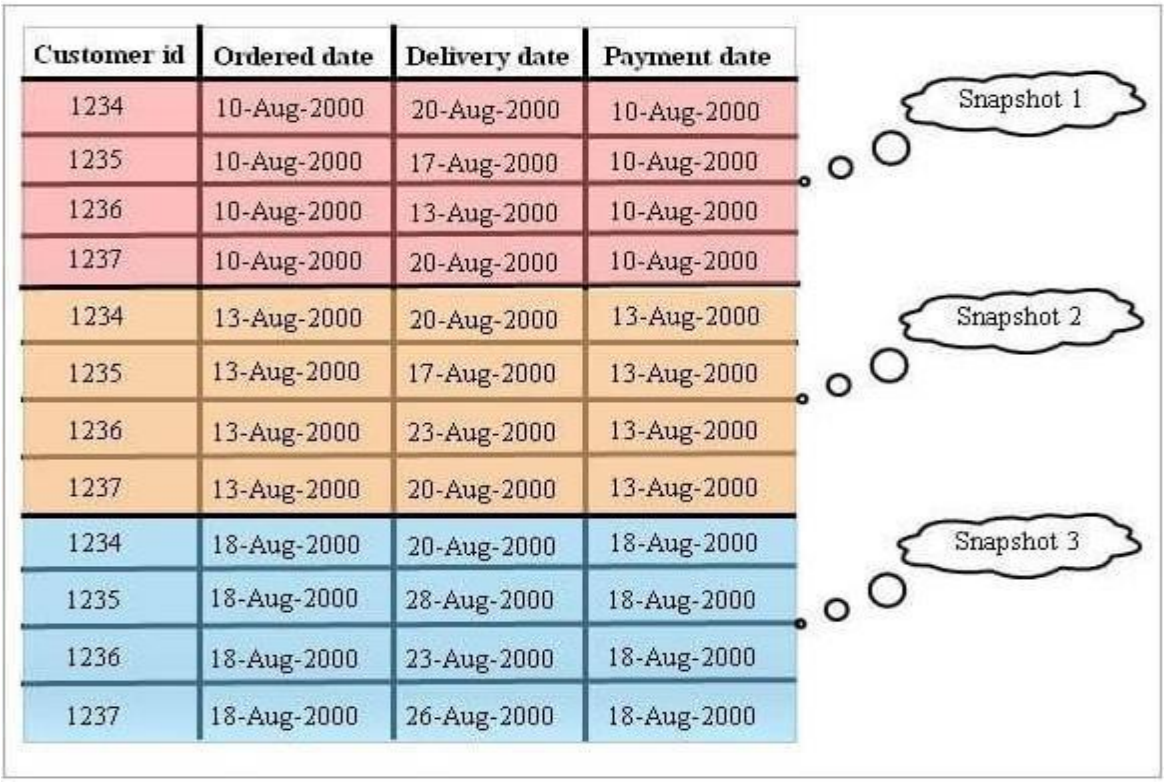


Customer id	Ordered amount	Order type	Ordered date
1234	1000	Purchase	10-Aug-2000
1235	5000	Purchase	10-Aug-2000
1236	1000	Refund	10-Aug-2000
1237	15000	Purchase	10-Aug-2000
1234	6000	Purchase	10-Sep-2000
1235	800	Purchase	10-Sep-2000
1236	12000	Purchase	10-Sep-2000
1237	9999	Purchase	10-Sep-2000
1234	980	Purchase	10-Oct-2000
1235	8799	Purchase	10-Oct-2000
1236	6540	Refund	10-Oct-2000
1237	43210	Refund	10-Oct-2000

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**.**

An example of an Accumulating Snapshot Fact Table.



Fact Table Types: Comparison

Fact tables types comparison.

Feature	Transaction	Periodic	Accumulating
Grain	1 row/transaction	1 row/time-period	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

Null Facts

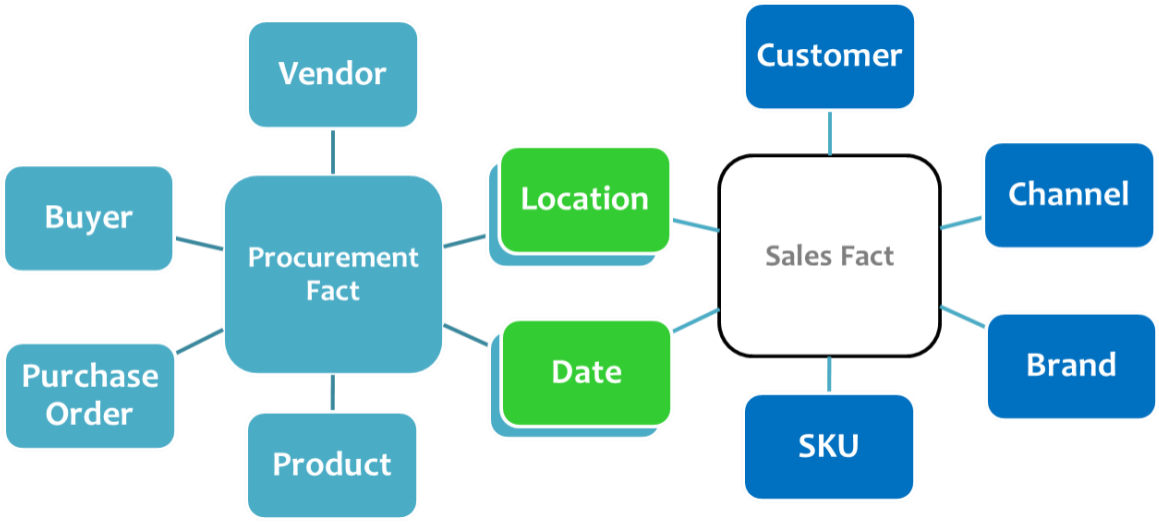
Null-valued measurements behave gracefully in fact tables. The aggregate functions (SUM, COUNT, MIN, MAX, and AVG) all do the “right thing” with null facts.

However, nulls must be avoided in the fact table's foreign keys because these nulls would automatically cause a referential integrity violation.

