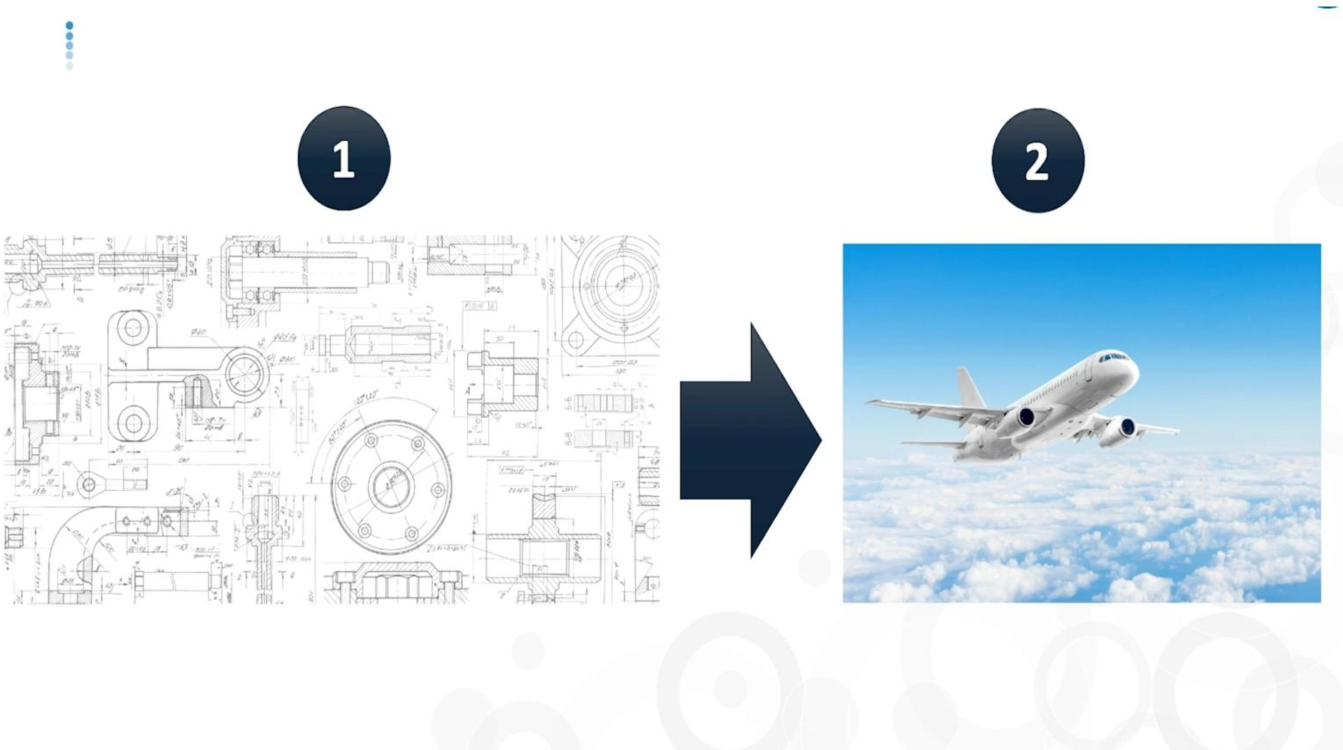
VISION BOARD

Data Warehouse and Dimensional Modeling Fundamentals

DeviKrishna R
+91 6235526324

What is a data model ?
-->It is a prototype built which helps us in getting to know the main system built based on this model (ex. Database, DWH, data mart)
-->Data models doesn't actually consists of any data in it.
-->We consider entities, relationship between entities, business logic for building data models

Example:
Let's consider we are about to build a real aero plane for a customer. Before we start off to build it , we first work on the design, each part functionality and come up with a prototype which after getting finalized can get started to build the actual plane

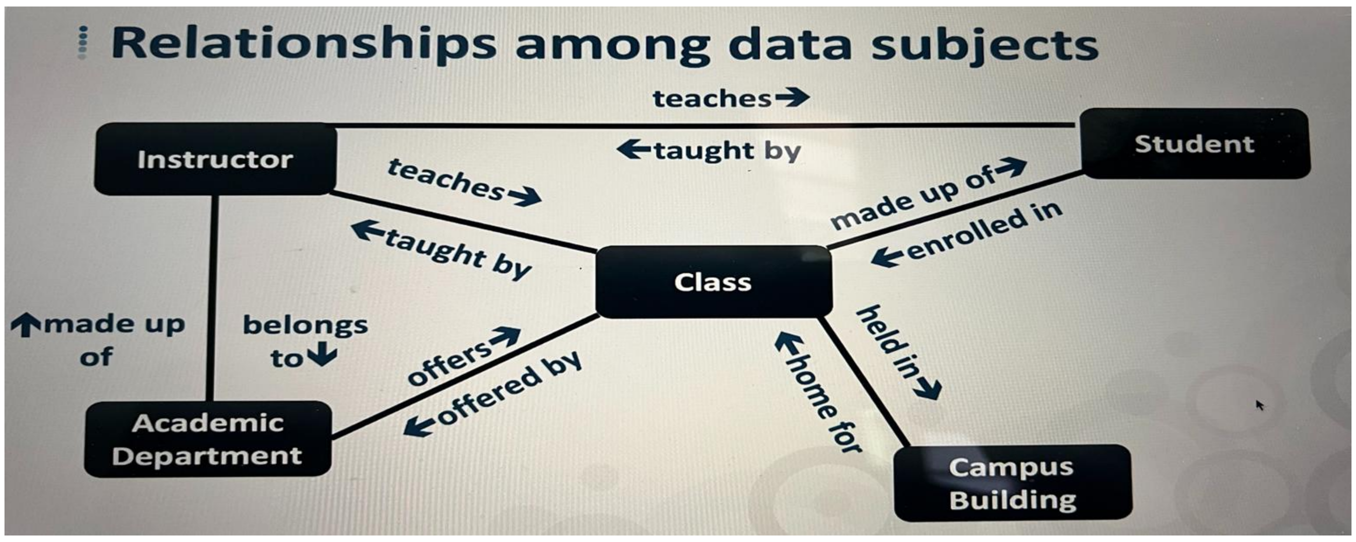


Same situation repeats here as well, before we try to build a real DWH/data mart, we get started to build a data model to understand our data in a better way.

Data modelling fundamentals:

Data models consists of the below:
1. Data subjects
-->Commonly referred to as entities. Is very familiar to "database tables"
2. Attributes of data subjects
-->Analogous to database columns
3. Relationship between data subjects
-->Tells about how tables are related to each other
-->Talks about various relationships we have at place :
single level - one table related to only one table

Multi-level - one table related to many tables
Hierarchy - one table divided into different sub tables which are related to the main table

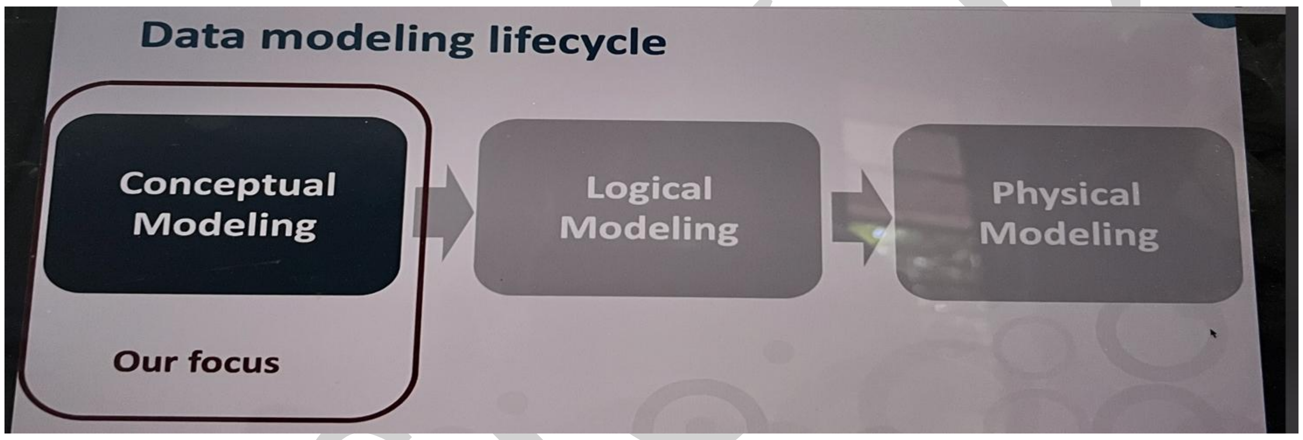


Relationships among data subjects

4. Business rules for our data
-->Some of the business rules to be followed:
4.1 Mandatory or optional relationships
4.2 Permissible attribute values (values that can include nulls)
4.3 Data change dynamics (column/table addition or removal)
4.4 Cardinality

