Maltem Consulting Group

Data Engineering and Big Data

Recap of Day 3

- The importance of Data Engineering
- Skills required for a Data Engineer
- The different phases of the Data Engineering Lifecycle
- Data storage topics:
  - Database
  - Data warehouse
  - ODS
  - OLTP
  - OLAP
  - Data Mart
  - Big Data
- The differences between DW, ODS, OLTP, OLAP, DM, BD

Recap of Day 3

Maltem Consulting Group

- Types of databases
  - Relational DB
  - NoSQL DB
    - Key-Value
    - Columnar
    - Document
    - Graph
- Interaction between MS SQL and MongoDB
- Big Data

Course outline – Day 3

Maltem Consulting Group

Data Modelling

- Key Concept of Relationships
- Introduction to data modelling
- Transaction data modelling
- Analytical data modelling
- Normalization and its types
- Denormalization
- Different types of modelling techniques
- Star schema
- Snowflake schema
- Facts
- Dimension
- SCD

Data Integration

- What is data integration
- Importance of data integration
- List of data integration tools
- ETL vs ELT

Data Integration details

- Introduction
- Operation
- Filter, Aggregation, Sort
- Join
- Union
- Business Logic

Big Data

- Hive
- Kafka
- Spark and demo

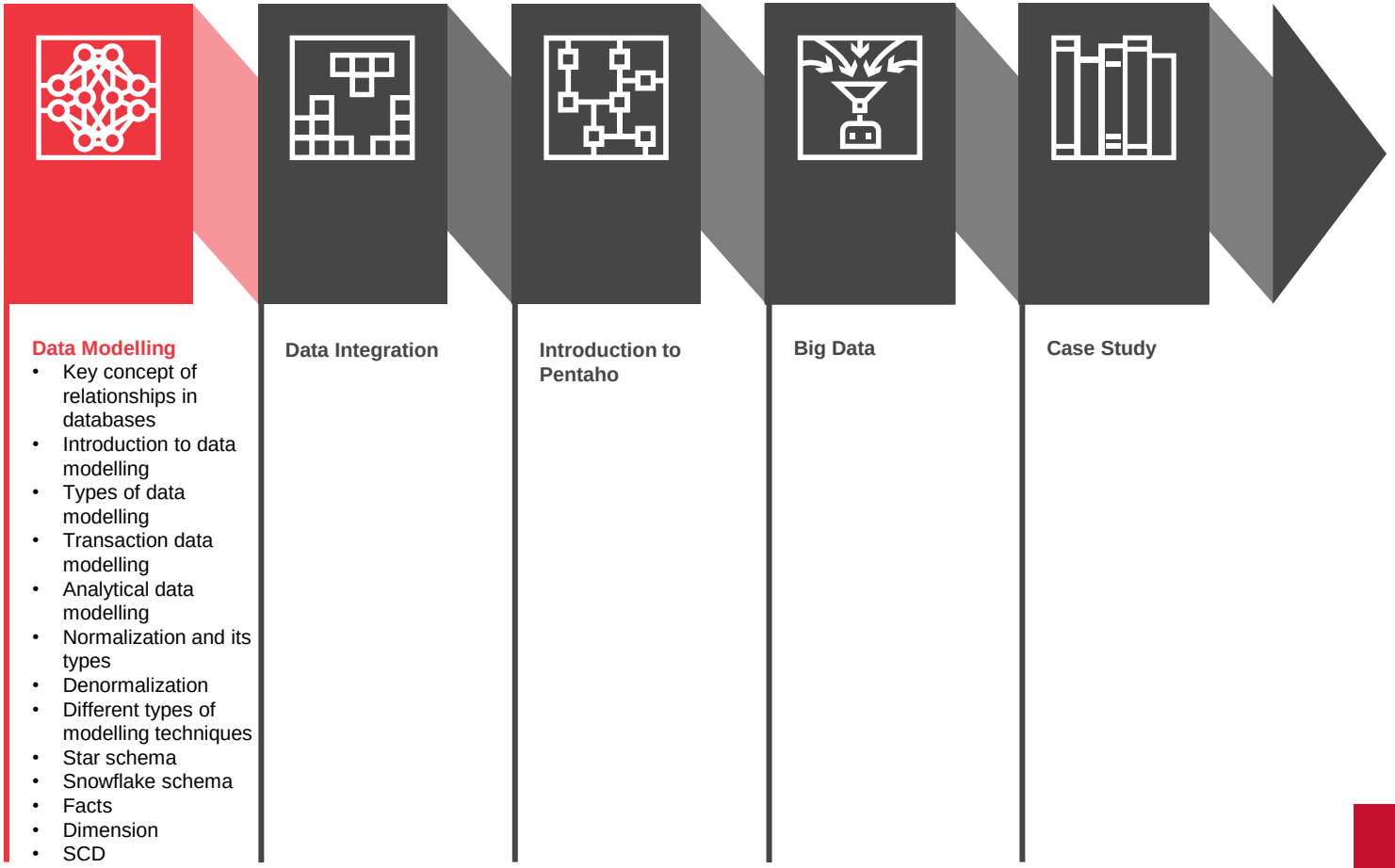
Timetable – Day 3

Maltem Consulting Group

Date: Wednesday, 29 September 2021		Time: 9:00 am to 05:00am	ITE Campus		
	Agenda	Presenter	Time		Duration (mins)
1	Catch up	Mahesh	9:00 AM	10:00 AM	60
2	Understanding Data Modelling	Mahesh	10:00 AM	10:30 AM	30
3	Break	Ben	10:15 AM	10:45 AM	30
4	Normalization	Mahesh	10:45 AM	12:15 PM	90
5	Lunch Break	Ben	12:15 PM	1:15 PM	60
6	Play with Pentaho	Ben	1:15 PM	2:15 PM	60
7	Types of modelling	Mahesh	2:15 PM	2:45 PM	30
8	Break	Ben	2:45 PM	3:15 PM	30
9	Fact Tables	Mahesh	3:15 PM	3:45 PM	30
10	Dimensional Tables	Mahesh	3:45 PM	4:45 PM	60

ITE

Day 2



Data Modelling

- Key concept of relationships in databases
- Introduction to data modelling
- Types of data modelling
- Transaction data modelling
- Analytical data modelling
- Normalization and its types
- Denormalization
- Different types of modelling techniques
- Star schema
- Snowflake schema
- Facts
- Dimension
- SCD

Data Integration

Introduction to Pentaho

Big Data

Case Study

What is a relationship in a database?

- Databases are split into different **tables** to store data related to **common entities** such as customers, orders, items
- In a database, a relationship can be defined as an **association** between **one or more database table** via **primary key** and **foreign key** references
- Different types of modelling techniques

There are various types of database relationships

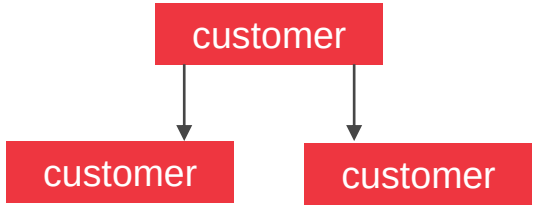
- Self-referencing
- One to one
- One to many
- Many to many

Database relationships contd.

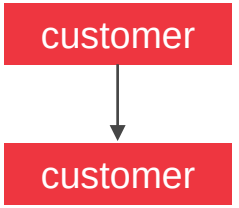
Self-referencing

- Relationship with itself
- Example: Employee referral program where employees refer each other

	customer_id	customer_name	reference_customer_id
1	1	Helen	3
2	2	Aditi	1
3	3	Ian	2



One employee referring many employees



One employee referring one employee

One to one relationship

- One table references another table via a reference key
- Relationship is defined via a primary and foreign key
- Example: Each address refers to only one customer

	customer_id	customer_name	address_id
1	1	Helen	500
2	2	Aditi	501
3	3	Ian	502
4	4	Dom	503
5	5	Peter	504

	address_id	address
1	501	marina one, 018936
2	502	marina one west tower, 018937
3	503	500 Dover Rd, Singapore 139651

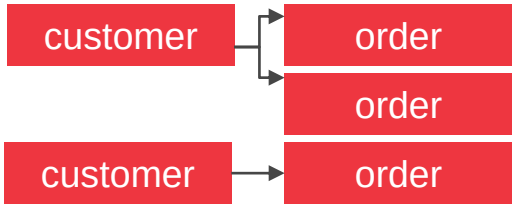
Database relationships contd.