Conformed fact table

- Conformed fact in a warehouse allows itself to have same name in separate tables. They can be compared and combined mathematically.
- Conformed dimensions can be used across multiple data marts. These conformed dimensions have a static structure.
- Any dimension table that is used by multiple fact tables can be conformed dimensions

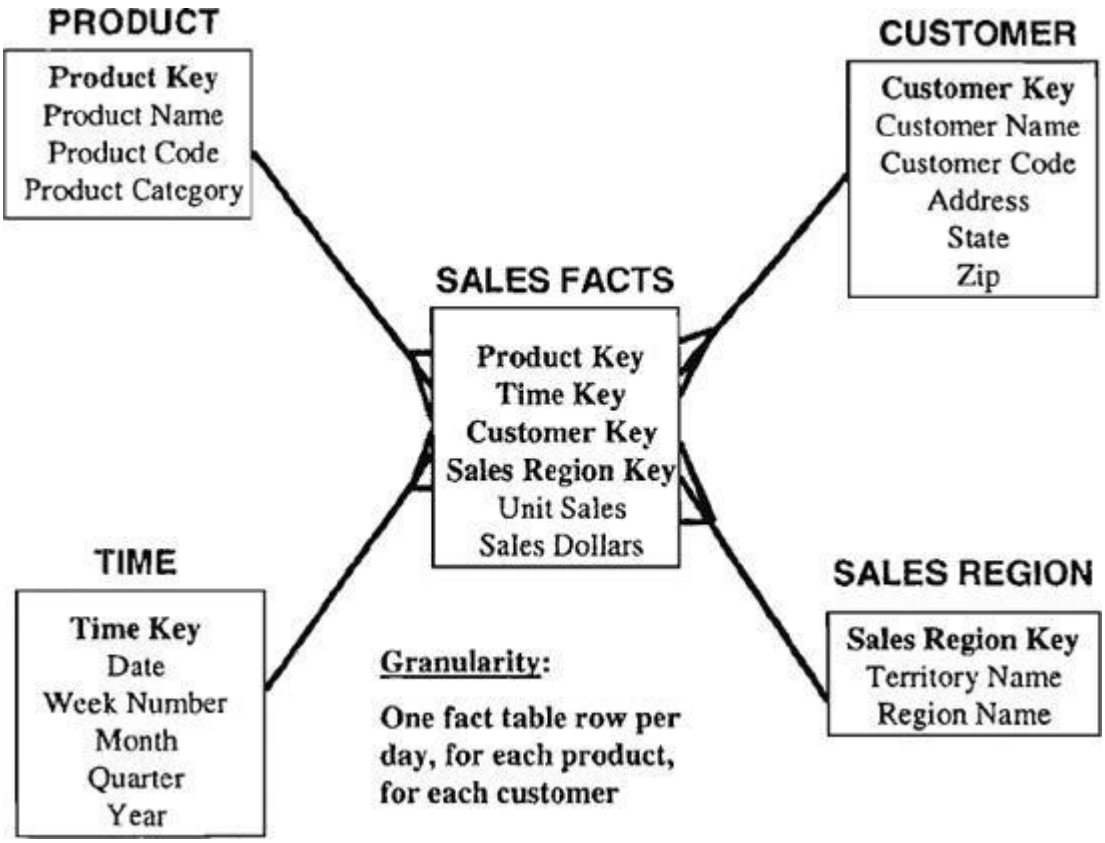
.



AGGREGATE FACT TABLES

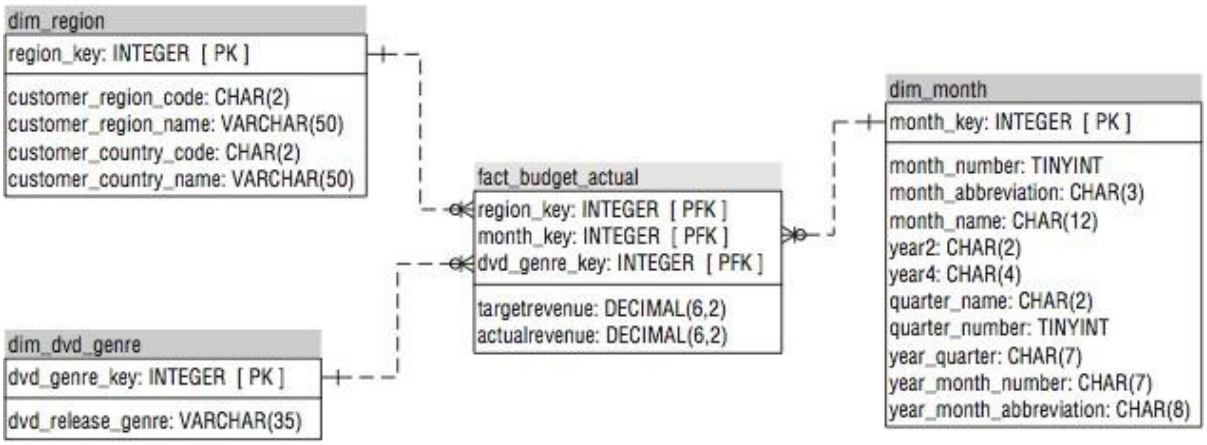
- Aggregates are precalculated summaries derived from the most granular fact table. These summaries form a set of separate aggregate fact tables.

- These queries retrieve hundreds and thousands of table rows, manipulate the metrics in the fact tables, and then produce the result sets.
- The manipulation of the fact table metrics may be a simple addition, an addition with some adjustments, a calculation of averages, or even an application of complex arithmetic algorithms.



Consolidated Fact Table

- It is often convenient to combine facts from multiple processes together into a single *consolidated fact table* if they can be expressed at the same grain.
- For example, sales actuals can be consolidated with sales forecasts in a single fact table to make the task of analyzing actuals versus forecasts simple and fast, as compared to assembling a drill-across application using separate fact tables.
- Consolidated fact tables add burden to the ETL processing, but ease the analytic burden on the BI applications. They should be considered for cross-process metrics that are frequently analyzed together.