Data Modelling Lifecycle:

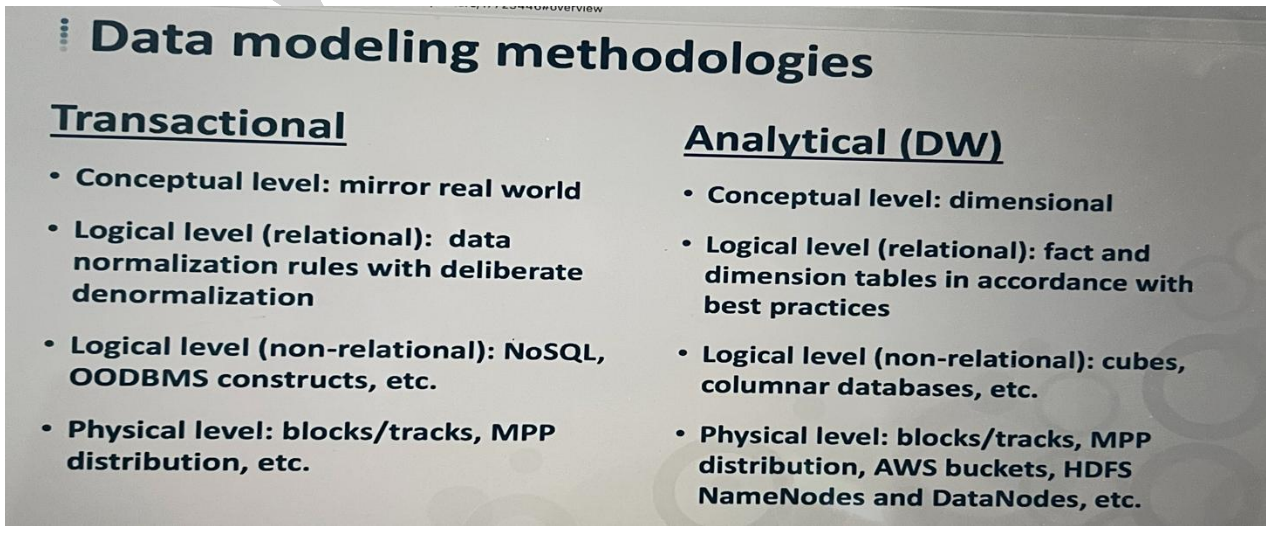
Conceptual Model ==> Logical Model ==> Physical Model



Data modeling lifecycle

Conceptual modelling: An overview of how the model can be built upon. We can decide upon tables and attributes that can be included in creation of the data model
Logical modelling : Built upon creating relationships among entities, key constraints to be included
Physical modelling : Final data model that will be used in building up the data warehouse which includes granular level of details like indexes, partition columns etc

Transactional Model VS Dimensional Model



Data modeling methodologies

Transactional

- Conceptual level: mirror real world
- Logical level (relational): data normalization rules with deliberate denormalization
- Logical level (non-relational): NoSQL, OODBMS constructs, etc.
- Physical level: blocks/tracks, MPP distribution, etc.

Analytical (DW)

- Conceptual level: dimensional
- Logical level (relational): fact and dimension tables in accordance with best practices
- Logical level (non-relational): cubes, columnar databases, etc.
- Physical level: blocks/tracks, MPP distribution, AWS buckets, HDFS NameNodes and DataNodes, etc.

https://intuit.udemy.com/course/data-warehouse-fundamentals-for-beginners

==
Introduction to Data warehouse Fundamentals
==

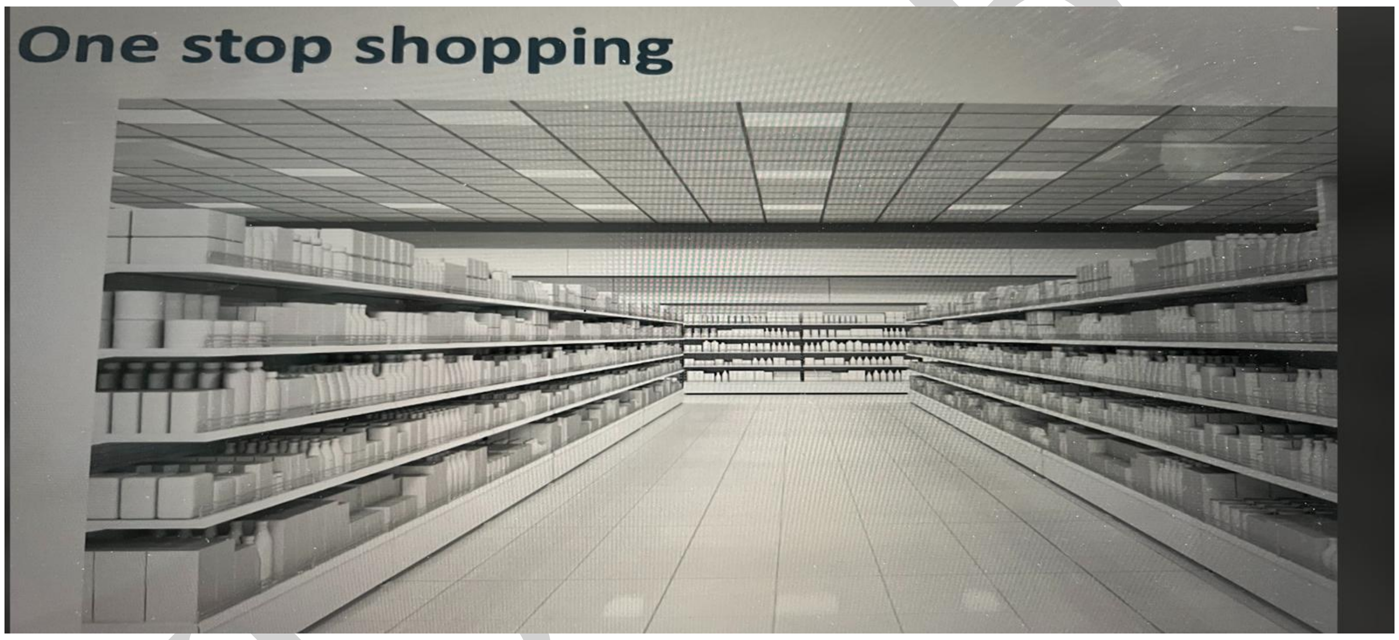
Data warehouse: Data (information) + warehouse(storage unit of data from various sources for various purposes)
-->it is analogous to warehouse concept where we store items for a long time and organized manner which serves a specific purpose.

Features:

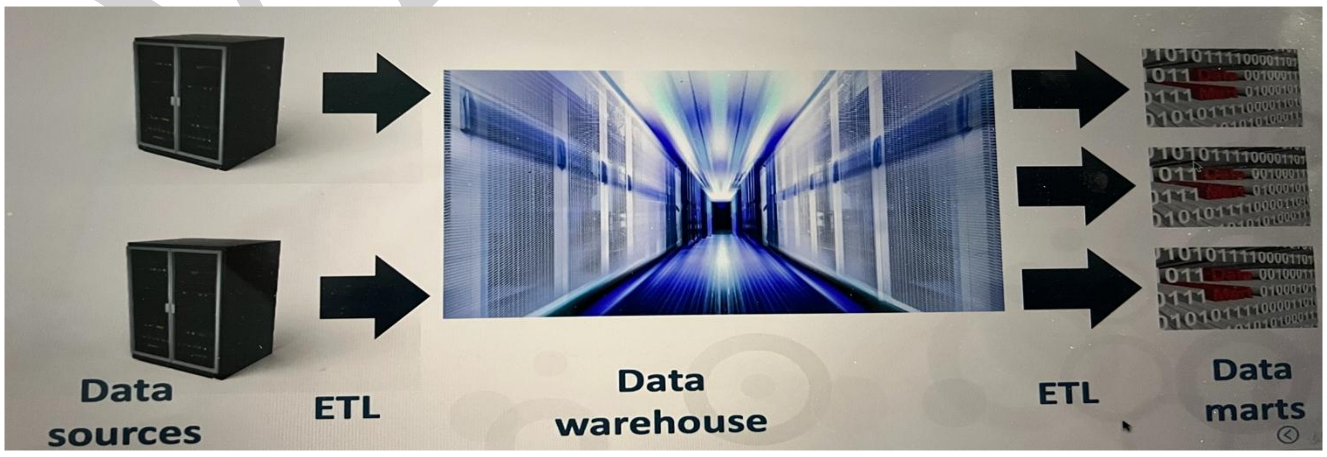
1. Integration from different data sources - Data from various sources can some to a common place where they can be stored in organized manner
2. Subject Oriented - Based on the data subject (kind of data) from a data source decides the way this data has to be stored
3. Time variant - Data in the DWH contains historical data not just current data
4. Non volatile - Data doesn't change a dynamically as in transactions systems.