One to many relationship

- One record that refers to multiple records in another table
- Example: Customer making multiple orders

	customer_id	customer_name	address_id
1	1	Helen	500
2	2	Aditi	501

	order_id	customer_id	amount	order_date
1	1	1	22.33	2016-08-07
2	2	1	54.21	2017-01-01
3	3	3	9.31	2018-05-22



Many to many relationship

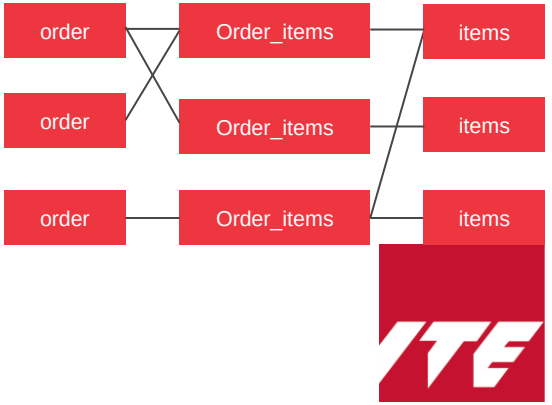
- Multiple records that refer to multiple records in another table
- Example: each order can have multiple items, each item can be in multiple orders
- The two tables are linked by a junction table which acts as a reference for the other two tables

	order_id	customer_id	amount	order_date
1	1	1	22.33	2016-08-07
2	2	1	54.21	2017-01-01
3	3	3	9.31	2018-05-22
4	4	4	64.39	2019-02-03
5	5	2	100.01	2019-04-17

Junction table

	order_id	item_id
1	1	601
2	2	602
3	3	603
4	4	604

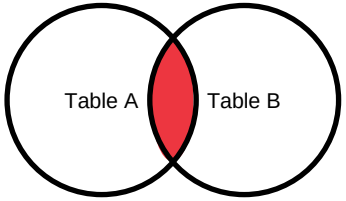
	item_id	item_name
1	601	chair
2	602	table
3	603	shirt
4	604	key



Joins in a database

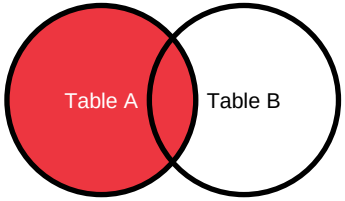
- Joins are methods of combining two datasets in a database
- There are 4 types of joins:
  - Inner Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

Inner Join



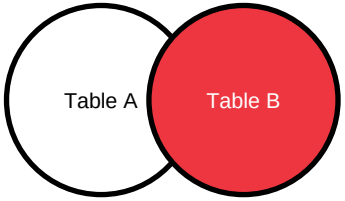
Select the records for table A and table B when the join condition matches

Left Join



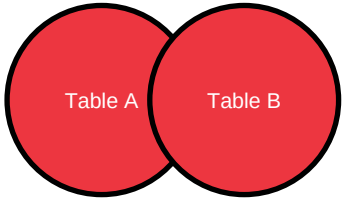
Select all the records for table A and only those records from table B which matches the join condition

Right Join



Select all the records for table b and only those records from table A which matches the join condition

Full Join



Select all the records for table A and table B irrespective of join condition satisfies. This is also called Cartesian product of tables

What is data modelling?

- **Data modelling** is the process of exploring **what kind of data** needs to be stored and creating corresponding data structures
- A data model **describes data**, provides data **semantics** and identifies database **relationships**
- Ensures **consistency** in **naming conventions**, default values and **security constraints**
- It answers **WHAT** the system contains (**conceptual data model**), **HOW** the system will be **implemented regardless of the DBMS** (**logical data model**), and HOW it will be implemented in a specific DBMS (**physical data model**)
- Modelling activities:
  - Identify relevant entities (customers, products)
  - Identify relevant attributes of entities
  - Define relationships between entities
  - Create tables to represent the entities

Two types of data modelling techniques:

- **Entity Relationship (E-R) Model**: is composed of **entity types** and specifies relationships that can exist between **entities**
- **UML (Unified Modelling Language)**: is a specialized language to standardize, construct, document, and architect of software

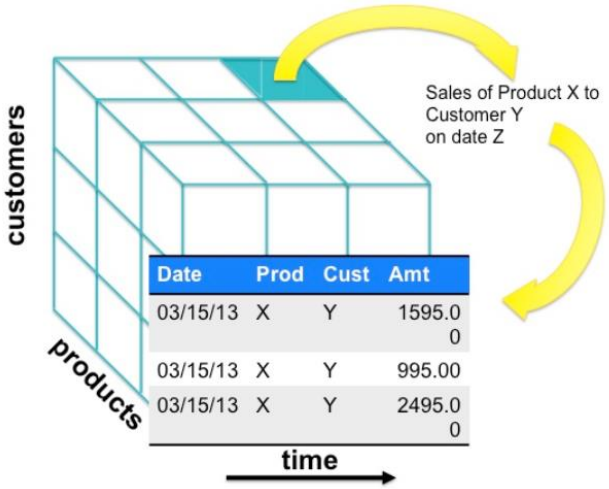
Types of data models

There are two types of data models whose use depends on the type of data stored and the operations performed

- Transactional data modelling
- Analytical data modelling

Transactional data modelling

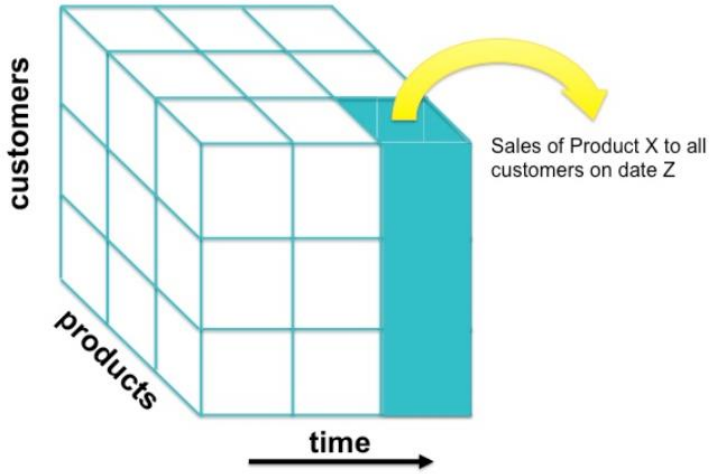
- A single record for each transaction
- Each record contains all information about a transaction
- The execution of each transaction is traceable throughout the model
- For example, if you are storing sales transaction data for a retail company, each transaction will be recorded as a single item in a transaction table
- From the transaction table, you can identify product, customers, amount sold, and the time of the transaction



https://help.gooddata.com/display/doc/Basics+of+Data+modelling

Analytical data modelling

- Data is stored as facts and dimensions
- The purpose the modelling is to provide context of transactions.
- Transaction information is read, summarized and analyzed in a data warehouse to get better insight into the business
- Models with dimensions provide distinct views on transactions
- For example, you can examine a transaction by product, customers, amount, and time through different dimensions of an analytical model



https://help.gooddata.com/display/doc/Basics+of+Data+modelling

Comparison of transactional and analytical data modelling

Transaction	Analytical
Used in transaction systems where many transactions are executed	Used in data warehouses to support decisions
Controls and runs core operational tasks	Helps in business planning and problem resolution
Data supports on-going business processes	Multi-dimensional views support different kind of business activities
Short and quicker updates	Long runtimes for batch data updates
Amount of data queried is small	Complex queries involving aggregations
Data is modelled after business processes	Data is modelled after the effect of business processes
Normalized with many tables	Denormalized with fewer tables
Has standard/well-defined processing	Ad-hoc processing

What is Normalization?