Basic Dimension Table Techniques:

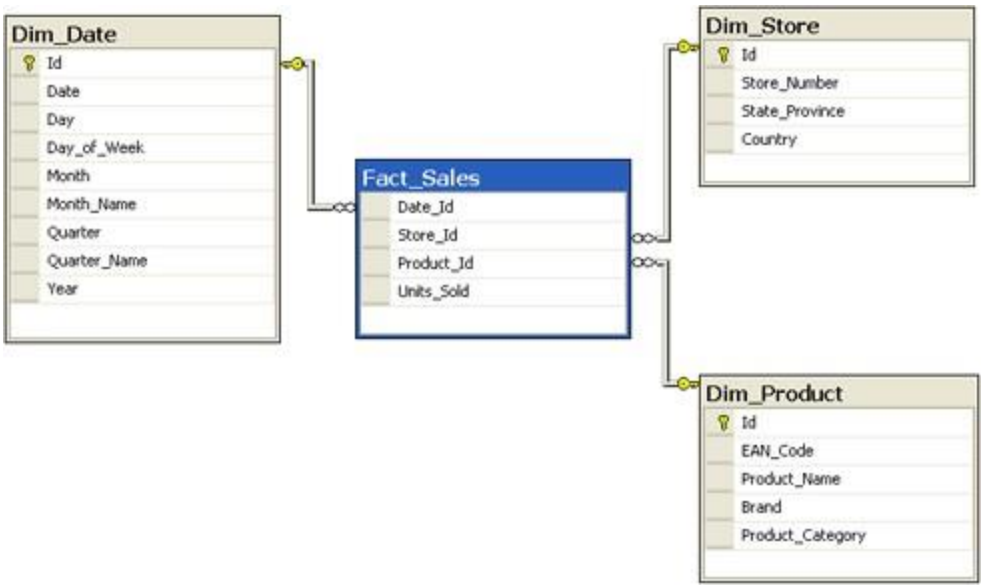
Dimension Table Structure:

- A Dimension Table is present in the star or snowflake schema.
- Dimension tables' help to describe dimensions i.e. dimension values, attributes and keys. It is generally small in size. Size can range from several to thousand rows.
- It describes the objects present in the fact table.
- Dimension Table refers to the collection or group of information related to any measurable event. They form a core for dimensional modelling.
- In data warehousing, a dimension table is one of the companion tables to a fact table in the star schema. Different from a fact table that contains measures or business facts, a dimension table contains the textual descriptor of the business.
- The fields of the dimension table are designed to satisfy these two important requirements:
    - ❖ Query constraining / grouping / filtering.
    - ❖ Report labeling

- It contains a column that can be considered as a primary key column which helps to uniquely identify every dimension row or record. It is being joined with the fact tables through this key
- Its help to store the history of the information or dimensional information.
- Its is easy to understand than the normalized tables.
- More columns can be added to the table without affecting the existing applications that are using those.

Dimension table example

In the schema below we have 3 dimension
tables Dim_Date, Dim_Store and Dim_Product surrounding the fact table Fact_Sales.



Surrogate keys in dimension tables

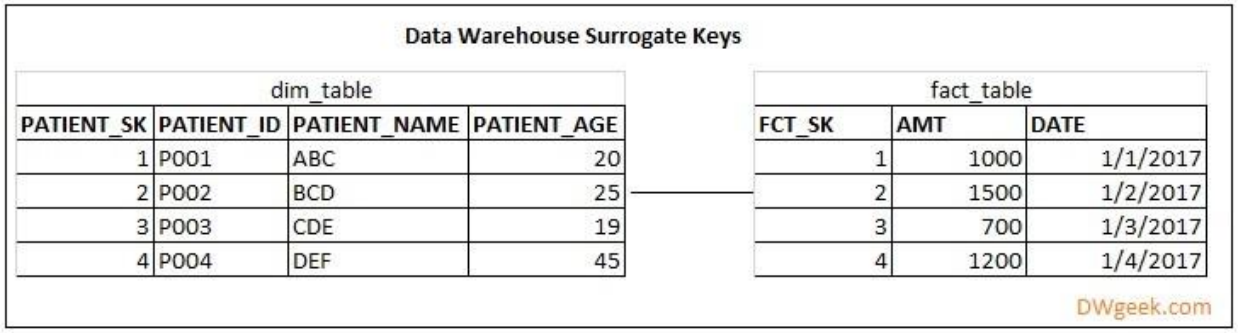
- It is critical that the primary key's value of a dimension table remains unchanged. And it is highly recommended that all dimension tables use surrogate keys as primary keys.
- Surrogate keys are key generated and managed inside the data warehouse rather than keys extracted from data source systems.

There are several advantages of using surrogate keys in dimension tables:

- ❖ **Performance** – join processing between dimension tables and fact table is much more efficient by using a single field surrogate key.
- ❖ **Integration** – in terms of data acquisition, the surrogate key allows integrating data from multiple data sources even if they lack consistent source keys.
- ❖ **Manage version of data** – keep track of changes in dimension field values in the dimension table.

- It is so important that the dimension tables should be designed in such a way that they can be shared between multiple data marts and cubes within a data warehouse. This ensures that the data warehouse provides consistent information for similar queries.

- Surrogate key must be used as the primary keys of dimension tables to enable the dimension tables to be shared easier.



Natural, Durable and Supernatural Keys:

- In data warehouse tables, natural keys are meaningful values that identify records, such as social security numbers that identify specific customers, calendar dates in a time dimension, or SKU numbers in a product dimension.
- In some cases, natural keys are unique identifiers and can serve as primary keys. Dates in a time dimension are reliable in this way.
- For example, the following records contain dates as unique natural keys:

2000-03-30 TH 14 MAR Q1_2000 2000
2000-03-31 FR 14 MAR Q1_2000 2000

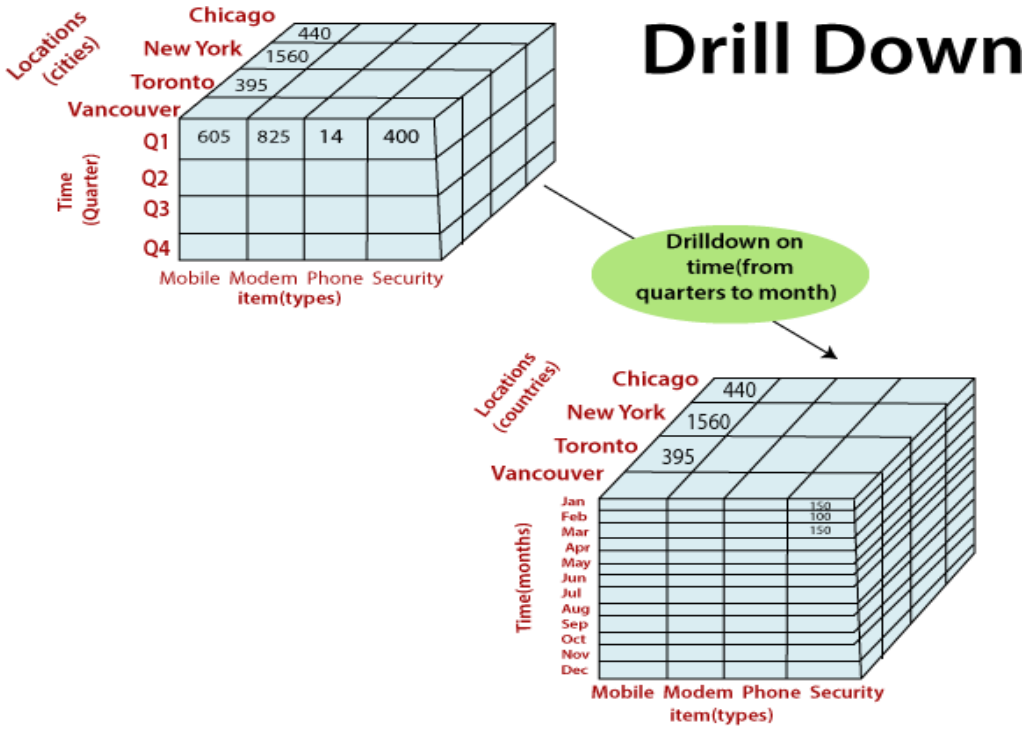
- some natural keys are not durable enough to serve as primary keys.
- Natural keys are not excluded from dimension entities, but they must be designed as business attributes.
- For instance, an employee number (natural key) may be changed if the employee resigns and then is rehired.
- When the data warehouse wants to have a single key for that employee, a new *durable key* must be created that is persistent and does not change in this situation. This key is sometimes referred to as a *durable supernatural key*.
- The best durable keys have a format that is independent of the original business process and thus should be simple integers assigned in sequence beginning with 1. While multiple surrogate keys may be associated with an employee over time as their profile changes, the durable key never changes.