WHY DWH :

1. Make data driven decisions - Based on past, present and future data. Try to find the unknown metrics which is used for analysis
2. One stop shopping - This is a common place where we can find data from various transactional DB, operational source all at one place



Sample end-to-end DWH :



DWH DESIGN :

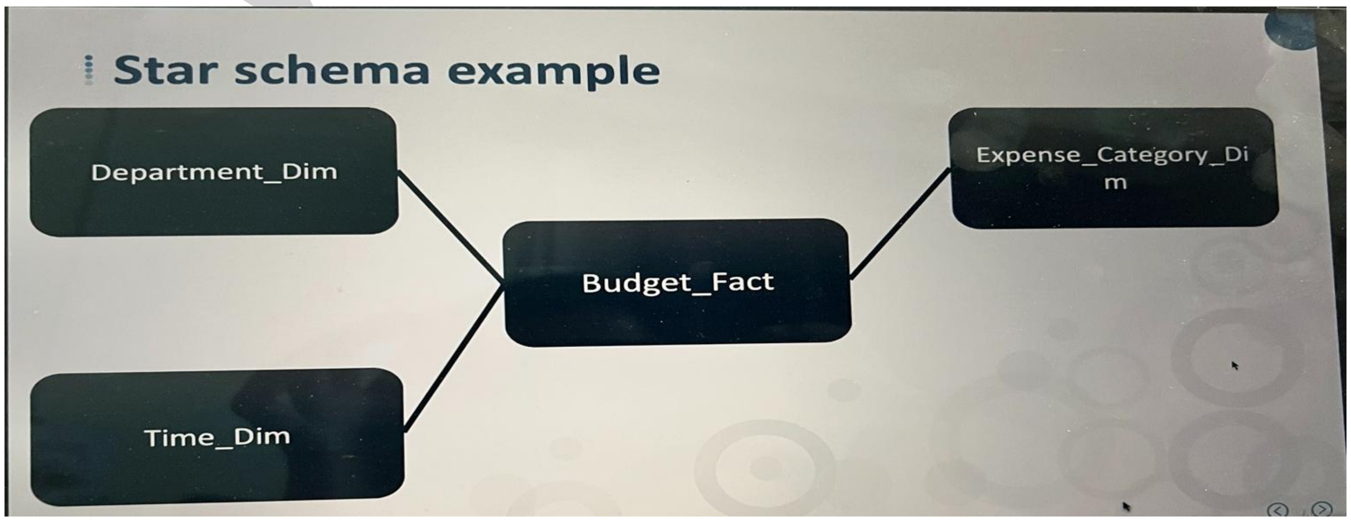
Dimensional Modelling :

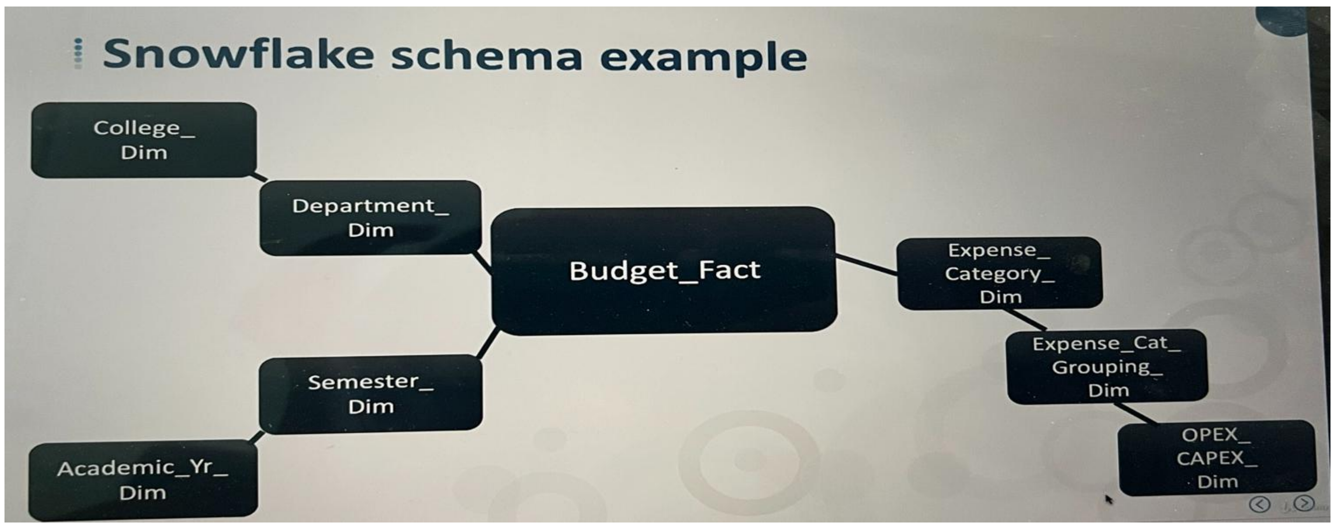
1. Storing data in a way which is easy to access and spot as well by introducing dimensions.
   Dimension = Measurement + Context
   100 => 100% => 100% attendance for computers class by 1st year students
   Chocolate at 2nd rack, 2nd from left and in the 2nd row
2. You mix out measurement along with the exact situation where this value is used. Based on which it will be easy to access this value in the DWH

Facts vs Dimensions

Facts	Dimensions
1. Facts are used to represent measurements 2. Facts are similar to metrics and quantifiable	1. Dimensions are used to measure dimensional context 2. Give details on how to calculate the fact

Star Schema	Snowflake Schema
1. All dimensions along with its hierarchy exist in a single dimension table. 2. Overall fewer joins exist as the dimensions are just one level away from fact 3. DB primary->foreign key relationship are straight forward 4. Denormalized dimension table*	1. Dimensions with its hierarchy go into a separate table 2. Overall more joins exist as extra level of hierarchy included 3. DB primary->foreign key relationships are more complex 4. Normalized dimensional table


Star schema example



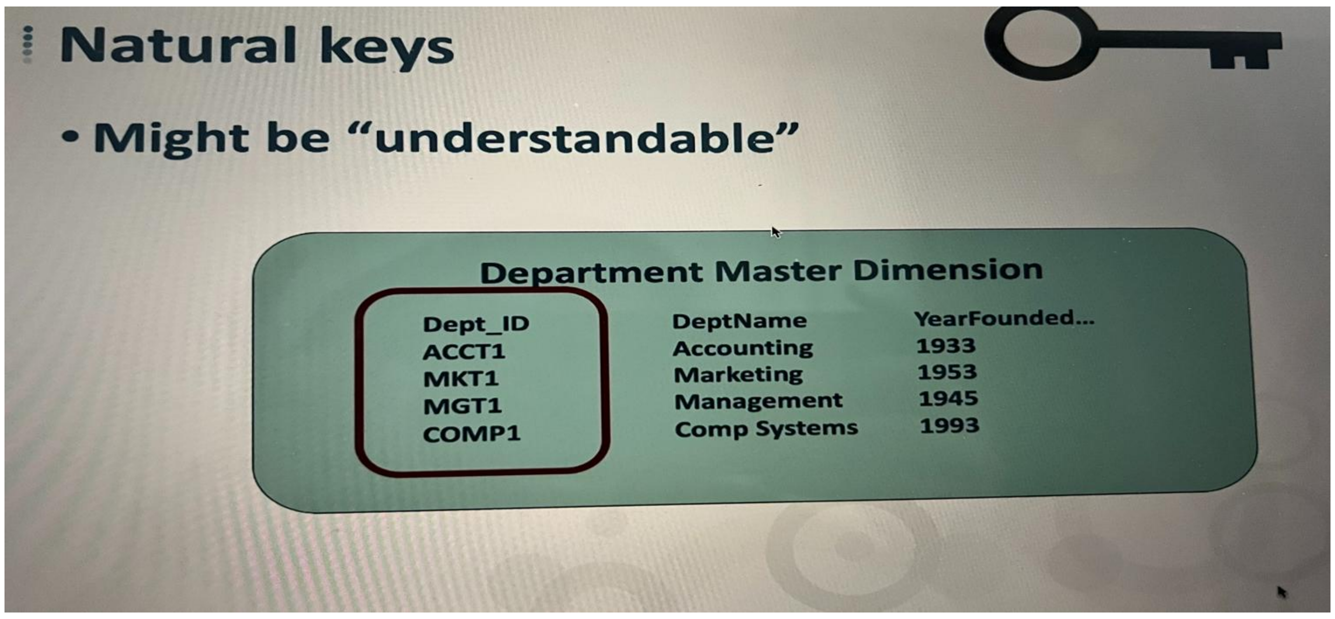
Database Keys for DWH:

Fundamentals :

1. Star and snowflake schema are implemented in RDBMS
2. RDBMS handle logical relationships to relate data across different tables
3. "official" relationships are handled through data

Primary Key	Foreign Key
1. Unique identifier of a row in a db table. 2. Has no business meaning attached to the ket	1. Key which is a primary key in another table 2. Used to form relationship between tables

Natural Key	Surrogate Key
1. Can be used as a unique identifier for a data row which has business meaning given to it. 2. Natural keys can come from source systems along with the actual data 3. In the below pic, as you see natural keys to be ACCT1 ==> Accounting MKT1 ==> Marketing You can see how they present business meaning	1. Unique identifier of data row which is generated by the db system. It has no business meaning. 2. Surrogate keys are generated in the DWH system and don't come from the source systems

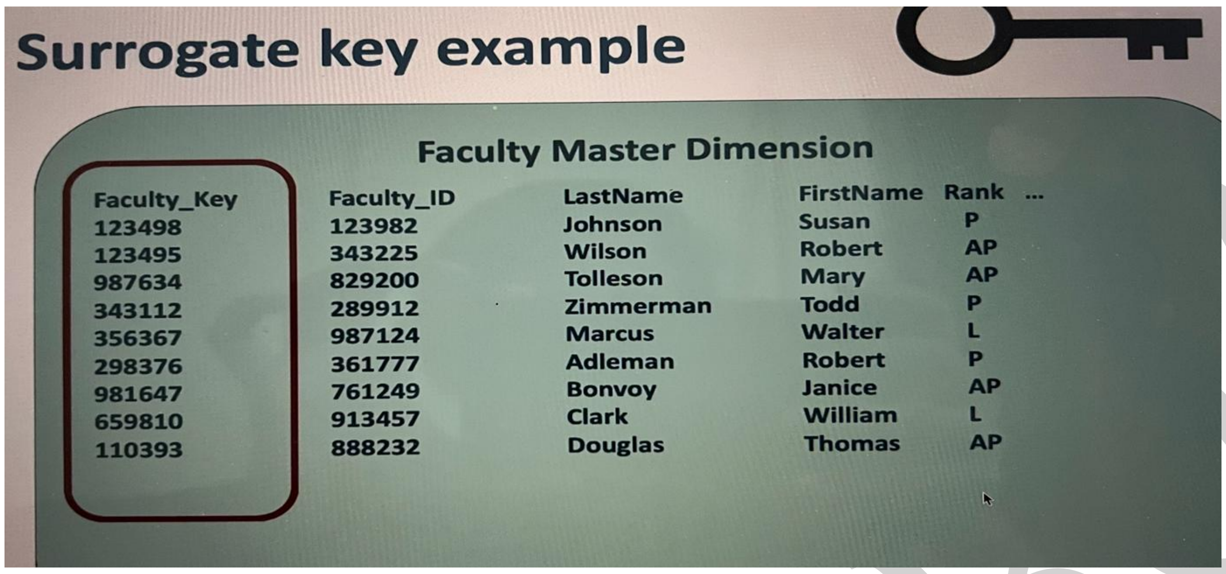


Natural keys

- Might be "understandable"

Department Master Dimension

Dept_ID	DeptName	YearFounded...
ACCT1	Accounting	1933
MKT1	Marketing	1953
MGT1	Management	1945
COMP1	Comp Systems	1993

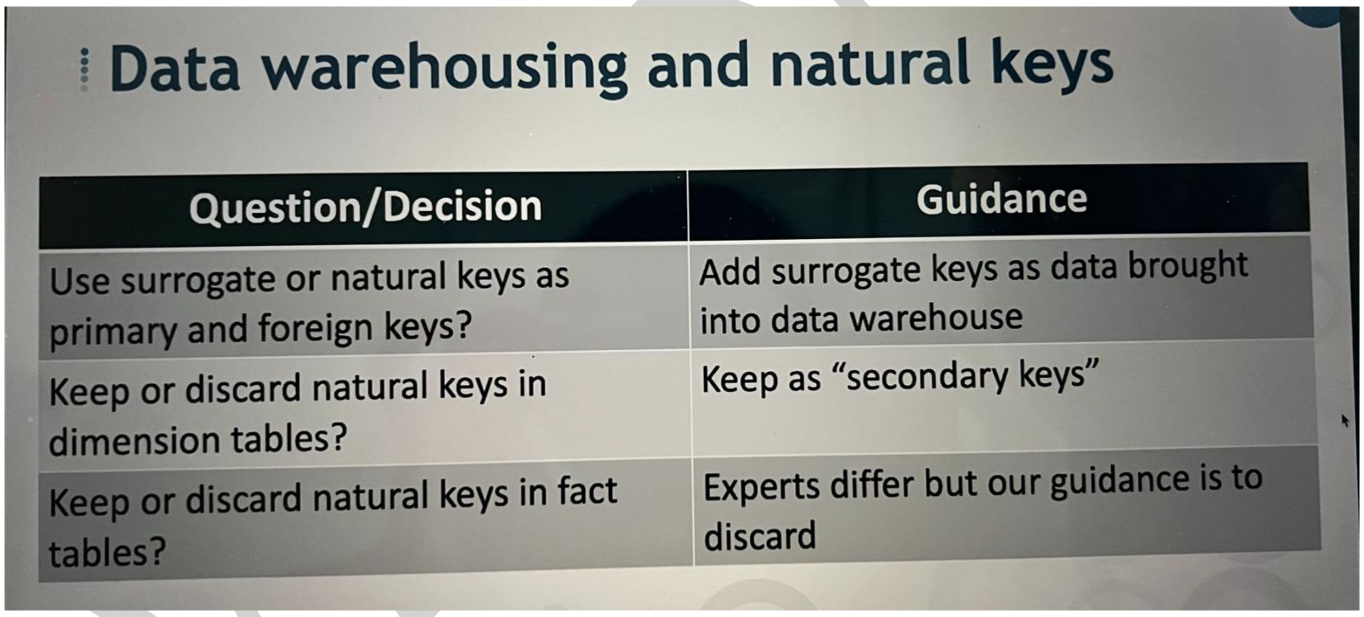


Surrogate key example

Faculty Master Dimension

Faculty_Key	Faculty_ID	LastName	FirstName	Rank	...
123498	123982	Johnson	Susan	P	
123495	343225	Wilson	Robert	AP	
987634	829200	Tolleson	Mary	AP	
343112	289912	Zimmerman	Todd	P	
356367	987124	Marcus	Walter	L	
298376	361777	Adleman	Robert	P	
981647	761249	Bonvoy	Janice	AP	
659810	913457	Clark	William	L	
110393	888232	Douglas	Thomas	AP	

Surrogate Keys: It is advised to use surrogate keys when building a DWH as it preserves data integrity.
Natural keys: They can be used as secondary keys



Data warehousing and natural keys

Question/Decision	Guidance
Use surrogate or natural keys as primary and foreign keys?	Add surrogate keys as data brought into data warehouse
Keep or discard natural keys in dimension tables?	Keep as "secondary keys"
Keep or discard natural keys in fact tables?	Experts differ but our guidance is to discard