- Process of **organizing** the data into **multiple related tables** to reduce data **redundancy**
- Date redundancy is the **repetition** of similar data at multiple places, which leads to an increase in the size of a database and difficulty maintaining integrity & consistency
- Inconsistency can be caused by **insertion**, **deletion** and **updating** anomalies. For example, a table stores two pieces of information, student and branch:
  - **Insertion anomaly:** If 10k student records are added , the same data (branch, hod, hod_phone_no) would get repeated in each row, increasing the data's size.
  - **Deletion anomaly:** Loss of related data when other data is deleted, such as unintentionally deleting some student information while deleting branch information

	student_id	student_name	branch	hod	hod_phone_no
1	1	Ian	computers	Mr.Adam	12345678
2	2	Dom	computers	Mr.Adam	12345678
3	3	Alex	computers	Mr.Adam	12345678
4	4	Martin	computers	Mr.Adam	12345678

  - **Updating anomaly:** a value change in a common field needs to be updated for all the records in which it is present, taking time and potentially leading to inconsistency. For example, when hod_phone_no gets changed it has to be updated for all records and if we miss one it leads to inconsistency

To solve the above problem, we **divide** the above table into **two tables**, **Student** and **Branch** and **establish a relationship between them**.

Different types of normalization techniques for transaction systems

- First Normal Form
- Second Normal Form
- Third Normal Form
- Others

First Normal Form (1 NF)

Maltem Consulting Group

- Each column contains a **single value**
- Each column should hold values of **the same data type**, no inter-mixing of different types of values
- Each column should have **unique name** to avoid confusion during data retrieval

Example: Payment table

film_id	title	rental_rate	length	description
1	ACADEMY DINOSAUR	0.99	86	A Epic Drama of a Feminist And a Mad Scientist ...
2	ACE GOLDFINGER	4.99	48	A Astounding Epistle of a Database Administrat...
3	ADAPTATION HOLES	2.99	50	A Astounding Reflection of a Lumberjack And a ...
4	AFFAIR PREJUDICE	2.99	117	A Fanciful Documentary of a Frisbee And a Lum...
5	AFRICAN EGG	2.99	130	A Fast-Paced Documentary of a Pastry Chef An...

- Each cell in this table only contains one individual value
- In this table, film_id serves as a primary key, which could be used to identify title, rental_rate, length and description

Second Normal Form (2 NF)

- The table is in 1st Normal Form
- All the non-key columns are **non-partially dependent** on the primary key
- If there is partial dependency relationship, create a **separate table** with the functionally dependent data and part of the key on which it depends

Example: Payment table

Not a second normal form

film_id	title	rental_rate	length	description
1	ACADEMY DINOSAUR	0.99	86	A Epic Drama of a Feminist And a Mad Scientist ...
2	ACE GOLDFINGER	4.99	48	A Astounding Epistle of a Database Administrat...
3	ADAPTATION HOLES	2.99	50	A Astounding Reflection of a Lumberjack And a ...
4	AFFAIR PREJUDICE	2.99	117	A Fanciful Documentary of a Frisbee And a Lum...
5	AFRICAN EGG	2.99	130	A Fast-Paced Documentary of a Pastry Chef An...

Second normal form

title	description
ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist ...
ACE GOLDFINGER	A Astounding Epistle of a Database Administrat...
ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a ...
AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lum...
AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...

film_id	title	rental_rate	length
1	ACADEMY DINOSAUR	0.99	86
2	ACE GOLDFINGER	4.99	48
3	ADAPTATION HOLES	2.99	50
4	AFFAIR PREJUDICE	2.99	117
5	AFRICAN EGG	2.99	130

- In the left hand side, the film_id serves as a primary key, title, rental_rate and length are dependent on film_id. However, this is not a 2nd Normal Form because description is partially dependent on title
- To create a second normal form, we need to separate the table into two tables shown in the right-hand side
- In the first table, description is dependent on title, and in the second table, title, rental_id and length are dependent on film_id

Third Normal Form (3 NF)

- The table is in 2nd Normal Form
- All attributes are **non-transitively dependent** on the primary key
- To eliminate the transitive dependency issue, we could create two tables in 3rd Normal Form

Example:

Not a third normal form

	film_id	title	rental_rate	length
	1	ACADEMY DINOSAUR	0.99	86
	2	ACE GOLDFINGER	4.99	48
	3	ADAPTATION HOLES	2.99	50
	4	AFFAIR PREJUDICE	2.99	117
	5	AFRICAN EGG	2.99	130

Third normal form

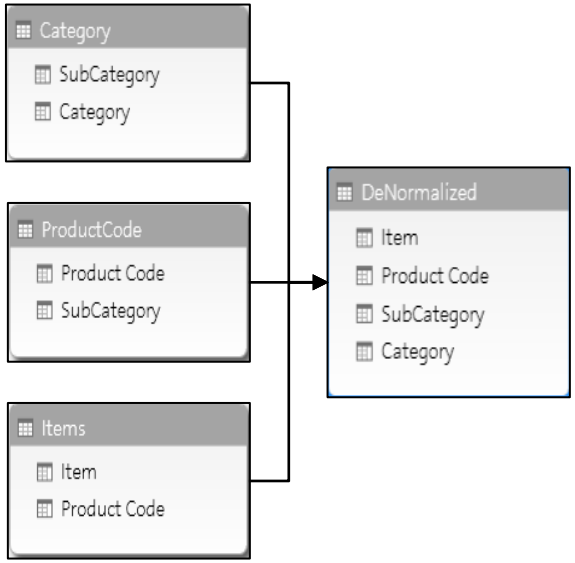
film_id	title	length
1	ACADEMY DINOSAUR	86
2	ACE GOLDFINGER	48
3	ADAPTATION HOLES	50
4	AFFAIR PREJUDICE	117
5	AFRICAN EGG	130

title	rental_rate
ACADEMY DINOSAUR	0.99
ACE GOLDFINGER	4.99
ADAPTATION HOLES	2.99
AFFAIR PREJUDICE	2.99
AFRICAN EGG	2.99

- In the left hand side, the film_id serves as a primary key, the title is dependent on the film_id, and the rental_rate is dependent on the title which transitively dependent on the film_id
- To create a third normal form, we could separate this table into two tables shown in the right-hand side. The first table contains film_id, title and length; the second table contains title and rental_rate
- The new created table eliminate the transitive dependency issue

What is Denormalization?

- Denormalization is the process of **adding data redundancy** in the form of **extra attributes** in the existing table, **creating a new table**, and creating an instance of a existing table
- Denormalization is done when you see a system is slowing down due to performance issues
- The goal is to **reduce the running time** of queries by making data more available to them
- **Reduces** joins in a relational database
- Increases query performance
- Helps in maintaining change histories by creating a table
- Expedites generation of reports that contain statistics
- Example: Data Warehouse

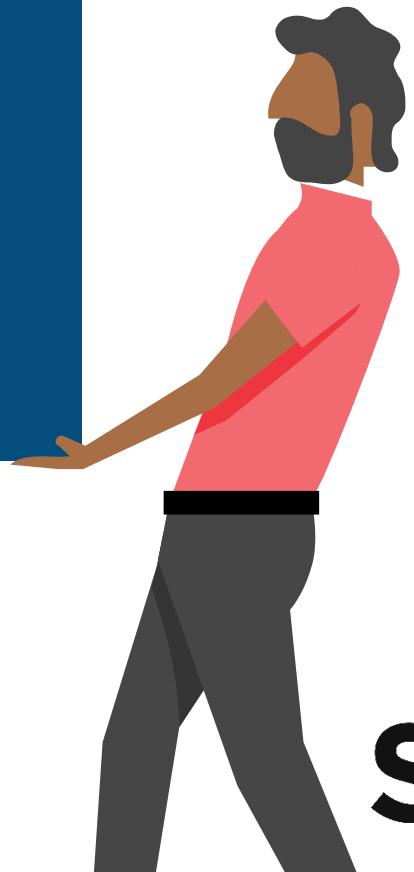


Normalization vs. Denormalization

Maltem Consulting Group

	Normalization	Denormalization
Definition	Data Normalization is the process of **organizing data**	Data Denormalization is the process of **combining data**
Purpose	**Reduce data redundancy and inconsistency**	**Make queries faster**
Disadvantage	Pieces of a **query may be distributed across many tables**	Values are stored **redundantly** in the system
When to use?	OLTP system, where **data CRUD** is faster	OLAP system, where **data search** could be done quickly
Redundancy	Eliminated	Added
Integrity	Maintained	May not be maintained
Number of tables	Increased	Decreased