Account_Key	Account_ID	Durable_Account_ID	Account Holder	State	Eff_Date	End_Date
3	8765	3	Smith	CA	2011-02-01	2011-05-10
7	8765	3	Smith	OR	2011-05-11	2011-10-23
23	9251	3	Smith	OR	2011-10-24	2011-12-31
55	9251	3	Smyth	OR	2012-01-01	9999-12-31

- The supernatural key is qualified as "durable" because it is an identifier that uniquely and reliably identifies the dimension entity across its attribute changes.
- The durable supernatural key keeps the same value in all rows that represent the different versions of the dimension over time.
- In some cases, the durable supernatural key can be an alternative for representing the dimension key in fact entities, such as with slowly changing facts.

Drilling down Dimension:

- The drill-down operation **(also called roll-down)** is the reverse operation of **roll-up**.
- Drill-down is like **zooming-in** on the data cube. It navigates from less detailed record to more detailed data.
- Drill-down can be performed by either **stepping down** a concept hierarchy for a dimension or adding additional dimensions.
- Figure shows a drill-down operation performed on the dimension time by stepping down a concept hierarchy which is defined as day, month, quarter, and year.
- Drill-down appears by descending the time hierarchy from the level of the quarter to a more detailed level of the month.
- Because a drill-down adds more details to the given data, it can also be performed by adding a new dimension to a cube. The following diagram illustrates how Drill-down works.



Degenerate Dimension–

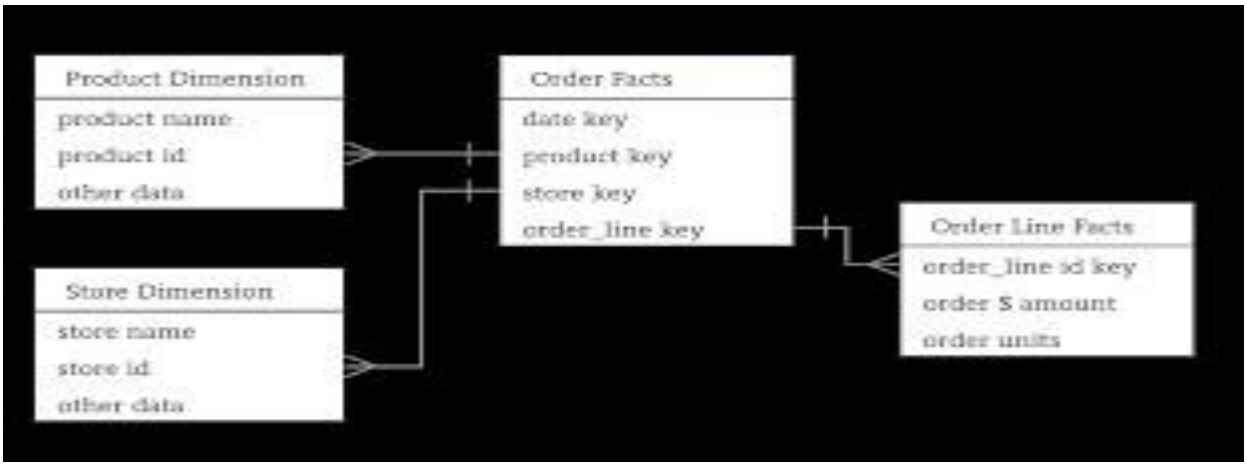
- A degenerate dimension is when the dimension attribute is stored as part of the fact table and not in a separate table.
- Product id comes from product dimension table. Invoice number is a standalone attribute and has no other attributes associated with it.
- An invoice number can be crucial since the business would want to know the quantity of the products.

Invoice_Number	Product_id	Quantity	Amount
11223344	12345	4	100
11223344	67892	3	200
44556677	11123	2	300
11223344	44567	1	400

Denormalized flattened dimensions:

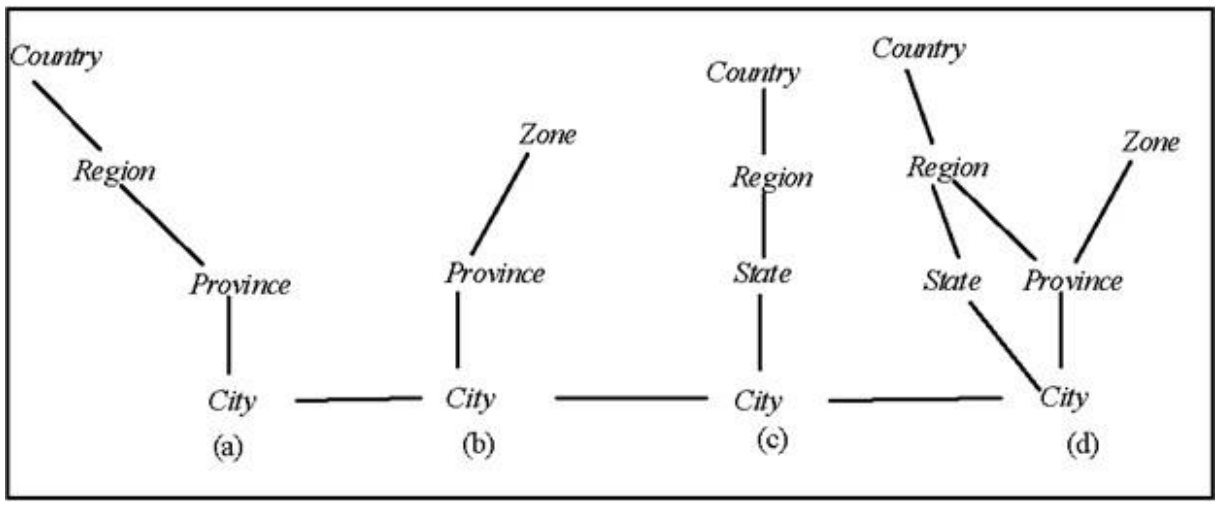
- Dimensional designers must resist the normalization urges caused by years of operational database designs and instead denormalize the many-to-one fixed depth hierarchies into separate attributes on a flattened dimension row.