======================
Designing Dimensions
======================

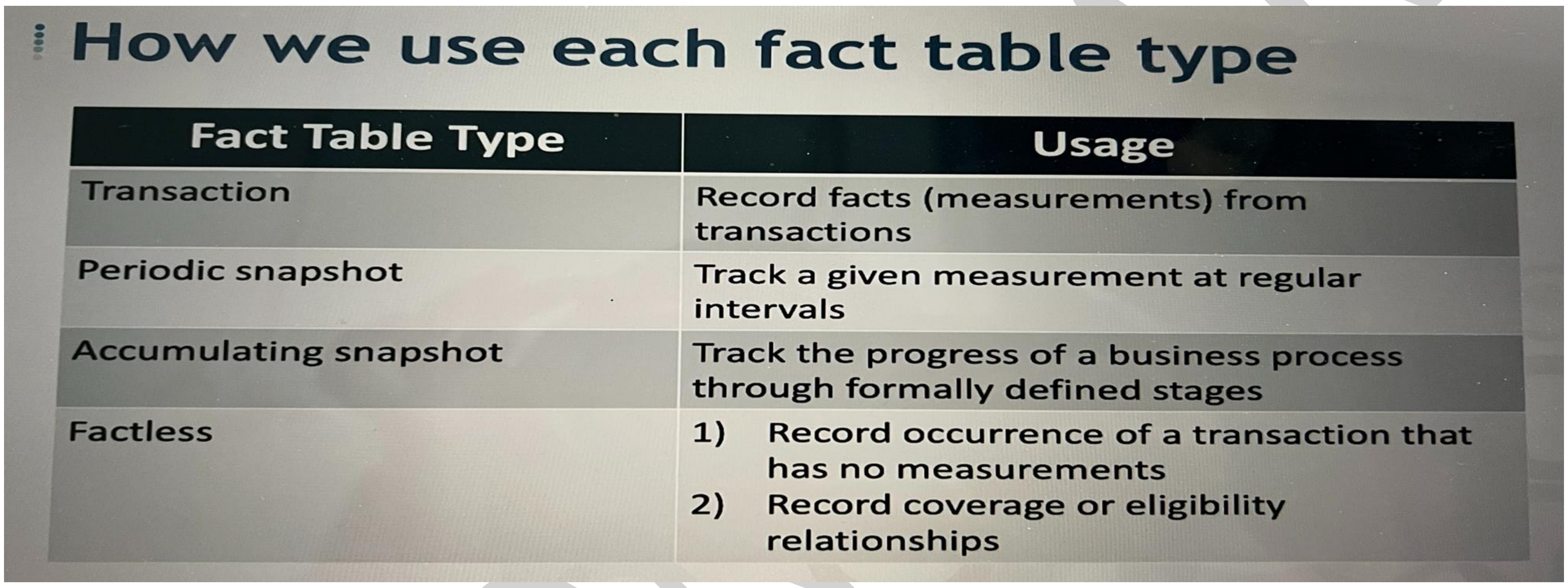
Concepts:

1. Each dimension should form a context to calculate measurements (facts)
2. Each dimension table can have many dimensions in it
3. Each table in DWH should have a primary key which should be ideally surrogate_key

Star Schema	Snowflake Schema
1. Flat dimensions	1. Hierarchical dimensions

2. Has one surrogate key and multiple natural keys for each of the sub-category	2. Has one surrogate key and one natural key in each of the dimension tables 3. They will have foreign keys in the sub category dimension tables College => department (college_key FK) ==>faculty (dept_key FK) Surrogate key became foreign key

Facts:
Types of facts:



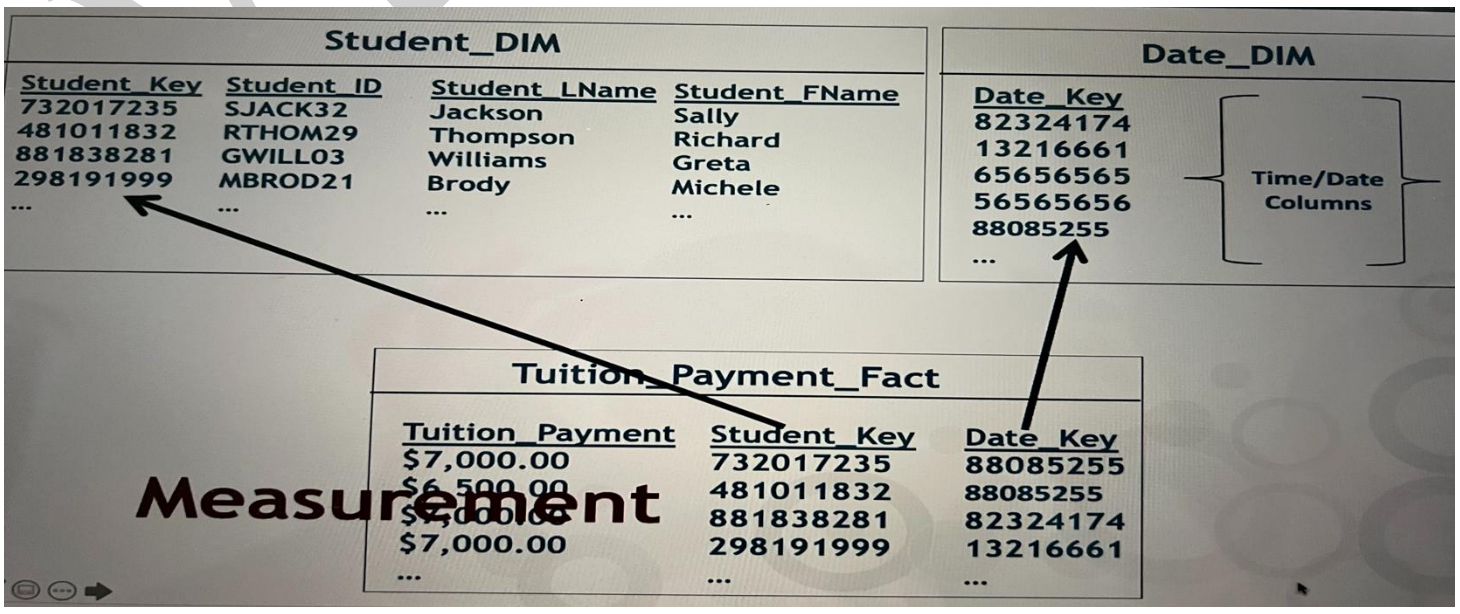
How we use each fact table type

Fact Table Type	Usage
Transaction	Record facts (measurements) from transactions
Periodic snapshot	Track a given measurement at regular intervals
Accumulating snapshot	Track the progress of a business process through formally defined stages
Factless	1) Record occurrence of a transaction that has no measurements 2) Record coverage or eligibility relationships

Transactional Facts:

1. Facts are derived based on transaction data
2. Where we store facts from our transactions
3. One or more facts are stored in the fact table based on rules that imply

Fact table: Measurement + surrogate_key from dimension tables
Surrogate_key = foreign key in fact table
Primary key for fact table = combination of surrogate_key from dimension tables (student_key, date_key)



Student_DIM

Student_Key	Student_ID	Student_LName	Student_FName
732017235	SJACK32	Jackson	Sally
481011832	RTHOM29	Thompson	Richard
881838281	GWILL03	Williams	Greta
298191999	MBROD21	Brody	Michele
...	...	...	...

Date_DIM

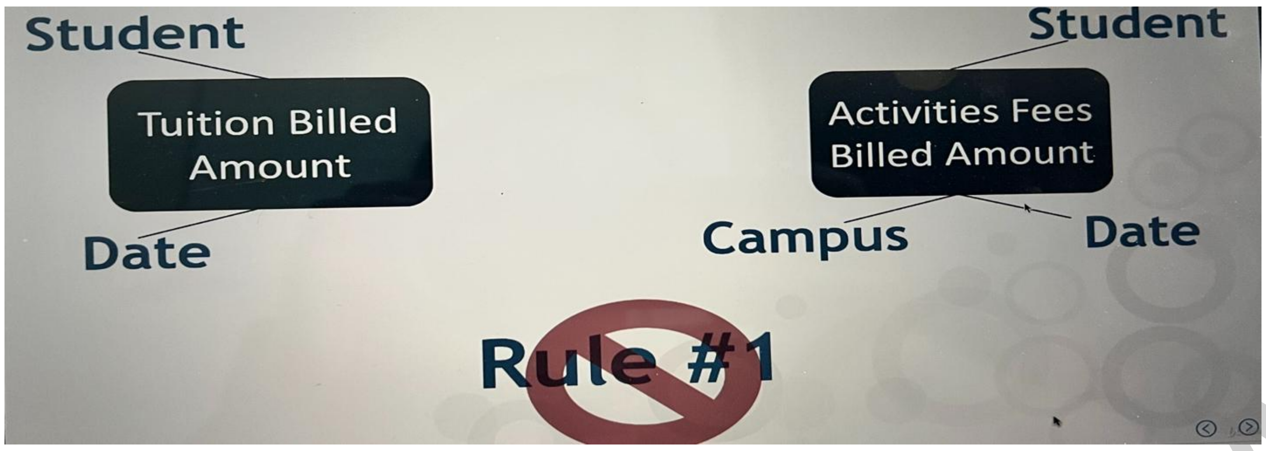
Date_Key
82324174
13216661
65656565
56565656
88085255
...

Tuition_Payment_Fact

Tuition_Payment	Student_Key	Date_Key
$7,000.00	732017235	88085255
$6,500.00	481011832	88085255
$7,000.00	881838281	82324174
$7,000.00	298191999	13216661
...	...	...

