Data Warehousing Fundamentals for Beginner

By

Alan Simon

Course Link

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

Notes By

Naveen Kumar M

https://www.linkedin.com/in/naveen-kumar-m-5a6960119/

Data Warehousing Concepts

What is Data Warehousing?

- Data warehousing as a discipline began right around 1990.
- Data warehousing is among the most durable, high impact disciplines we've put to work.
- Born out of necessity when nineteen eighties era attempts to get a handle on proliferation of organisational data that was dispersed among dozens or hundreds or even thousands of systems.
- Data warehousing came of age during the nineteen nineties alongside other movements of the day, such as enterprise resource planning or ERP, the Internet and e-commerce, customer relationship management, or CRM, and the growth and explosion of distributed technology itself.
- Data warehousing helps us corral and integrate and organize our data from around the enterprise and even outside our enterprise. And then business intelligence and its modern analytics successor's help us make data driven decisions from that integrated information.
- Think of the entire discipline of data warehousing as a fundamental underpinning to data driven decision making that most users don't fully see.

We can think of data warehouse as a fairly large warehouse filled with data rather than physical products.

Data warehouse is not the same as database but it’s build on top of database.

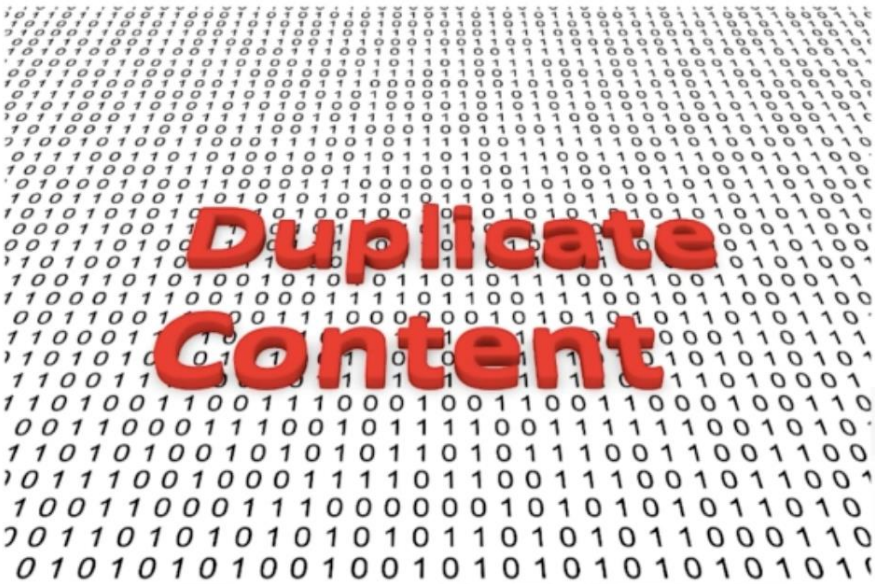
So, here database is a platform and data warehouse is usage.

Data comes from elsewhere

Data inside the data warehouse comes from the operational systems or other external sources. We don't create a data for the first time in data warehouse. Instead, data from different source are sent to the data warehouse.

Possibly dozens of data sources

Data copied...not moved



Know the Rules

Integrated Environment

- Data from different sources are sent into the data ware housing.

Subject Oriented

- Should organize the data with subject regardless of the source system.

Time Variant

- Historical Data

Non volatile

- Data warehouse should be stable to periodic update. Data can be in refreshable in batches in millions.

Improvement

- Re-structure and re-organizable.

Decision

- Making the decision based on the data.

Reason for you to build the data warehouse

Two primary reason

- Making a data driven decisions
- One stop shopping

Making a data drive decisions

- Past data
- Present data
- Future data
- The unknown data

Business Intelligence + Data warehousing = Best Value

One Stop Shopping

We can explain one stop shopping by explaining what if don't have a one stop shopping.

Data driven decision making back in old became very tedious and problematic because of data scattered all over the place.

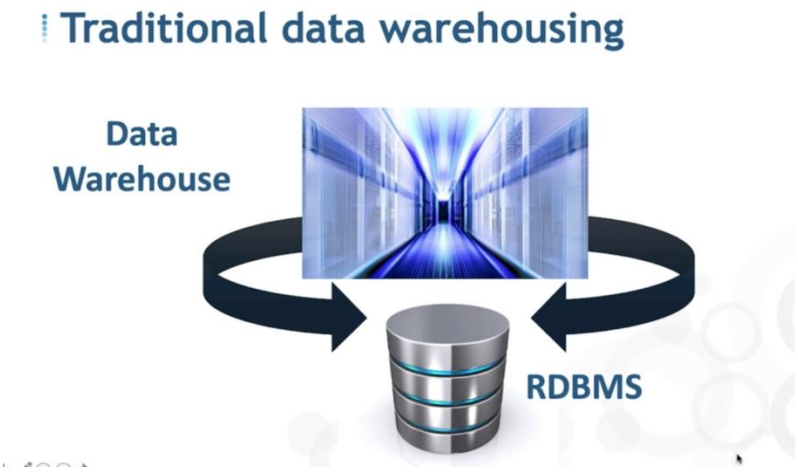
If we have a one stop shopping, we can concentrate on the analysis of the data rather than gathering the data over and over again.

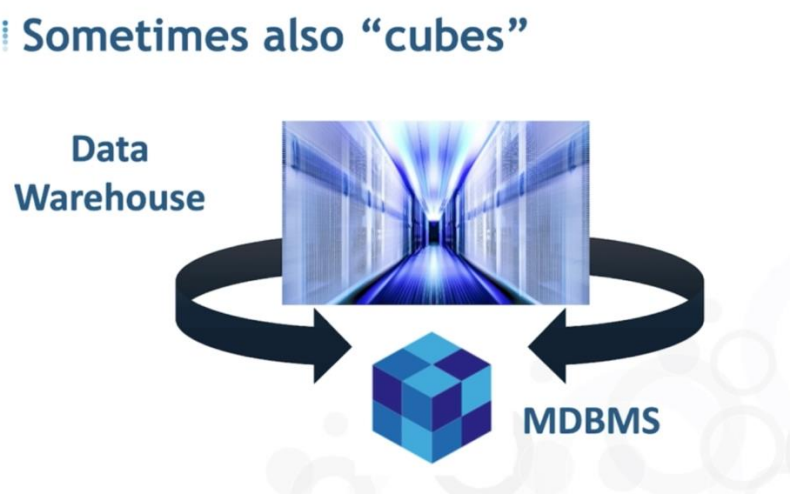
Business Intelligence and data ware housing, the sibling discipline provide the immerse value.

Compare a Data Warehouse to a Data Lake

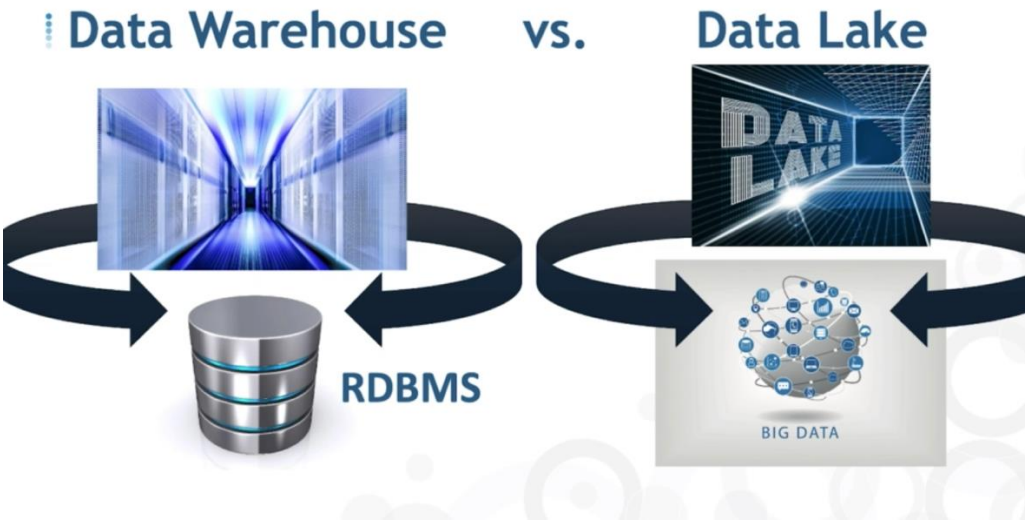
Data warehouse – Back 1990
Data Lake – more recent times

Similarity and Differences





Data warehousing build on top of the RDBMS and also Multi-Dimensional Data base.



Data lake build on top of the big data environment rather than traditional relational database.

3 V’s of Big Data

Volume

- Big data manages the extremely large volumes of data.

Velocity

- Rapid intake of data.

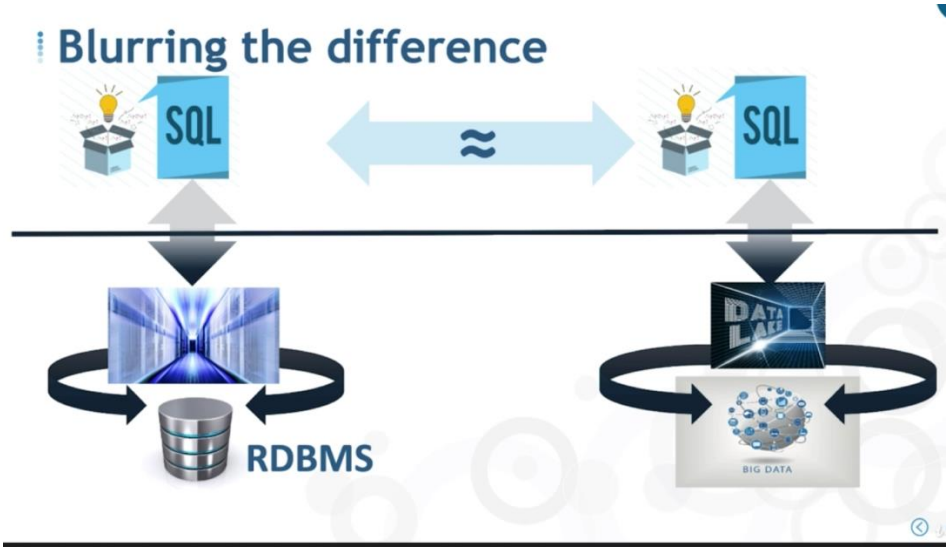
Variety

- Supports semi and unstructured data with structured data.
- Semi structure data includes – text messages, email, blob.
- Unstructured data - Audio and Video

Next Generation

3v's expand the data driven decision making capabilities.

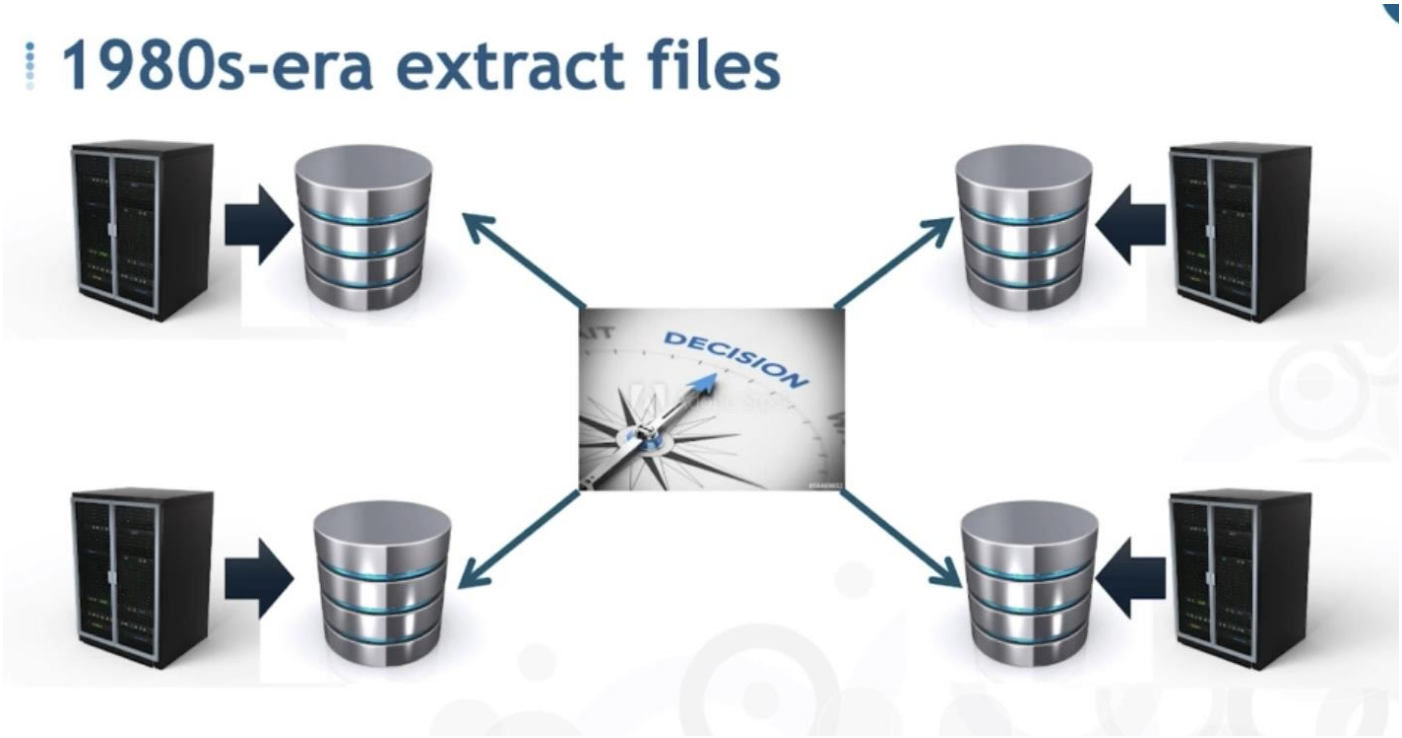
Blurring the difference



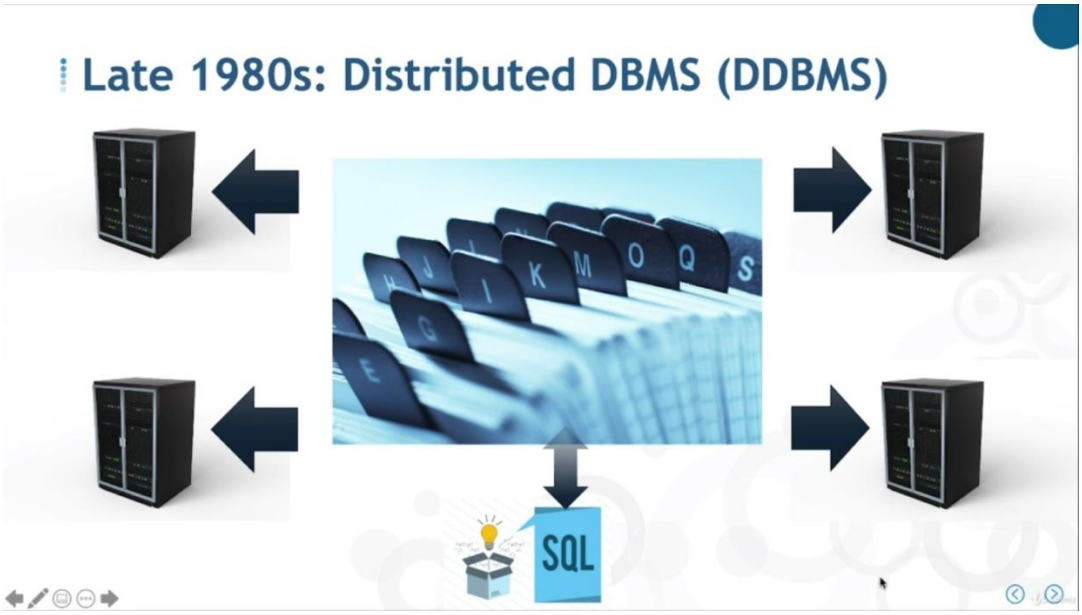
SQL Endpoint can also be built on top of the Data Lake similar to RDBMS.

Compare Data Warehouse to Data Virtualization

History

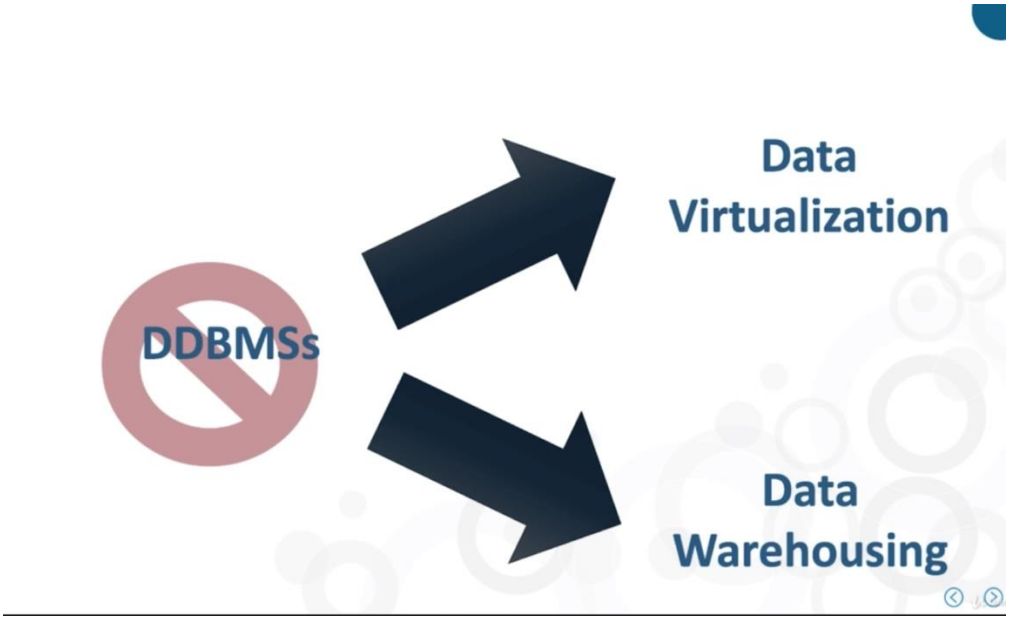


Before data warehousing, keep on extracting the data for decision making.



Earlier in 1980s, there is no powerful enough technology to sustain the concept of distributed DBMS.

So, DDBMS is a failure one. This failure diverse into two different path.



Data Virtualization

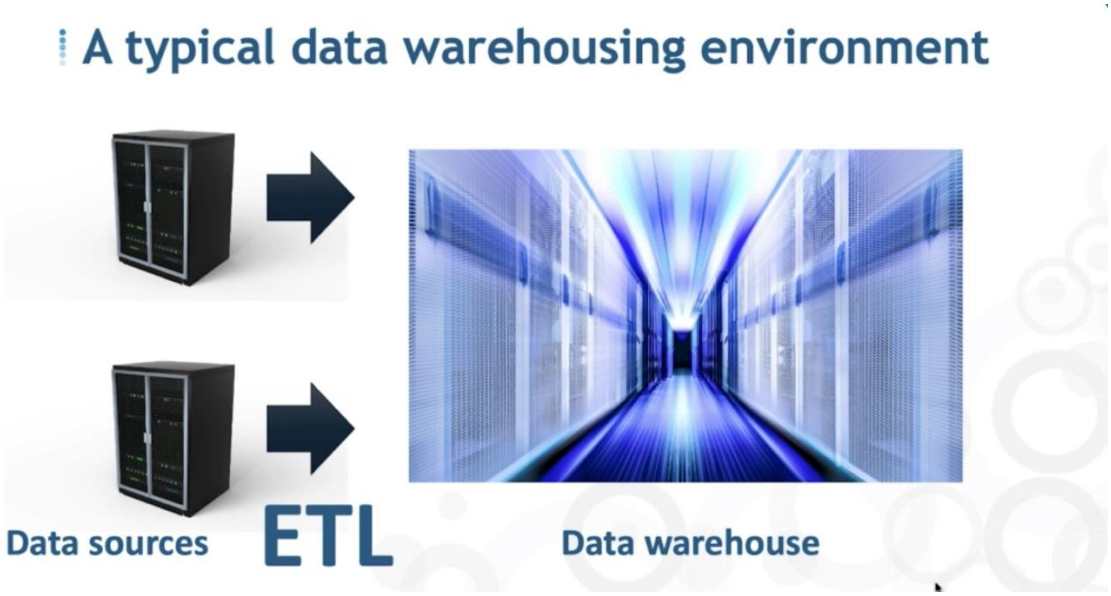
- Read only DDBMS.
- In-place data access. So no duplicate content.
- Many names over the years.
    - Virtual Data warehousing
    - Enterprise Information Integration (EII)
    - Enterprise Data Access (EDA)
    - Data Virtualization

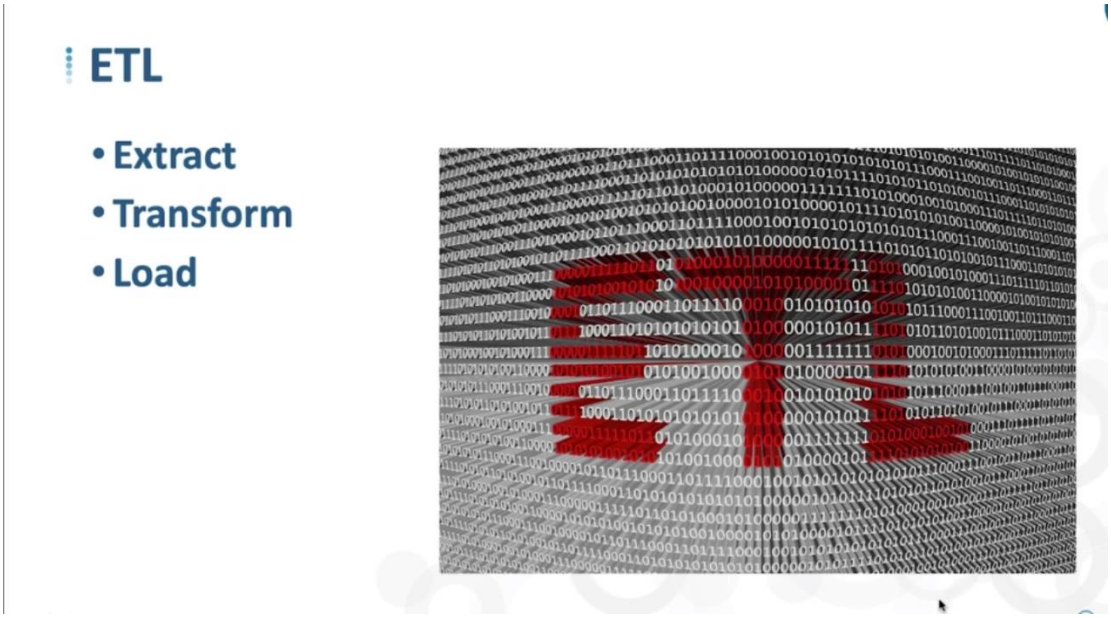
Data virtualization use cases

- Simple transformation
- Smaller number of data sources
- Relaxed response time

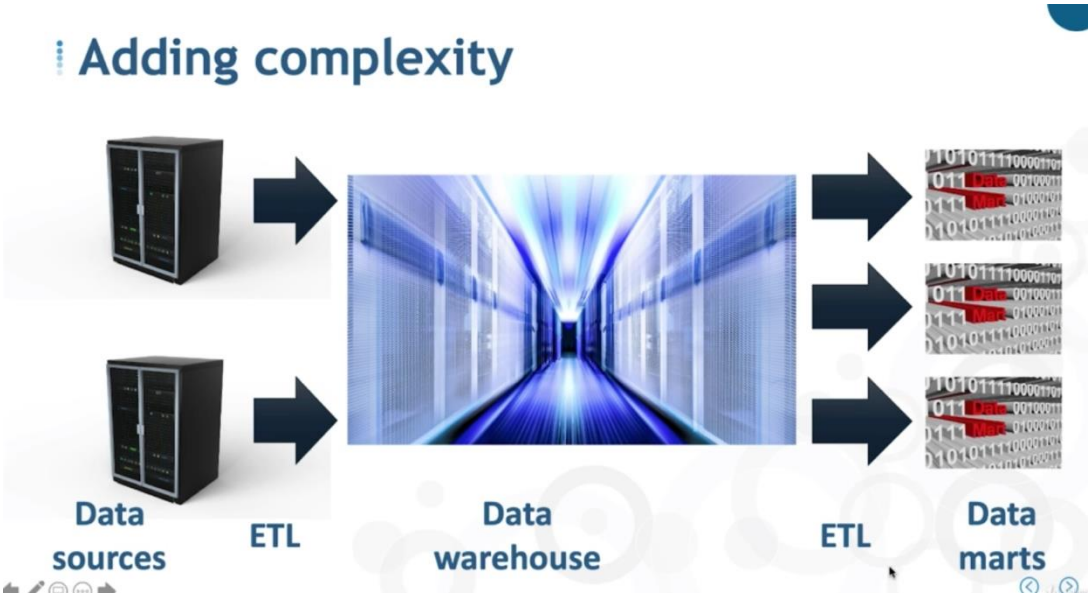
Simple End to End Data Warehousing environment

ETL





Adding complexity



Smaller Environment = Data marts

Analogy

Data marts is subset of data from data warehouses which is tailored to serve the certain business use cases.

Data Warehousing Architecture

Introduction to Data warehousing architecture

Architectural Option

Centralized Data Warehouse

- Straightforward
- Building a single central database for storing all the data for our application.
- Home for our data
- To support business intelligence and analytics.

Data Marts

- Smaller scale or more narrowly focused data warehouse.

Component Based Data Warehousing

- Opposite of centralized approach.
- Multiple component of data warehousing and data marts operate together to comprise to serve overall data warehousing environment.

Cube

Operational Data store

Staging layer.

- Persistent
- Non persistent

Build a Centralized Data Warehouse

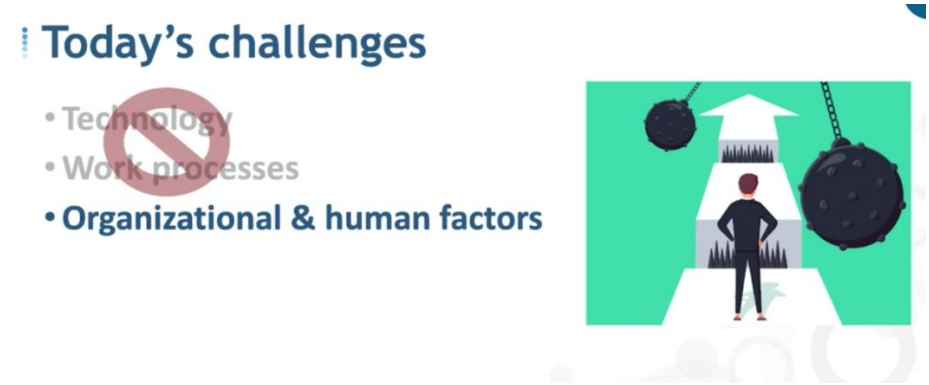
Single Database

- Data from all the sources are feed into single database.

One Stop Shopping

- Support for one stop shopping.
- All the data needed for reporting, business intelligence and analytics are from single data store.

Historical Challenges

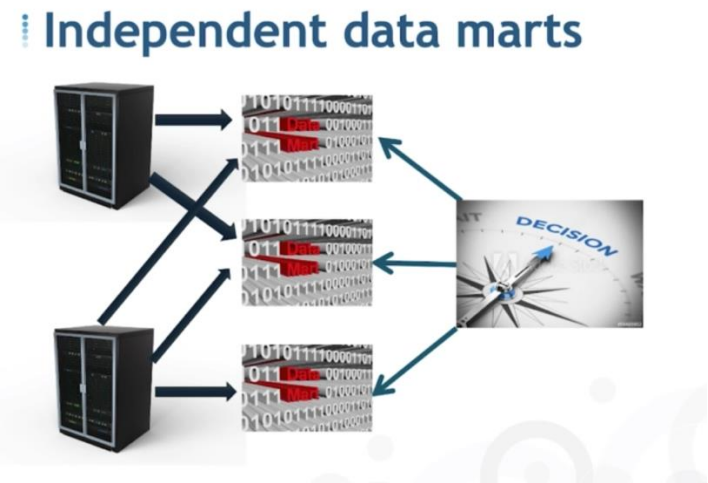


Compare a data warehouse to a data mart

Dependant Data Mart

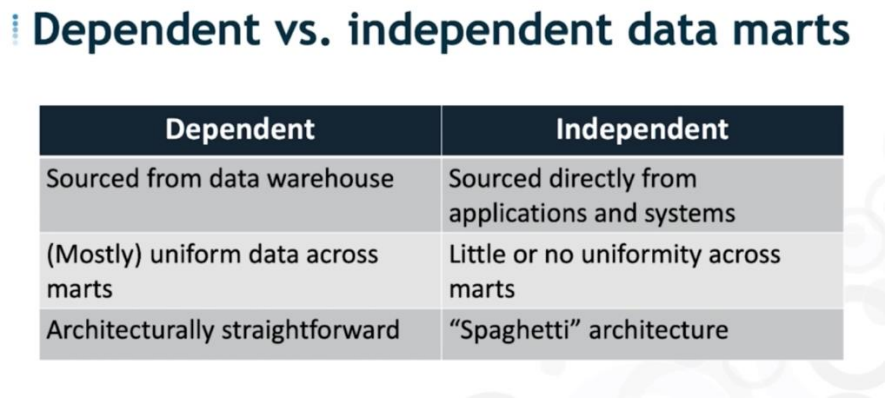
- Here, data marts are directly dependent on data warehouse.
- Which means, without data warehouse data marts do not exists.

Independent data marts

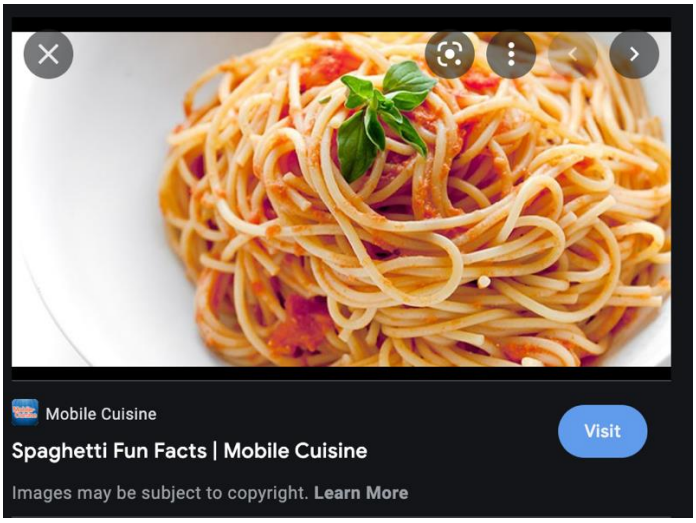


- Data warehouse does not required.
- Data marts are built from one or more data sources.
- An Independent data marts are similar to 1980s-era extract files with exception that data in data marts are organized dimensionally.
- Small scale data warehouse with organized the data.