- Dimension denormalization supports dimensional modeling's twin objectives of simplicity and speed.



Multiple hierarchies in dimensions:

- Multiple hierarchies can be defined for dimensions containing level-based hierarchies.
- You create multiple hierarchies for a dimension when you want to organize dimension members in different ways. For example, in a Time dimension, you can create hierarchies for Calendar year and Fiscal year.
- Because dimension members in separate hierarchies can be used to represent the same entity, each hierarchy should contain the same lowest level members.
- For example, in a Time dimension, the Calendar hierarchy might have Year, Month, and Day levels. The Fiscal hierarchy might have Year, Quarter, and Day levels. The lowest level in both dimensions is the Day level.
- Hierarchies that are modeled using a shared level can be optimized during query execution to remove non-intersecting values. To do this, you must ensure the **Remove non-existent tuples** property is set in a dynamic cube.



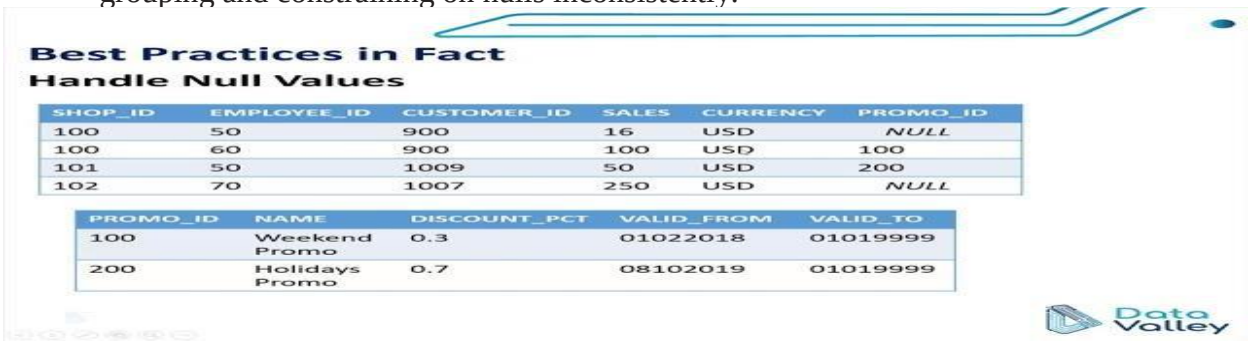
Flags and indicators as dimension attributes:

- Cryptic abbreviations, true/false flags, and operational indicators should be supplemented in dimension tables with full text words that have meaning when independently viewed.
- Operational codes with embedded meaning within the code value should be broken down with each part of the code expanded into its own separate descriptive dimension attribute.

True	False
Yes	No
Y	N
1	0

Null Attributes in Dimensions

- Null-valued dimension attributes result when a given dimension row has not been fully populated, or when there are attributes that are not applicable to all the dimension's rows.
- In both cases, we recommend substituting a descriptive string, such as Unknown or Not Applicable in place of the null value.
- Nulls in dimension attributes should be avoided because different databases handle grouping and constraining on nulls inconsistently.



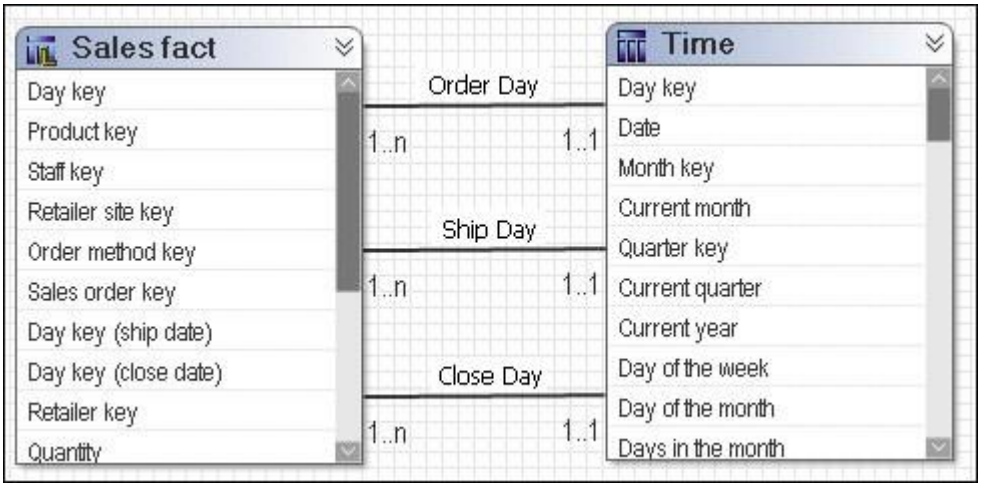
Calendar Date Dimensions:

- A date dimension contains a continuous range of dates that cover the entire date period required for the analysis.
- It also includes columns that will allow a user to filter the data by almost any date logic.It can include the day of the week, workdays, weekends, quarters, months, years, or seasons.
- *Calendar date dimensions* are attached to virtually every fact table to allow navigation of the fact table through familiar dates, months, fiscal periods, and special days on the calendar.
- To facilitate partitioning, the primary key of a date dimension can be more meaningful, such as an integer representing YYYYMMDD, instead of a sequentially-assigned surrogate key.