Rules to have more than 1 fact in fact table:

Rule 1:
Both the facts should have the same grain
From below tuition_billed_amount => student + date
Activities_fees_billed_amount => student + date + campus
-->Both the facts are derived from different dimensions due to which they can't be together
-->As these facts have different analytics to be performed and have different business processes they can't be together

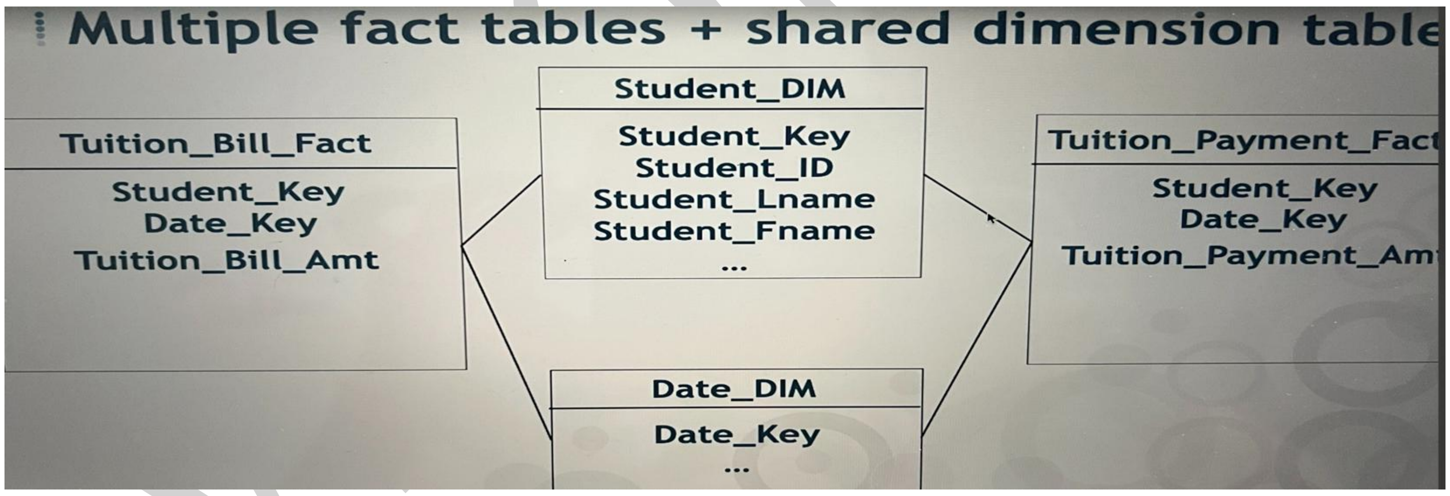


Rule 2:
Both the facts has to occur at the same time

From above, the amount billed to students occurs at the same time
Tuition bill and activities bill are sent to student at the same time

Multiple facts having common dimensions:

-->From below we observe that Rule 1 exists but Rule 2 doesn't
-->One way to fix it is through having 2 fact tables joined by common dimension table



Multiple fact tables + shared dimension table

Student_DIM
Student_Key
Student_ID
Student_Lname
Student_Fname
...

Tuition_Bill_Fact
Student_Key
Date_Key
Tuition_Bill_Amt

Tuition_Payment_Fact
Student_Key
Date_Key
Tuition_Payment_Am

Date_DIM
Date_Key
...

==========================
Slowly Changed Dimensions
==========================

SCD :

1. Dimensions that change slowly over time
2. Dimensions that store and manage both current and historical data over time in a DWH
3. Techniques to manage history within DWH

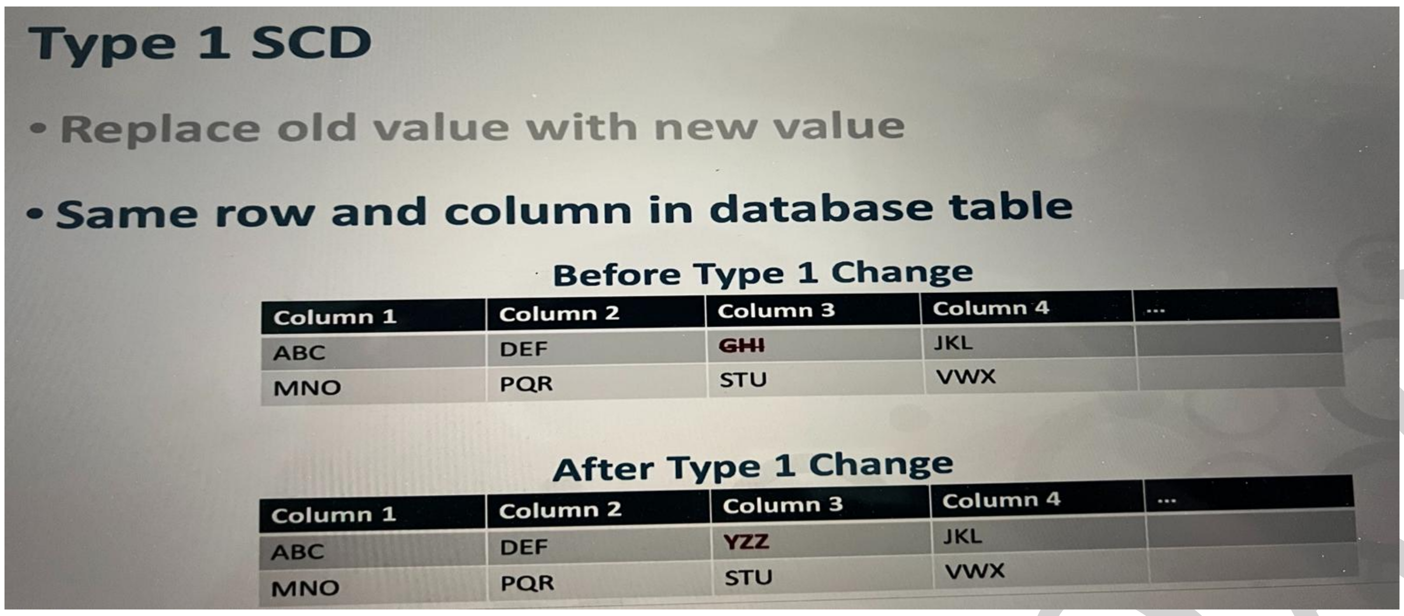
SCD TYPES:

TYPE 1 Overwrite old data and no history retention

TYPE 2	Maintain unlimited history ie all the history versions will be available
TYPE 3	Maintain limited history

TYPE 1 SCD:

1. The row value which needs to be changed is updated with the new value
2. Old value is permanently deleted
3. Mainly useful for correcting errors