Lunch Time



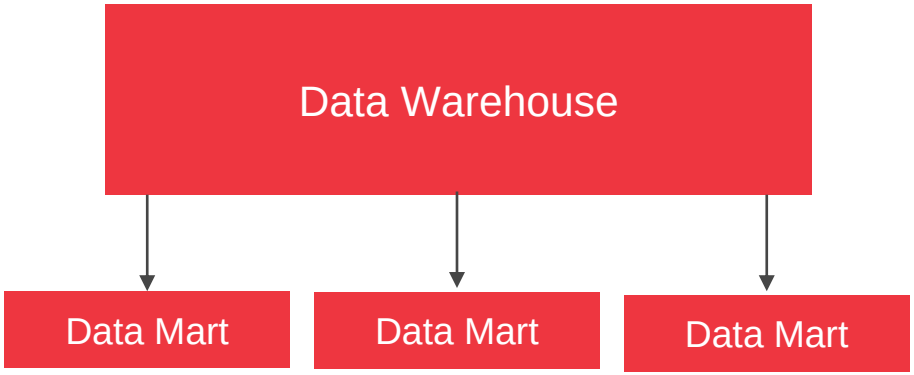
SP PACE ACADEMY

Different types of modelling techniques for analytical system

- Inmon – Top Down Approach
- Kimball – Bottom Up Approach
- Standalone

Inmon

- The process of building data warehouse starts from the **corporate data model and key business entities**
- The next step is to build the **data warehouse**, which captures all business details including keys, attributes, dependencies and relationship
- **Data marts** will be built **specifically for each department**, such as Finance, Sales, etc.
- Any data which comes into data warehouse is integrated and the **data warehouse** is **the only source** for data marts



Advantages and Disadvantages of Inmon

Advantages

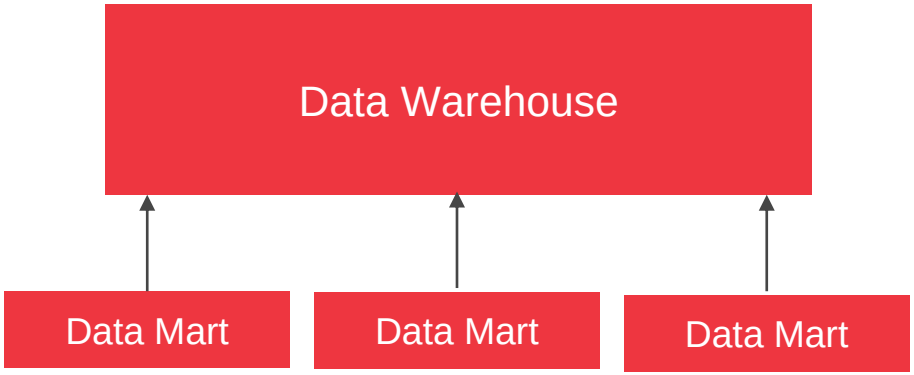
- Data warehouse serves as a **single source of truth** as it is the only source for data marts
- The business process could be **understood easily** as its top-down
- It is very **easy to update** the data warehouse as and when business requirement changes

Disadvantages

- **More ETL work** is needed as the data marts are built form the data warehouse
- Building **model could be complex** as it involves more tables and joins
- The initial set-up and delivery is quite **time-consuming**

Kimball

- The process of building the data warehouse starts from **identifying key business process** and **business questions** which data warehouse needs to answer
- Data is loaded into a **dimensional model**, which is not normalized
- **Multiple star schemas** will be used to satisfy different business reporting requirements
- The star schema will be used to build the **data warehouse**



Advantages and Disadvantages of Kimball

Advantages

- The star schema is easy for business users and is **easy to use** for reporting
- Works well for **department metrics** and **KPI tracking**, as data marts are for business reporting
- Quick and easy to set up and build

Disadvantages

- The data is **not integrated** before serving the reporting need
- **Redundant data** may cause data anomalies over time
- Cannot meet the **enterprise reporting needs** because the model is oriented towards the business process rather than the enterprise

Standalone

- Aside from Dimensional Modelling, there are other modelling techniques that also utilise denormalization. These can be considered standalone as they are utilised for one-off use cases, such as reporting tables
- Reporting tables hold data sets that are used for specific reports. All of the required data has already been joined and aggregated as necessary to meet the reporting needs.

Advantages

- Reduced processing for reports, leading to faster response times
- Simpler report development

Disadvantages

- Redundant data storage
- Inflexible structure supports a very limited set of analytics

What is dimensional modelling?

- Dimensional modelling is a technique of **defining** the **data structure** of a data warehouse
- The concept of modelling was developed by **Ralph Kimball**
- Data is stored in **denormalized** form
- The goal of designing dimensional modelling is to read, aggregate, analyse numerical data to understand business trends
- Normally queries retrieve large **number of records** and summarize it in multiple ways **for generating reports** and **analytical decisions**
- **Subject-oriented** data which tends to answer , “What products are not selling well?”, “What time of the year and which products sell best?”, “In what region are sales best”?
- Data is represented in **facts** and **dimensions**
- **Easily understandable** by **business users** as it is based on business terms so the users know what each dimension, fact and attributes means

What is a Fact and Fact Table?

Fact is the **measurement/metrics** and **fact** in the business process. Facts are normally in numeric values.

Examples of fact:

Retail Business: monthly sales revenue

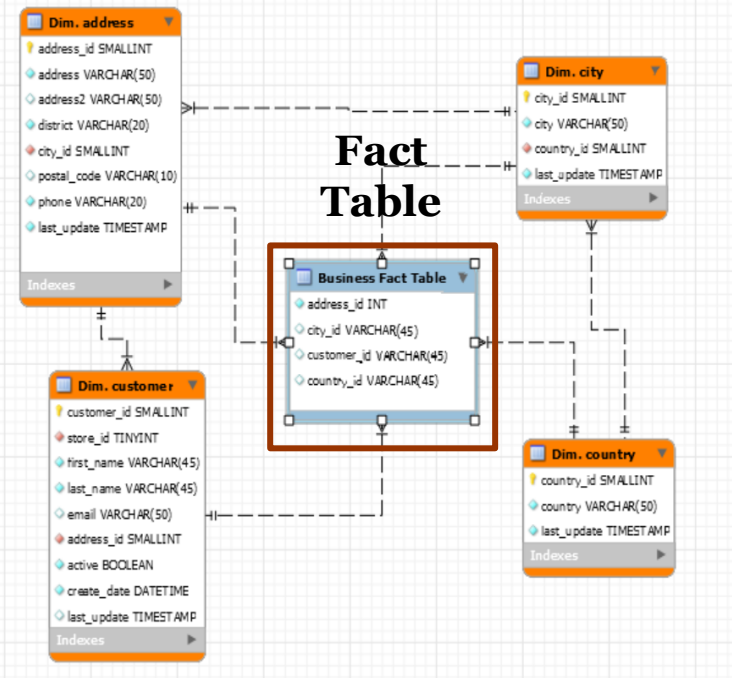
Manufacturing company: quarterly overhead cost

HR Department: monthly employee payroll

Fact Table: a table containing all **measurements** and **facts**, which stores quantitative information for analysis

It contains **foreign keys** which connect the fact table with the dimension table

A fact table works with dimension tables. A fact table holds the data items to be analyzed, and a dimension table stores detailed context of data so that we could analyse data stored in the fact table



What is a Dimension and what are its types?