Dependent vs Independent data marts



Dependent	Independent
Sourced from data warehouse	Sourced directly from applications and systems
(Mostly) uniform data across marts	Little or no uniformity across marts
Architecturally straightforward	"Spaghetti" architecture

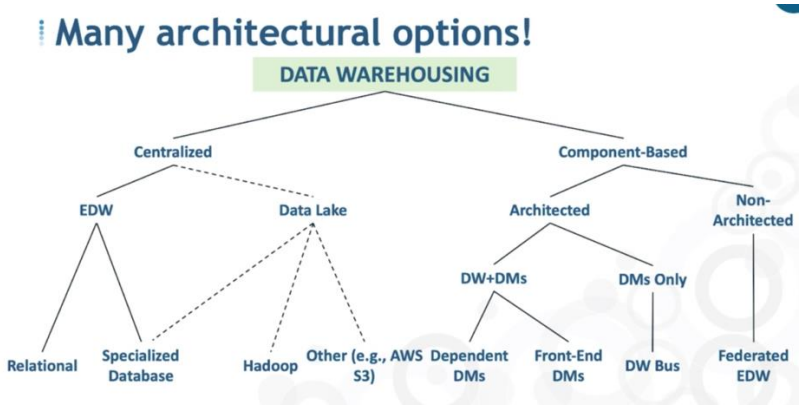


Data warehouse vs Independent data mart

Data warehouse vs. independent data mart

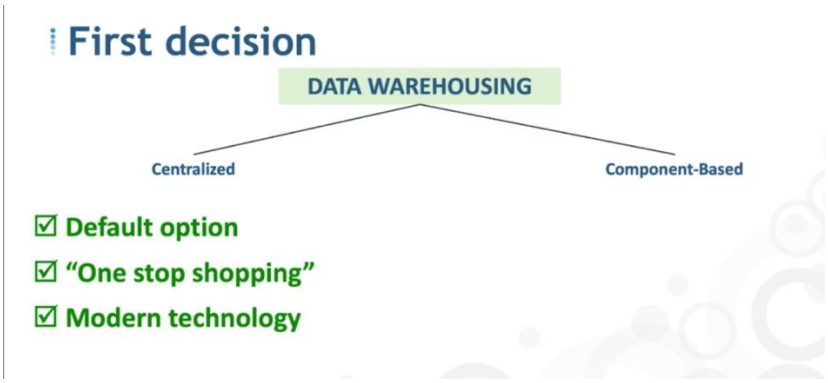
Data Warehouse	Independent Data Mart
Many sources	One or more sources
ETL from sources	ETL from sources
Probably large data volumes	Possibly large data volumes
Dimensionally organized data	Dimensionally organized data

Data Warehousing Architectural Options

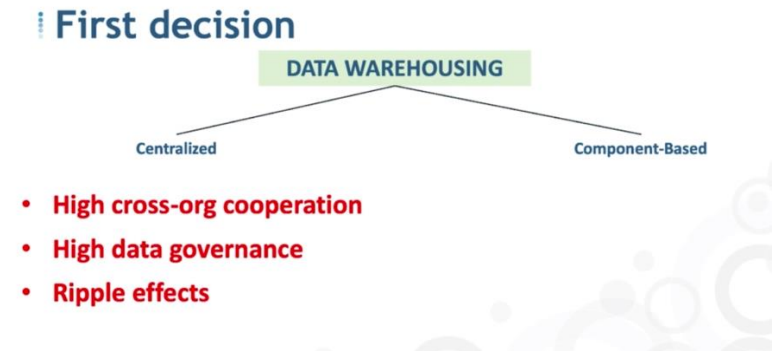


Centralized Data Warehouse

Advantage

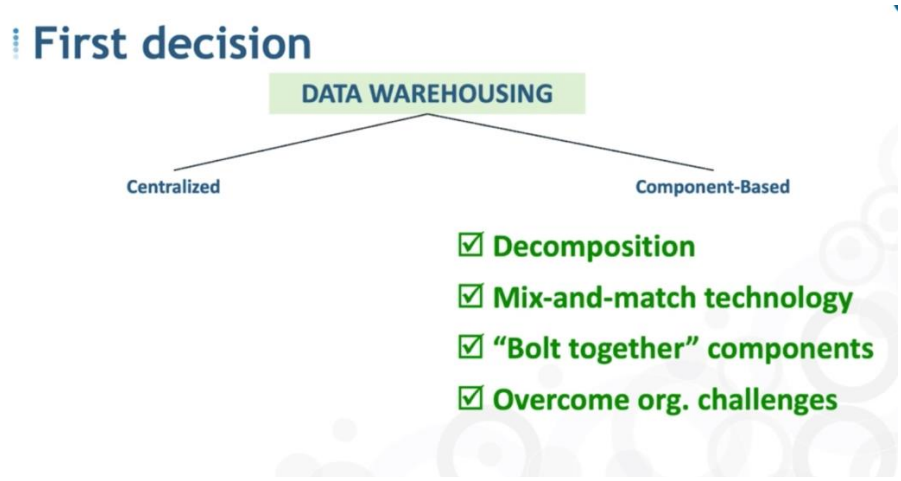


Disadvantage

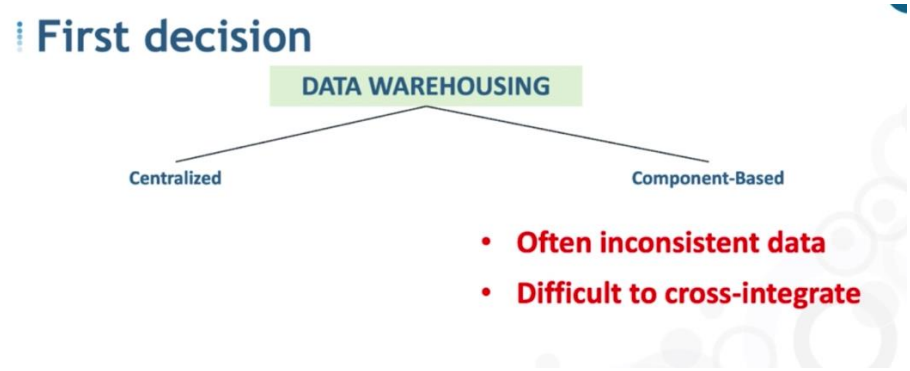


Component Based Data Warehouse

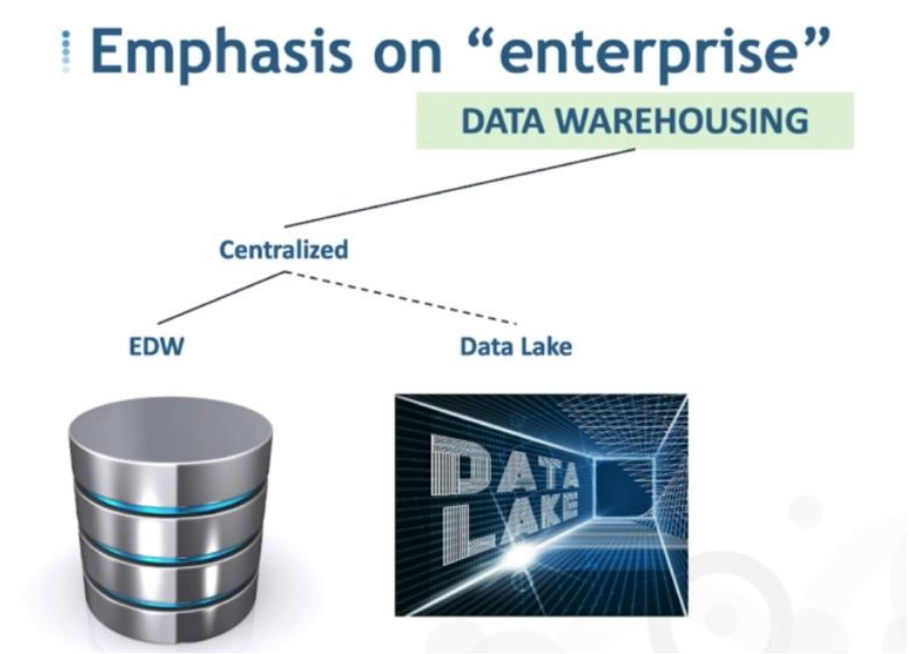
Advantage



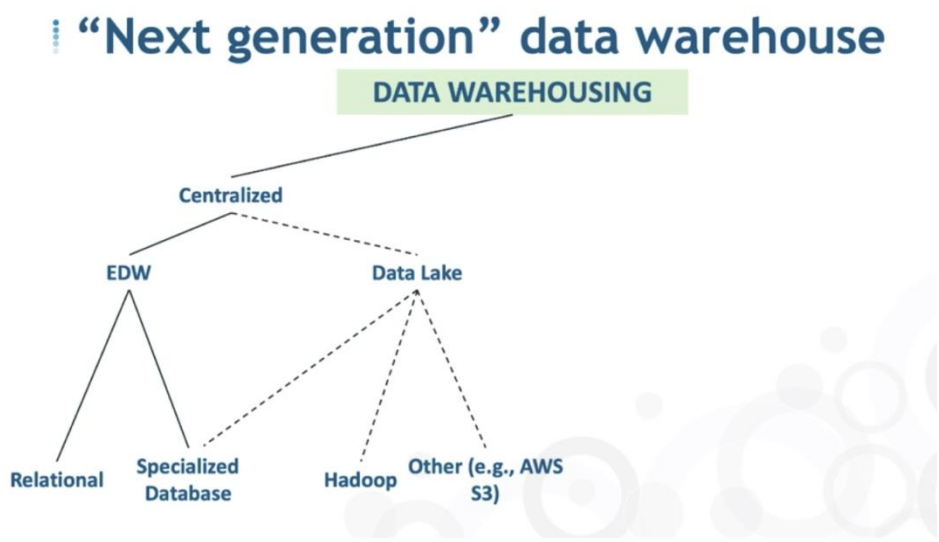
Disadvantage



Centralized Data Warehouse Options

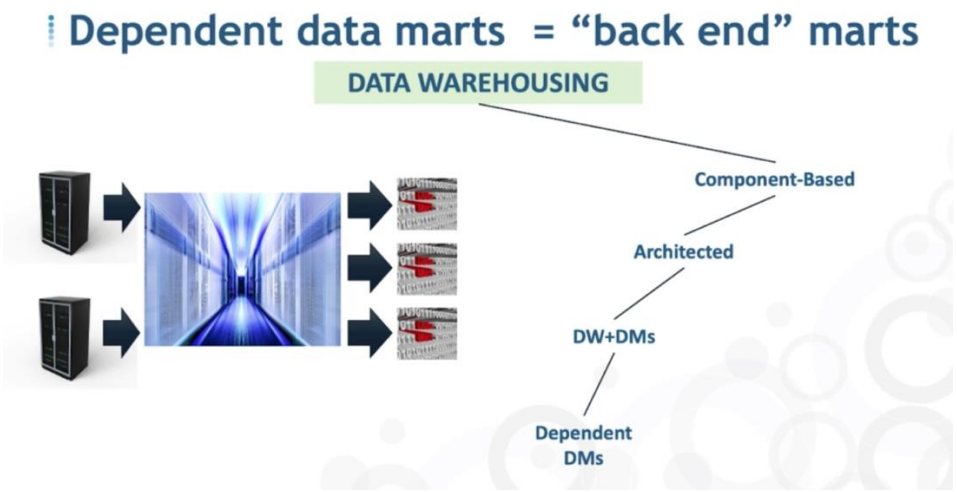


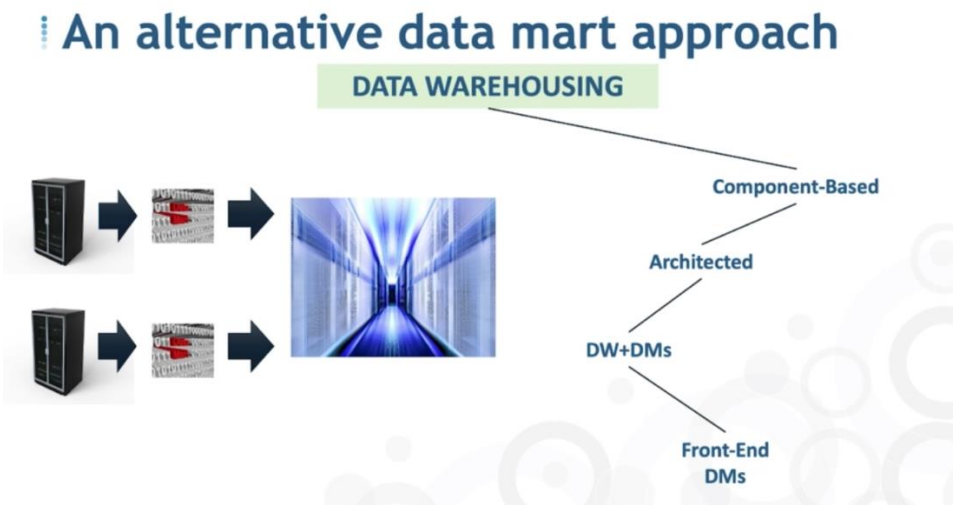
EDW – Enterprise Data Warehouse



Dependent DM = Cooperate Data Information Factory

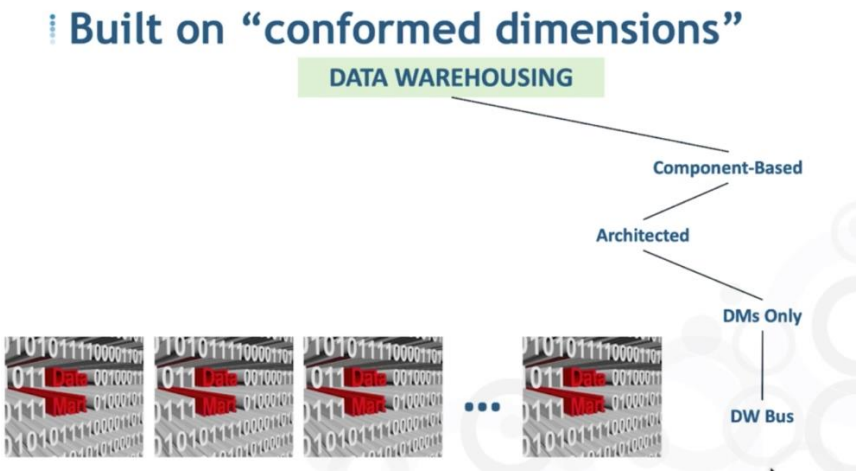
Component Based Data Warehouse Options



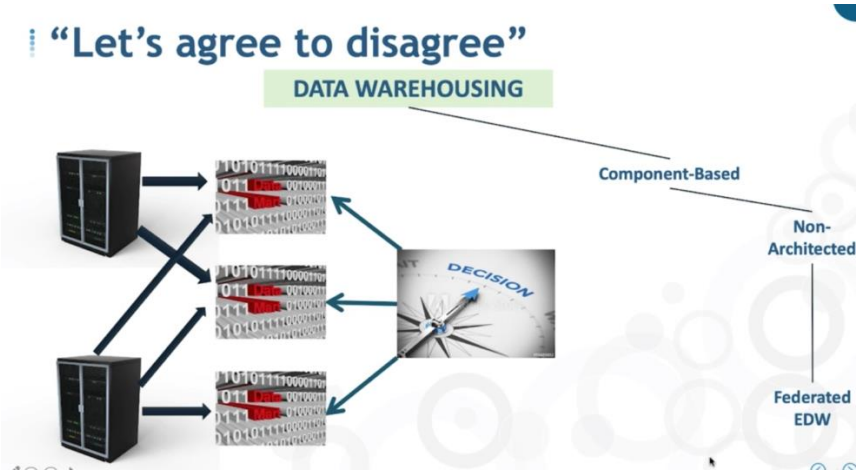


Here, Data marts are in front of data warehouse.

Data Marts Only

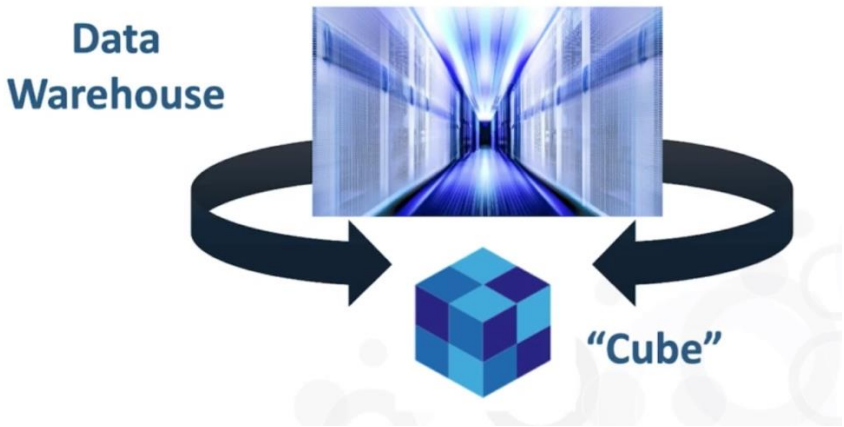


Federated Enterprise Data Warehouse



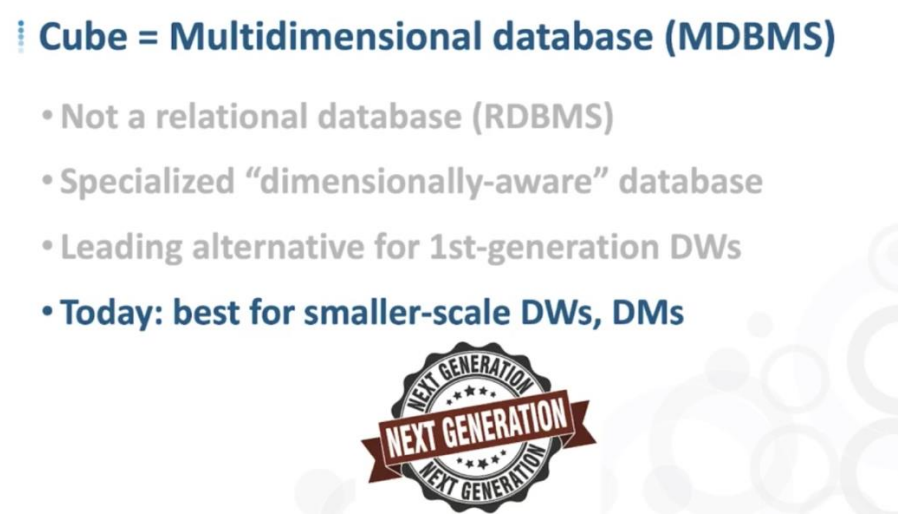
This option is last resort. Because, something is better than nothing.

Including Cubes in Data Warehousing Environment

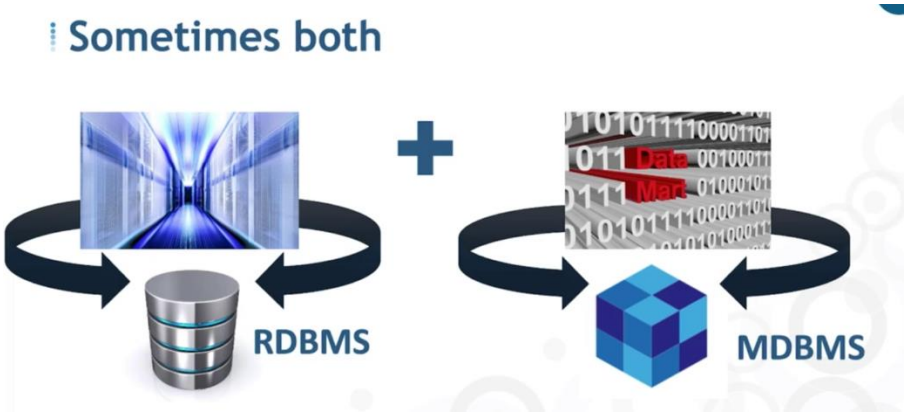


Data warehousing can be built on top of cube.

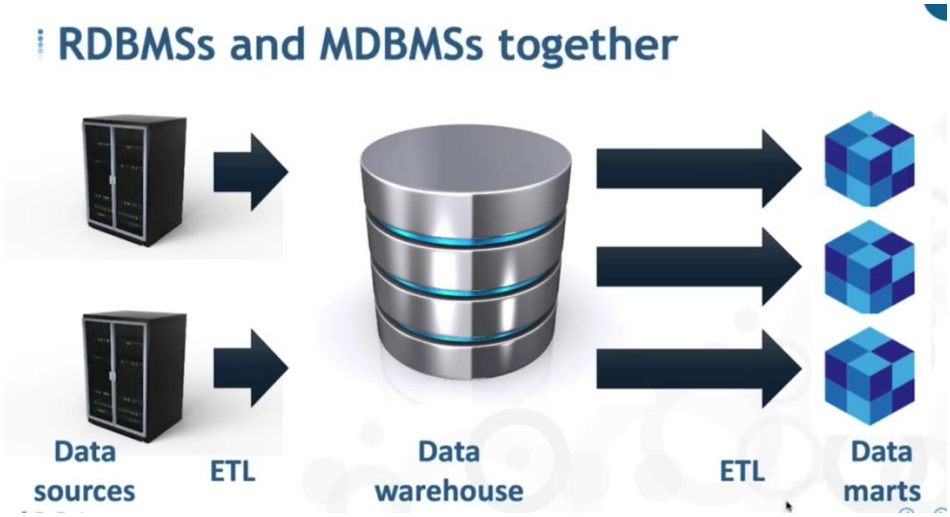
What is Cube ?

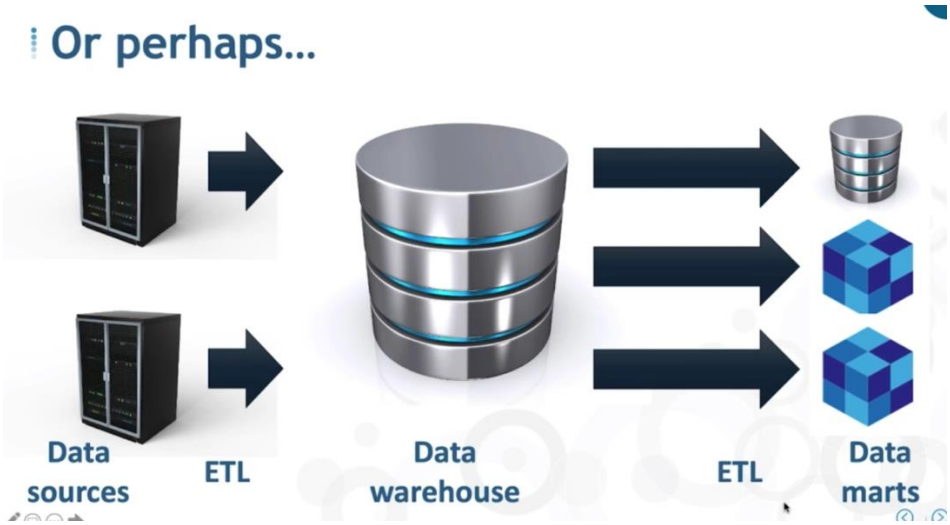


Cube Advantages and Disadvantages

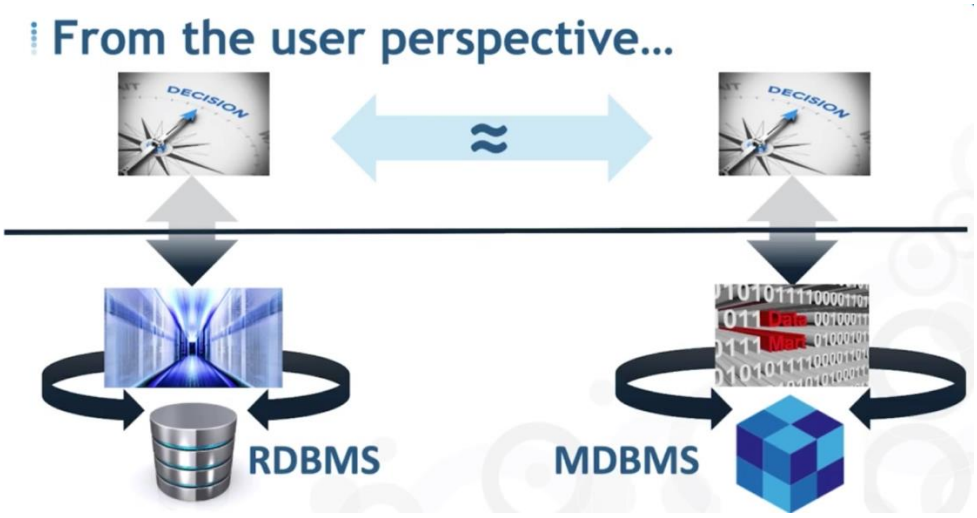


RDBMSs and MDBMSs together

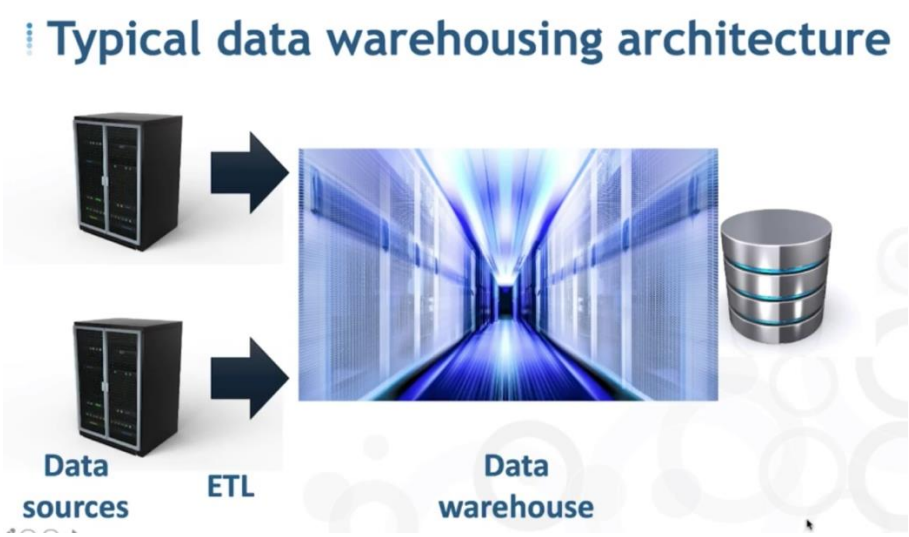


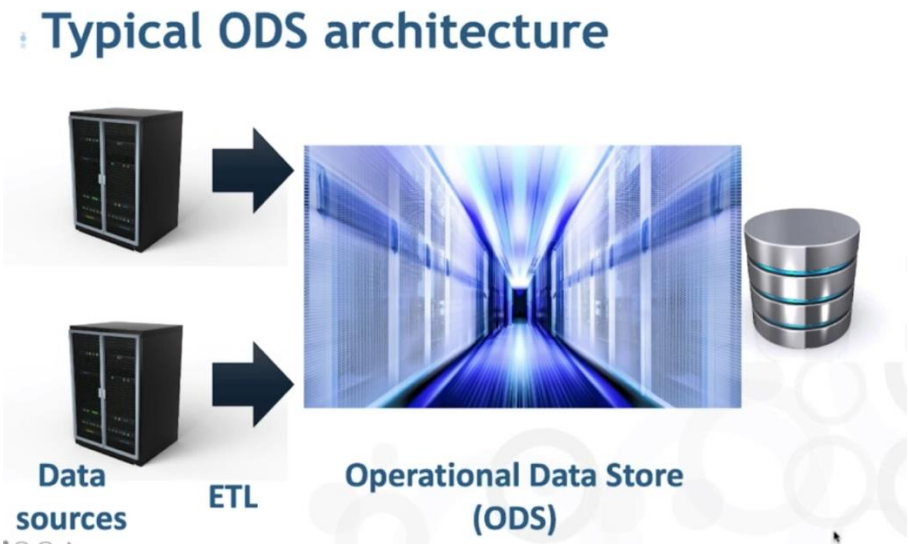


User Perspective on RDBMS and MDBMS



Including Operation Data Stores in Data Warehousing Environment

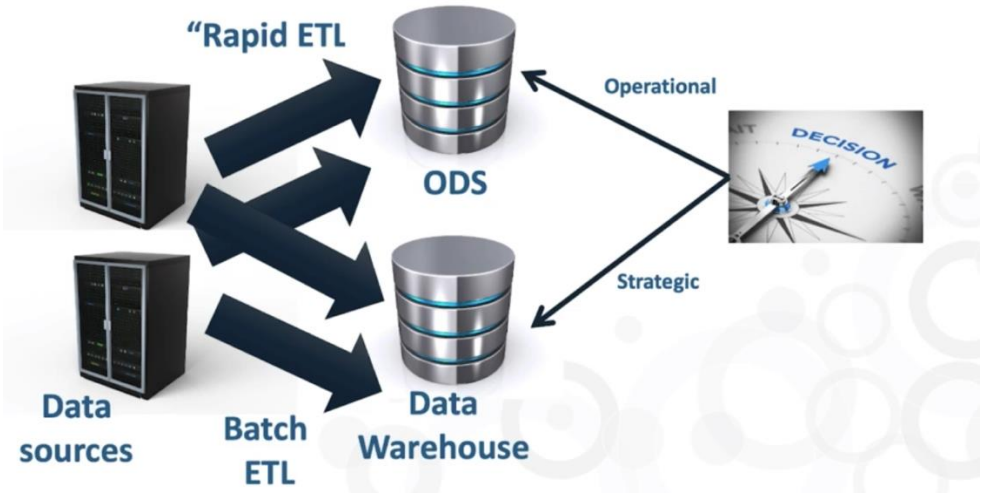


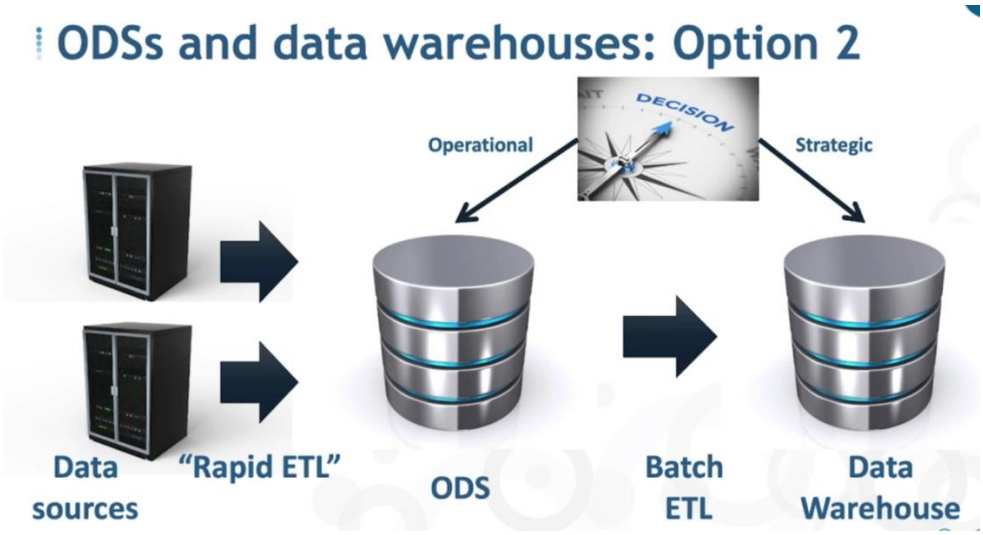


Then what is the difference between data warehouse and operational data store.

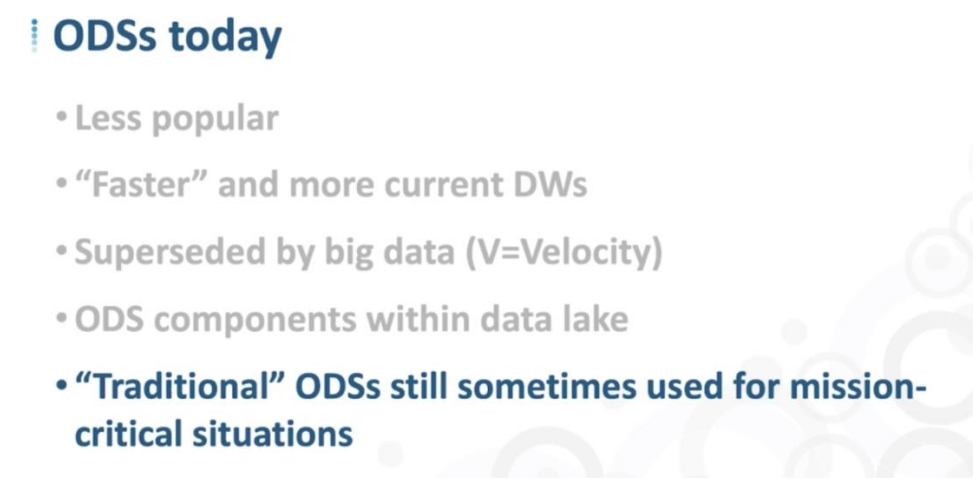
What is an ODS ?