Date	Year	Quarter	QuarterID	MonthNum	MonthID	MonthName	DayOfMonth	DayofWeek	DayName
2021-01-01	2021	1	2021Q1	1	202101	Jan	1	5	Friday
2021-01-02	2021	1	2021Q1	1	202101	Jan	2	6	Saturday
2021-01-03	2021	1	2021Q1	1	202101	Jan	3	0	Sunday
...									
2021-12-27	2021	4	2021Q4	12	202112	Dec	27	1	Monday
2021-12-28	2021	4	2021Q4	12	202112	Dec	28	2	Tuesday
2021-12-29	2021	4	2021Q4	12	202112	Dec	29	3	Wednesday
2021-12-30	2021	4	2021Q4	12	202112	Dec	30	4	Thursday
2021-12-31	2021	4	2021Q4	12	202112	Dec	31	5	Friday

Role-Playing Dimensions:

- A table with multiple valid relationships between itself and another table is known as a role-playing dimension. This is most commonly seen in dimensions such as Time and Customer.
- For example, the Sales fact has multiple relationships to the Time query subject on the keys Order Day, Ship Day, and Close Day.



Junk Dimensions:

- A Junk Dimension is a dimension table consisting of attributes that do not belong in the fact table or in any of the existing dimension tables.
- The nature of these attributes is usually text or various flags, e.g. non-generic comments or just simple yes/no or true/false indicators.

- The have low cardinality and usually don't come under SCD.

Junk Dimension Example (Cont.) [3]

FACT_TABLE

CUSTOMER_ID
PRODUCT_CD
TXN_ID
STORE_ID
JUNK_ID
TXN_AMT

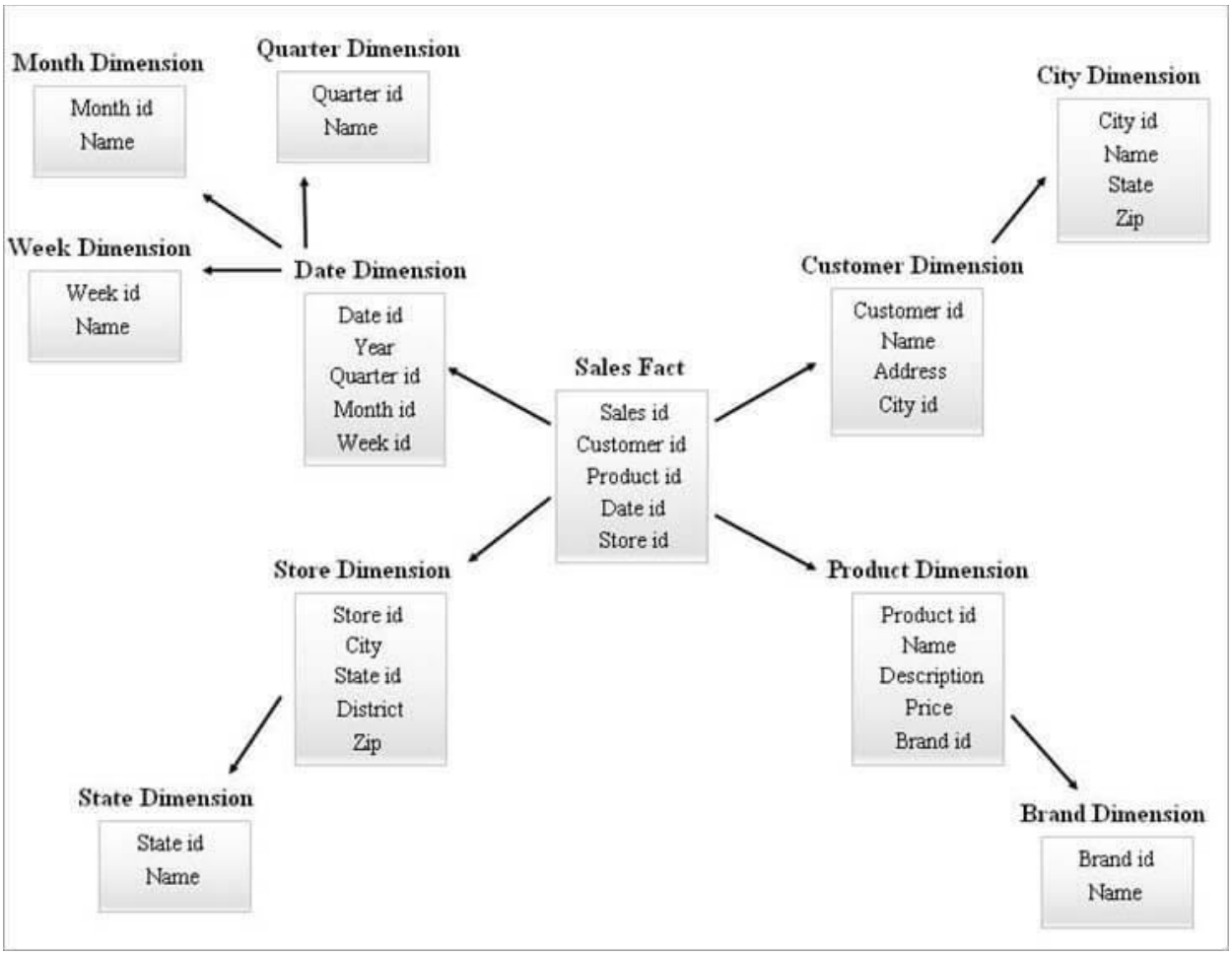
DIM_JUNK

JUNK_ID	TXN_CODE	COUPON_IND	PREPAY_IND
1	1	Y	Y
2	2	Y	Y
3	3	Y	Y
4	1	Y	N
5	2	Y	N
6	3	Y	N
7	1	N	Y
8	2	N	Y
9	3	N	Y
10	1	N	N
11	2	N	N
12	3	N	N

72

Snowflaked Dimensions:

- A **snowflake schema** is a multi-dimensional data model that is an extension of a **star schema**, where dimension tables are broken down into subdimensions.
- Snowflake schemas are commonly used for business intelligence and reporting in OLAP data warehouses, data marts, and relational databases.
- In a snowflake schema, engineers break down individual dimension tables into logical subdimensions.
- This makes the data model more complex, but it can be easier for analysts to work with, especially for certain data types.
- It's called a snowflake schema because its entity-relationship diagram (ERD) looks like a snowflake, as seen below.