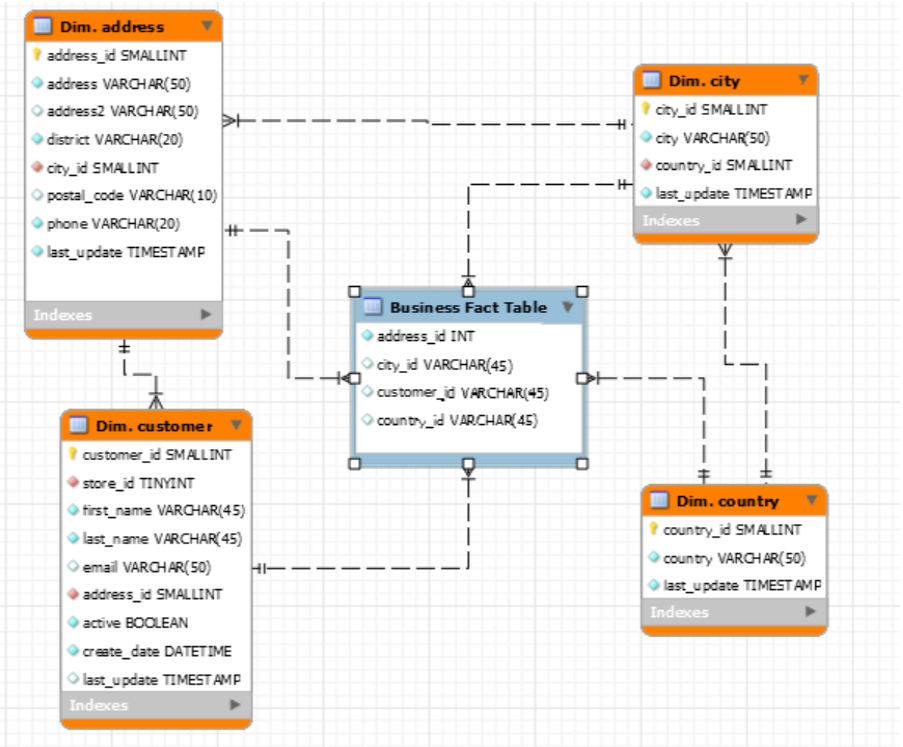
- Dimension provides **context** of a business event
- Dimension describes **when, where** and **what** about a business event

For example, in a retail business process, dimensions could be:

When – the time of the transaction

Where – the location of the store

What – the product the customer bought



What is a Star Schema?

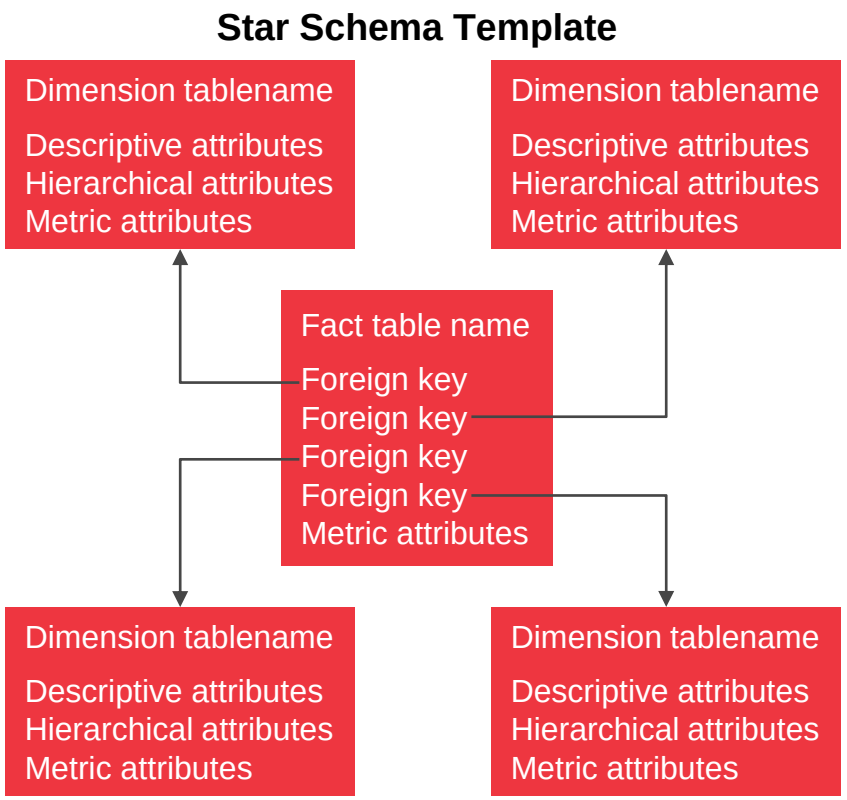
- 1 fact table: n dimensional table

Fact table:

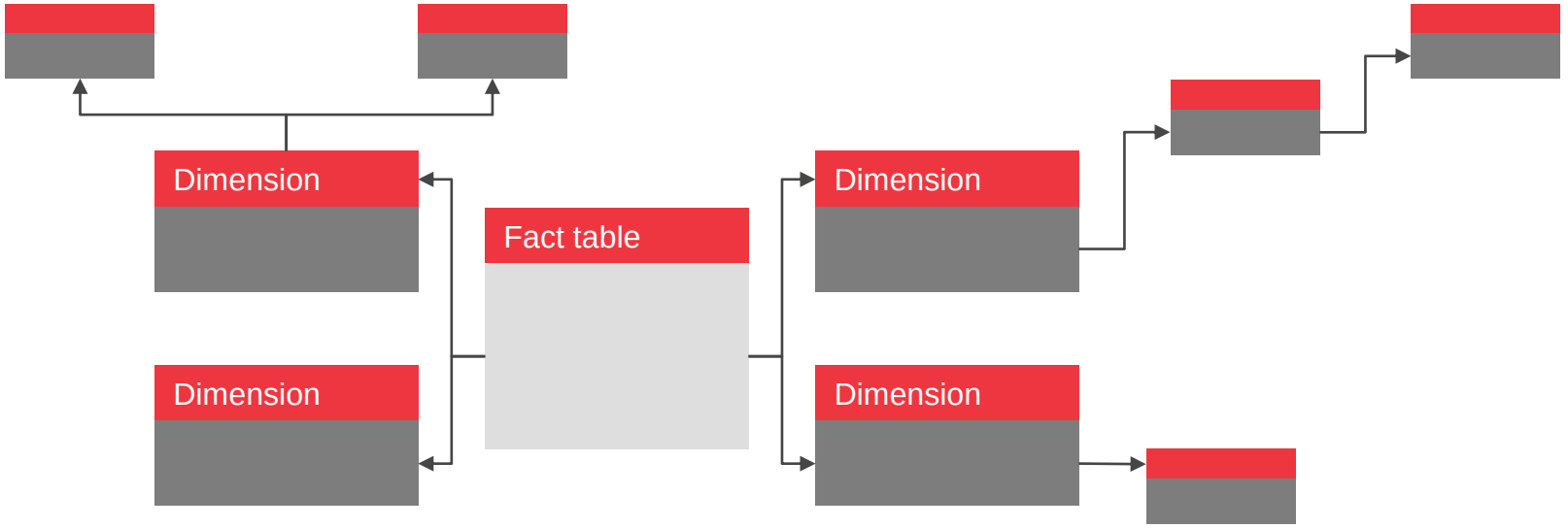
- Holds quantitative business process data
- Records metrics of specific event
- Record numeric values and foreign keys to dimensional data where descriptive information is kept

Dimension tables:

- Hold descriptive attributes related to data in the fact table
- Define a wide variety of characteristics, like time, geography, product description



What is a Snowflake Schema?



- Some dimension tables are normalized
- Due to normalization, the redundancy is reduced
- Data maintenance is easy

For example:

A fact table for ‘supply’ has one of it’s attribute as ‘Location_key’ which is defined by a dimension table, say ‘Location’. The ‘Location’ dimension table has one of it’s attribute as ‘Site_Code’.

The attribute ‘Site_Code’ is described by another dimension table.

Fact details — Three types of measures

Additive fact: the most flexible and useful measure. Additive facts could be summed up **across all dimensions** in the fact table.

An example of a fully additive measure is sales data of a movie rental company. We could add up sales data across different time periods (Jan, Feb, March) or across different stores or regions (Singapore, Vietnam, and India).