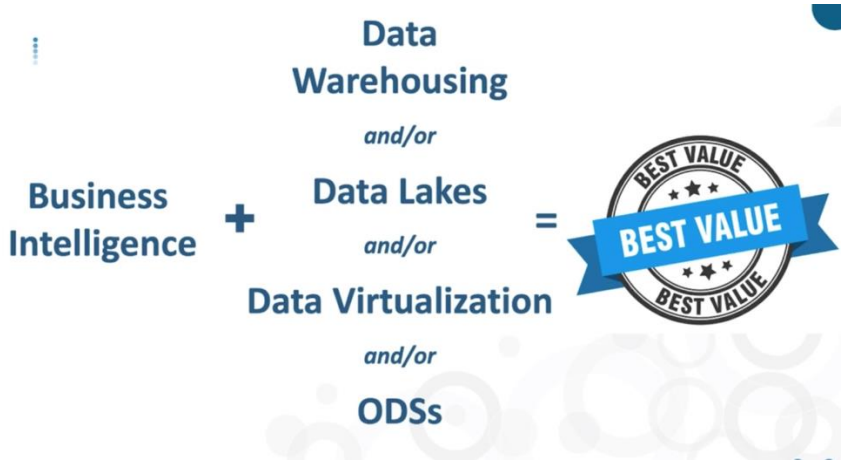
- Does not maintain the history.
- Real time data feed.
- Operational Decision making.


ODSs and data warehouses: Option 1
"Rapid ETL
Operational
DECISION
ODS
Strategic
Data sources
Batch ETL
Data Warehouse


ODSs and data warehouses: Option 2
Operational
DECISION
Strategic
Data sources
"Rapid ETL"
ODS
Batch ETL
Data Warehouse

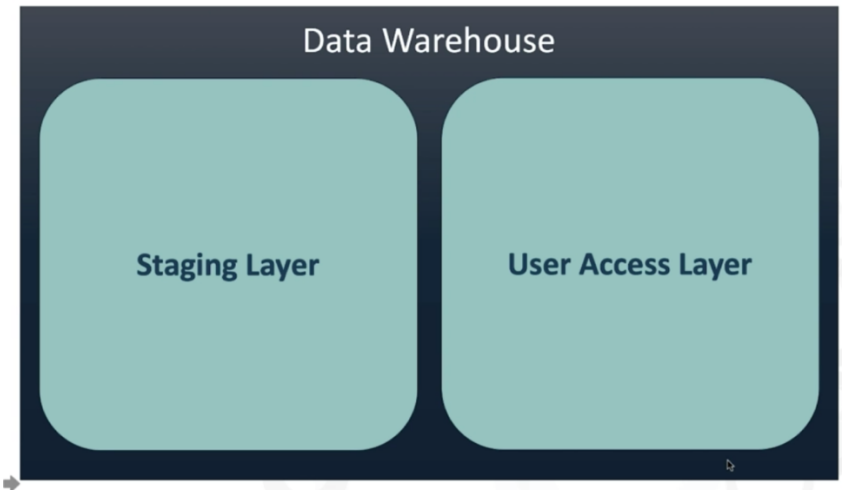
ODS Today


ODSs today
• Less popular
• "Faster" and more current DWs
• Superseded by big data (V=Velocity)
• ODS components within data lake
• "Traditional" ODSs still sometimes used for mission-critical situations



Role of Staging Layer in Data Warehouse

When look into the data warehouse, there are two layers



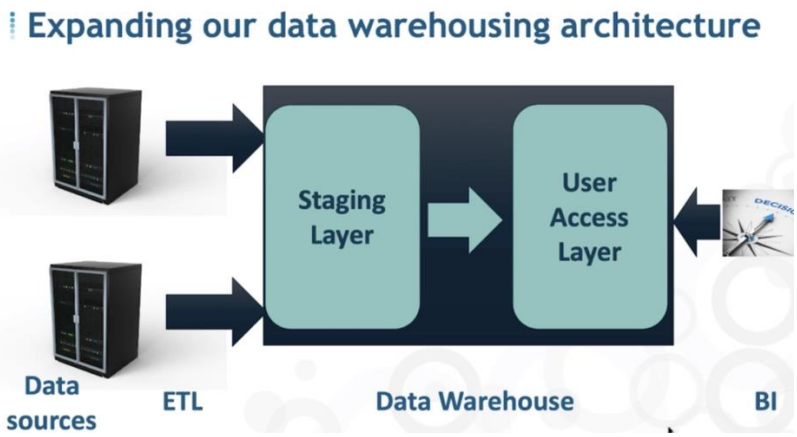
Staging Layer

- Landing Zone
- Extraction part in ELT
- 2 variations

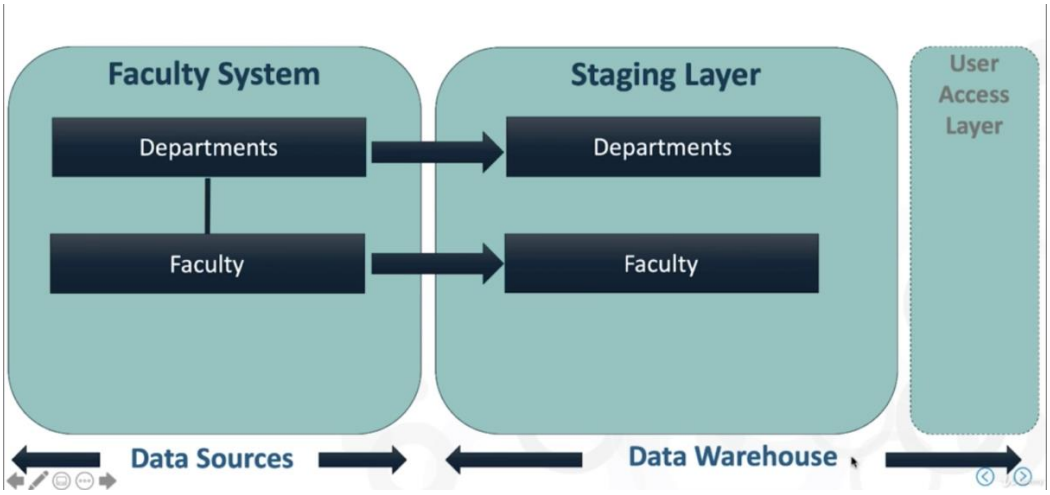
User Access Layer

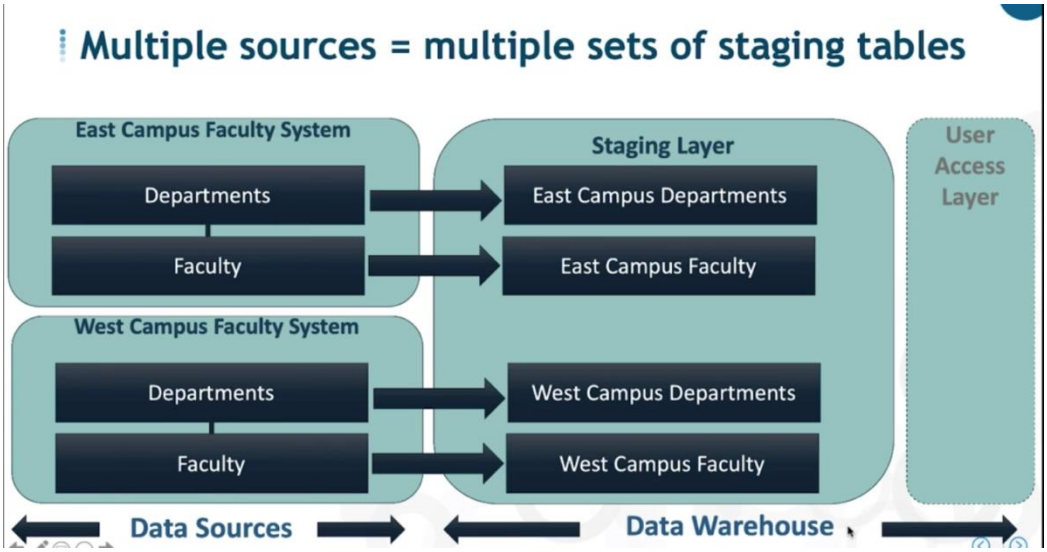
- This where user go for data
- Dimensional data

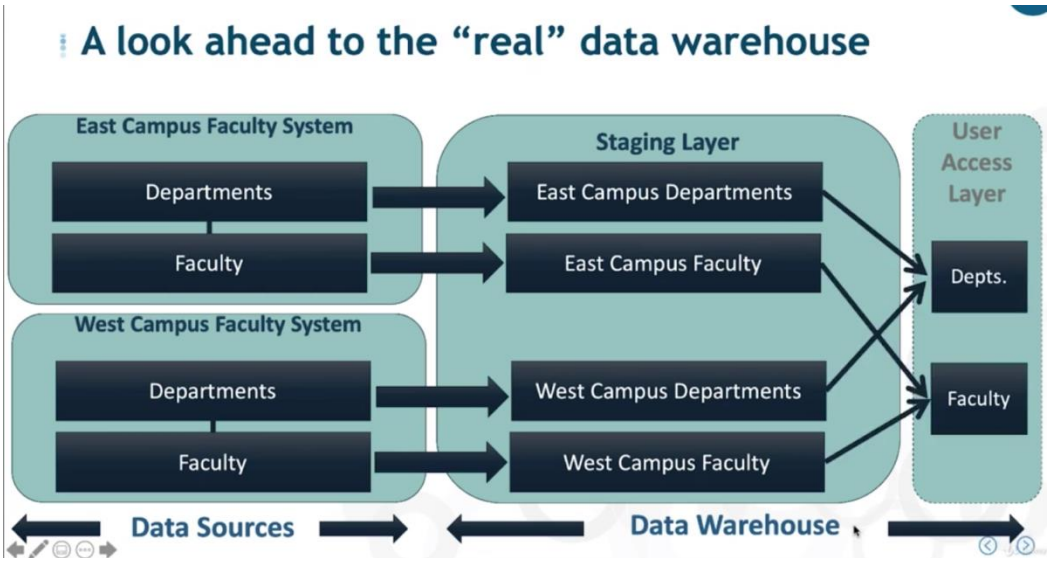
Looking into Data Warehousing Architecture

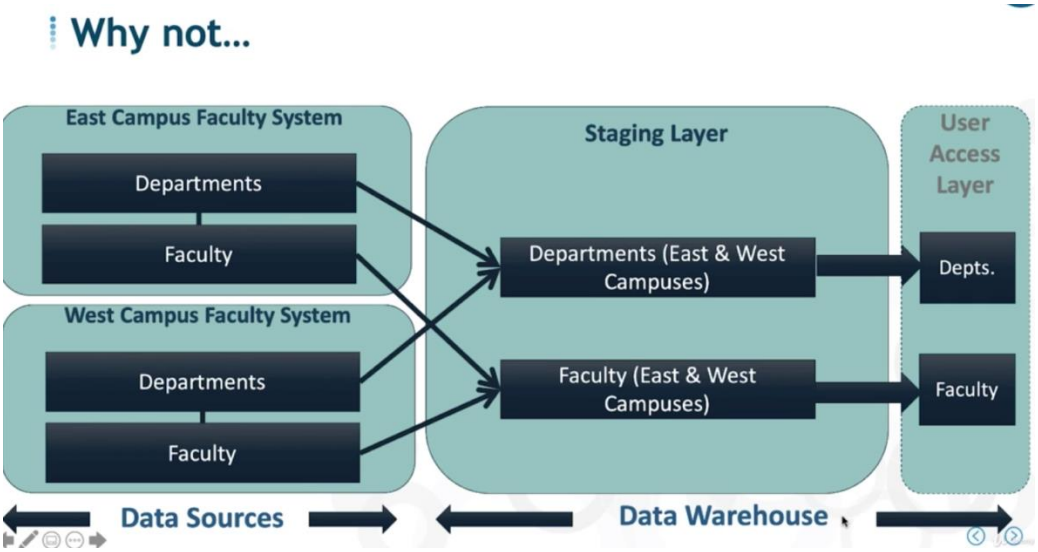


Looking into Staging Layer

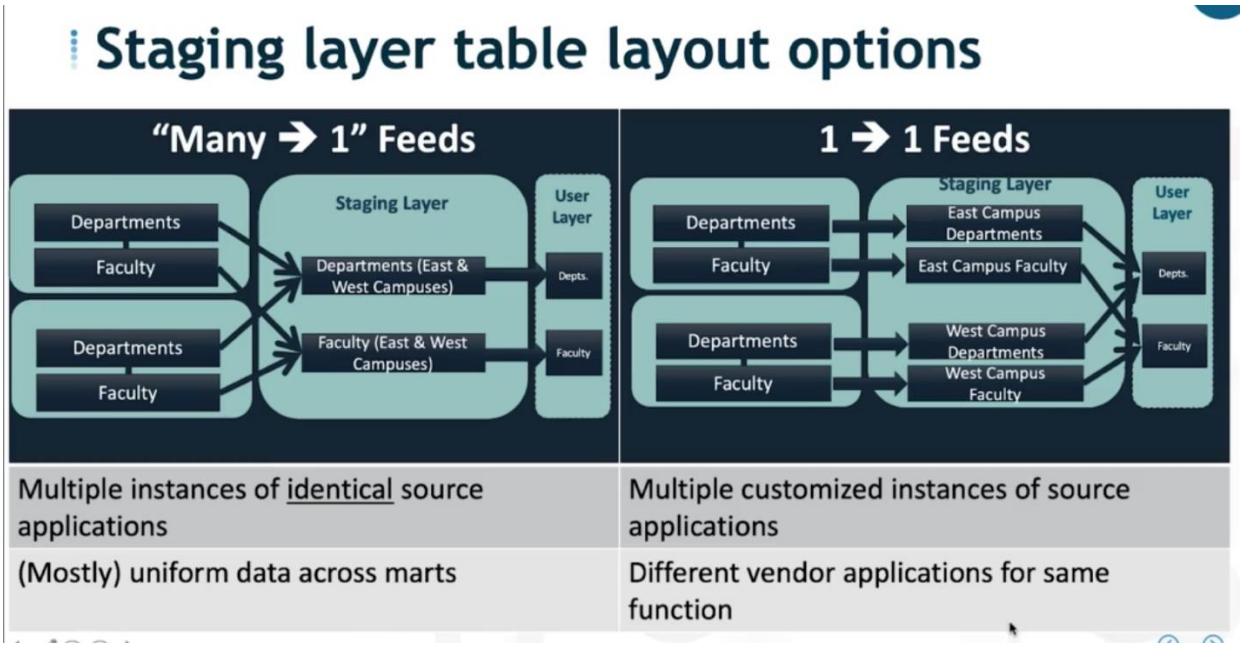




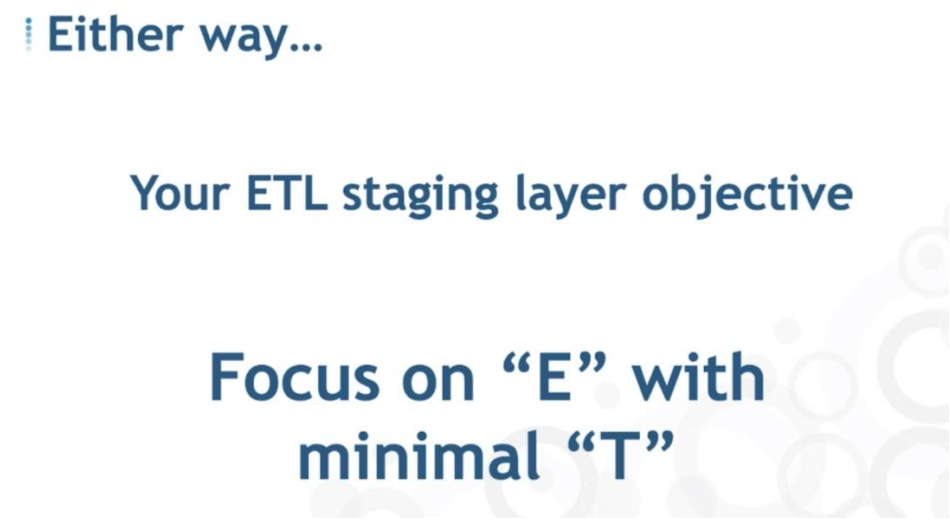




We can possibly could do that when data are aligned properly.

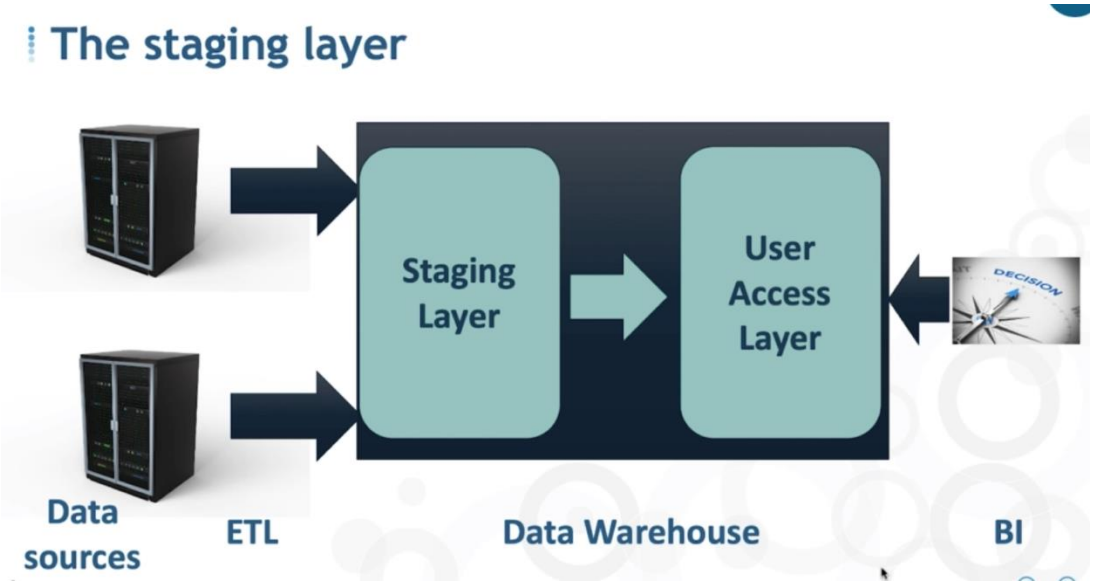


Staging layer objective



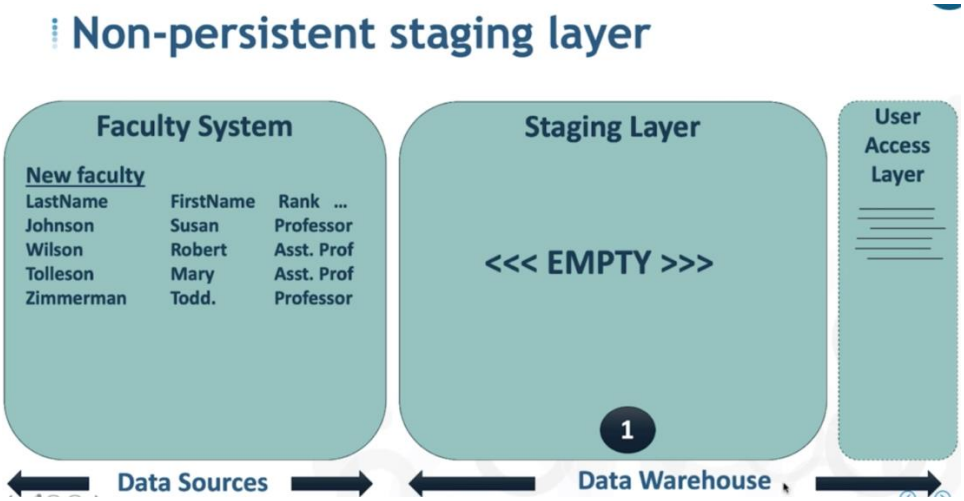
Comparing Two Types of Staging Layer

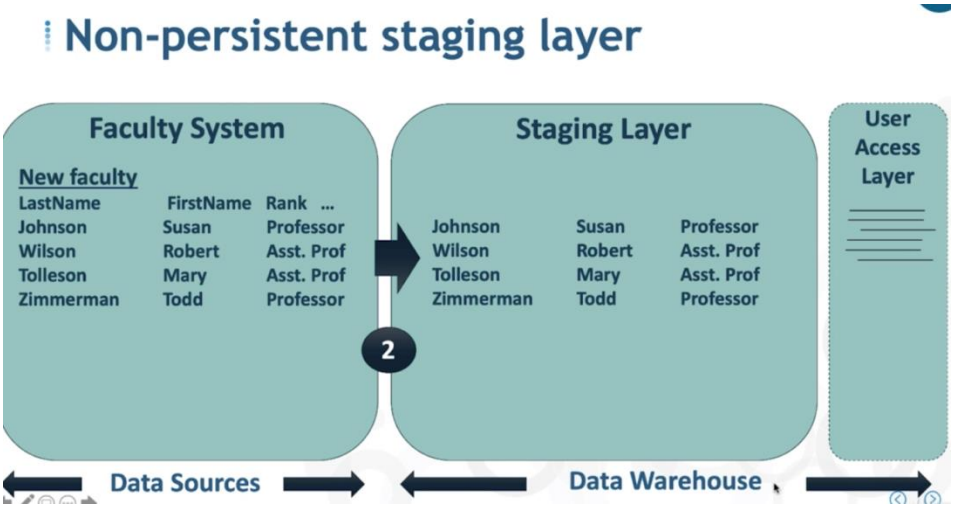
- Non-persistent staging layer
- Persistent staging layer

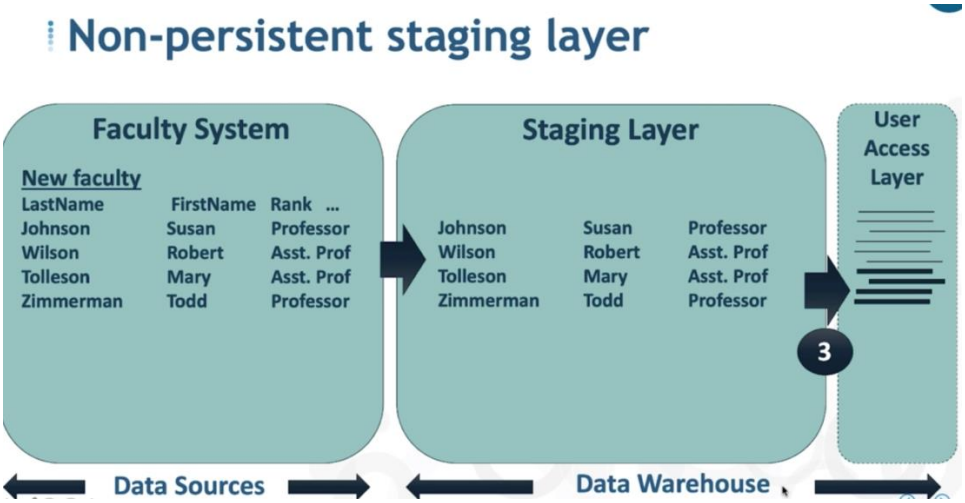


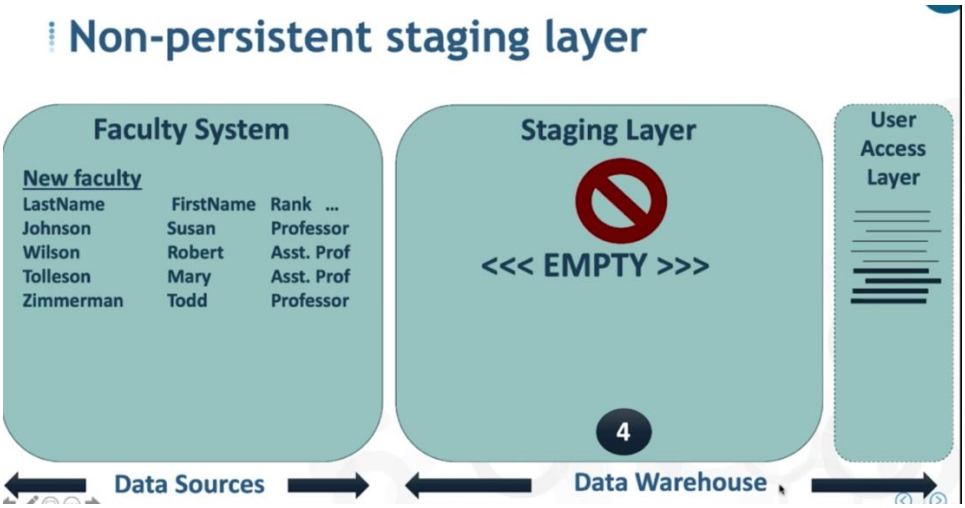
Non Persistent Staging Layer

- We will empty the staging layer once the data is moved to user access layer.

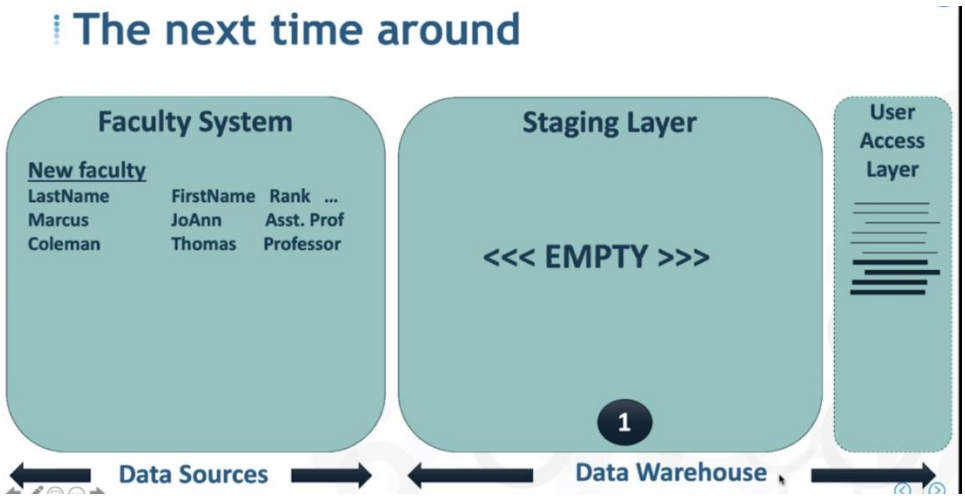


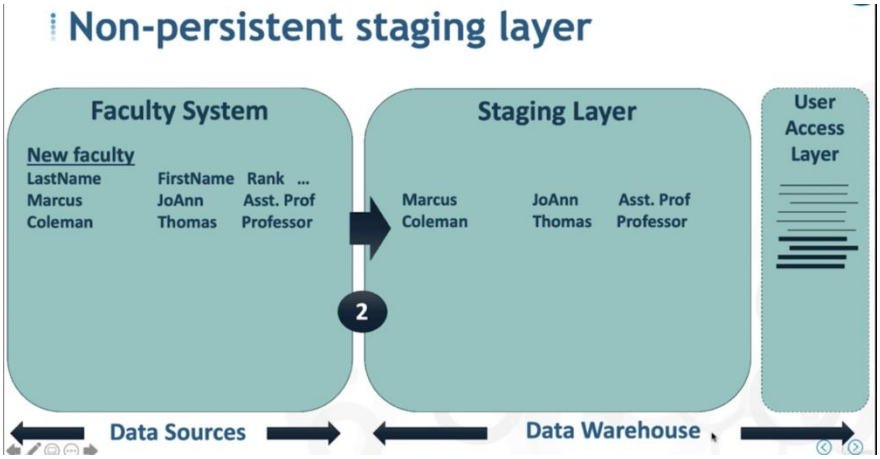


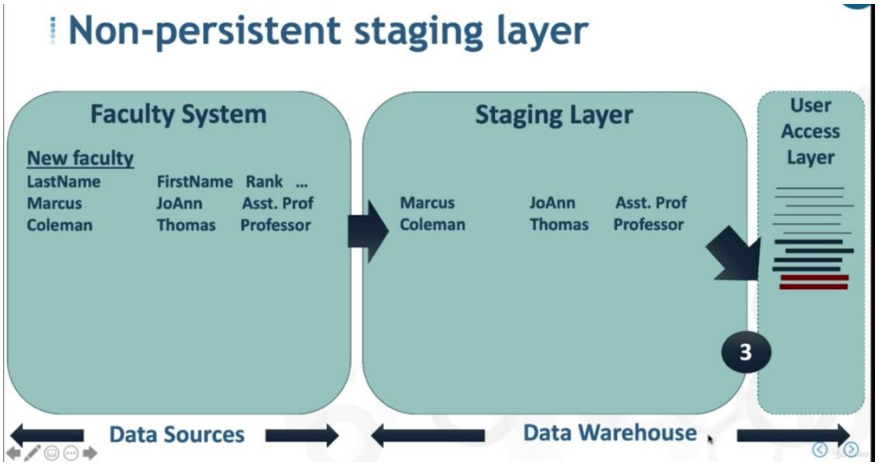


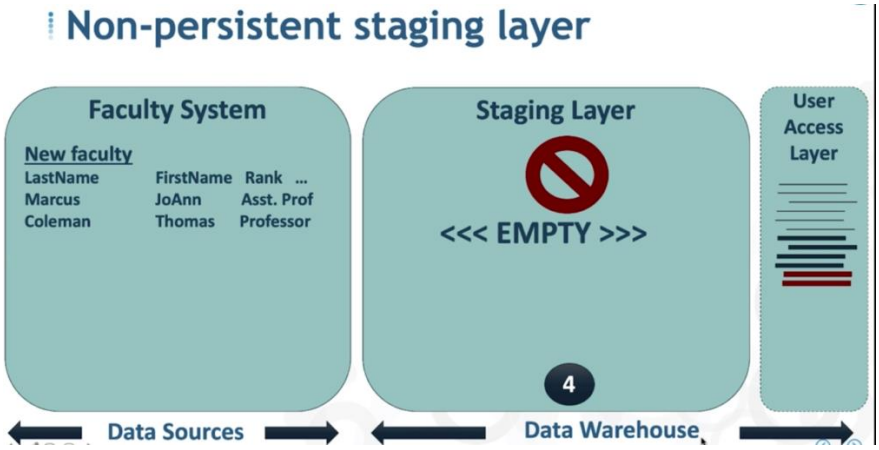


From the next time on wards, we will bring the new data to the staging layer.