Benefits of snowflake schemas

- Fast data retrieval
- Enforces data quality
- Simple, common data model for data warehousing

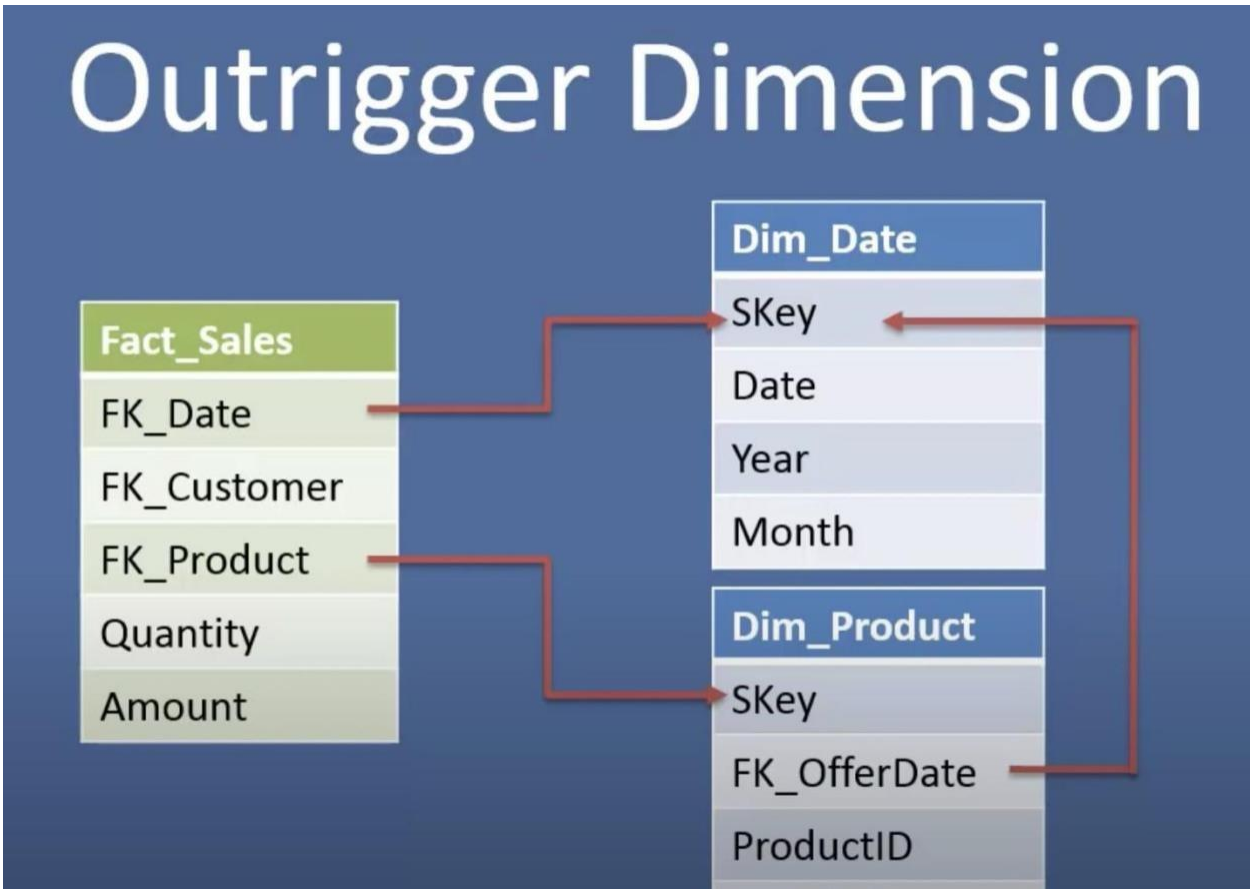
Drawbacks of snowflake schemas

- Lots of overhead upon initial setup
- Rigid data model
- High maintenance costs

Outrigger dimensions:

- An *outrigger* is a dimension table or entity that is joined to other dimension tables in a star schema. Outriggers are used when a dimension table is snowflaked.
- Outriggers are tables or entities that are shared by more than one dimension.
- A table or entity that is included in a hierarchy but is not directly related to the fact table are known as outriggers.

- Outriggers are often used when a dimension table or entity is referenced by another dimension.
- The primary key of an outrigger is referenced by the foreign key of a dimension table or entity.



Slowly Changing Dimension Techniques

- A **slowly changing dimension** (**SCD**) in data management and data warehousing is a dimension which contains relatively static data which can change slowly but unpredictably, rather than according to a regular schedule.
- Some examples of typical slowly changing dimensions are entities as names of geographical locations, customers, or products.
- Some scenarios can cause referential integrity problems.rather than changing regularly on a time basis.
- Slowly changing dimensions are the dimensions in which the data changes slowly, rather than changing regularly on a time basis.

There are total 6 types of SCD that are widely used in the DWH implementation. They are as follows:

- Type 0: This is a fixed dimension.
- Type 1: Maintain only current state.
- Type 2: History is stored.
- Type 3: Current and previous states are stored.
- Type 4: Combination of types 1 and 2.
- Type 6: Hybrid type.

In real world. Only Type 1, 2 and 3 are used. Other types of SCDs are used rarely.

SCD Type 0

- Type 0 is a fixed dimension.The data in this dimension table never changes. The data into this dimension table is loaded one time at the beginning of the project.

- An example for Type 0 is business users data assigned to particular regions. These business users will never change their location. So the data in this dimension never changes. The total sales done by each business user can be generated.

Id	Business User	Location
1	Adam	USA
2	Mark	UK
3	Henry	India

SCD Type 1

- SCD type 1 methodology is used when there is no need to store historical data in the dimension table.

- This method overwrites the old data in the dimension table with the new data. It is used to correct data errors in the dimension.

As an example, assume the customer table with the below data.

surrogate_key customer_id customer_name Location

---

1 1 Mark Chicago

Here the customer Location is Chicago and the customer moved to another location New York. If you use type1 method, it just simply overwrites the data. The data in the updated table will be.

surrogate_key customer_id customer_name Location

---

1 1 Mark New York

- The advantage of type1 is ease of maintenance and less space occupied.
- The disadvantage is that there is no historical data kept in the data warehouse.

SCD Type 2

- SCD type 2 stores the entire history the data in the dimension table. With type 2 we can store unlimited history in the dimension table.
- In type 2, you can store the data in three different ways. They are:
  - Versioning
  - Flagging
  - Effective Date

SCD Type 2 Versioning

- In versioning method, a sequence number is used to represent the change.
- The latest sequence number always represents the current row and the previous sequence numbers represents the past data.

As an example, let's use the same example of customer who changes the location. Initially the customer is in Illinois location and the data in dimension table will look as.

surrogate_key customer_id customer_name Location Version

---

1 1 Marston Illinois 1

- The customer moves from Illinois to Seattle and the version number will be incremented. The dimension table will look as

surrogate_key customer_id customer_name Location Version

---

1 1 Marston Illinois 1

2 1 Marston Seattle 2

Now again if the customer is moved to another location, a new record will be inserted into the dimension table with the next version number.

surrogate_key	customer_id	customer_name	Location	Version
1	1	Mark	Illinois	1
2	1	Mark	Seattle	2

SCD Type 2 Flagging

- In flagging method, a flag column is created in the dimension table.
- The current record will have the flag value as 1 and the previous records will have the flag as 0.