Semi-additive fact: semi-additive facts could be summed up **across some but not all dimensions.**

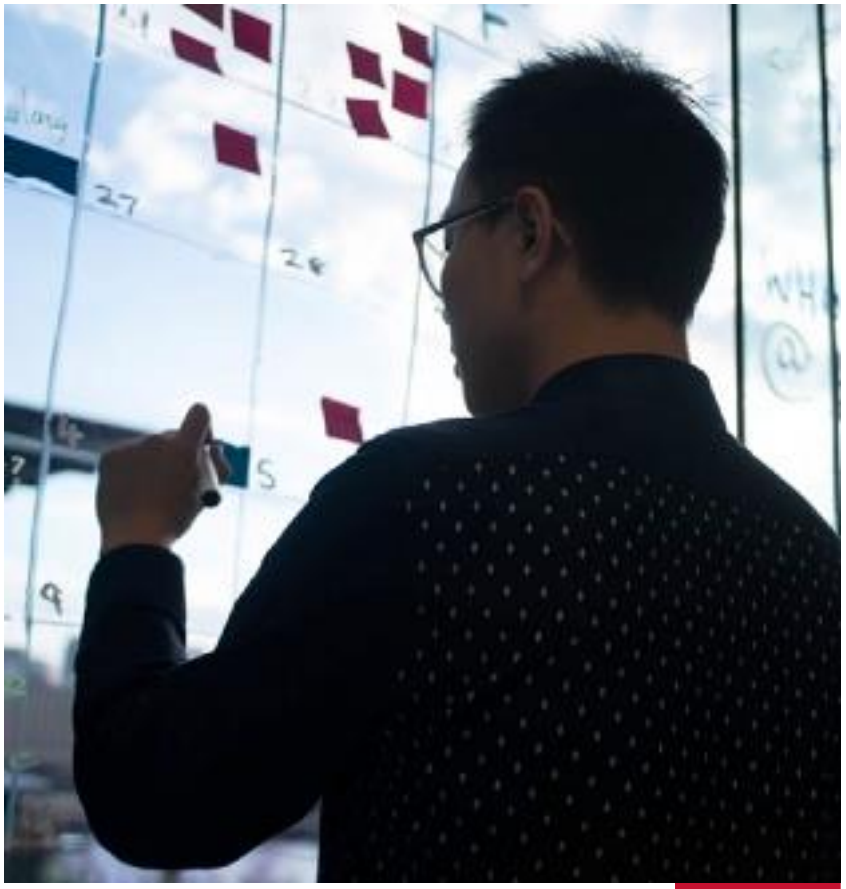
An example of a semi-additive measure is the total asset data of a movie rental company. We could add total asset across different subsidiaries but it does not make sense to sum up the asset across time periods.

Non-Additive fact: Non-additive facts **could not be summed up** across any dimension in the fact table.

An example of non-additive measure is cost/sales ratio.

Fact details — Three types of fact tables

- Transaction
- Periodic snapshots
- Accumulative snapshots



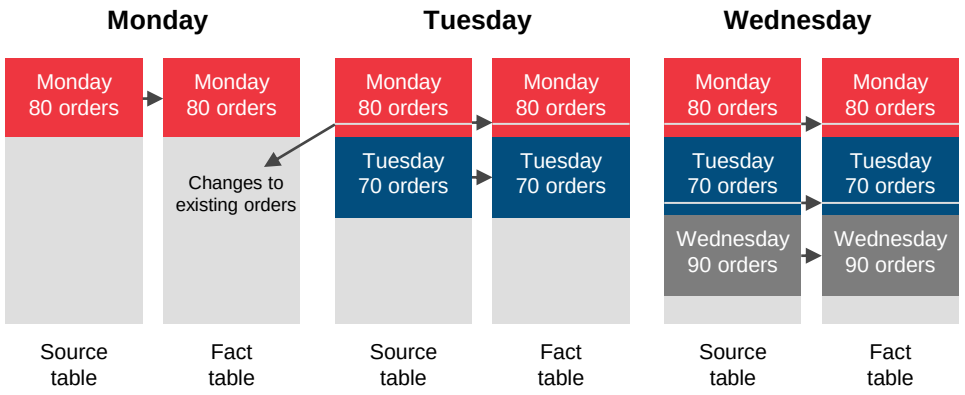
Fact details — Transaction fact table

- Each event is stored in the fact table only **once**
- It has a **date column** indicating when the event occurred
- It has an **identifier column** which identifies each event
- The number of rows is the same as the source data

For example, in the source system, we have a data warehouse containing all of customer order data. On Monday we had 80 orders, Tuesday 70 orders, Wednesday 90, and so on. Therefore, on Monday night we load the 80 rows into the data warehouse, on Tuesday night we load the 70 rows, and on Wednesday night we load the 90 rows. In the data warehouse we store the customer orders in the Sales Fact Table, which is a transactional fact table.

Four key features of this fact table:

- Each order is stored in the fact table only once
- There is a date column indicating when the customer order happens (Mon. Tues. Wed.)
- The customer order number or the ticket number could be used to identify each order specifically
- On Wednesday night, after the warehouse load is finished, we have the total number of orders stored in the data warehouse 90+70+70=230, which equals the total rows in the source data



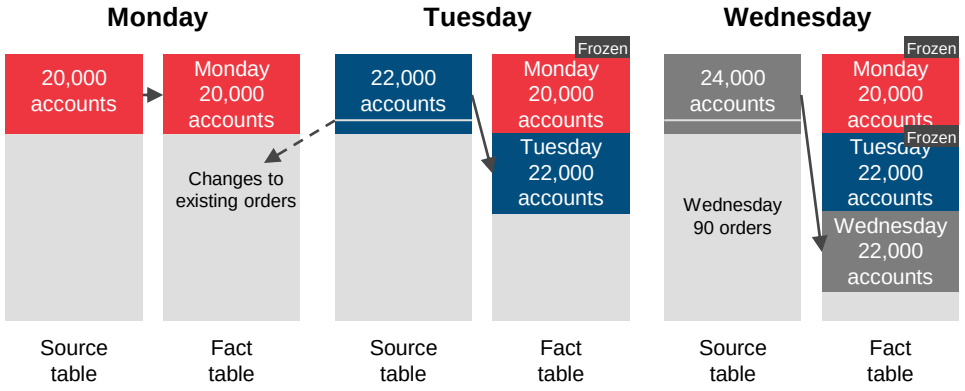
Fact details — Periodic snapshot fact tables

- The **whole source system** is **copied** into the fact table regularly
- The same event is stored **multiple times**
- It has a **date column** indicating when a copy of the source table was created

For example, at the end of each day, the balance of each customer accounting stored in the bank system will be uploaded to the bank account balance system. There are 20, 000 customers on Monday, 22,000 customers on Tuesday and 24, 000 customers on Wednesday.

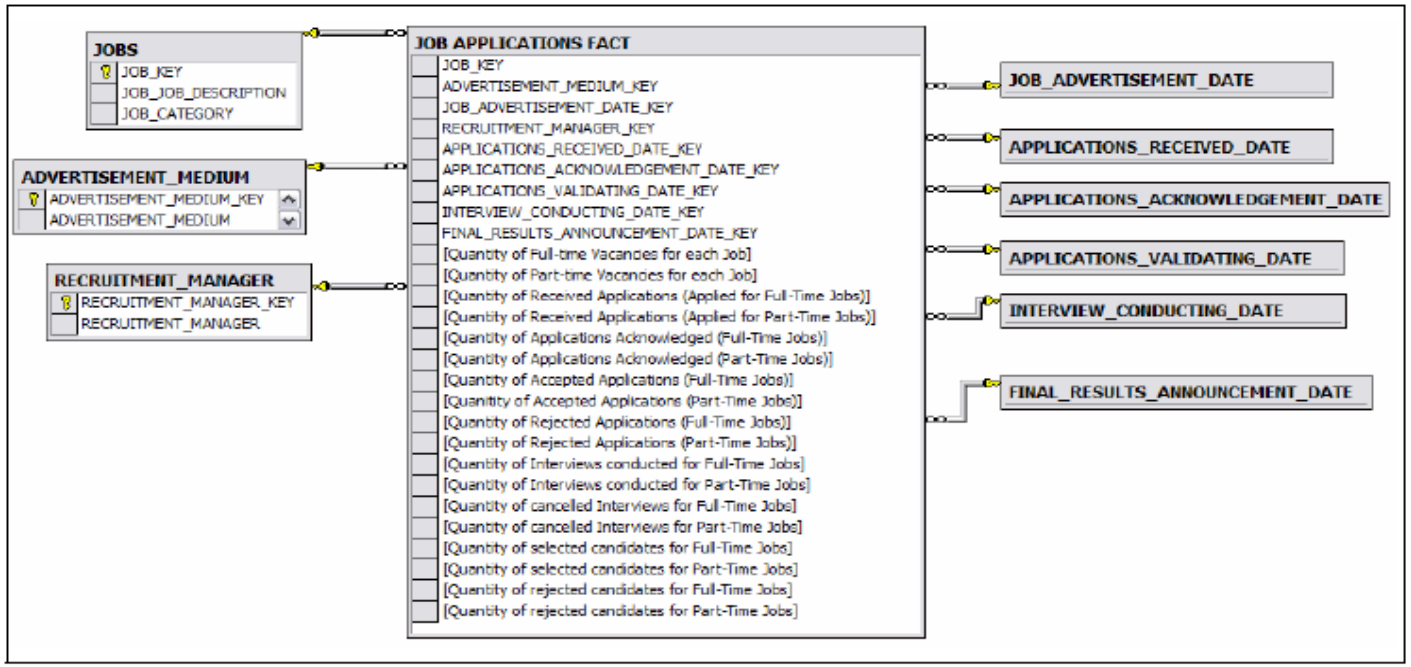
Three key features of this fact table:

- Everyday, we update the customer account details to the fact table
- For customer account, it may be copied everyday to the fact table, each day with a different account balance
- There is a data column (Mon. Tues. Wed.) indicating in which day the customer account is updated.



Fact details — Accumulative snapshots fact tables

- Snapshot fact tables are similar to the transaction fact table in design
- Snapshot fact table samples **data** at **fixed points in time** or as a **result of an event**
- An accumulating fact table stores one row for the **entire lifetime of an event**
- Example: records data of job vacancy advertised by a company



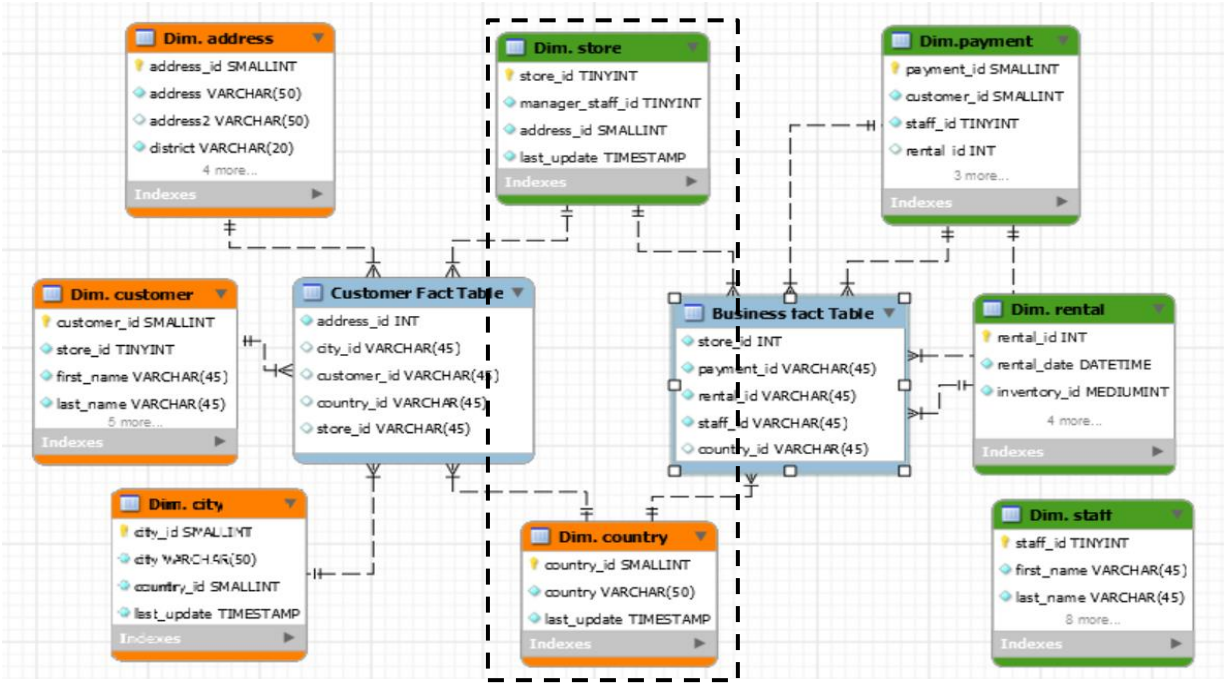
Dimension details — Types of dimensions

- Conformed Dimension – used in multiple facts
- Junk Dimension – combination of different and unrelated attributes
- Degenerate Dimension – part of fact table as it has other dimension attributes associated with it
- Role-playing Dimension – has multiple foreign key relations with fact table

Dimension details — Conformed dimension

- A conformed dimension is a dimension that has the **same meaning** as every fact with which it relates
- Conformed dimension endures **consistency** across multiple facts
- It may be a single dimension table relating to multiple fact tables within the same data warehouse

Conformed dimension

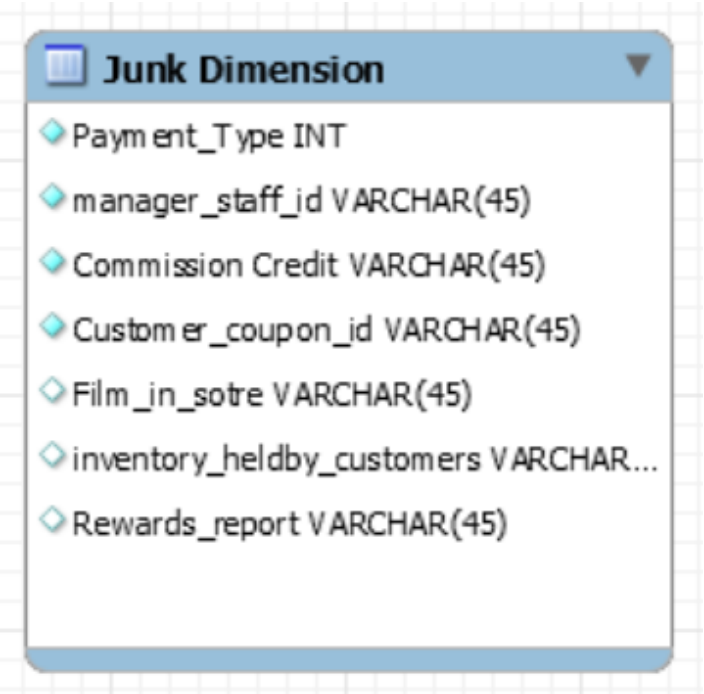


The store and county are conformed dimensions relating to both customer fact table and business fact table, as stores and country data has same meaning in customer and business database

- Dimension table for customer fact table
- Dimension table for business fact table

Dimension details — Junk dimension

- A junk dimension is a collection of random transactional codes flags or text attributes that are unrelated to any particular dimension.
- It provides a convenient place to store the junk attributes
- Example: sample items in junk dimensions