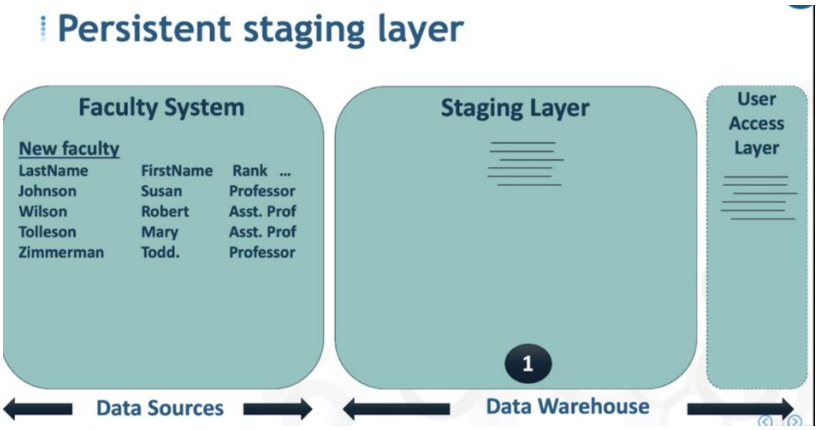

Non-persistent staging layer
Faculty System
New faculty
LastName FirstName Rank ...
Marcus JoAnn Asst. Prof
Coleman Thomas Professor
Staging Layer
Marcus JoAnn Asst. Prof
Coleman Thomas Professor
User Access Layer
2
Data Sources
Data Warehouse

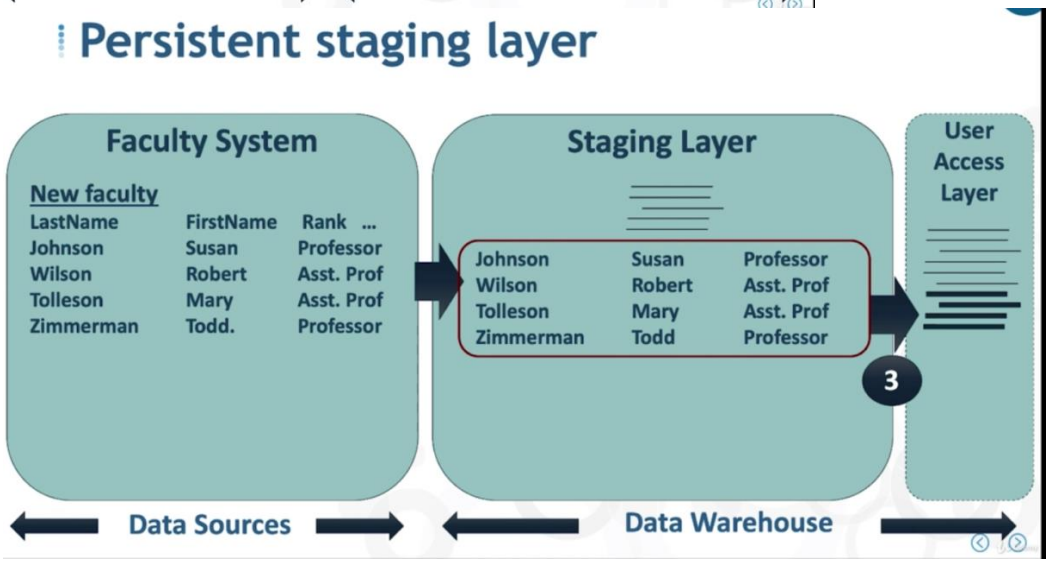

Non-persistent staging layer
Faculty System
New faculty
LastName FirstName Rank ...
Marcus JoAnn Asst. Prof
Coleman Thomas Professor
Staging Layer
Marcus JoAnn Asst. Prof
Coleman Thomas Professor
User Access Layer
3
Data Sources
Data Warehouse

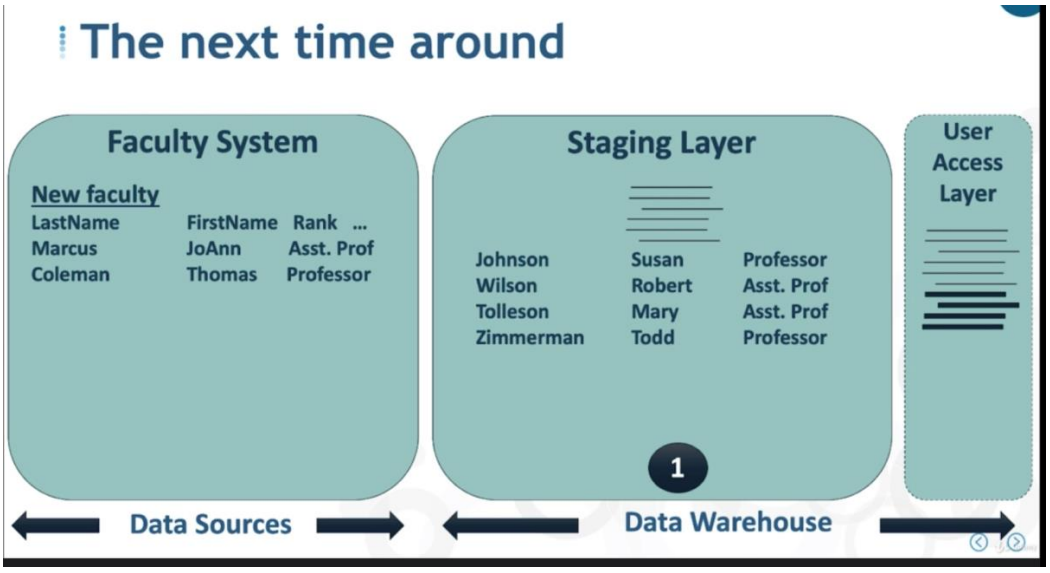

Non-persistent staging layer
Faculty System
New faculty
LastName FirstName Rank ...
Marcus JoAnn Asst. Prof
Coleman Thomas Professor
Staging Layer
<<< EMPTY >>>
User Access Layer
4
Data Sources
Data Warehouse

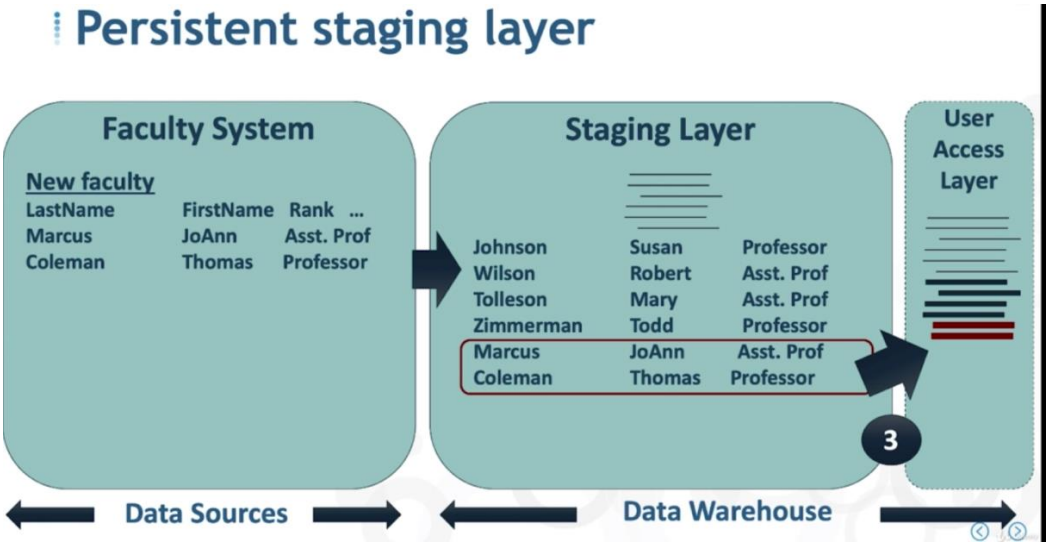
Persistent Staging Layer

We don’t remove the data from staging layer even after the data moved to the user access layer.









Non-persistent staging layer Advantage and Disadvantage

Non-persistent staging layer

Advantages	Disadvantages
Less storage space	Need to go back to source systems to rebuild user layer
Data already moved to user layer	Data QA also requires source systems

Persistent staging layer Advantage and Disadvantage

Persistent staging layer

Advantages	Disadvantages
Rebuild user layer without source systems	More storage space
Data QA: compare staging layer with user layer	Risk of “ungoverned” access

Assignments

You will be presented with a scenario (described below) as part of a university system's efforts to build a data warehousing environment.

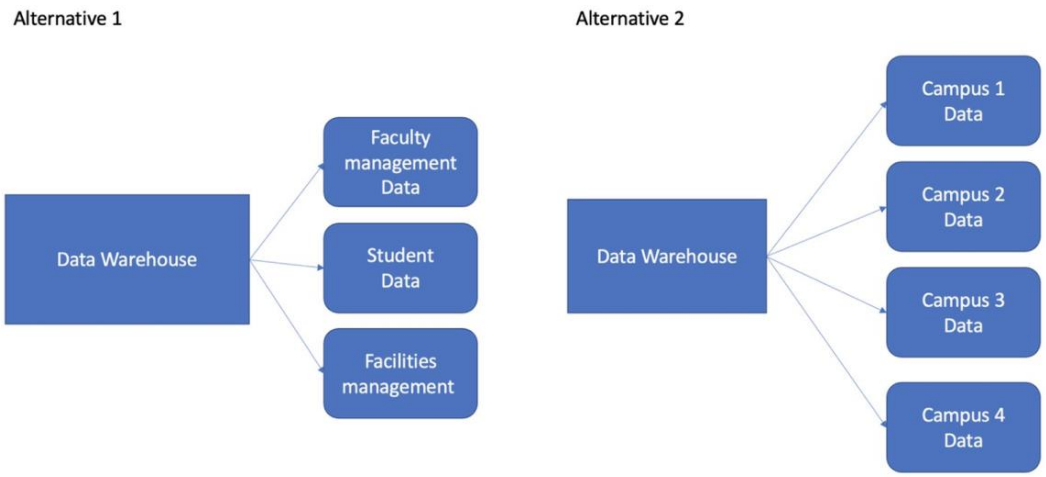
Based on the details of the scenario, you will draw diagrams to specify two different architectural alternatives that could be used.

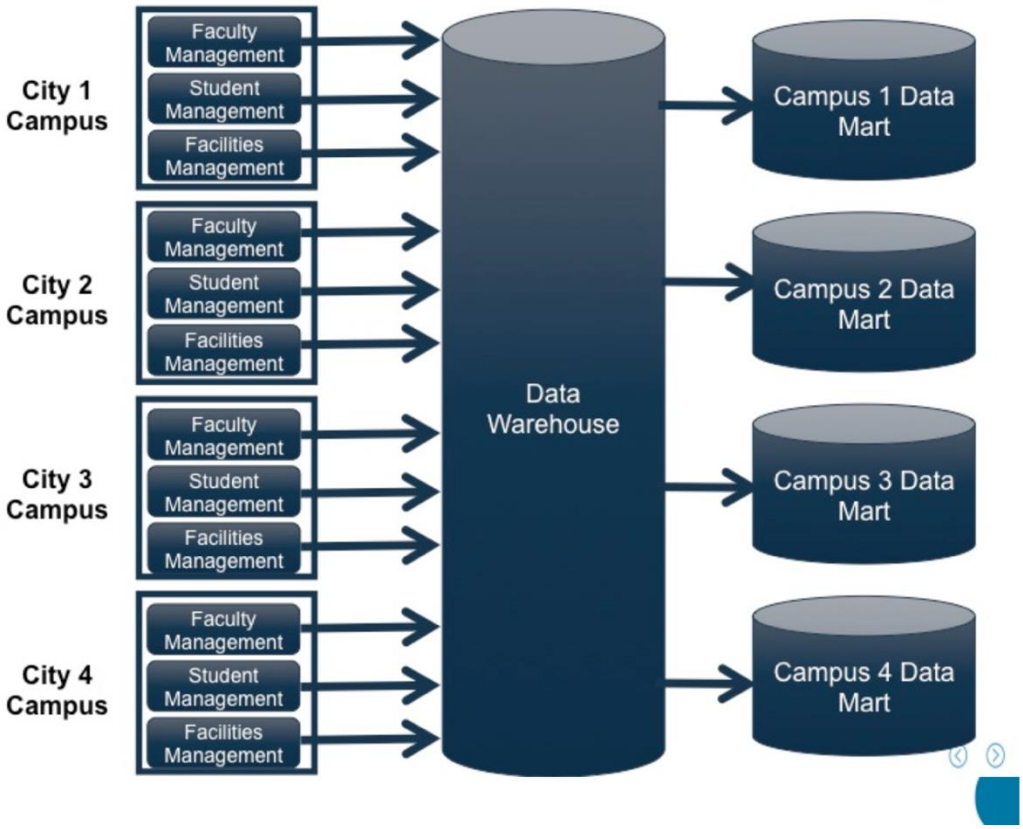
For purposes of this assessment, each of your alternatives will be a component-based approach using data marts rather than a centralized, single-database data warehouse.

The scenario:

A large state university system has four different campuses located all across the state. Each of these campuses operates as, essentially, its own self-contained university, which means:

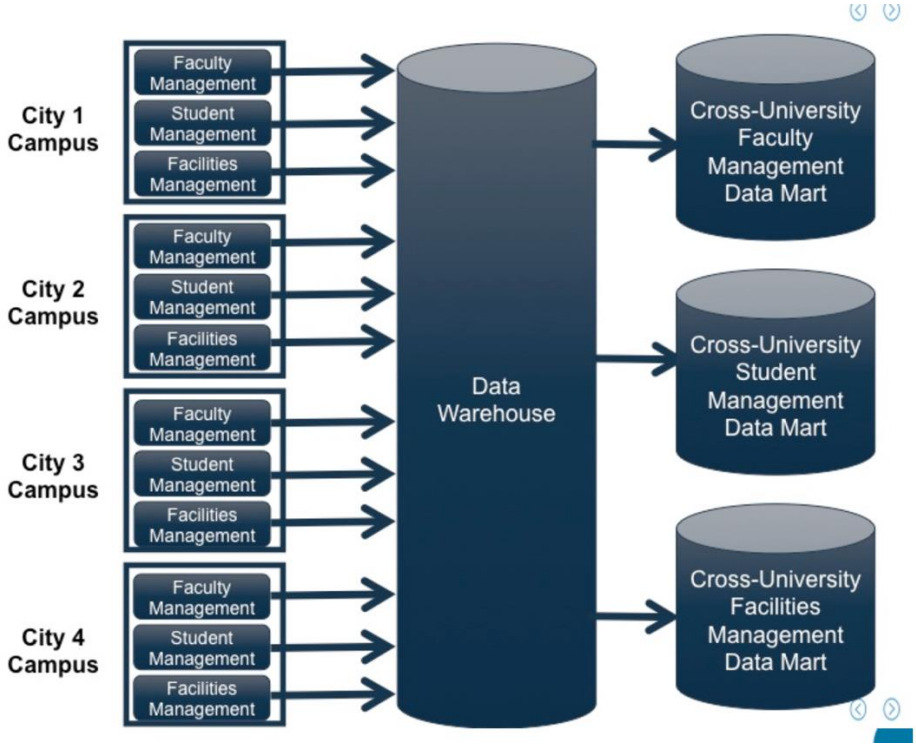
- Each campus has its own IT organization that operates and maintains its own applications for that campus only
- Each campus' applications include the following key software systems for:
  - Faculty management
  - Student admissions, grading, and administration
  - Facilities management (e.g., buildings and classrooms)
- The state university's "main campus" is one of the four individual campuses, and is located in a small town in the center of the state
- The main campus IT organization has responsibilities for not only that campus, but also support for the entire state university system as a whole
- The main campus IT organization is just starting on a data warehousing initiative that is intended to serve the needs of 1) individual campuses as well as 2) the state university system as a whole.





With campus-based data marts:

- The administrators at each campus have easy access to "everything going on at THEIR campus"
- Each campus-specific data mart can be customized and extended to any individual needs at that campus with minimal impact on any other campuses
- Any broader analysis that would require data from beyond that campus could be performed in the data warehouse rather than the campus-specific data mart



With function-based data marts:

- The overall university system administrators responsible for functional areas – e.g., faculty management, student management, and facilities management (and potentially others) – have ready access to their relevant data with minimal impact on other functional areas
- Each function-specific data mart can be customized and extended to any individual needs of that particular functional area
- Any broader analysis that would require data from beyond that particular could be performed in the data warehouse rather than the function-specific data mart

A final thought

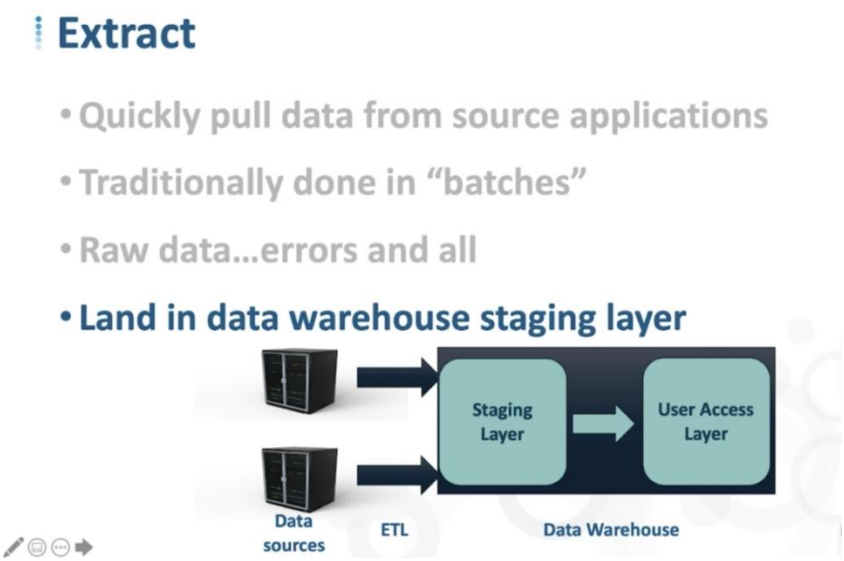
- Especially today, your objective should be to build a centralized data warehousing environment rather than a component-based one to avoid unnecessary data fragmentation
- Still, the realities of most organizations lead to the building and deployment of at least some dependent data marts (i.e., those that are provided data from a data warehouse rather than directly from source systems)

ETL

Compare ETL to ELT

Bringing data from multiple sources to data warehouse environment.

Extract



Transform

Transform

- "Apples to apples"
- Prepare for uniform data in user access layer
- Can be very complex

Load

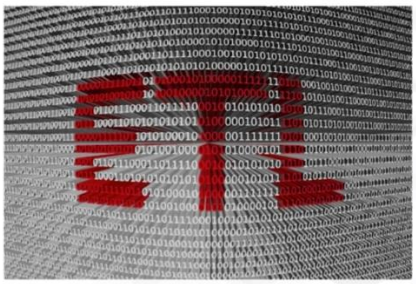
Load

- Final stop along the data pathway
- Store uniform data in user access layer

Challenges with traditional ETL

Challenges with traditional ETL

- Significant business analysis before storing data
- Significant data modeling before storing data

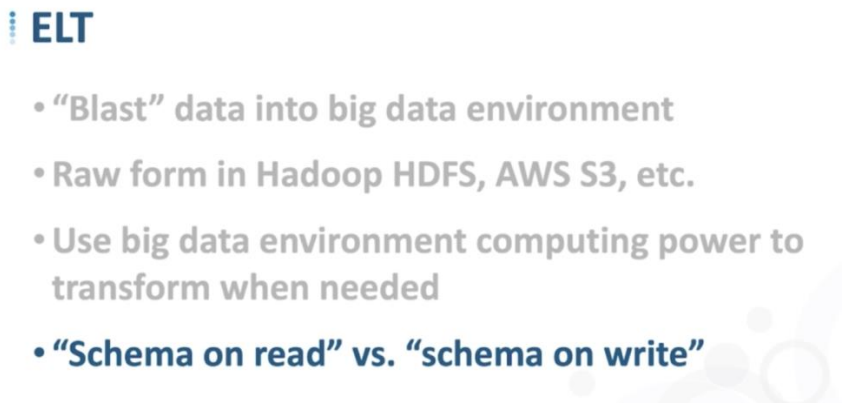


ELT

Change the order

vs.

Do the load first and then perform transform operation.



Schema does not have to be exist while ingesting the data. Schema on write is converse to the ELT.

- For tradition Datawarehouse, we stick with ETL.
- If we are doing the data lake, there we can have a ELT.

Design the Initial Load ETL

Two variations of ETL

- Initial
- Incremental

Initial ETL

Bring only relevant data to data warehousing environment. Whereas in big data environment, bring all possible data.

What to bring into data warehouse

- Data definitely needed for BI and analytics

What to bring into data warehouse

- Data definitely needed for BI and analytics
- Data probably needed for BI and analytics
- **Historical data**

Initially we won't be getting an historical data. Once the data warehousing up and running, later in the period we will be getting the historical data.

After the initial load, if we are required to keep the data warehouse up to date, we need to periodic update. That's we incremental ETL comes in.

Compare the different model for incremental ETL

Incremental ETL

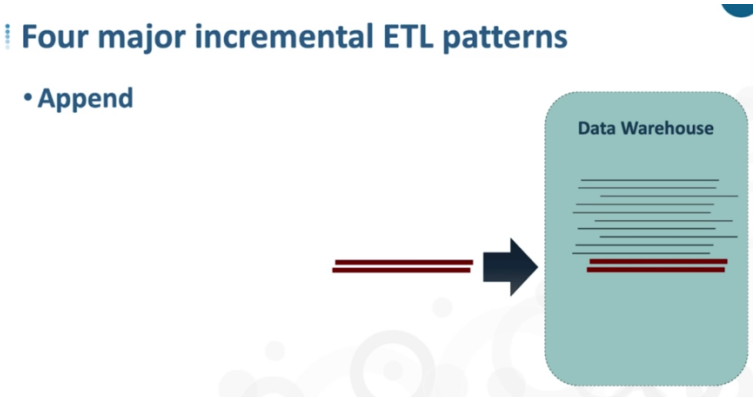
Incremental ETL

- Incrementally "refreshes" the data warehouse
- New data: employees, customers, products, …
- Modified data: employee promotions, product price change, …
- **Special handling for deleted data: customer drops from a subscription plan, …**

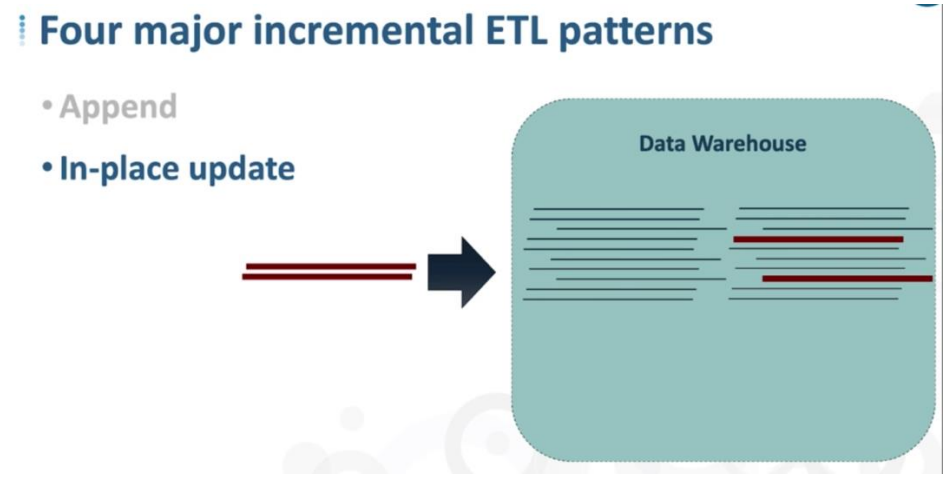
Bring the data warehouse up to date.

Four major incremental ETL Patterns

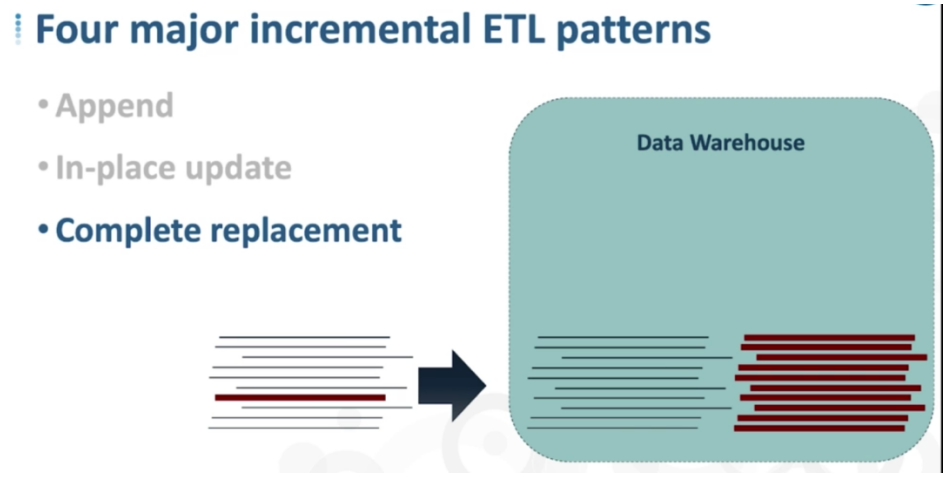
Append



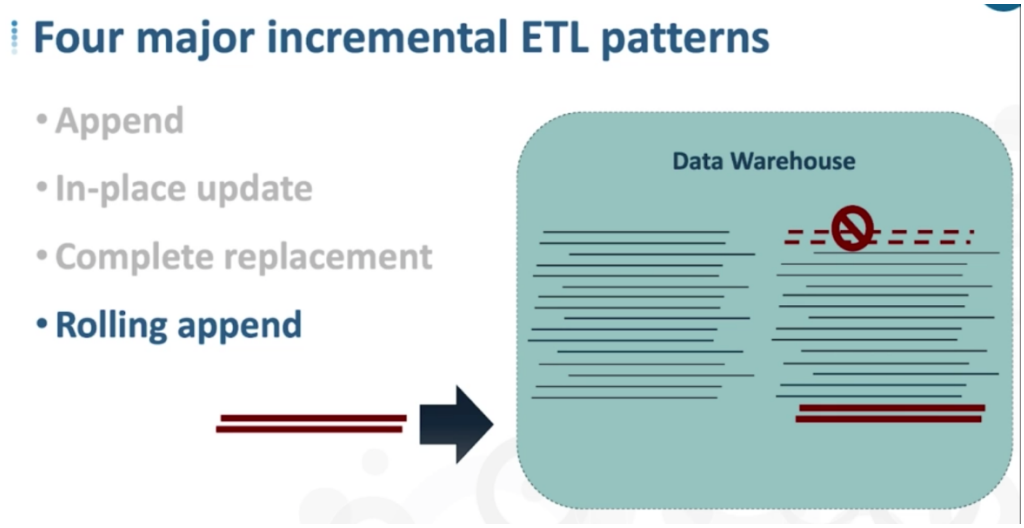
In-place update



Complete replacement

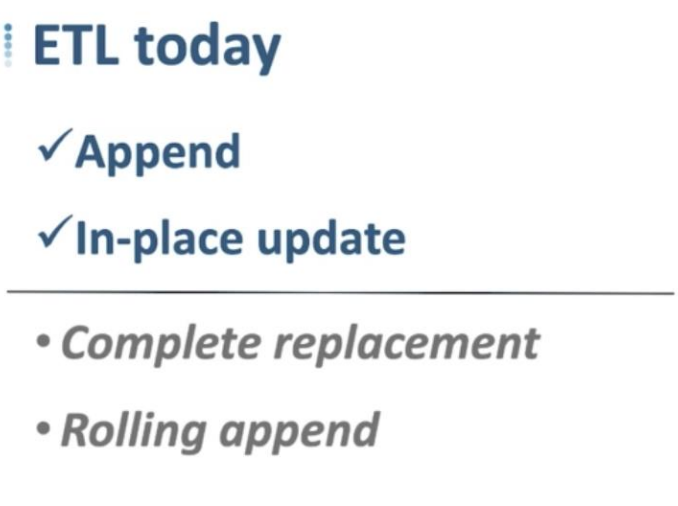


Rolling append



Old data will be deleted on new data append. For example, if we have constraint like keeping only the 6 months of data.

ETL Today



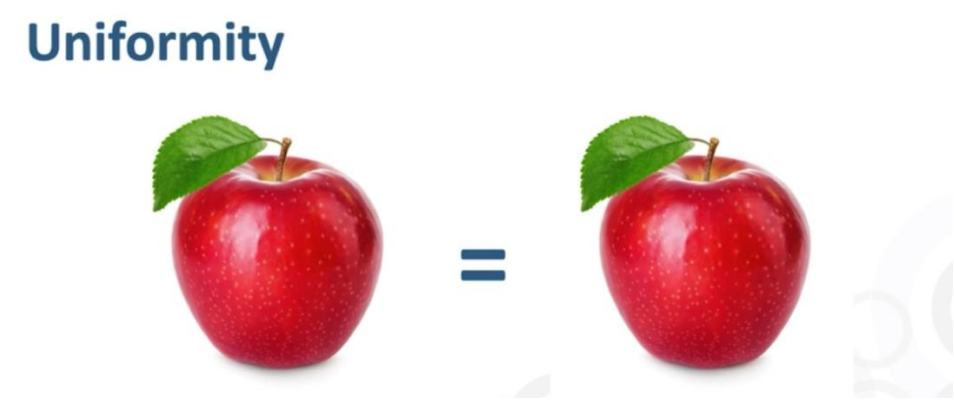
ETL today mostly uses the Append and In-place update.

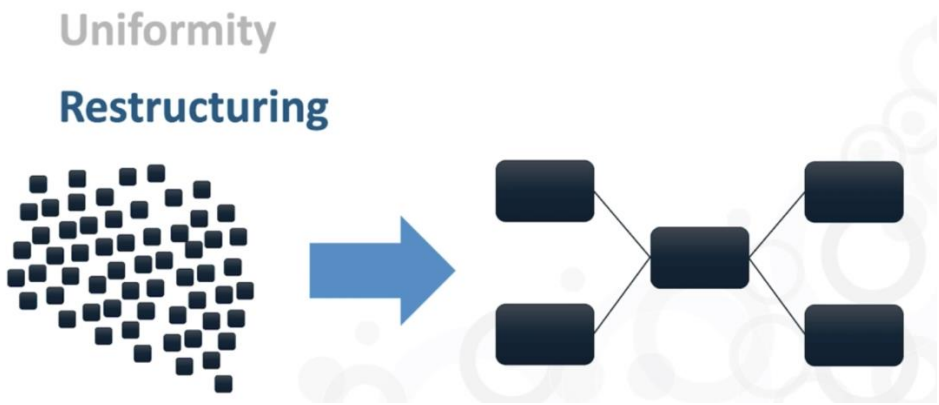
Explore the role of Data Transformation

Data Transformation - Key part of making the data warehouse functional.

Data transformation overarching goals

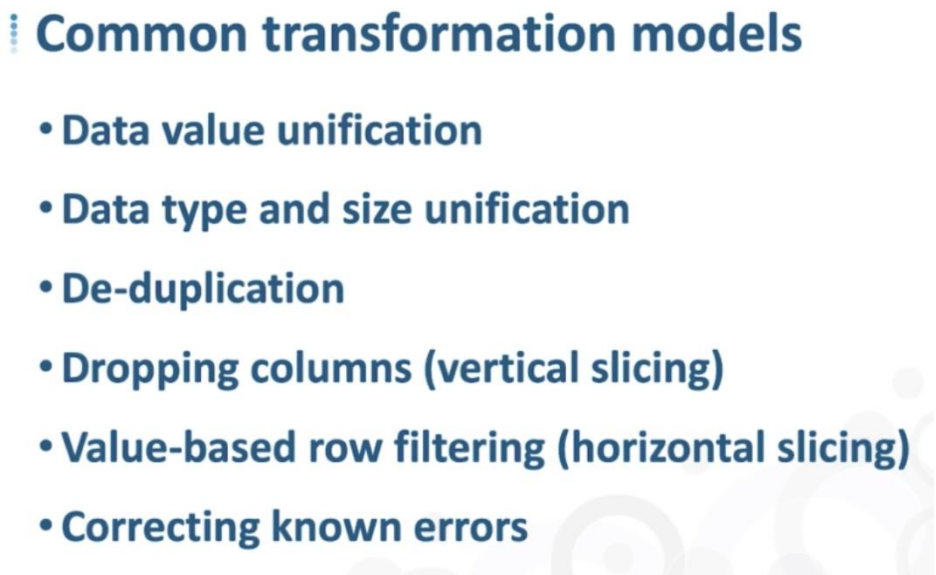
We may get a data from different source system, at the end after the transformation, everything should look similar.