Type 1 SCD

- Replace old value with new value
- Same row and column in database table

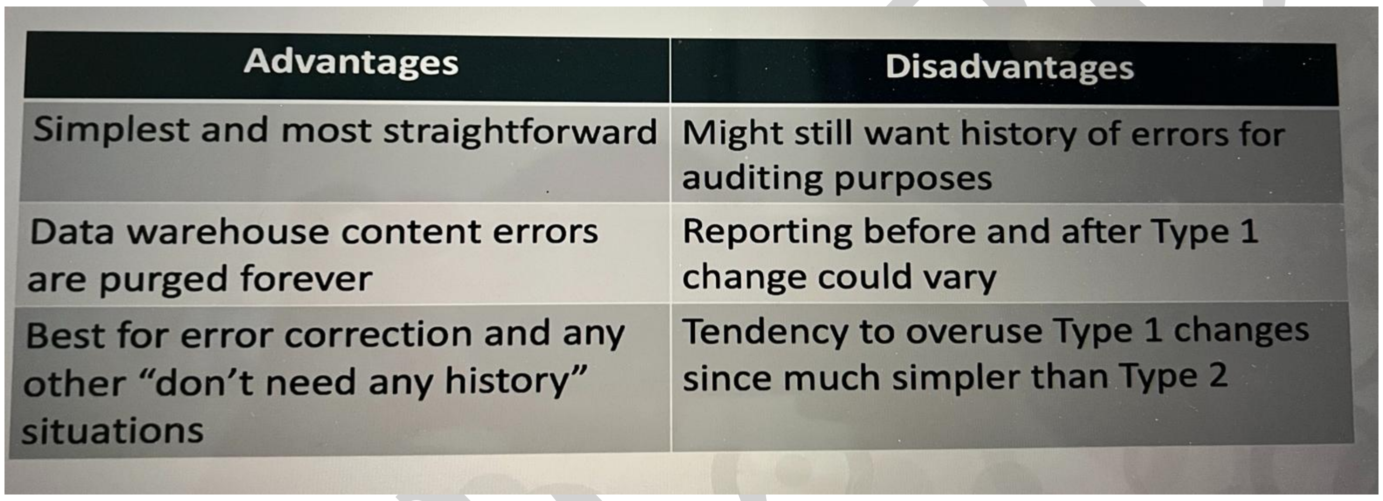
Before Type 1 Change

Column 1	Column 2	Column 3	Column 4	...
ABC	DEF	~~GHI~~	JKL	
MNO	PQR	STU	VWX	

After Type 1 Change

Column 1	Column 2	Column 3	Column 4	...
ABC	DEF	YZZ	JKL	
MNO	PQR	STU	VWX	

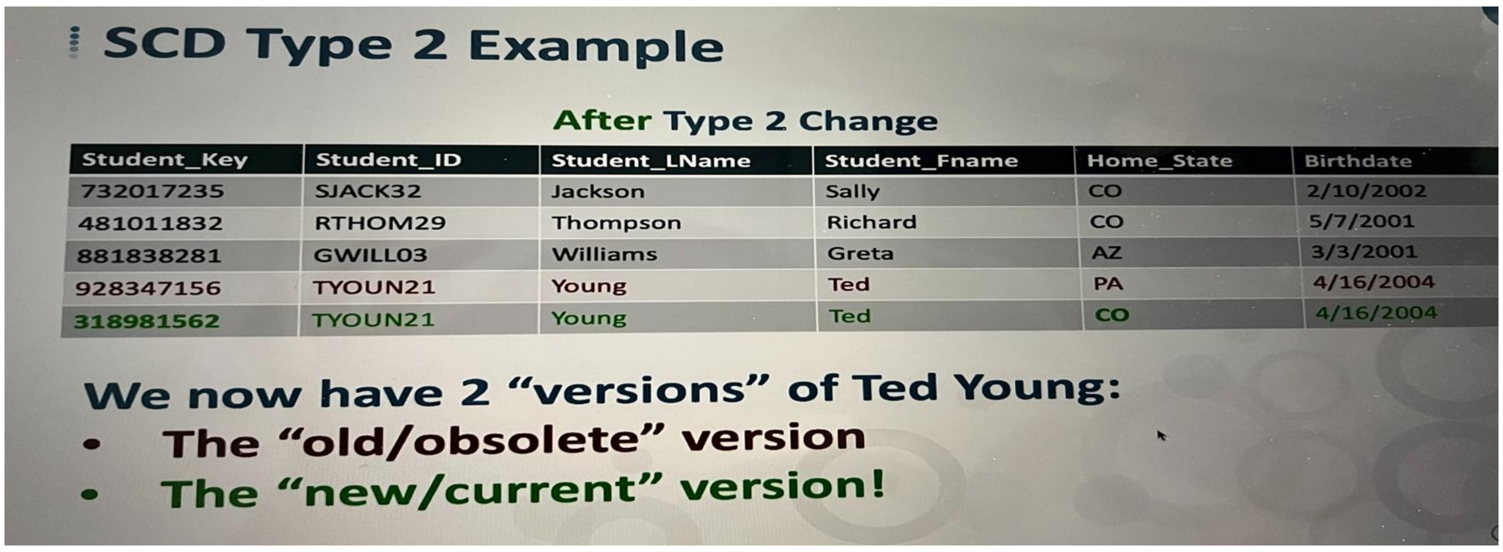
Pros and Cons:



Advantages	Disadvantages
Simplest and most straightforward	Might still want history of errors for auditing purposes
Data warehouse content errors are purged forever	Reporting before and after Type 1 change could vary
Best for error correction and any other "don't need any history" situations	Tendency to overuse Type 1 changes since much simpler than Type 2

TYPE 2 SCD:

1. The data column value which is updated is stored as a new row and the old value also exists in a different row
2. A new surrogate key is generated for the updated data row
3. Reports and analytics data before and after Type 2 SCD will give accurate results
4. Analytics done on old and new data changes can be captured accurately
5. Historical analysis done on historical data



SCD Type 2 Example

After Type 2 Change

Student_Key	Student_ID	Student_LName	Student_Fname	Home_State	Birthdate
732017235	SJACK32	Jackson	Sally	CO	2/10/2002
481011832	RTHOM29	Thompson	Richard	CO	5/7/2001
881838281	GWILL03	Williams	Greta	AZ	3/3/2001
928347156	TYOUN21	Young	Ted	PA	4/16/2004
318981562	TYOUN21	Young	Ted	CO	4/16/2004

We now have 2 "versions" of Ted Young:

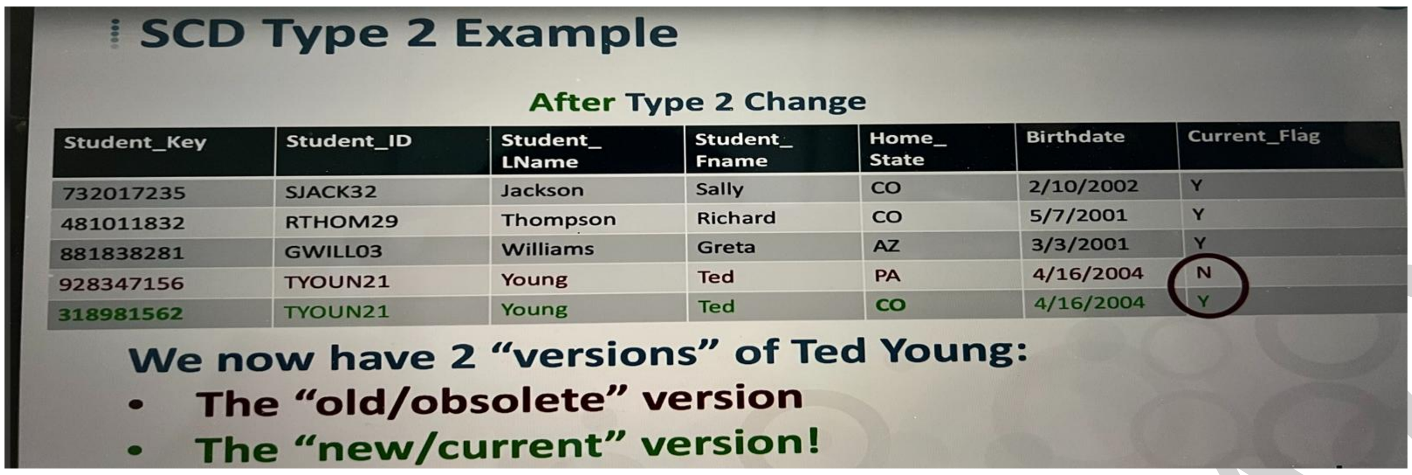
- The "old/obsolete" version
- The "new/current" version!

Cons:

1. Huge storage in fact tables as all history versions of data is retained
2. Additional column fields to be included in fact table to identify the old and new versions of data.
3. Include natural keys from dimension tables for better identification of rows which differ only in surrogate keys but all details remain the same

Ways to handle SCD-2:

1. Include a new column called "current_flag" which will reflect if any changes done on the data row



SCD Type 2 Example

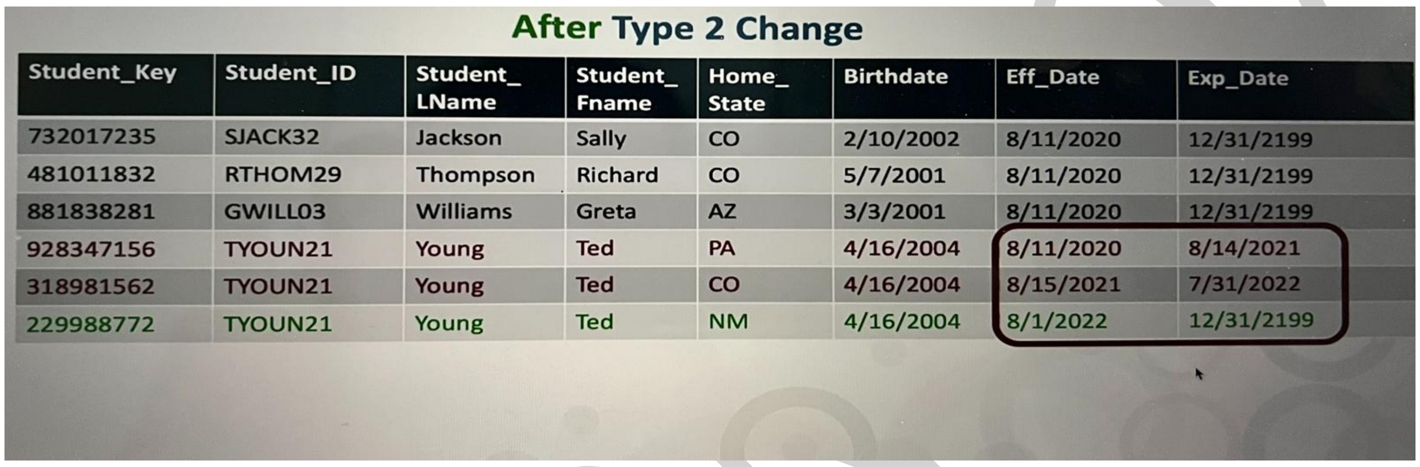
After Type 2 Change

Student_Key	Student_ID	Student_ LName	Student_ Fname	Home_ State	Birthdate	Current_Flag
732017235	SJACK32	Jackson	Sally	CO	2/10/2002	Y
481011832	RTHOM29	Thompson	Richard	CO	5/7/2001	Y
881838281	GWILL03	Williams	Greta	AZ	3/3/2001	Y
928347156	TYOUN21	Young	Ted	PA	4/16/2004	N
318981562	TYOUN21	Young	Ted	CO	4/16/2004	Y