Now for the first time, the customer dimension will look as.

surrogate_key customer_id customer_name Location flag

---

1 1 Marston Illinois 1

Now when the customer moves to a new location, the old records will be updated with flag value as 0 and the latest record will have the flag value as 1.

surrogate_key customer_id customer_name Location flag

---

1 1 Marston Illinois 0

2 1 Marston Seattle 1

surrogate_key	customer_id	customer_name	Location	Flag
1	1	Mark	Illinois	0
2	1	Mark	Seattle	1

SCD Type 2 Effective Date

- In Effective Date method, the period of the change is tracked using the start_date and end_date columns in the dimension table.

surrogate_key customer_id customer_name Location Start_date End_date

---

1 1 Marston Illinois 01-Mar-2010 20-Feb-2011

2 1 Marston Seattle 21-Feb-2011 NULL

- The NULL in the End_Date indicates the current version of the data and the remaining records indicate the past data.

surrogate_key	customer_id	customer_name	Location	Start_Date	End_Date
1	1	Mark	Illinois	01-Mar-10	20-Feb-11
2	1	Mark	Seattle	21-Feb-11	Null

SCD Type 3

- In type 3 method, only the current status and previous status of the row is maintained in the table. To track these changes two separate columns are created in the table.

- The customer dimension table in the type 3 method will look as

surrogate_key customer_id customer_name Current_Location previous_location

---

1 1 Marston Illinois NULL

Let say, the customer moves from Illions to Seattle and the updated table will look as

surrogate_key customer_id customer_name Current_Location previous_location

---

1 1 Marston Seattle Illinois

Now again if the customer moves from seattle to NewYork, then the updated table will be

surrogate_key customer_id customer_name Current_Location previous_location

---

1 1 Marston NewYork Seattle

The type 3 method will have limited history and it depends on the number of columns you create.

surrogate_key	customer_id	customer_name	Current_Location	Previous_Location
1	1	Marston	Illinois	Null
2	1	Marston	Seattle	Illinois
3	1	Marston	NewYork	Seattle

SCD Type 4

- The scd type 4 is also called as fast growing dimension. Imagine tracking all these changes and storing them in a single dimension (using type3).

- It takes so much time to generate a report when this dimension table is joined with the fact table. To generate the report faster, the data in the dimension table should be minimal.

	Id	Customer	Location
	1	Mark	Seattle

surrogate_key	Id	Customer	Location
1	1	Mark	NewYork
2	1	Mark	Seattle

- In Type 4, the current data is maintained in the dimension table and the history is stored in another table. This improves the performance when generating the report. However it adds an overhead of maintaining the historical data in a separate table.

Type 5

- The type 5 technique builds on the type 4 mini-dimension by embedding a "current profile" mini-dimension key in the base dimension that's overwritten as a type 1 attribute. This approach is called type 5 because 4 + 1 equals 5.

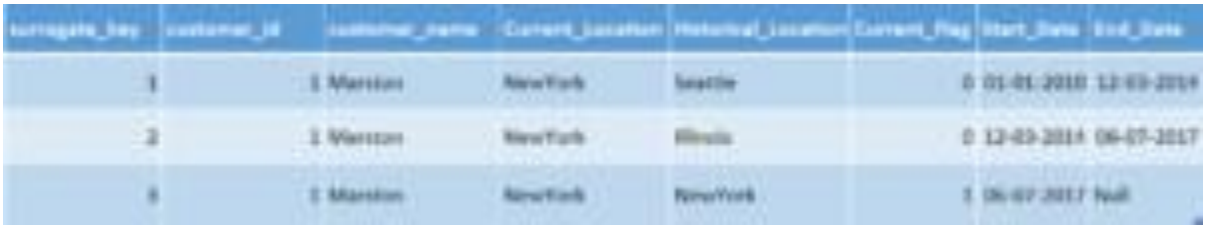
- The type 5 slowly changing dimension allows the currently-assigned mini-dimension attribute values to be accessed along with the base dimension's others without linking through a fact table.

- The outrigger attributes should have distinct column names, like "Current Income Level," to differentiate them from attributes in the mini-dimension linked to the fact table.

- The ETL team must update/overwrite the type 1 mini-dimension reference whenever the current mini-dimension changes over time. If the outrigger approach does not deliver satisfactory query performance, then the mini-dimension attributes could be physically embedded (and updated) in the base dimension.[3]

SCD Type 6

- This is a combination of Type 1, 2 and 3. This is also called as Hybrid type. In this dimension, the current data is stored in all the historical record in a current column.



- This type of dimension adds a lot of complexity. Implementing this SCD type is bit hard and also stores a lot of redundant data. However, this provides an easy way to compare current data with historical data.

Type 7: Hybrid[4] - Both surrogate and natural key[edit]

An alternative implementation is to place *both* the surrogate key and the natural key into the fact table.[5] This allows the user to select the appropriate dimension records based on:

- the primary effective date on the fact record (above),
- the most recent or current information,
- any other date associated with the fact record.

This method allows more flexible links to the dimension, even if one has used the Type 2 approach instead of Type 6.

Here is the Supplier table as we might have created it using Type 2 methodology:

Supplier_Key	Supplier_Code	Supplier_Name	Supplier_State	Start_Date	End_Date	Current_Flag
123	ABC	Acme Supply Co	CA	2000-01-01T00:00:00	2004-12-21T00:00:00	N
124	ABC	Acme Supply Co	IL	2004-12-22T00:00:00	2008-02-03T00:00:00	N

125	ABC	Acme Supply Co	NY	2008-02-04T00:00:00	9999-12-31T23:59:59	Y

To get current records:

```
SELECT
  delivery.delivery_cost,
supplier.supplier_name,
supplier.supplier_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_code = supplier.supplier_code
WHERE supplier.current_flag = 'Y';
```

To get history records:

```
SELECT
  delivery.delivery_cost,
supplier.supplier_name,
supplier.supplier_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_key = supplier.supplier_key;
```

To get history records based on a specific date (if more than one date exists in the fact table):

```
SELECT
  delivery.delivery_cost,
supplier.supplier_name,
supplier.supplier_state
FROM delivery
INNER JOIN supplier
  ON delivery.supplier_code = supplier.supplier_code;
  AND delivery.delivery_date BETWEEN supplier.Start_Date AND supplier.End_Date
```