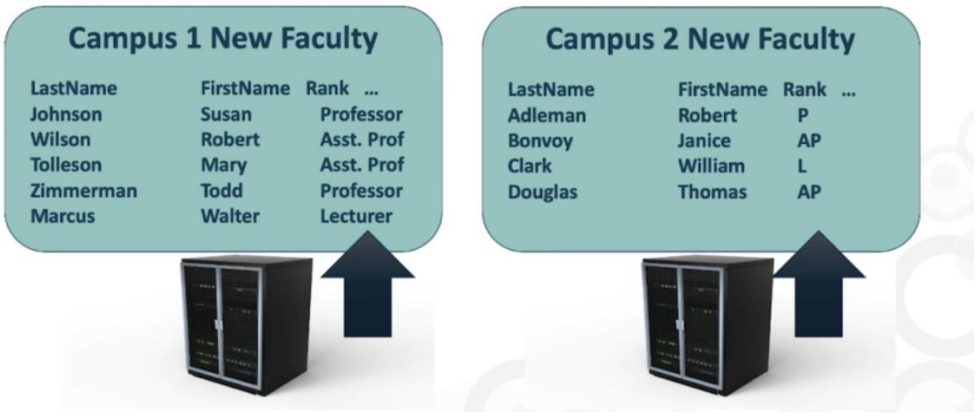
Restructuring the different form of data.

Common Transformation Models

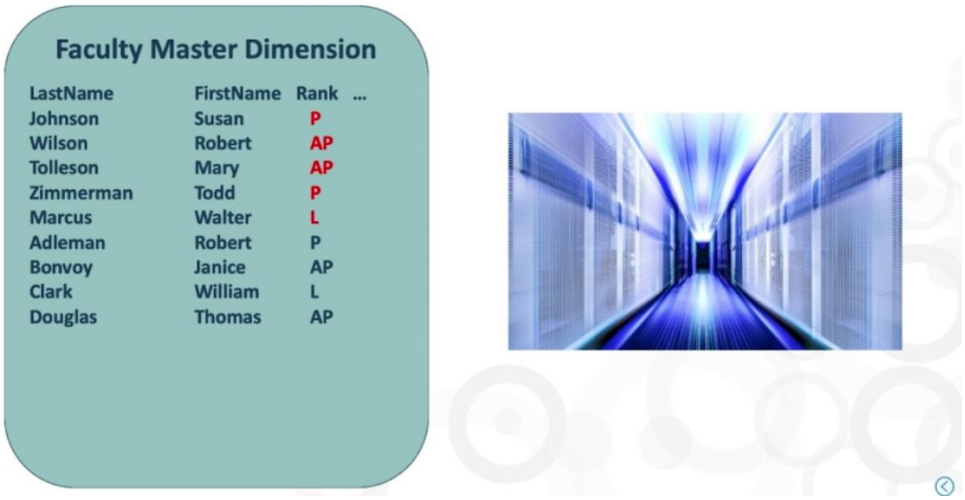


Data value unification

Data value unification

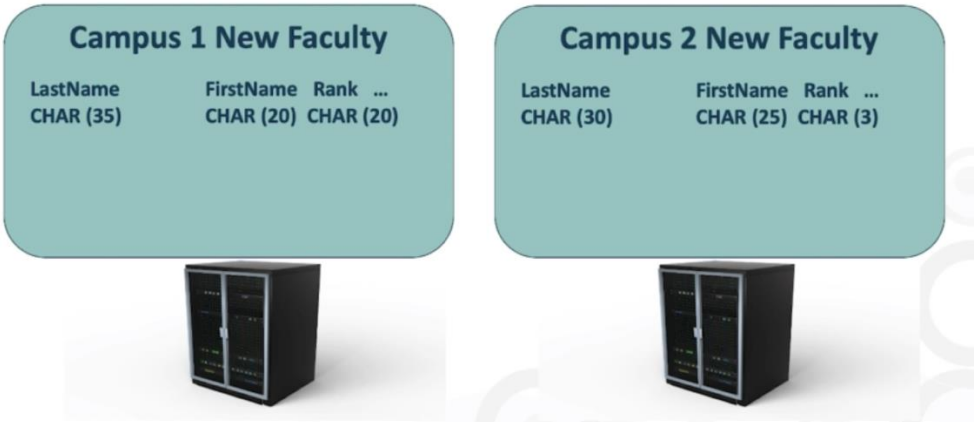


Data value unification

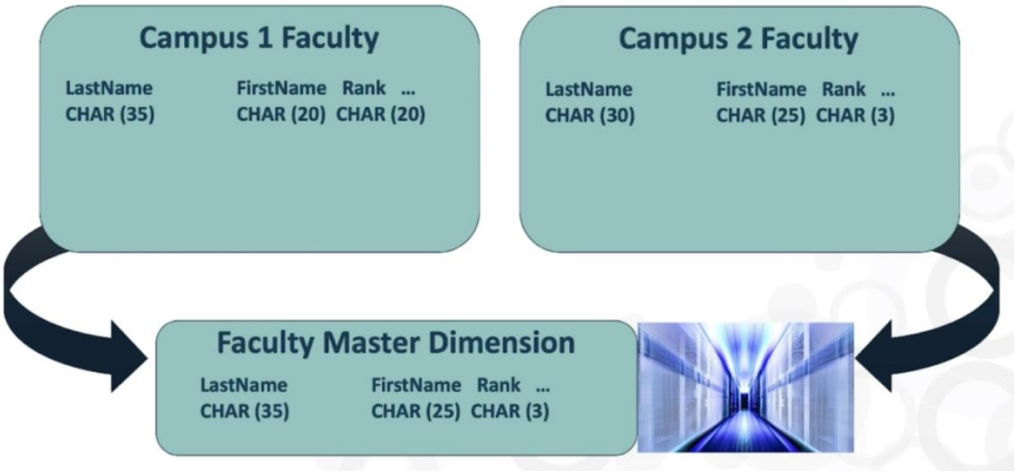


Data type and size unification

Data type and size unification

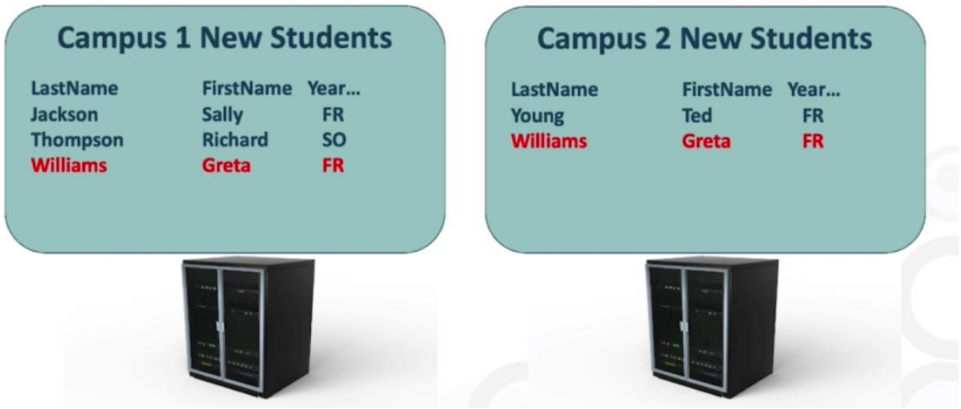


Data type and size unification

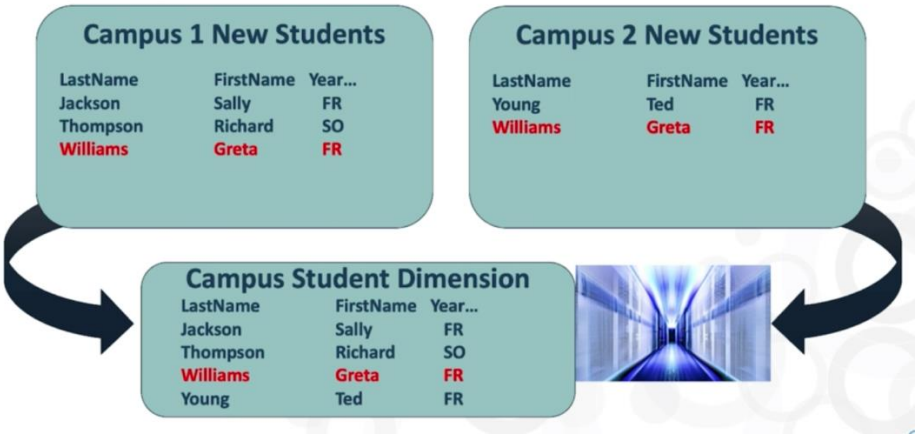


De-duplication

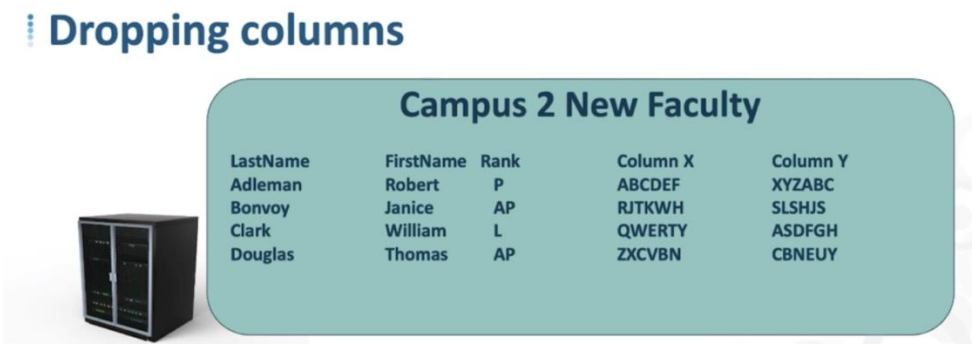
De-duplication



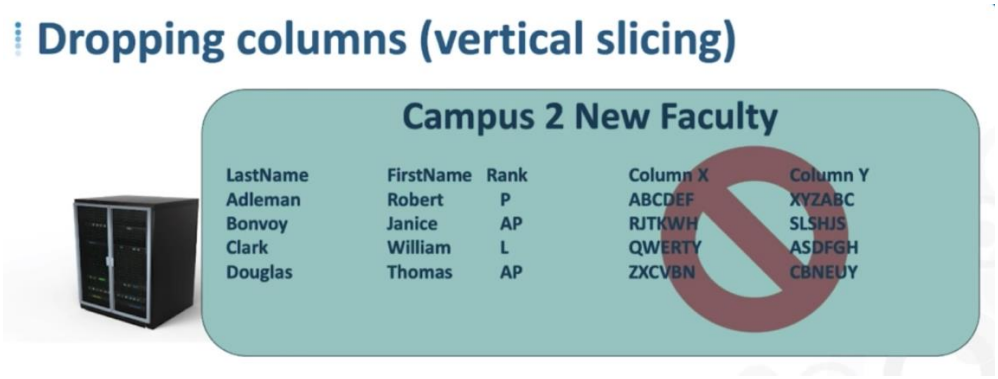
De-duplication

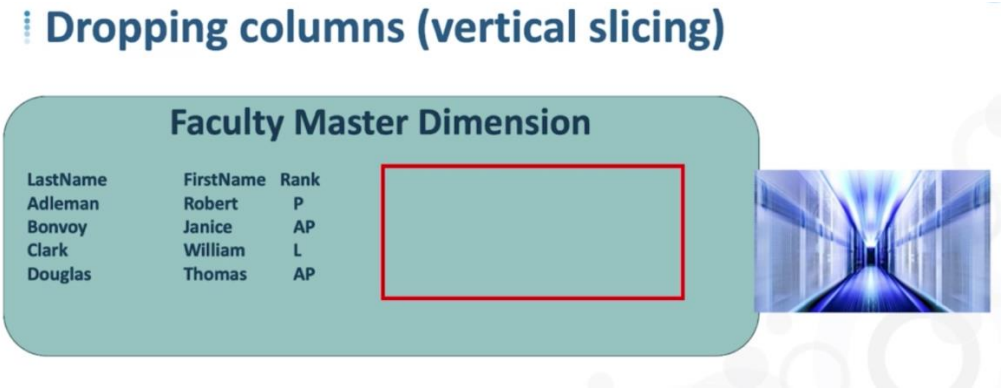


Dropping columns (vertical slicing)

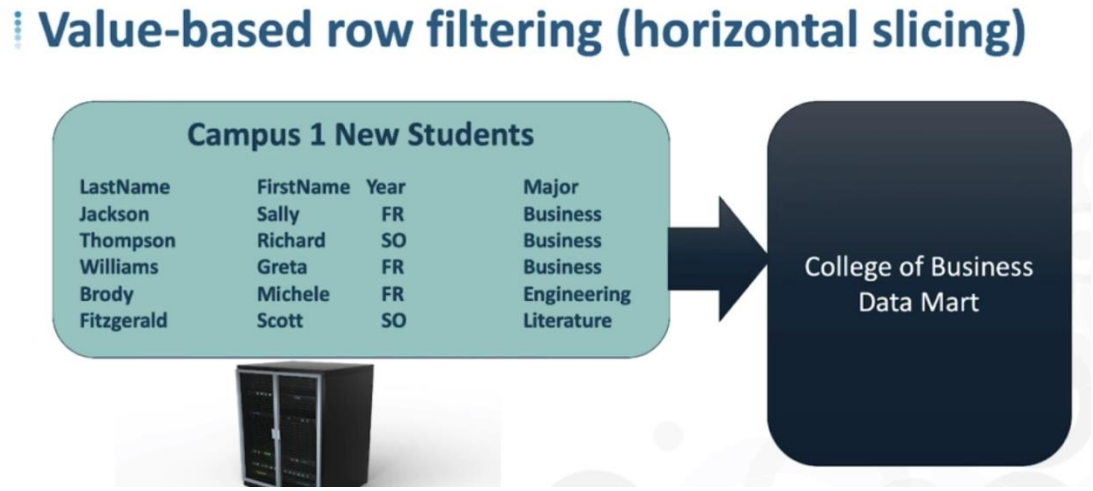


We decided that we don't need column x and column y. We are removing the same from the table.



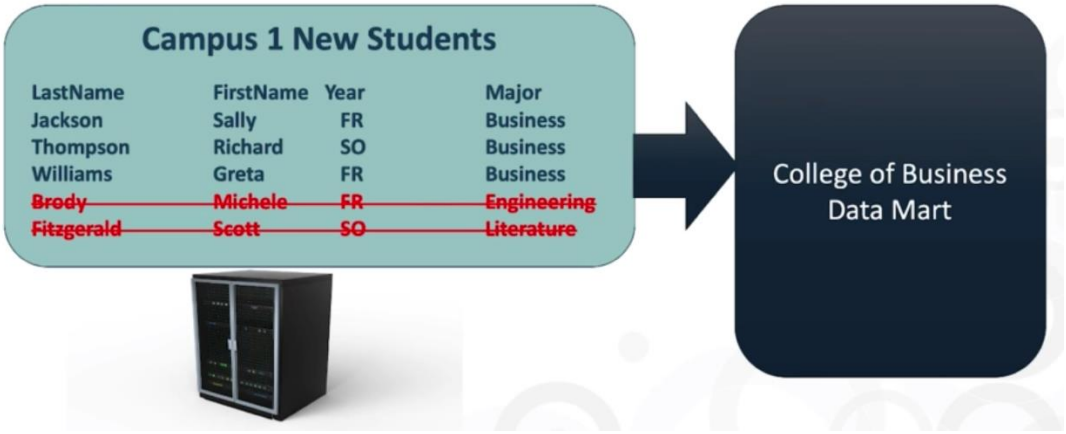


Value based row filtering (horizontal slicing)



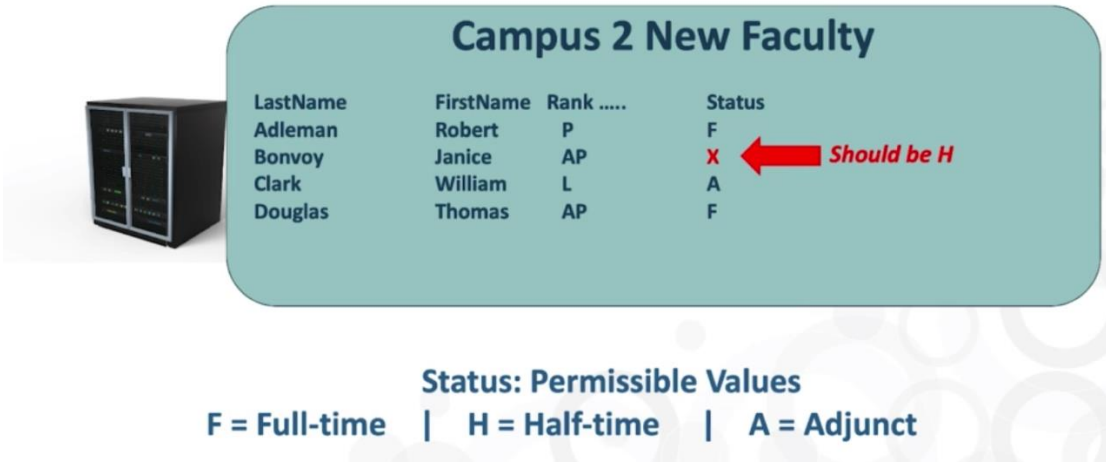
We are building the business data mart. So, we are filtering the data based on 'business' major.

Value-based row filtering (horizontal slicing)

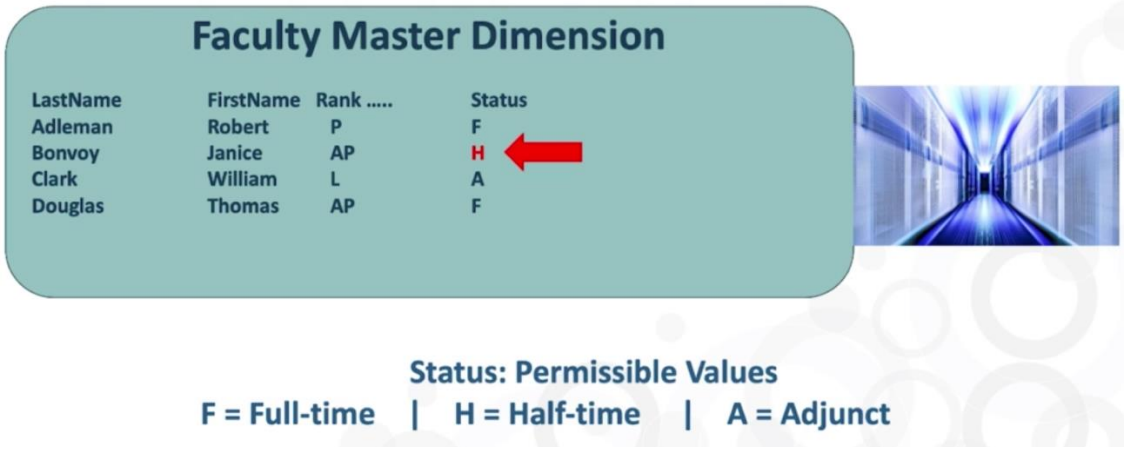


Correcting known errors

Correcting known errors



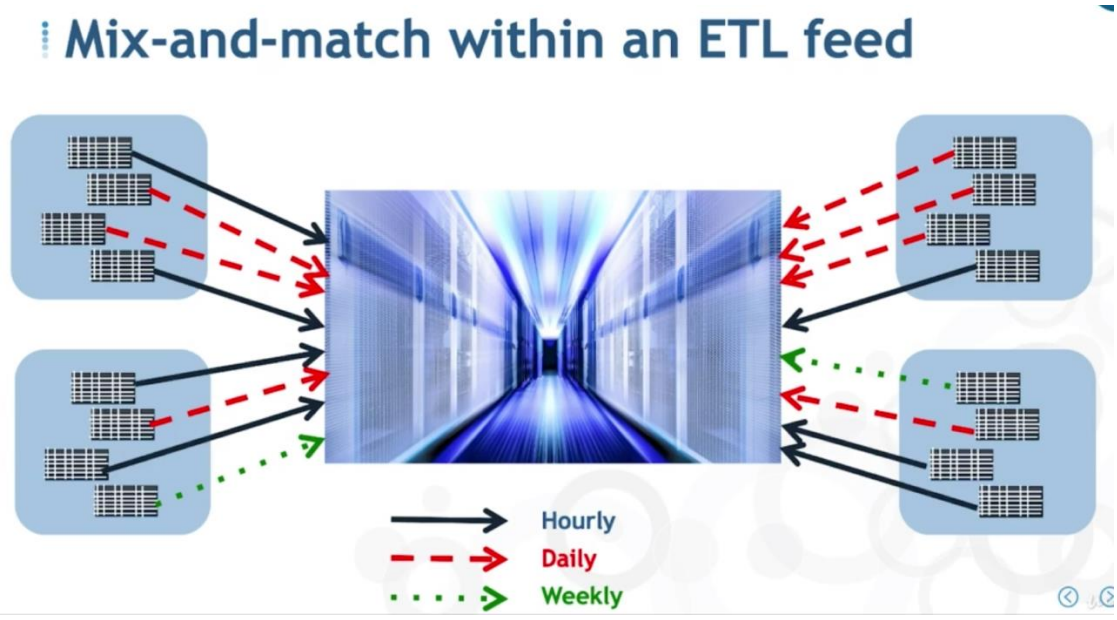
Correcting known errors



Implement Mix and Match Incremental ETL

Frequency of data update differs from source system.

Follows different pattern on different source system.



Frequency of data update differs among table in single source system.

Assignments

1. The university where you work has two different applications that contain student data: 1) a student tuition billing system and 2) a student academics and enrollment system.

   You are responsible for the integration work necessary to build a single "master list of students" in the data warehouse.

   A key part of your integration work will be to specify the data transformations necessary as part of the ETL.

   Relevant data fields in the **student tuition billing system**:

   Last Name: STRING (40)

   First Name: STRING (25)

   Academic Status: STRING (3) with the following permissible values:

   "FGS" – Full-time student in good standing

   "PGS" – Part-time student in good standing

   "FAP" – Full-time student on academic probation

   "PAP" – Part-time student on academic probation

   Relevant data fields in the **student academics and enrollment system**:

   Last Name: STRING (35)

   First Name: STRING (15)

   Enrollment Status: STRING (10) with the following permissible values:

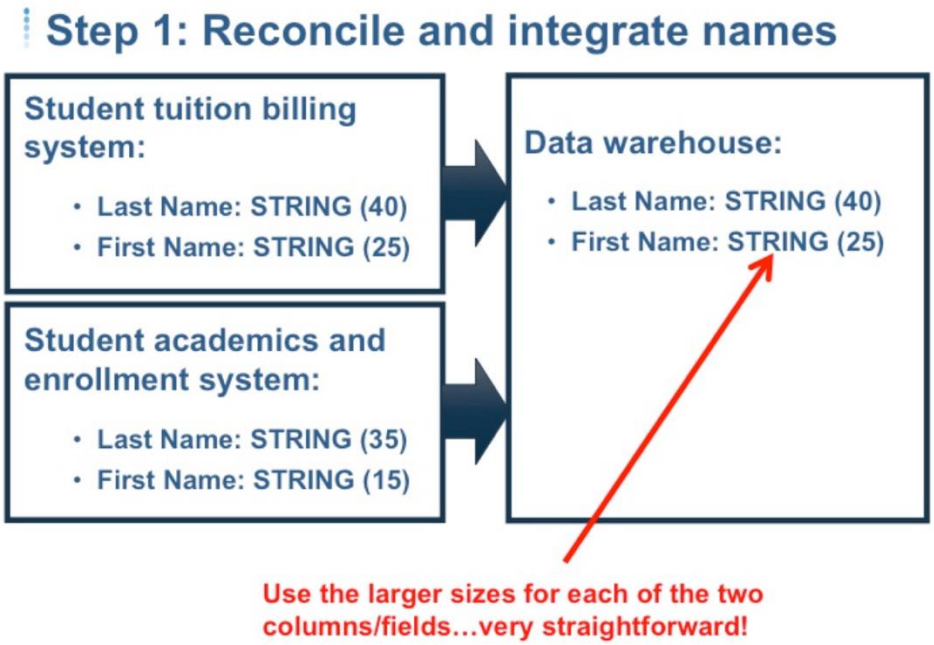
   "Full-time"

   "Part-time"

   Academic Status: STRING (13) with the following permissible values:

   "Good standing"

   "Probation"



Step 2: Reconcile and integrate enrollment and academic status

Student tuition billing system:

- Academic Status: STRING (3) with permissible values:
  - "FGS" – FT/good standing
  - "PGS" – PT/good standing
  - "FAP" – FT/probation
  - "PAP" – PT/probation

Student academics/enrollment system:

- Enrollment Status: STRING (10) with permissible values:
  - "Full-time"
  - "Part-time"
- Academic Status: STRING (13) with permissible values:
  - "Good standing"
  - "Probation"

- First consideration: tuition billing system combines two different concepts into the same column/field, while the academics/enrollment system has separate column/field for each concept
- Decision for data warehouse: combine or separate?
- Your best choice: SEPARATE so each can be analyzed by itself with less complication (e.g., data field substringing)

Step 2: Reconcile and integrate enrollment and academic status (more)

Student tuition billing system:

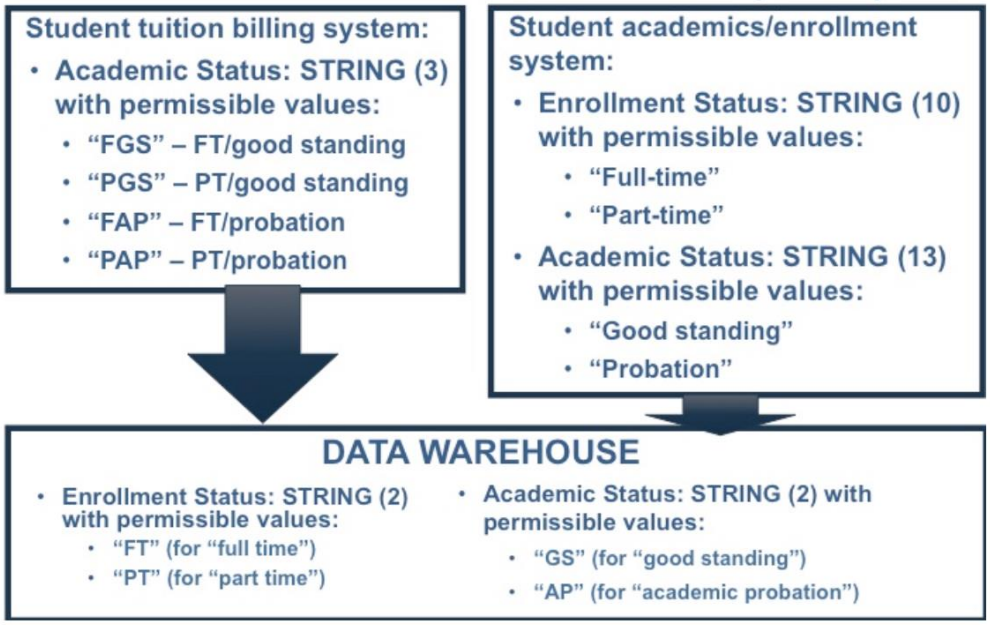
- Academic Status: STRING (3) with permissible values:
  - "FGS" – FT/good standing
  - "PGS" – PT/good standing
  - "FAP" – FT/probation
  - "PAP" – PT/probation

Student academics/enrollment system:

- Enrollment Status: STRING (10) with permissible values:
  - "Full-time"
  - "Part-time"
- Academic Status: STRING (13) with permissible values:
  - "Good standing"
  - "Probation"

- Second consideration: how to represent each field?
  - Status codes?
  - Full words?
- No 100% always-correct answer, but in general you can save space with status codes…remember the BI tool can substitute full words for the status code (e.g., "FT" becomes "Full-time" if that's what is desired on a report)

Step 2: Reconcile and integrate enrollment and academic status (more)



Data Warehousing Building Block

BI category drives data model

BI Category	Data Model
Basic reporting	Dimensional
Online analytical processing (OLAP)	Dimensional
Predictive analytics	Data mining/specialized
Exploratory analytics	Data mining/specialized

Our focus

BI Category	Data Model
Basic reporting	**Dimensional**
Online analytical processing (OLAP)	**Dimensional**
Predictive analytics	Data mining/specialized
Exploratory analytics	Data mining/specialized

The Basic Principle of Dimensionality

In Dimensional Data model, data should be organized dimensionally to make the data driven decision.

Making a data driven decision in a dimensional sense require,

- One or more measurements
- Dimensional context for each measurement

Example,

6644

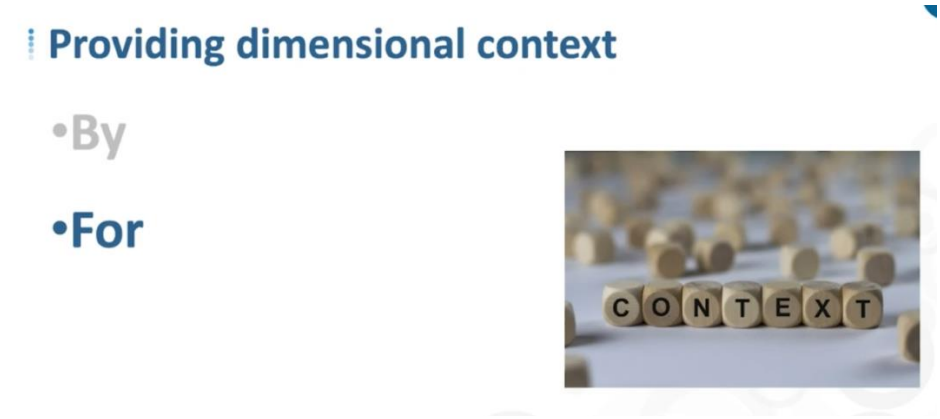
Here, 6644 is some measurement, we don't know what measurement is.

Adding a dollar sign in front of the measurement add some details. But we don't know what this is for. Is it a salary or price of something. So we don't know the context here.

Now, we know the $6644 is payment. But still it requires more context. We don't know what this payment is for. Still there is not enough context.

So, absence of context is the problem. So we need to make sure for every measurement there is enough context.

Providing a dimensional context



Bring dimensional context to our measurement gives us the data driven insights that we can use for decision making in a business intelligence.

Typical dimensional insights

What about…

- What is the average annual faculty salary for assistant professors, by department, by academic year?
- What is the average student loan balance for engineering majors, by class year, by campus?

Dimensional Context

Dimensional context: "by" vs. "for"

Wording	Usage*
By	"Sliced and grouped" by values of the entire dimension
For	One or more specific values from within the entire dimension

*almost always

Implied Wording

Implied wording

- What is the average annual faculty salary this academic year for assistant professors, by department?

Dimensional Context

Implied wording

- What is the average annual faculty salary for this academic year for assistant professors, by department?

Dimensional Context

Implied wording

- What is the average annual faculty salary for this academic year for assistant professors, by department?

Dimensional Context

- What is the total number of vacation days taken last academic year by administration employees?