We now have 2 "versions" of Ted Young:

- The "old/obsolete" version
- The "new/current" version!

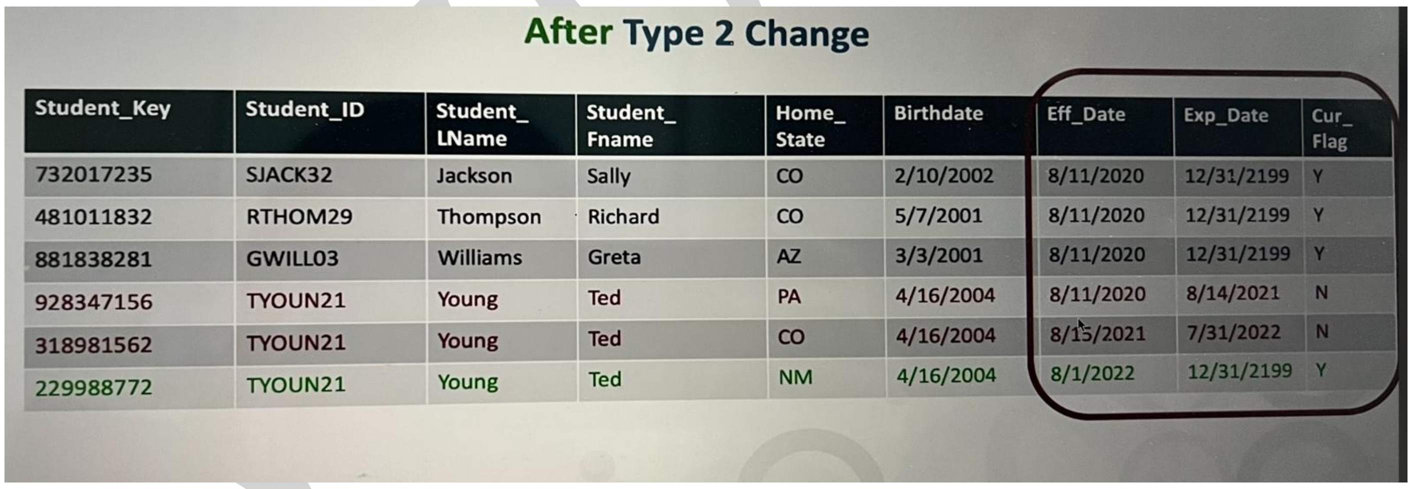
2. Include columns of "effective_date" and "expiry_date"



After Type 2 Change

Student_Key	Student_ID	Student_ LName	Student_ Fname	Home_ State	Birthdate	Eff_Date	Exp_Date
732017235	SJACK32	Jackson	Sally	CO	2/10/2002	8/11/2020	12/31/2199
481011832	RTHOM29	Thompson	Richard	CO	5/7/2001	8/11/2020	12/31/2199
881838281	GWILL03	Williams	Greta	AZ	3/3/2001	8/11/2020	12/31/2199
928347156	TYOUN21	Young	Ted	PA	4/16/2004	8/11/2020	8/14/2021
318981562	TYOUN21	Young	Ted	CO	4/16/2004	8/15/2021	7/31/2022
229988772	TYOUN21	Young	Ted	NM	4/16/2004	8/1/2022	12/31/2199

3. Combining both the above solutions



After Type 2 Change

Student_Key	Student_ID	Student_ LName	Student_ Fname	Home_ State	Birthdate	Eff_Date	Exp_Date	Cur_ Flag
732017235	SJACK32	Jackson	Sally	CO	2/10/2002	8/11/2020	12/31/2199	Y
481011832	RTHOM29	Thompson	Richard	CO	5/7/2001	8/11/2020	12/31/2199	Y
881838281	GWILL03	Williams	Greta	AZ	3/3/2001	8/11/2020	12/31/2199	Y
928347156	TYOUN21	Young	Ted	PA	4/16/2004	8/11/2020	8/14/2021	N
318981562	TYOUN21	Young	Ted	CO	4/16/2004	8/15/2021	7/31/2022	N
229988772	TYOUN21	Young	Ted	NM	4/16/2004	8/1/2022	12/31/2199	Y

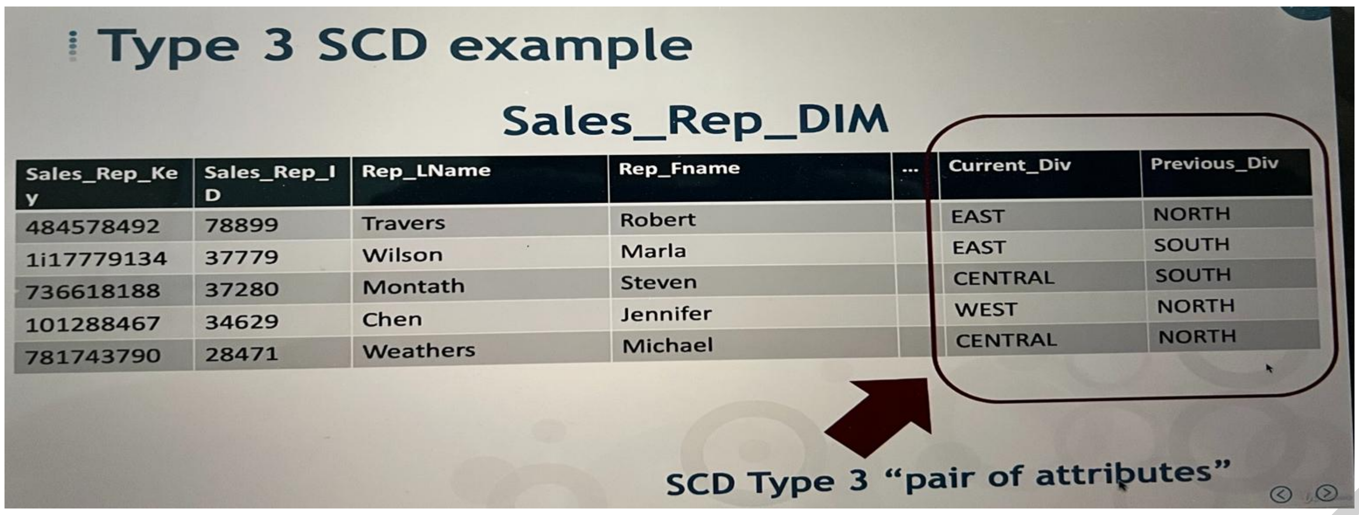
Type 3 SCD:

1. Add a new column rather than a new row to reflect the changes done
2. Column for "old value" and "new value"
3. Supports back and forth switching for effective reporting

Let's take an example when we have a re-organization taking place. Older versions are having divisions to be north and south.
Newer versions are having divisions to be east, west and central

Cons:

1. It is not suitable for dwh where various columns are changed like place, country, address, pincode
2. It is suitable only for use cases where changes are limited



Type 3 SCD example

Sales_Rep_DIM

Sales_Rep_Key	Sales_Rep_ID	Rep_LName	Rep_Fname	...	Current_Div	Previous_Div
484578492	78899	Travers	Robert		EAST	NORTH
1i17779134	37779	Wilson	Marla		EAST	SOUTH
736618188	37280	Montath	Steven		CENTRAL	SOUTH
101288467	34629	Chen	Jennifer		WEST	NORTH
781743790	28471	Weathers	Michael		CENTRAL	NORTH

SCD Type 3 “pair of attributes”