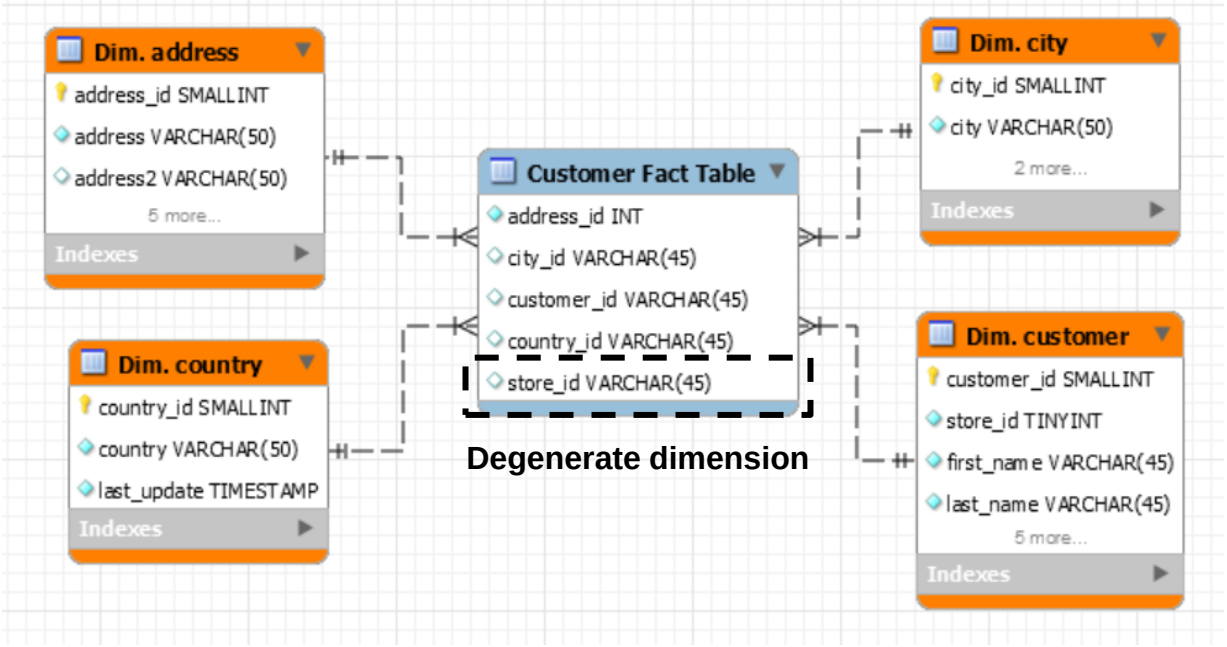


This junk dimension lists some sample junk attributes, which cannot be classified in other existing dimension tables.

- Data Profiling is useful in identifying junk dimension candidates

Dimension details — Degenerate dimension

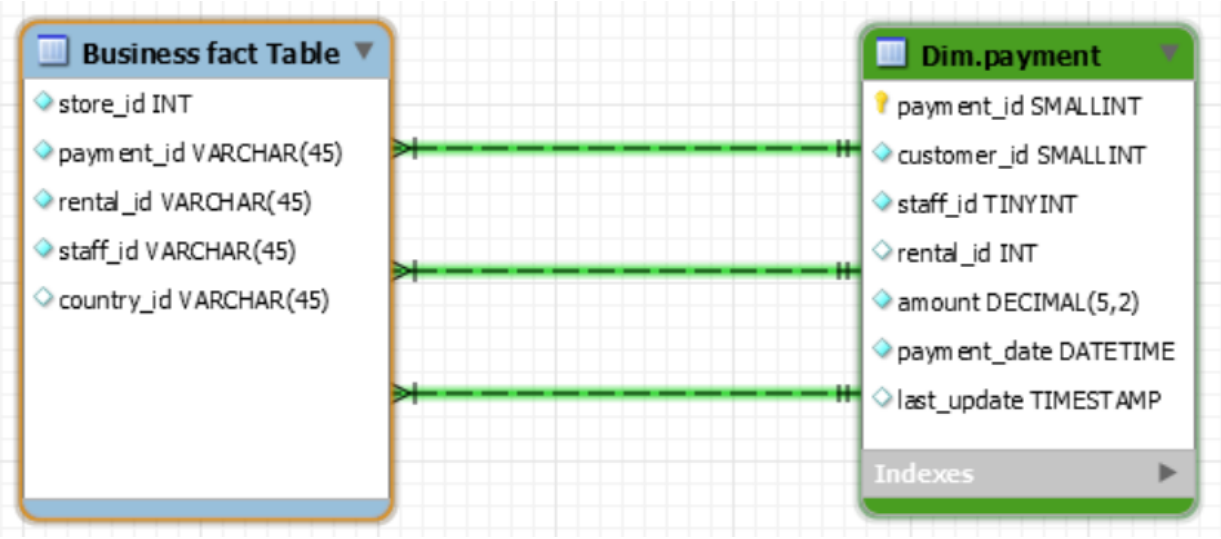
- A degenerate dimension is a dimension key in the fact table that **does not have its own dimension table**
- Degenerate dimensions are stored in the fact table where the grain of the table is the line item itself



The store_id stored in the customer fact table is a degenerate dimension, as it does not have its own dimension table.

Dimension details — Role-playing dimension

- A role-playing dimension is a dimension with **multiple relationships between itself and another table**
- Example of a role-playing dimension



The Business Fact has multiple relationships to the Payment dimension table on the primary key payment_id, staff_id, and rental_id.

Slowly Changing Dimensions (SCD) and its types

- Dimension that varies over time rather than changing on regular time basis
- Stores and manage both current and historical data
- Helps businesses get better insight
- Example: a product name changes over time, people changes their name for some reason

Type 0 - Type 0 dimension attributes keep **original values.** There is no value change

Type 1 - Type 1 SCD **overwrites existing data** value with the new attribute value

Type 2 - Type 2 SCD adds a record to capture the new attribute values **marks the original record inactive**

Type 3 - Type 3 SCD **adds a new column** to store changed data

Type 6 - Type 6 SCD **combines** type 1, type 2 and type 3

SCD Type 1 with Example

- Overwrite the **existing data values** with new data values
- Easy to update the dimensions and limit the growth of the dimension table to only new records
- But the historical value of the data will be lost because the dimension contains the current values for each attribute

Example:

Original Record:

film_id	title	length	rental_rate
141	CHICAGO NORTH	185	4.99
182	CONTROL ANTHEM	185	4.99

Updated Record:

film_id	title	length	rental_rate
141	CHICAGO NORTH	185	4.99
182	CONTROL ANTHEM	185	5.99

Note: as a result of update information, all of the history data before the region changes will be lost. This change will skew the historic data and the report run before the update will be different from the report run after the update for the same timeframe.

SCD Type 2 with Example

Maltem Consulting Group

- **Adds** a **new record** encompassing the changes.
- The current/original record marked as closed and new record become active
- Effective time and the expiration time is used to identify the time period when the record is active
- Version number is also used to track the history of changes

Example:

Original Record:

film_id	title	length	rental_rate
1	ACADEMY DINOSAUR	36	0.99
11	ALAMO VIDEOTAPE	126	0.99

Updated Record:

film_id	title	length	rental_rate
1	ACADEMY DINOSAUR	36	0.99
11	ALAMO VIDEOTAPE	100	0.99
11	ALAMO VIDEOTAPE	126	0.99

Note: to indicate the data change, one active record will be added to show updated information.

SCD Type 3 with Example

- **Add a second column** to store the most recent past value of the columns. When the data is updated, the previous past value and the new value is placed into reportable columns
- It allows people to look back what previous value is, but it makes uploading/loading data more challenging

Example:

Original Record:

film_id	title	length	rental_rate
1	ACADEMY DINOSAUR	36	0.99
11	ALAMO VIDEOTAPE	126	0.99

Updated Record:

film_id	title	length	rental_rate	Previous rental rate
141	CHICAGO NORTH	185	4.99	4.99
182	CONTROL ANTHEM	185	5.99	4.99

Note: to indicate the data change, one more column will be added to show previous rental_rate

SCD Type 6 with Example

- SCD Type 6 combines SCD Type 1, SCD Type 2 and SCD Type 3 (1 + 2 + 3 = 6). To create a Type 6 SCD, we would start with a Type 2, add columns for the records we wish to capture the current value as well as the historical value
- It allows people to filter or group on the Type 2 value in effect when the measure occurred or the current attribute value

Example:

Original Record:

film_id	title	length	rental_rate
141	CHICAGO NORTH	185	4.99
182	CONTROL ANTHEM	185	4.99

Updated Record:

film_id	title	length	rental_rate	Previous length
1	ACADEMY DINOSAUR	36	0.99	36
11	ALAMO VIDEOTAPE	100	0.99	126
11	ALAMO VIDEOTAPE	126	0.99	

Quick Quiz

What are the types of dimensions?

Why do we use SCDs?

End of Day 3