"by administration employees"

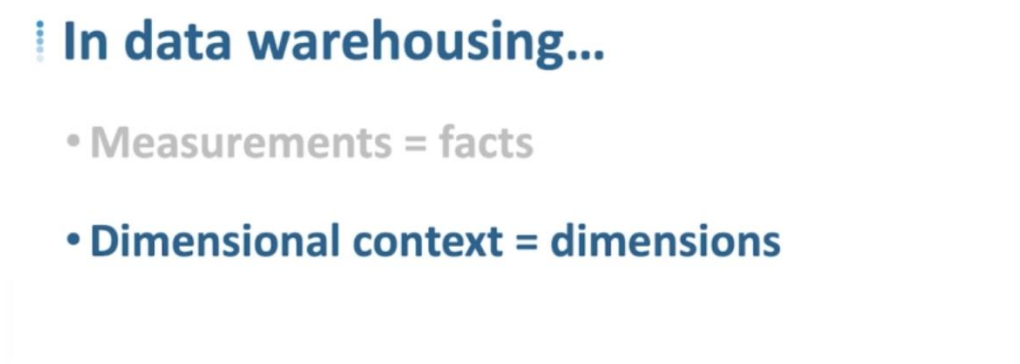
sounds more natural than

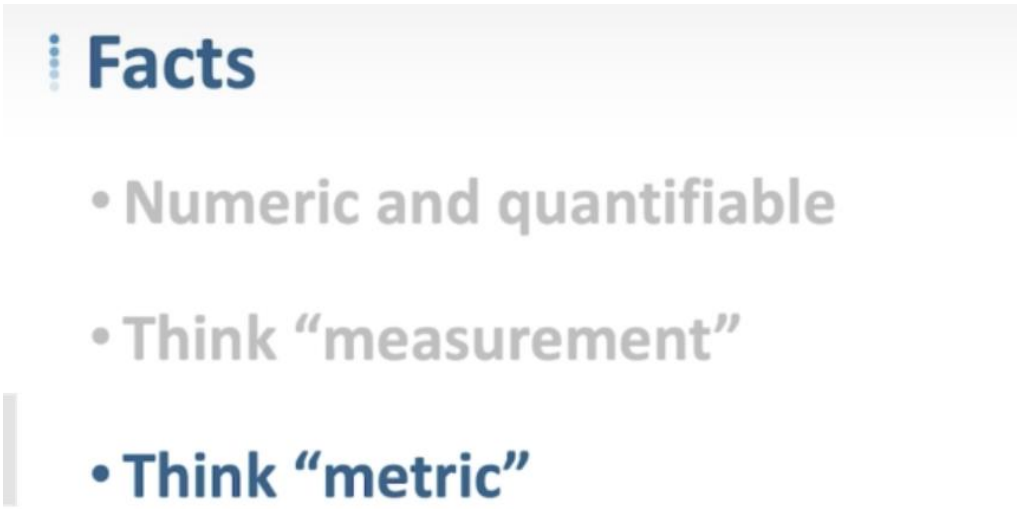
"for administration employees"

The critical lesson

Compare Facts, Fact Tables, Dimensions and Dimension Tables

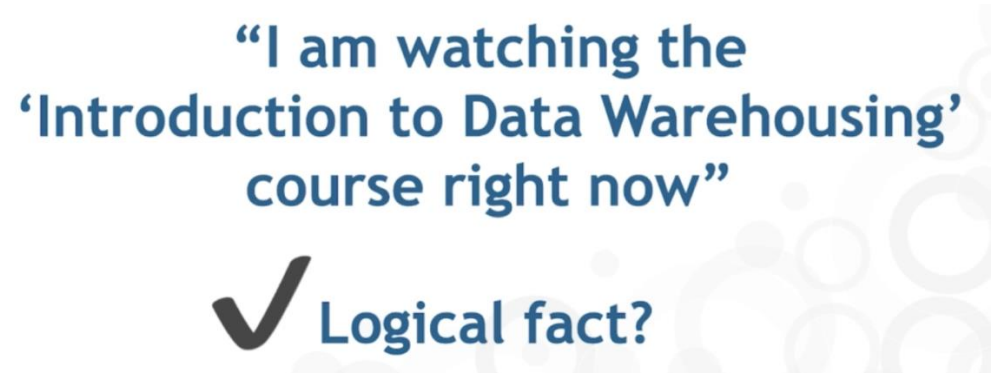
Fact

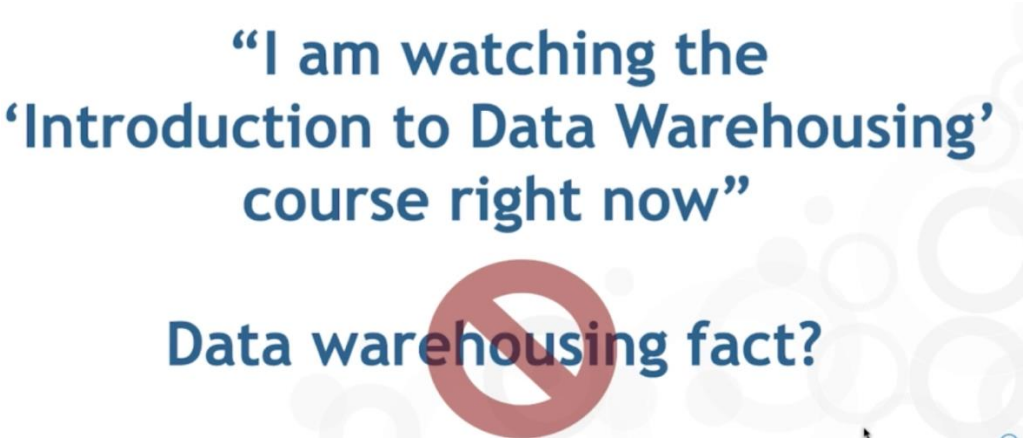




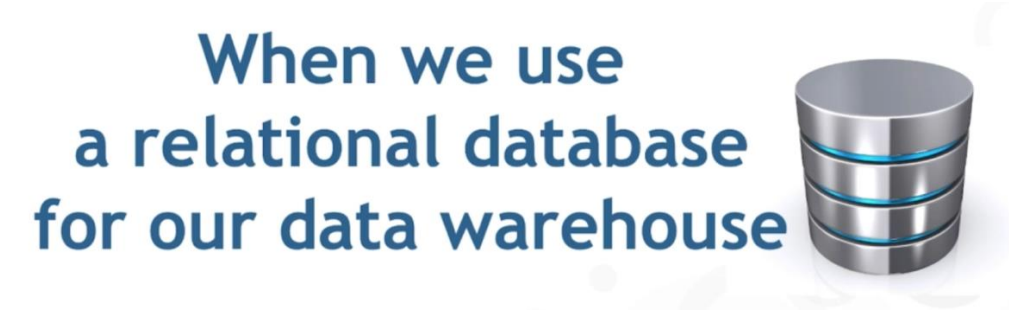
Dimension facts are not the same as a logical fact.

Differ from logical facts

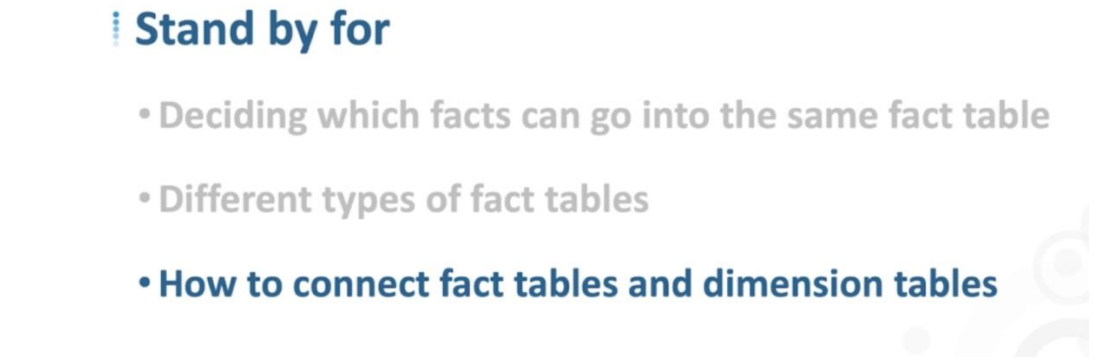




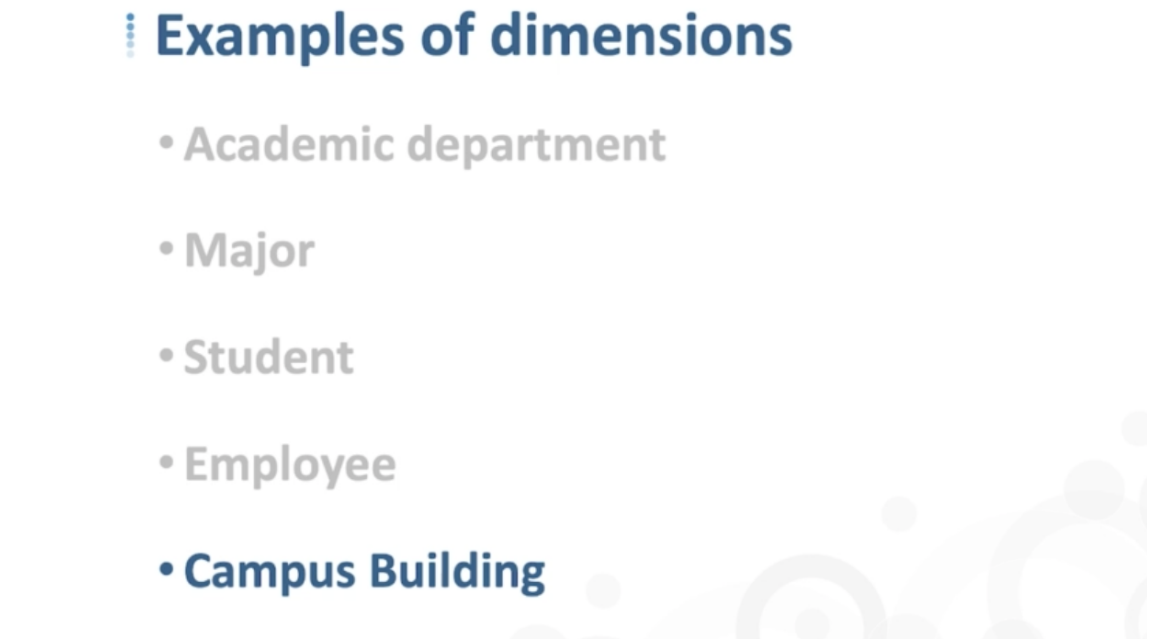
Fact Tables



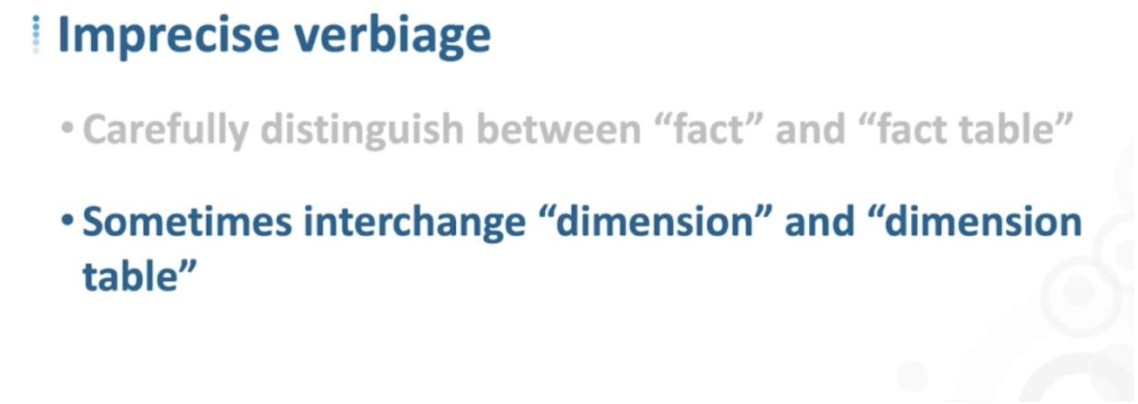
Fact and Fact Table Rules

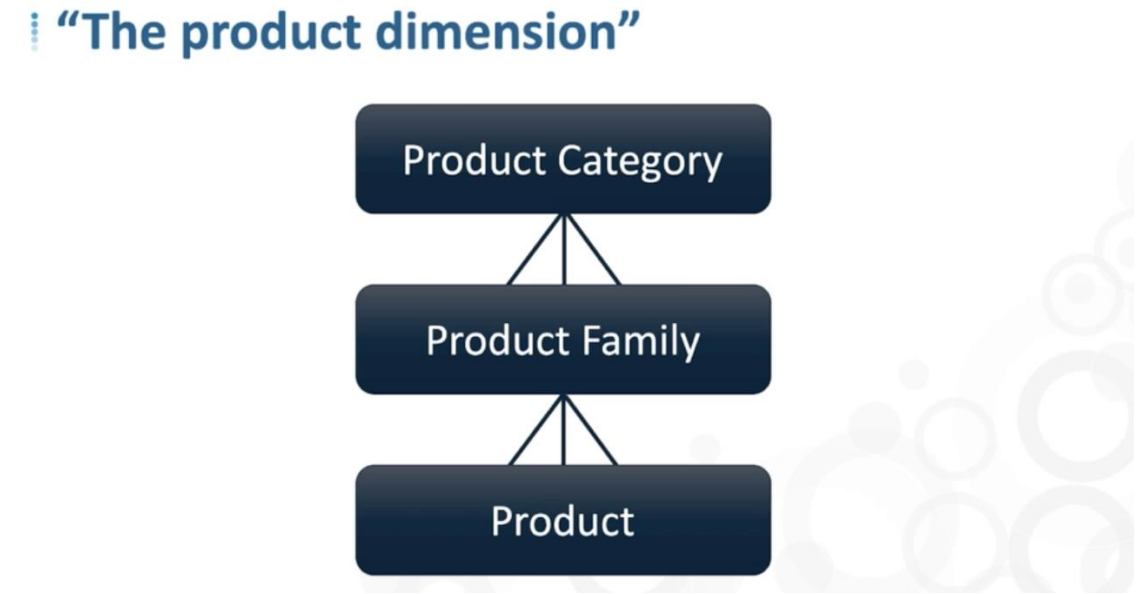


Dimensions

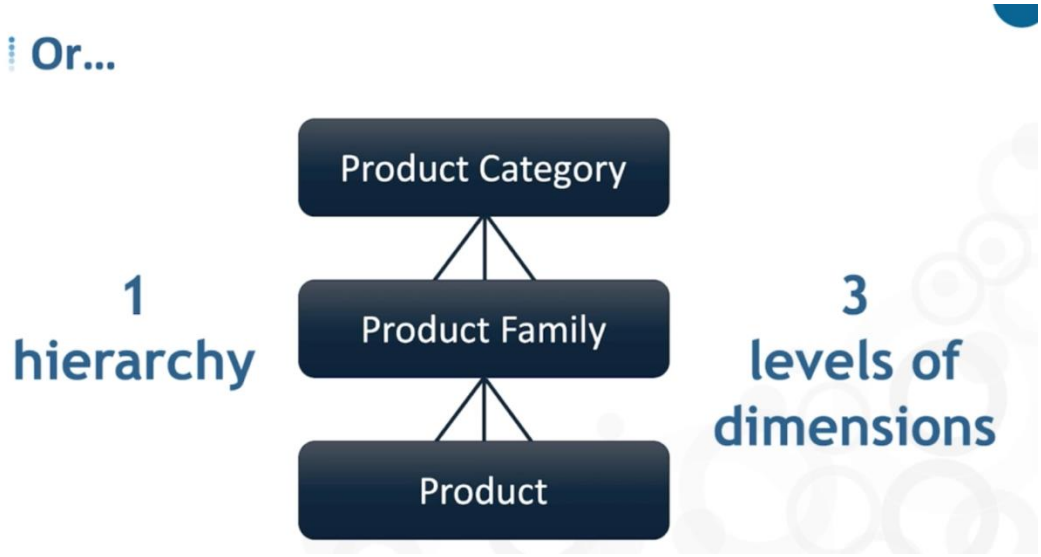


Imprecise verbiage





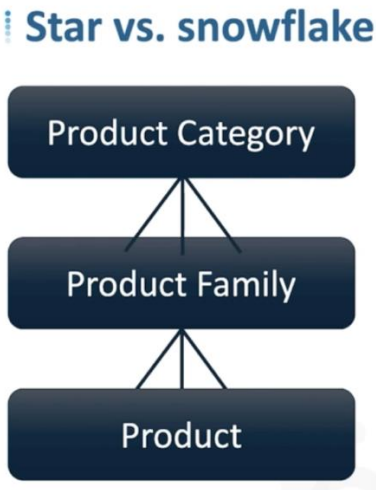
Here we may think that, product dimension means entire set of object. But,

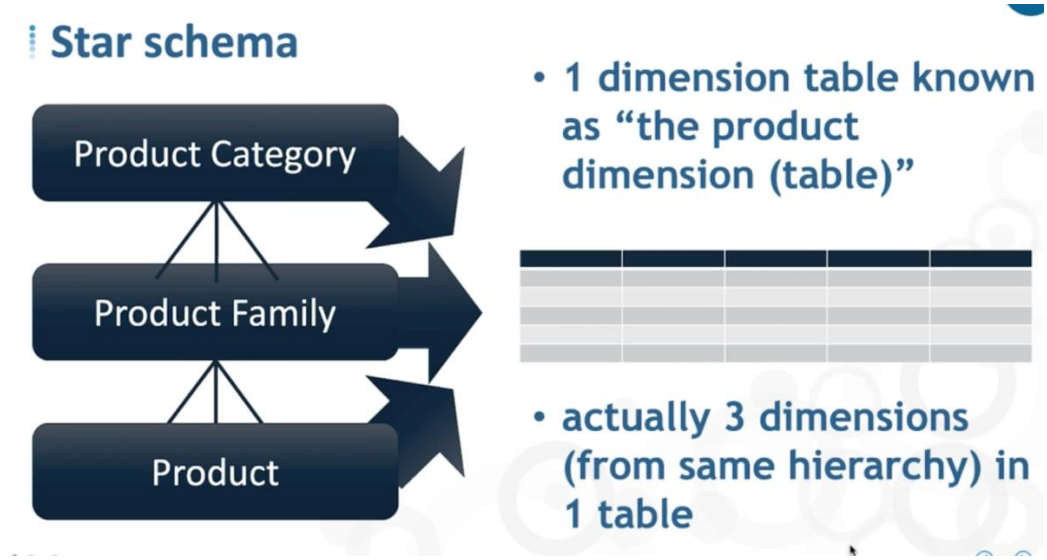


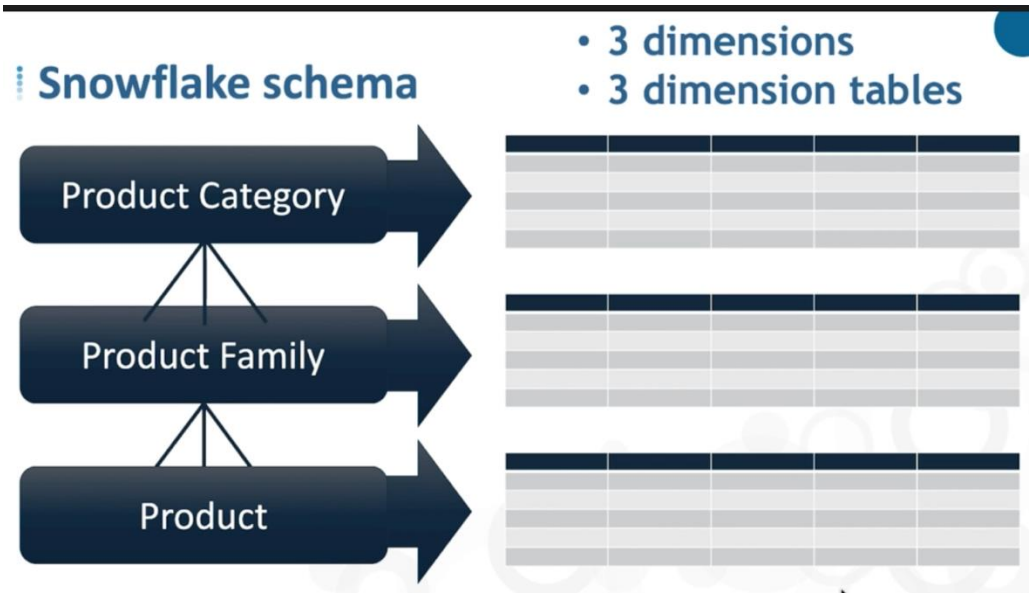
Why it is precise for facts and fact table ?
Why it is imprecise for dimension and dimension tables ?

The reason is difference between star vs snowflake schema.

Start vs snowflake schema difference







Compare Different Forms of Additivity in Facts

Additive Fact

An additive fact can be added under all circumstances

Examples of additive facts

- A faculty member's salary

Faculty Salaries

LastName	FirstName	Rank ...	Salary
Johnson	Susan	Professor	250,000
Wilson	Robert	Asst. Prof	150,000
Tolleson	Mary	Asst. Prof	125,000
Zimmerman	Todd	Professor	200,000
Marcus	Walter	Lecturer	100,000
			825,000

Examples of additive facts

- A faculty member's salary

Faculty Salaries

LastName	FirstName	Rank ...	2020 Salary	2019 Salary	2018 Salary	2017 Salary
Johnson	Susan	Professor	250,000	240,000	240,000	150,000

Professor Johnson total salary 2017-2020: $880,000

Examples of additive facts

- A faculty member's salary
- A student's credit hours completed

Student Credit Hours Tracking

LastName	FirstName	Year...	2020-2021 Credit Hours Completed
Jackson	Sally	FR	30
Thompson	Richard	SO	36
Williams	Greta	FR	34
Young	Ted	FR	28
			128

Examples of additive facts

- A faculty member's salary
- A student's credit hours completed

Student Credit Hours Tracking

			Credit Hours Completed		
LastName	FirstName	Year...	2020-2021	2021-2022	2022-2023
Jackson	Sally	FR	30	36	37

Sally Jackson total credit hours Fall 2020 – Spring 2023: 103

Non-additive fact

Example of a non-additive fact

- Grade point average (GPA)

Student Credit Hours Tracking

LastName	FirstName	Year...	Fall 2020 GPA
Jackson	Sally	FR	3.3
Thompson	Richard	SO	3.2
Williams	Greta	FR	2.8
Young	Ted	FR	4.0
			13.3

Example of a non-additive fact

- Grade point average (GPA)

Student Credit Hours Tracking

LastName	FirstName	Year...	Fall 2020 GPA
Jackson	Sally	FR	3.3
Thompson	Richard	SO	3.2
Williams	Greta	FR	2.8
Young	Ted	FR	4.0
			13.3

Couldn't we calculate average GPA?

Yes...but additivity relates solely to "ability to add"

Typical non-additive facts

- Margins
- Ratios
- Percentages
- Calculated average (e.g., GPA)

With non-additive facts

- Store underlying components in fact tables
- Possibly store non-additive fact also for individual row easy access (minimal calculations)
- **Calculate aggregate averages, ratios, percentages, etc. from totals of underlying components**

Semi-additive facts

Semi-additive facts

- Sometimes you can add these facts
- But other times you can't add them
- Typically used in periodic snapshot fact tables
- **We'll look at an example when we get to periodic snapshots**

Compare a star schema to a snowflake schema

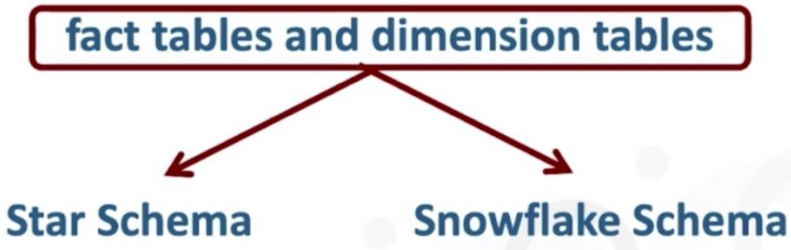
Recapping some key points

Recapping some key points

- A data warehouse drives analytical decision-making
- Business intelligence (BI) is a key facet of analytical decision making
- OLAP is a key facet of BI
- **OLAP = dimensional analysis**

Additional key points

- Dimensional analysis supported by underlying dimensional data
- Dimensional data = facts and dimensions, fact tables and dimension tables



Star schema vs snowflake schema

Star schema vs. snowflake schema (1)

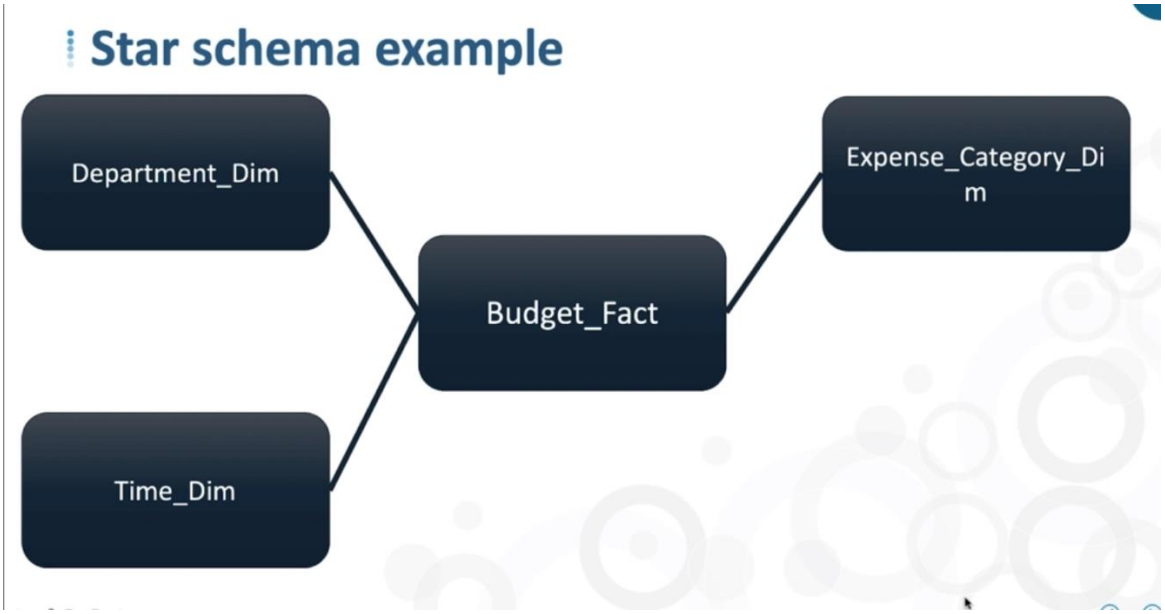
Star Schema	Snowflake Schema
All dimensions along a given hierarchy in one dimension table	Each dimension/dimensional level in its own table
Only one level away from fact table along each hierarchy	One or more levels away from fact table along each hierarchy
With one fact table visually resembles a star	With one fact table visually resembles a snowflake

Star schema vs. snowflake schema (2)

Star Schema	Snowflake Schema
Overall fewer database joins for drilling up/down	Overall more database joins for drilling up/down
Database primary->foreign key relationships straightforward	Database primary->foreign key relationships more complex
Typically more database storage needed for dimensional data	Typically less database storage needed for dimensional data
"Denormalized" dimension table data*	"Normalized" dimension table data*

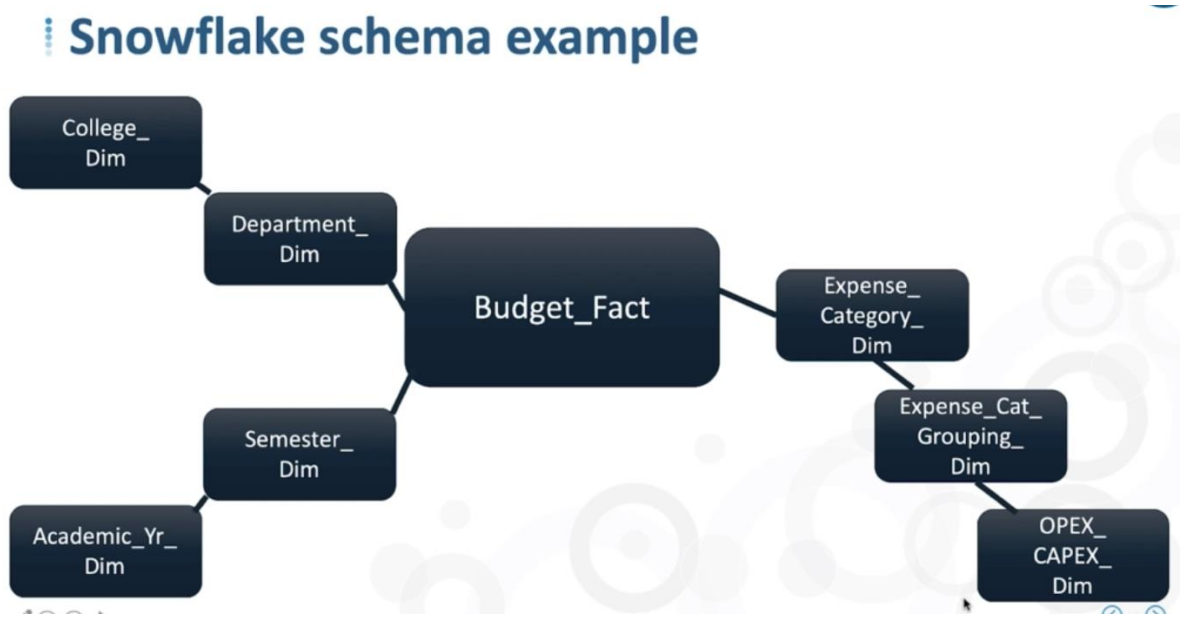
*Normalization/denormalization loosely defined compared to transactional database: 1NF, 2NF, 3NF, etc.

Star schema example

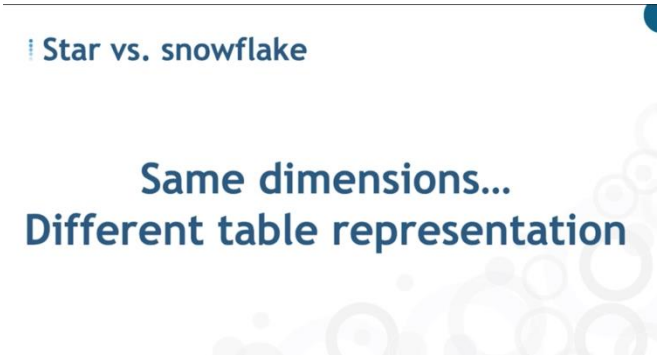


- As we can see, Dimension tables are one level away from the fact tables.
- Department and Expense Category can have hierarchy, but everything is squished into single dimension table.

Snowflake schema



In snow flake schema, hierarchy of dimension are maintained in different dimension table.



Database Keys for Data Warehousing

Fundamentals

- Star and snowflake schemas implemented in RDBMSs
- RDBMSs use logical relationships to relate data across table
- Technically any data can be related to any other data
- **"Official" relationships handled through keys**

RDBMS key types

- Primary vs. foreign keys
- **Natural vs. surrogate keys**

Primary Keys

Primary key

- A unique identifier for each row in a database table
- Could be a single database column (field)
- **Might require more than one database column**

Primary key example

Faculty Master Dimension

Faculty_ID	LastName	FirstName	Rank	...
123982	Johnson	Susan	P	
343225	Wilson	Robert	AP	
829200	Tolleson	Mary	AP	
289912	Zimmerman	Todd	P	
987124	Marcus	Walter	L	
361777	Adleman	Robert	P	
761249	Bonvoy	Janice	AP	
913457	Clark	William	L	
888232	Douglas	Thomas	AP	

Primary key example

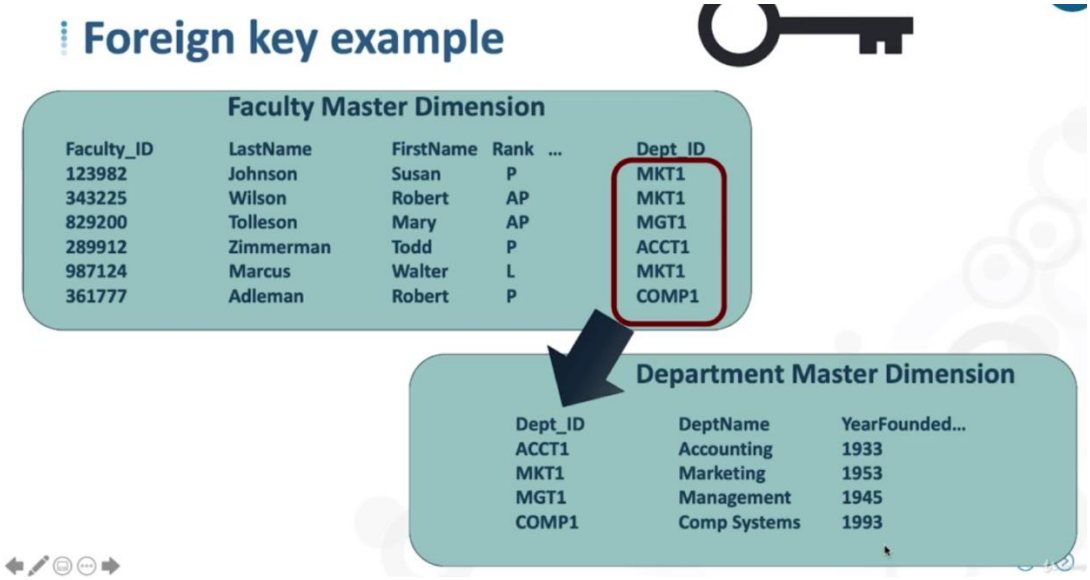
Department Master Dimension

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

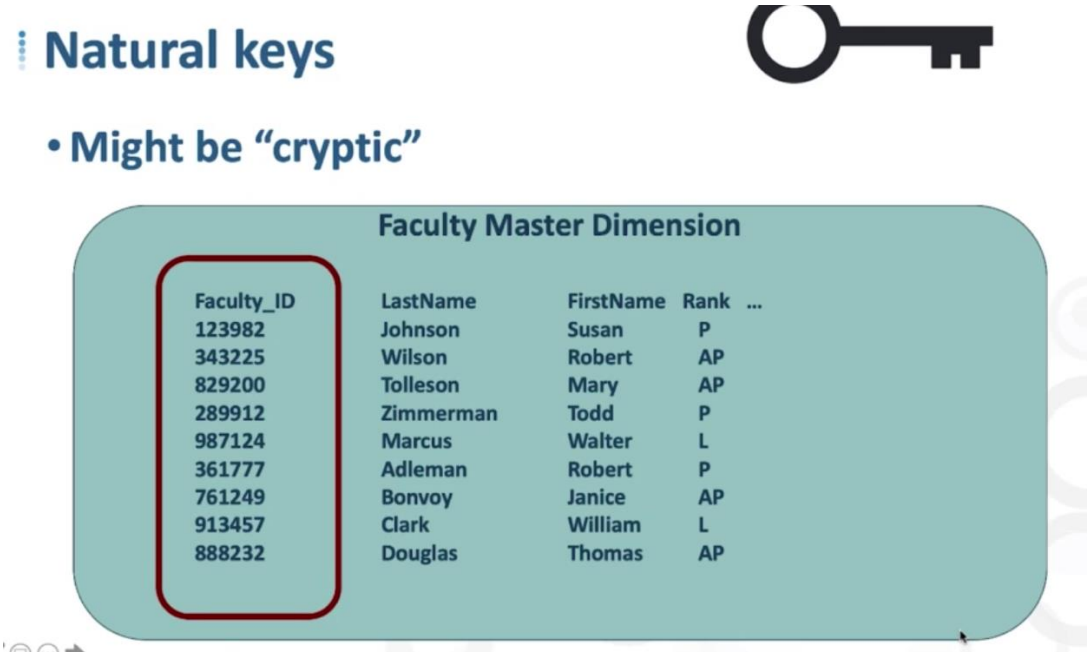
Foreign Key

Foreign key

- "Some other table's primary key"
- Used to indicate logical relationship
- Helps improve query performance

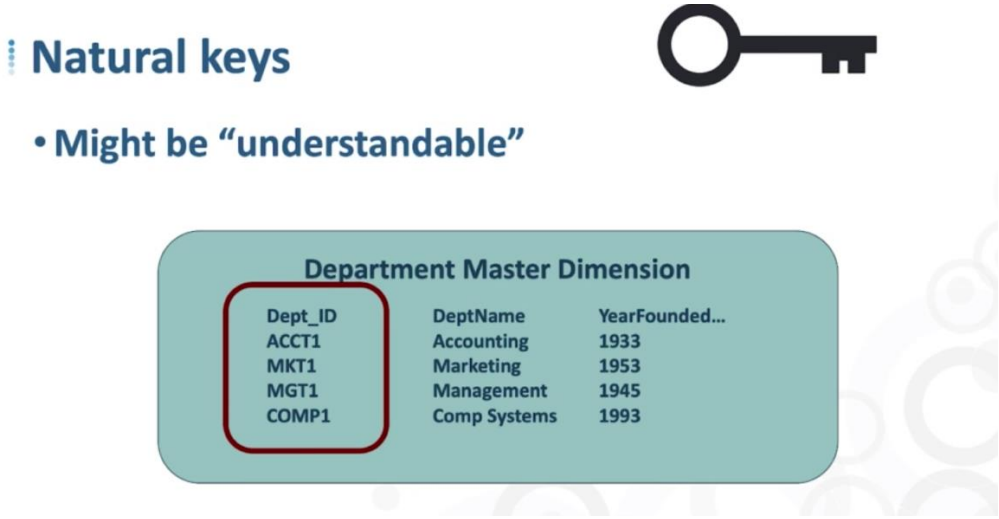


Natural Keys



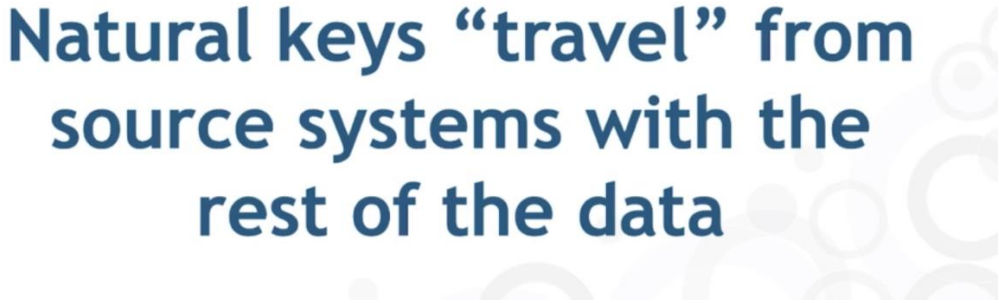
Here, cryptic represent,

- the Faculty_ID which are random six digits number and not related to LastName and FirstName ans so.
- It means, we cannot infer any other information like LastName, FirstName from Faculty_ID column. It's just a random six digits number.



Here, understandable represent,

- We can infer the DeptName from Dept_ID

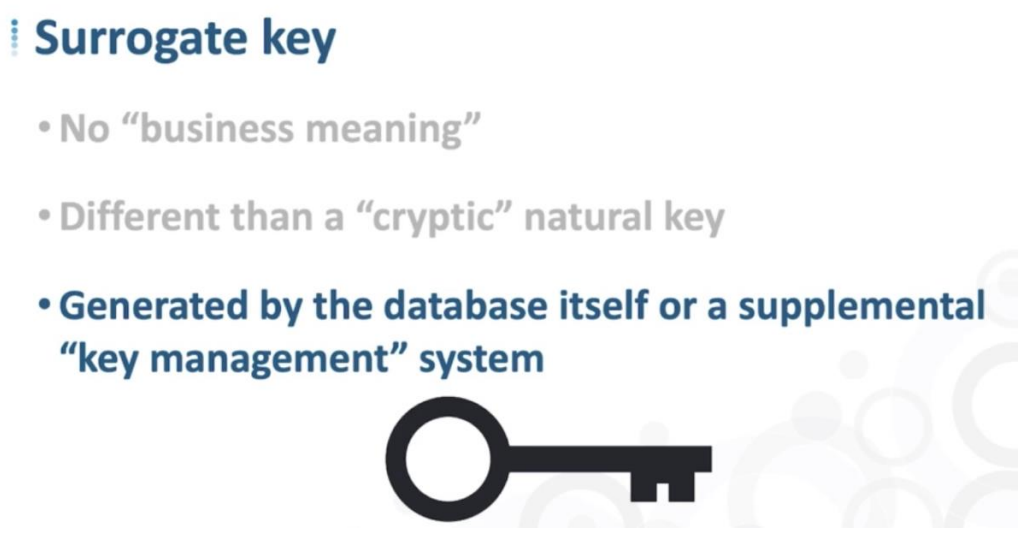


In Either way Natural keys are,

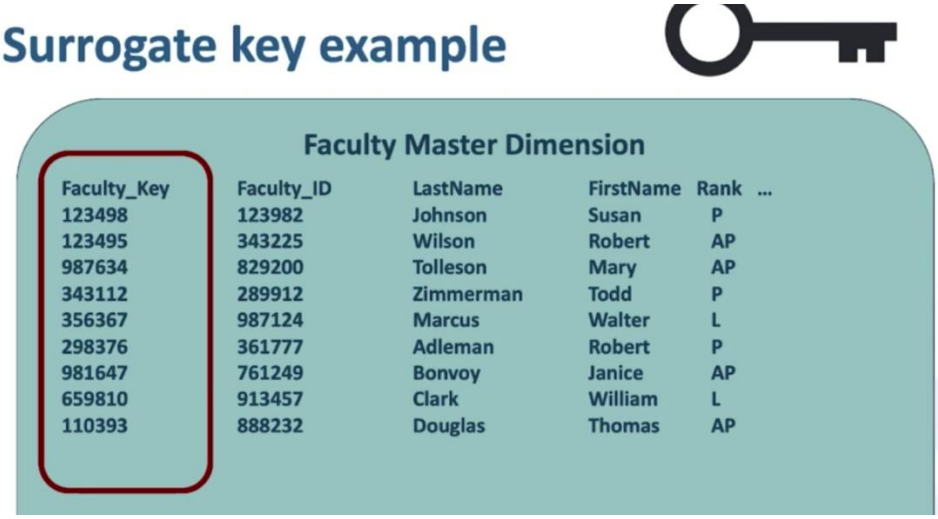
- Comes from source system
- It’s not generated in data warehouse.

Surrogate keys

In data warehouse, it is best practice to use surrogate keys to relate data across tables.



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	

If we use ‘Key’ in column name, example ‘Faculty_Key’, it means that it is a surrogate key. So, it is a best practises to follow.

‘Faculty_Key’ can be used to relate the data instead of ‘Faculty_ID’

Data warehousing and natural 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

Design Facts, Fact Tables, Dimensions and Dimension Tables

Introduction to Dimension Modelling

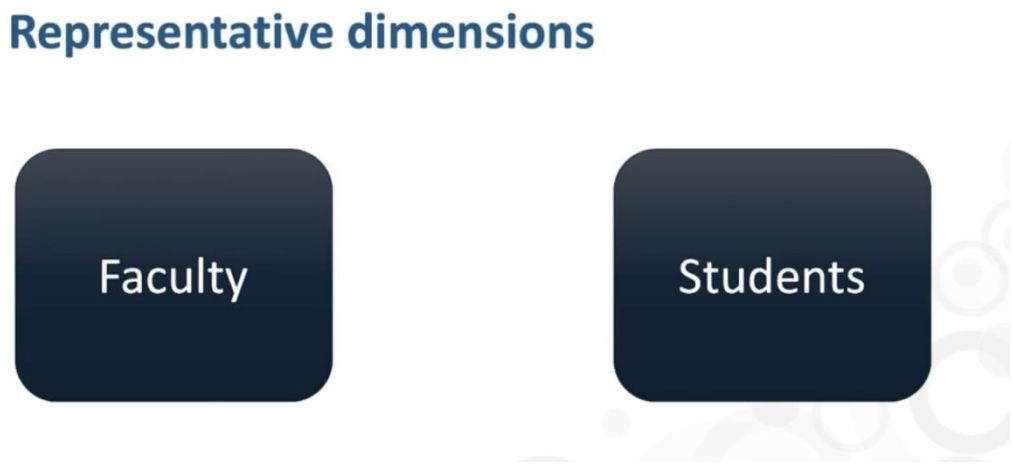
Four Type of Fact Tables

- Transaction-Grained
- Periodic Snapshot
- Accumulating Snapshot
- Factless Fact Table

Structural differences in dimension tables when we are building star schema vs snowflake schema.

Design Dimension Tables for Star and Snowflake Schemas

Representative Dimensions

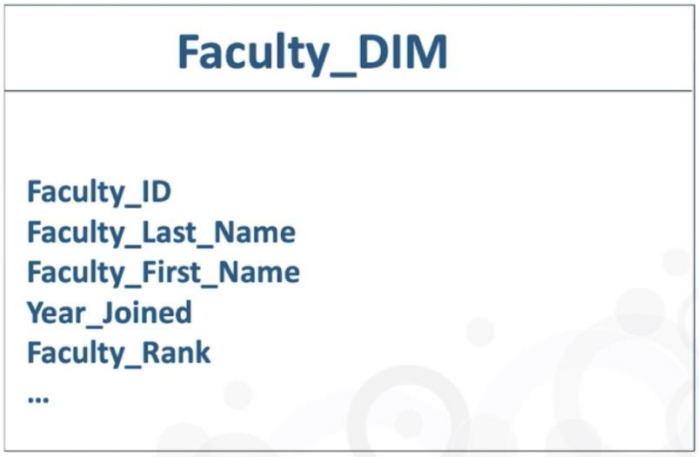


Faculty Dimensions

Representing “Faculty”

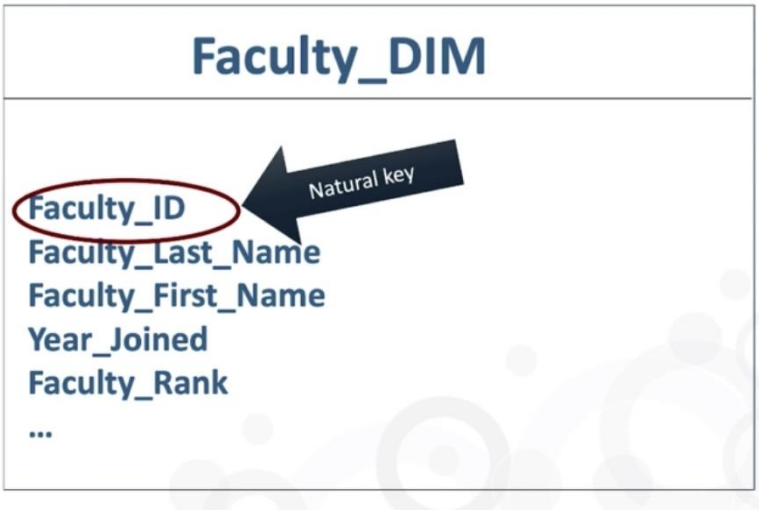
It is a best practice to have a ‘dim’ in table to represent a dimension table.

Columns (fields)



Shows available columns in faculty dimension table.

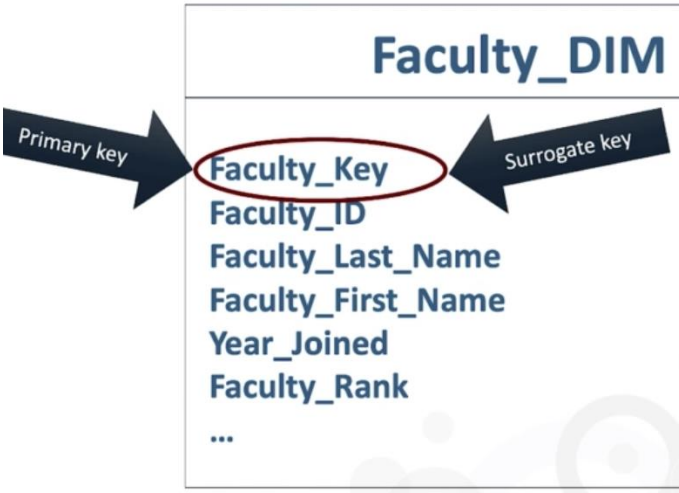
Columns (fields)



Remember:

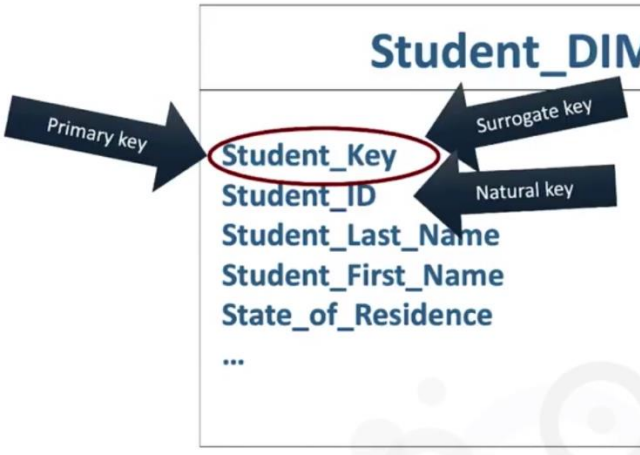
- Dimension → context for measurements (facts)
- Dimension ≠ dimension table
- "Table" → relational database (RDBMS)
- RDBMS table requires primary key
- **Data warehouse primary key = surrogate key**

Columns (fields)

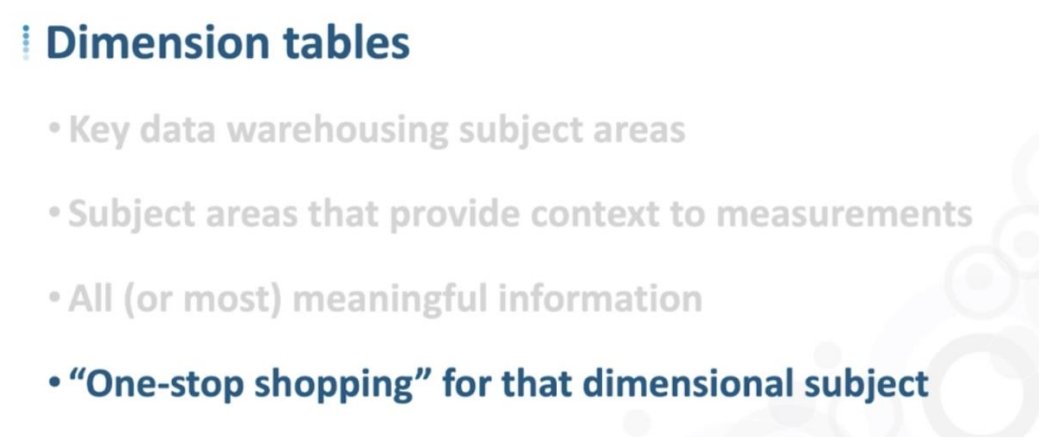


Student Dimensions

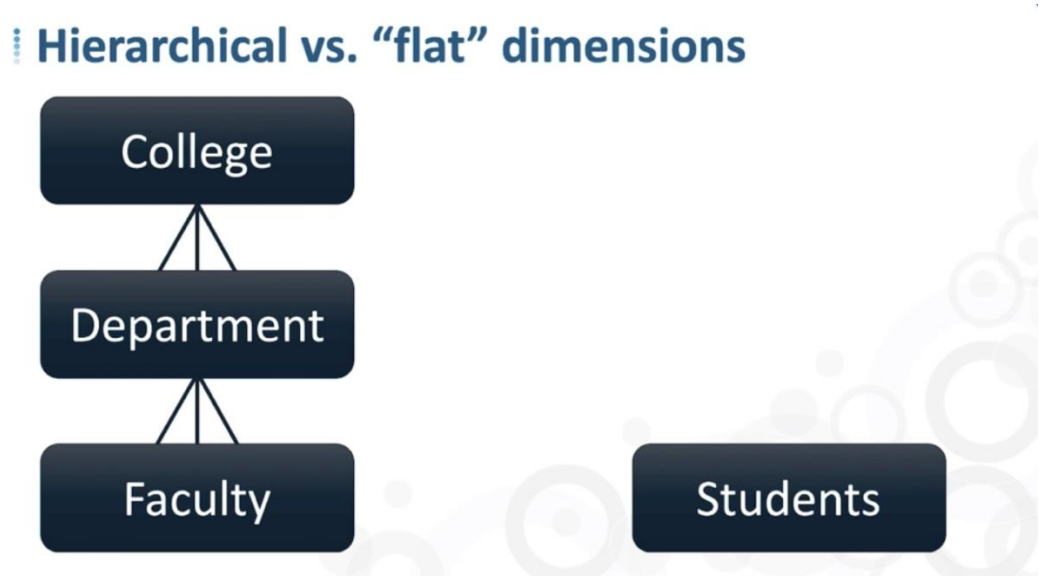
Columns (fields)



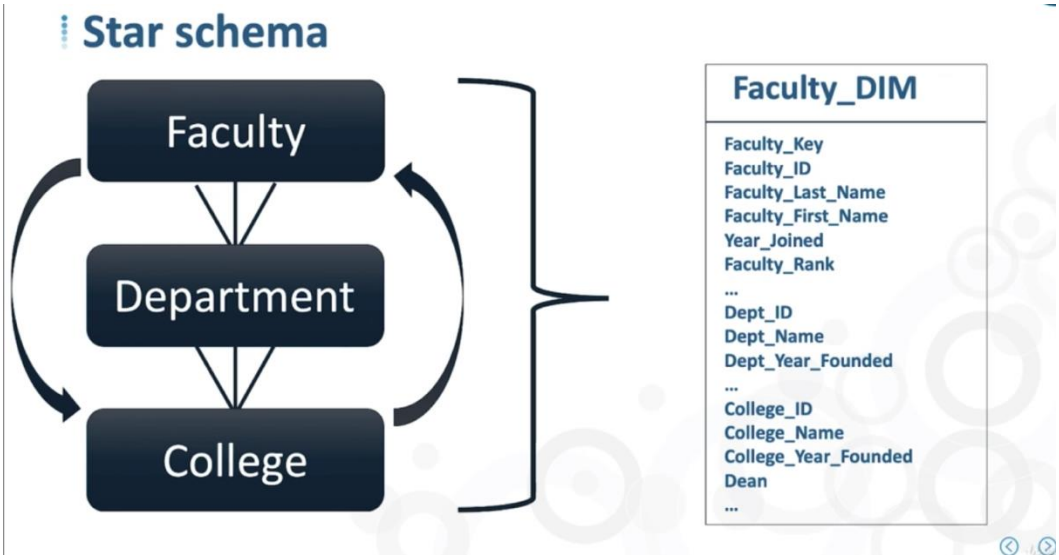
Dimension Tables



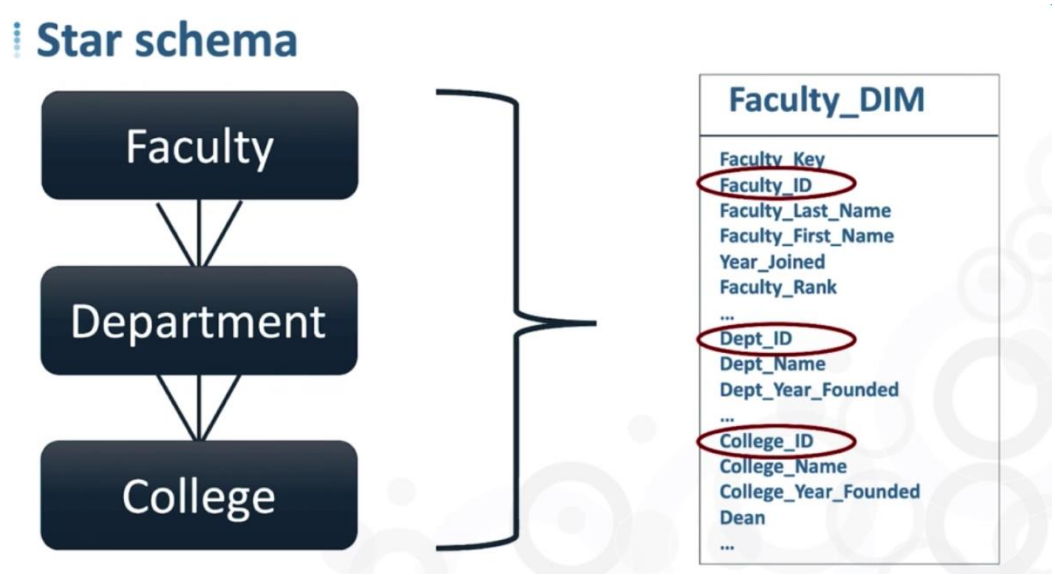
Hierarchical vs flat dimensions



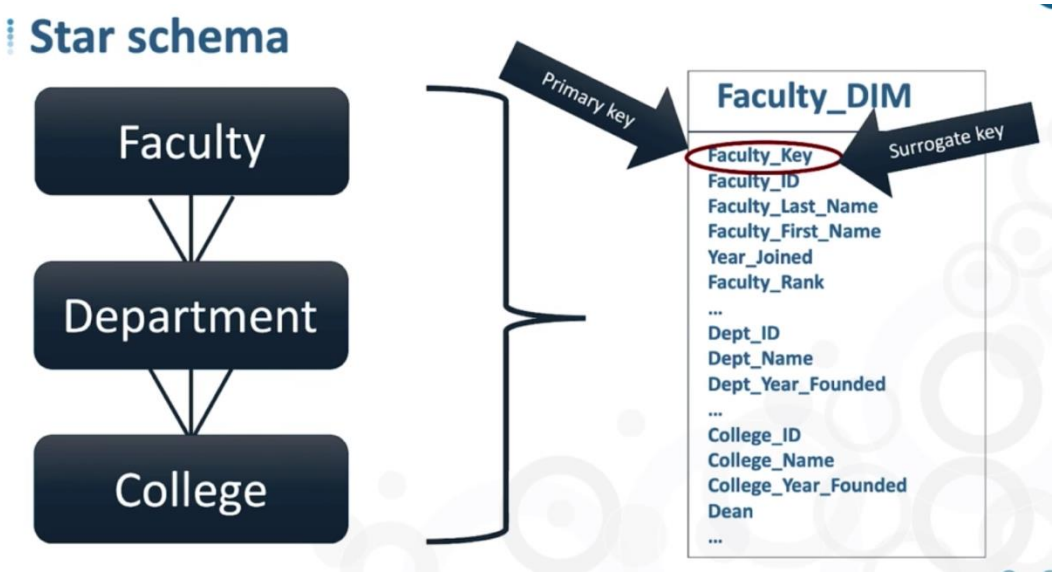
Star Schema



- Flipping the hierarchy upside down when building the table.

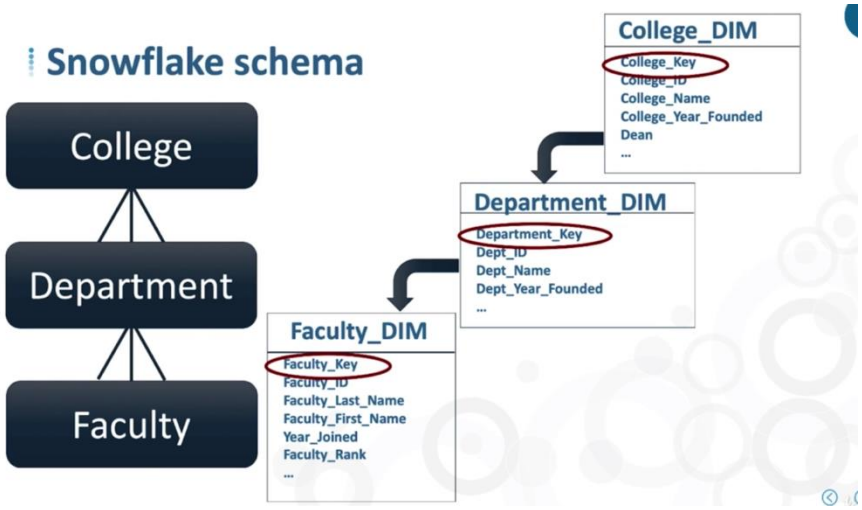


- We have three IDs within faculty dimension table.
- These IDs are natural keys.

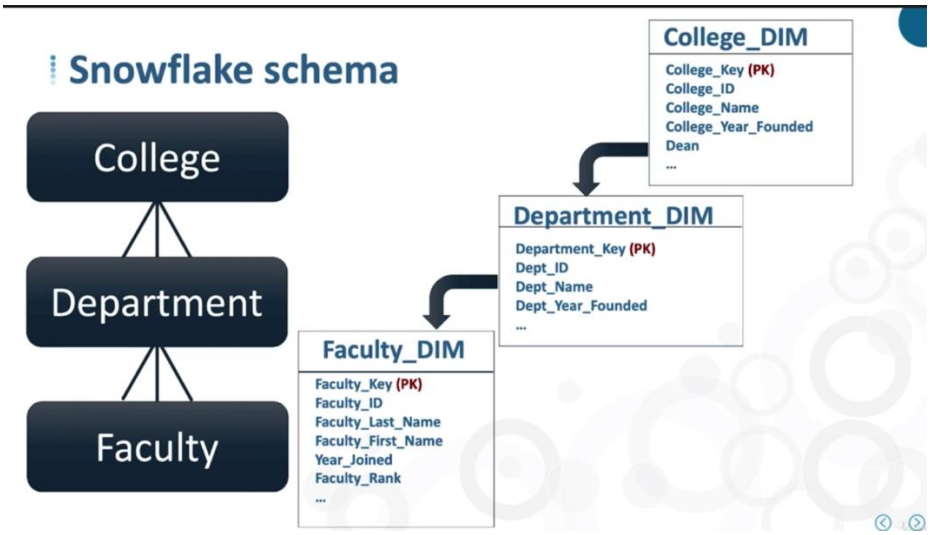


Here, surrogate key is primary key of 'Faculty_DIM' table

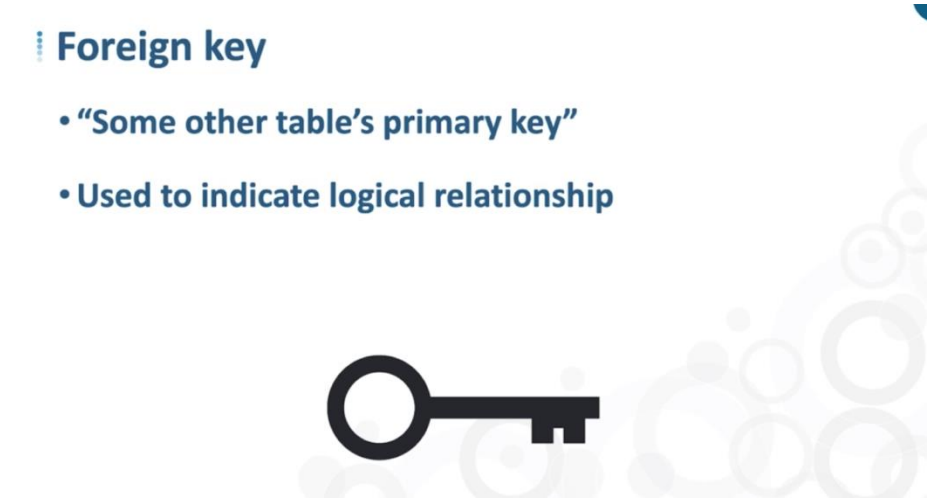
Snowflake schema

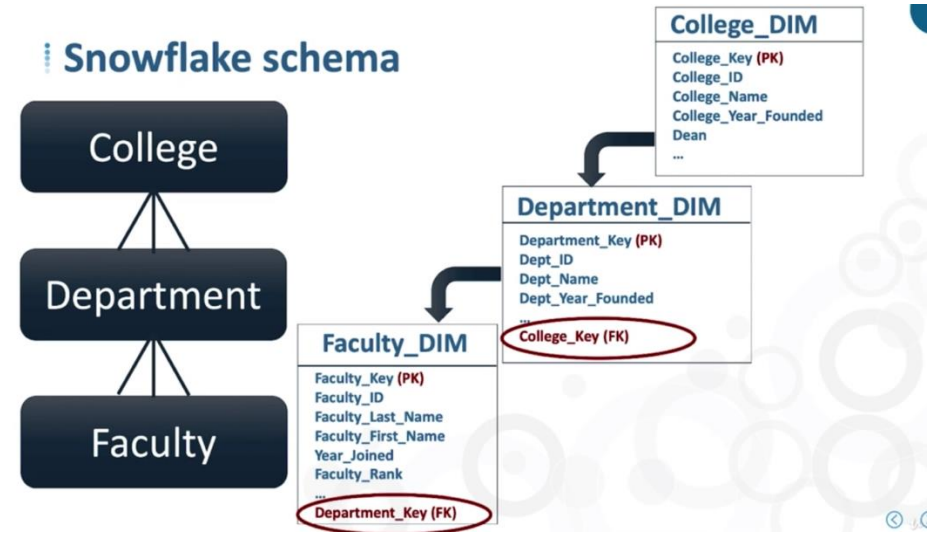


- In Snowflake schema, each table having its own surrogate key.
- Those surrogate keys are respective tables primary key.



Foreign Key

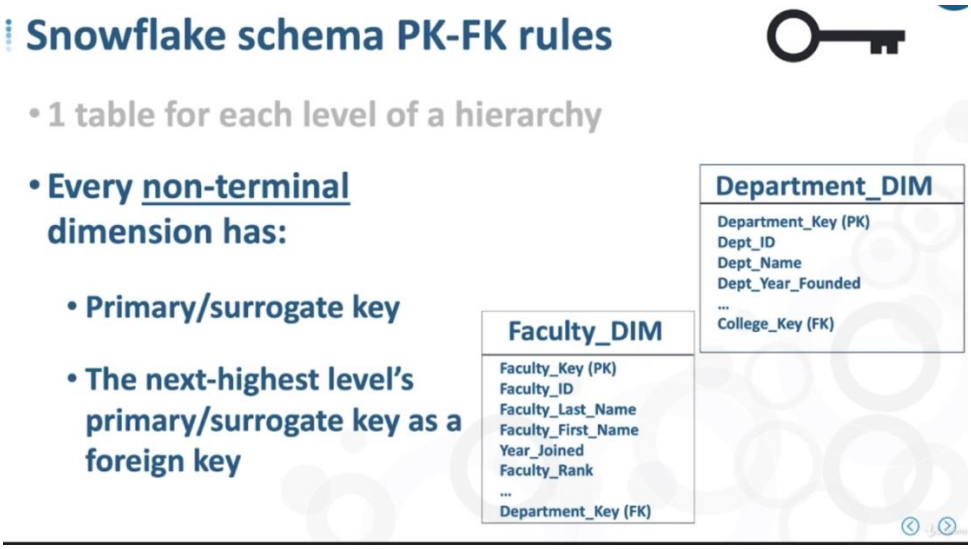


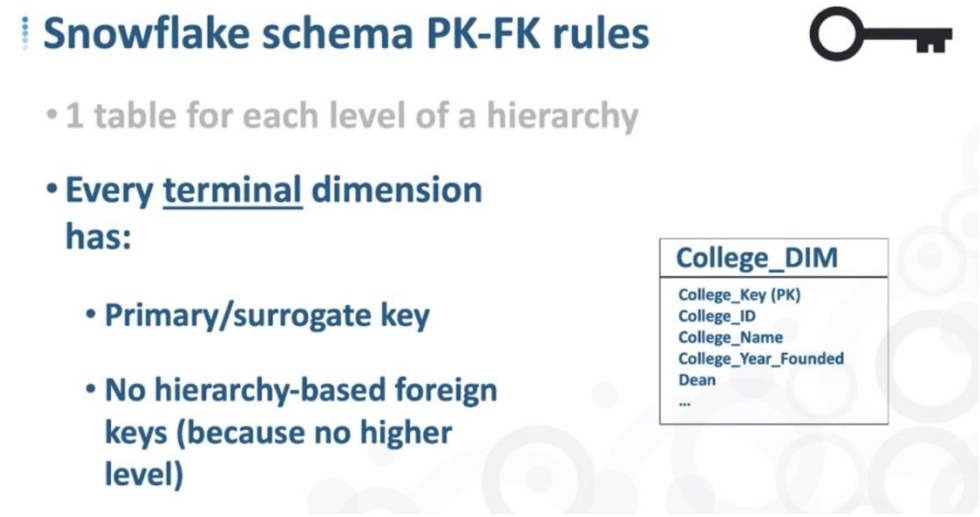


In Snowflake schema, for higher level table, we don't have a foreign key.

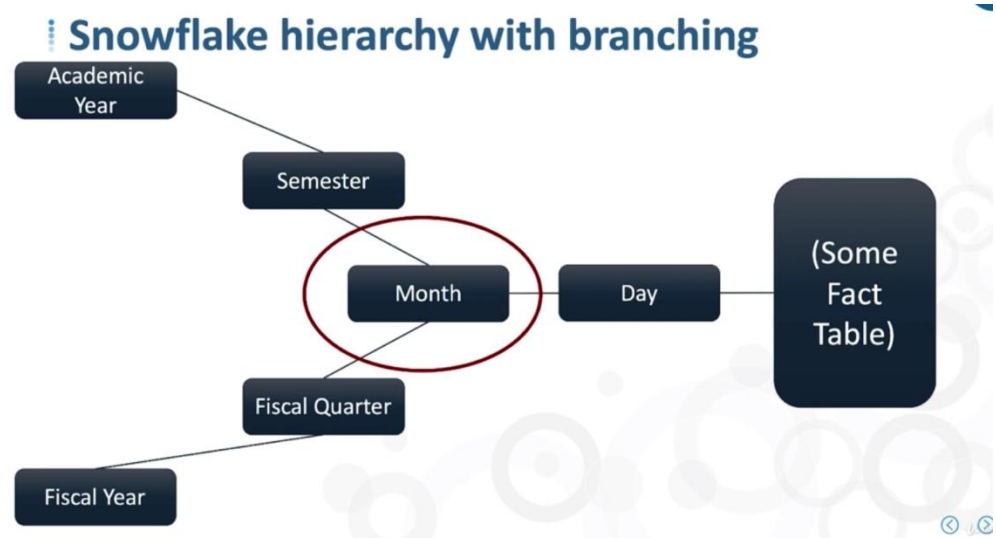
Snowflake schema PK-FK Rules

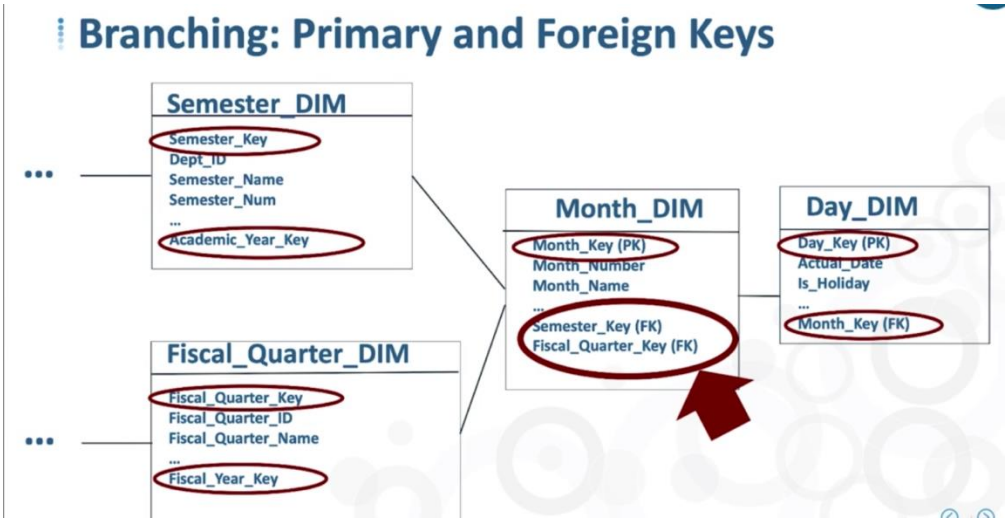
Terminal dimensional table is the one in higher level of hierarchy.





Snowflake hierarchy with branching





Four Main Types of Data Warehousing Fact Tables

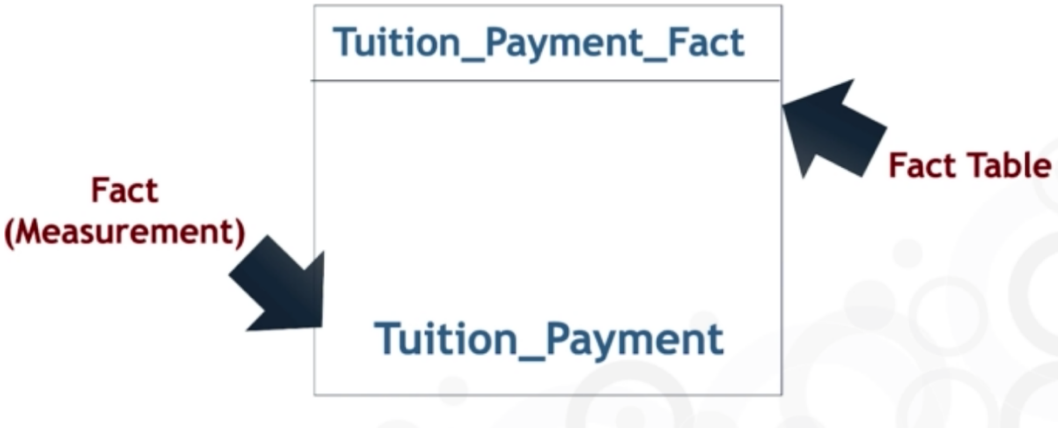
How we use each fact table type

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

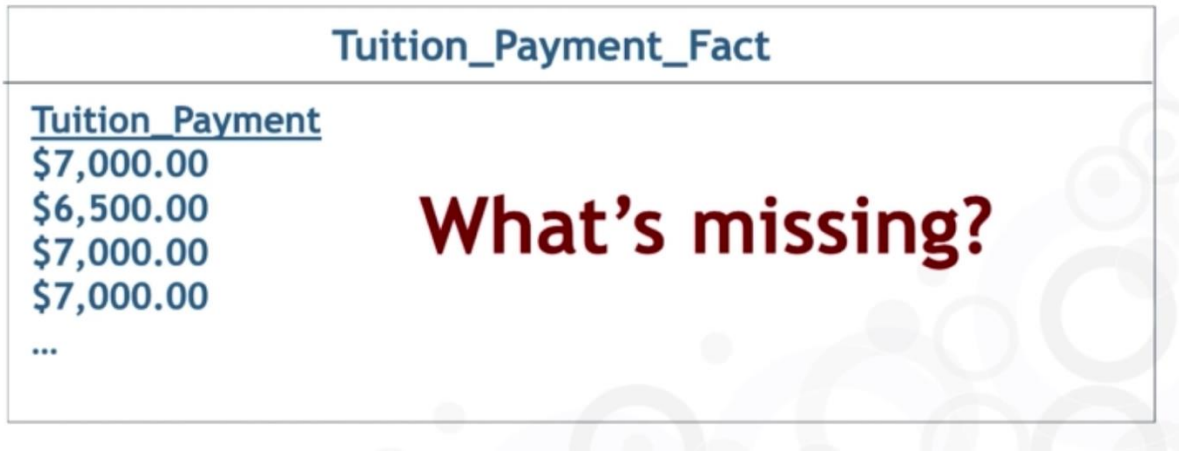
The Role of Transaction Fact Tables

Transaction fact tables

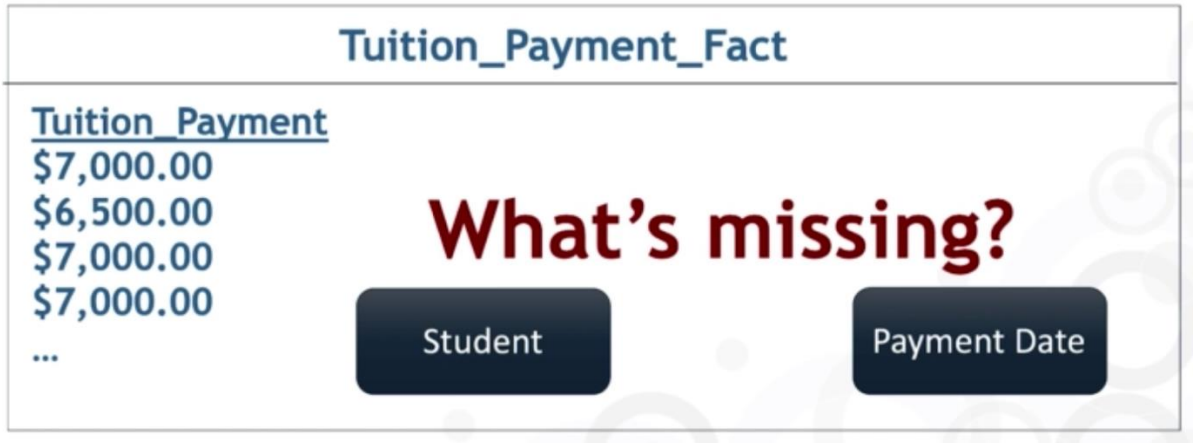
- Formally: transaction-grained fact tables
- Heart of dimensional models
- Literally: a table where we store facts from our transactions
- One or more facts stored together in a fact table
- **Rules tell us how many facts in each fact table**



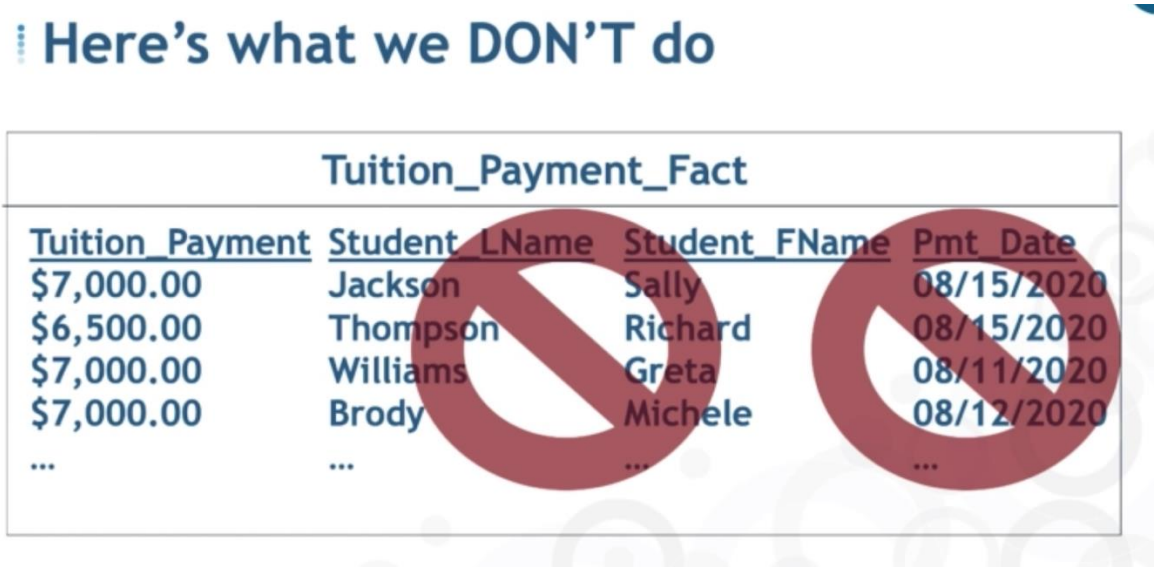
'Fact' will be included with Fact table name to represent it's a fact table

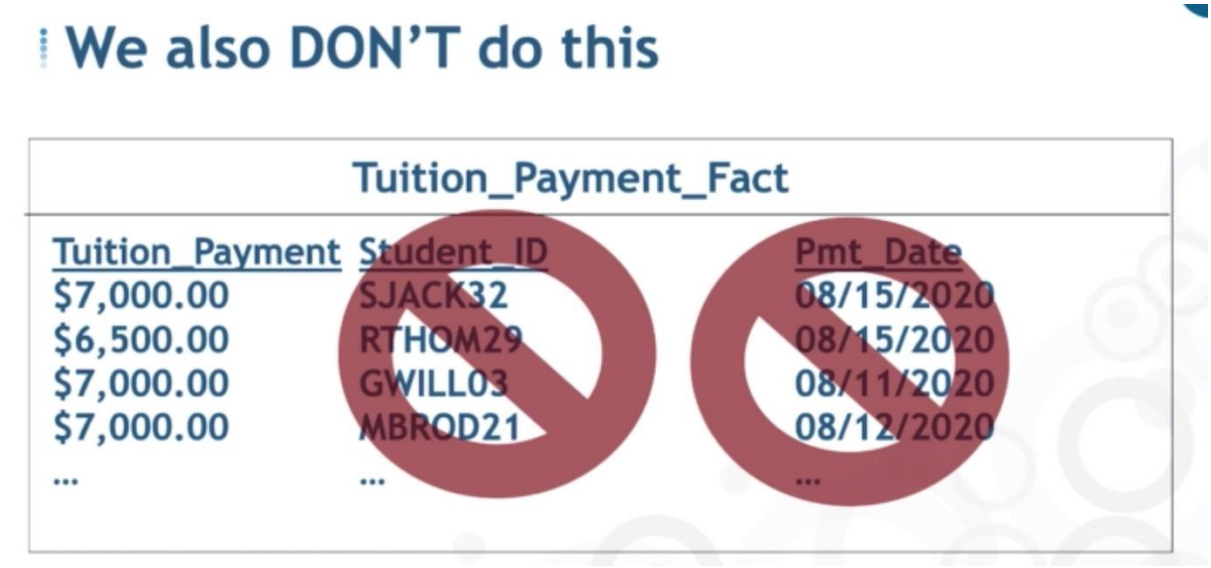


Here, we are missing the context. So context can be,



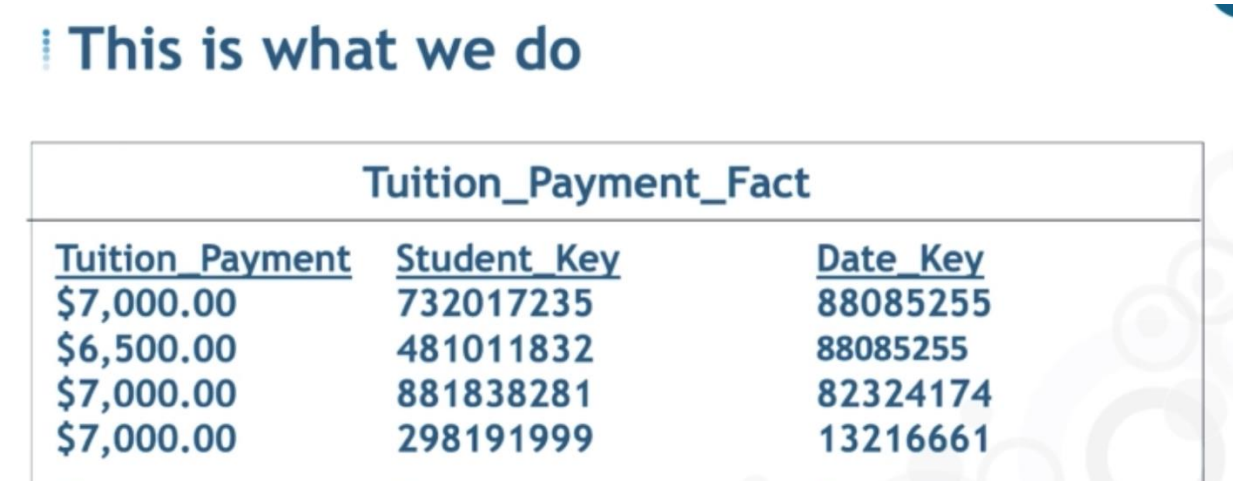
What we don't do in fact table

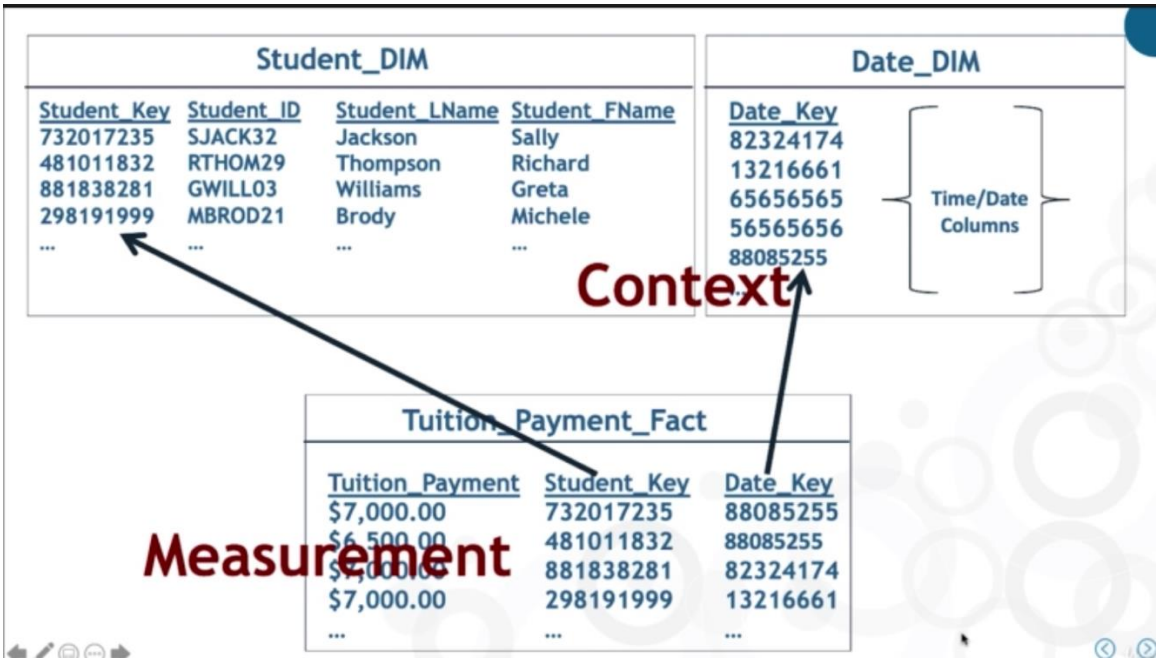




What we do in fact table

We will include the surrogate key in fact table.





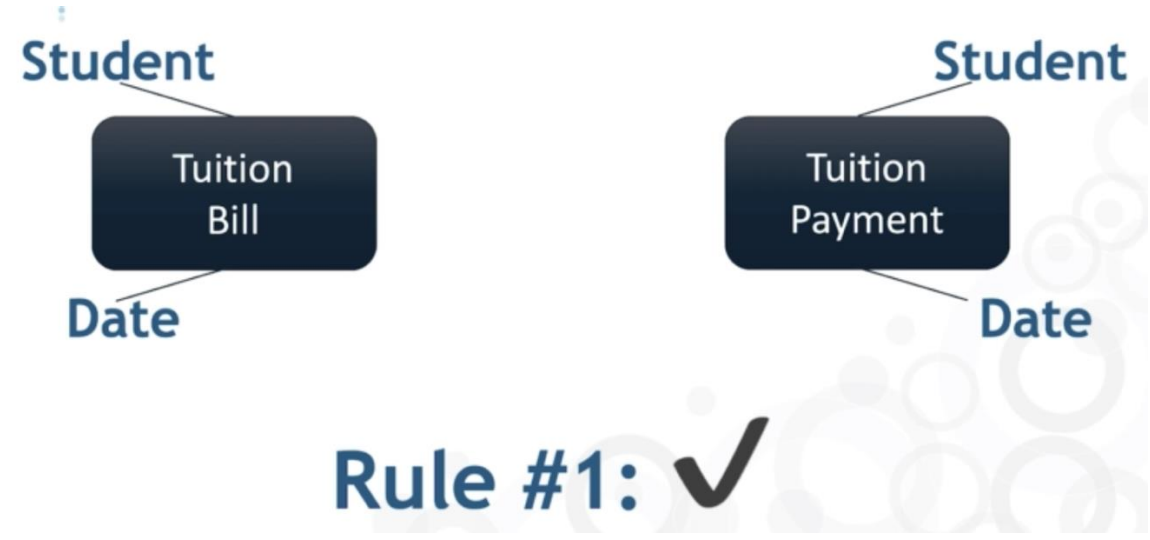
The Rules Governing Facts and Transaction Fact Tables

Rules

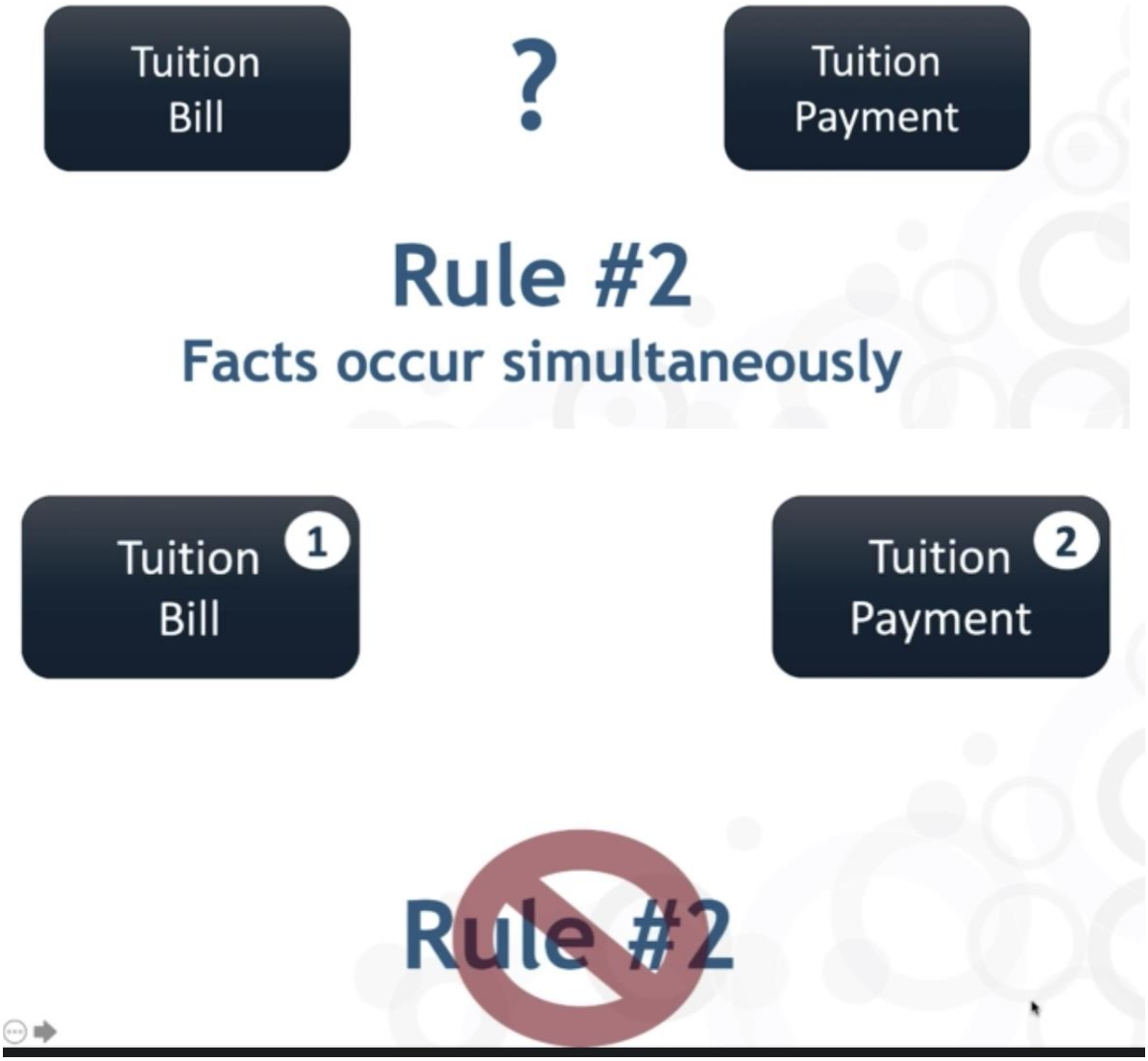
Any 2 or more facts can be stored in the same fact table, if both of the following rules apply

1. Facts available at same grain (level of detail)
2. Fact occur simultaneously

We can tell, if the fact satisfy the rule 1 by looking at the dimensions.

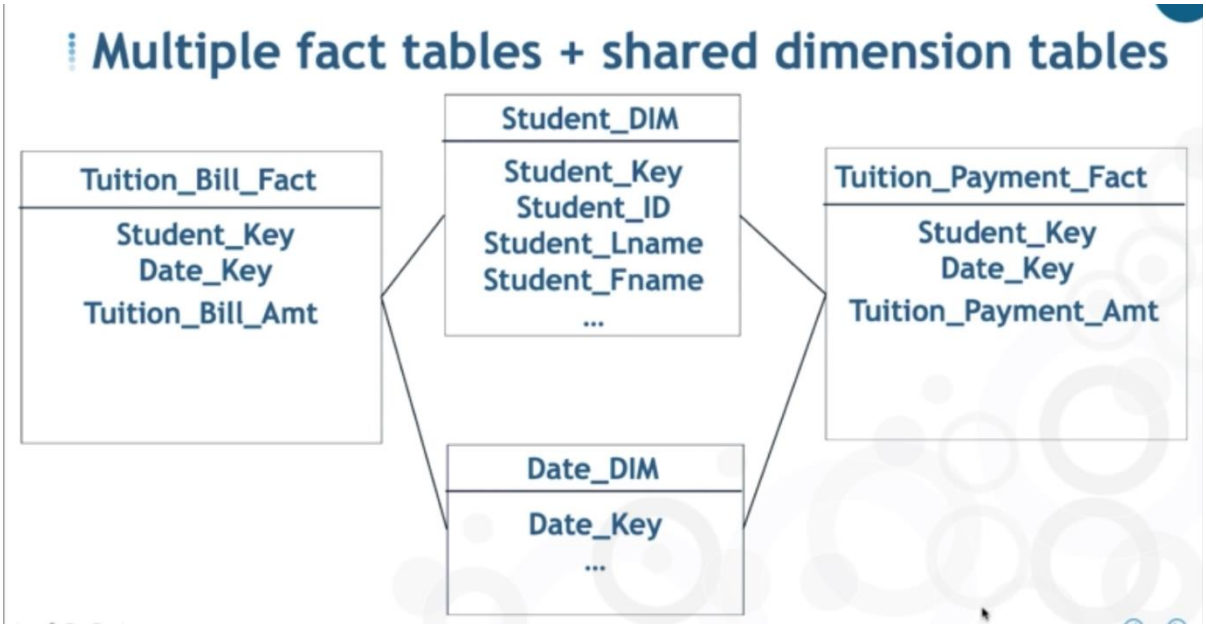


Here, Tuition Bill and Tuition Payment can be analysed at same grain which is Student and Date.

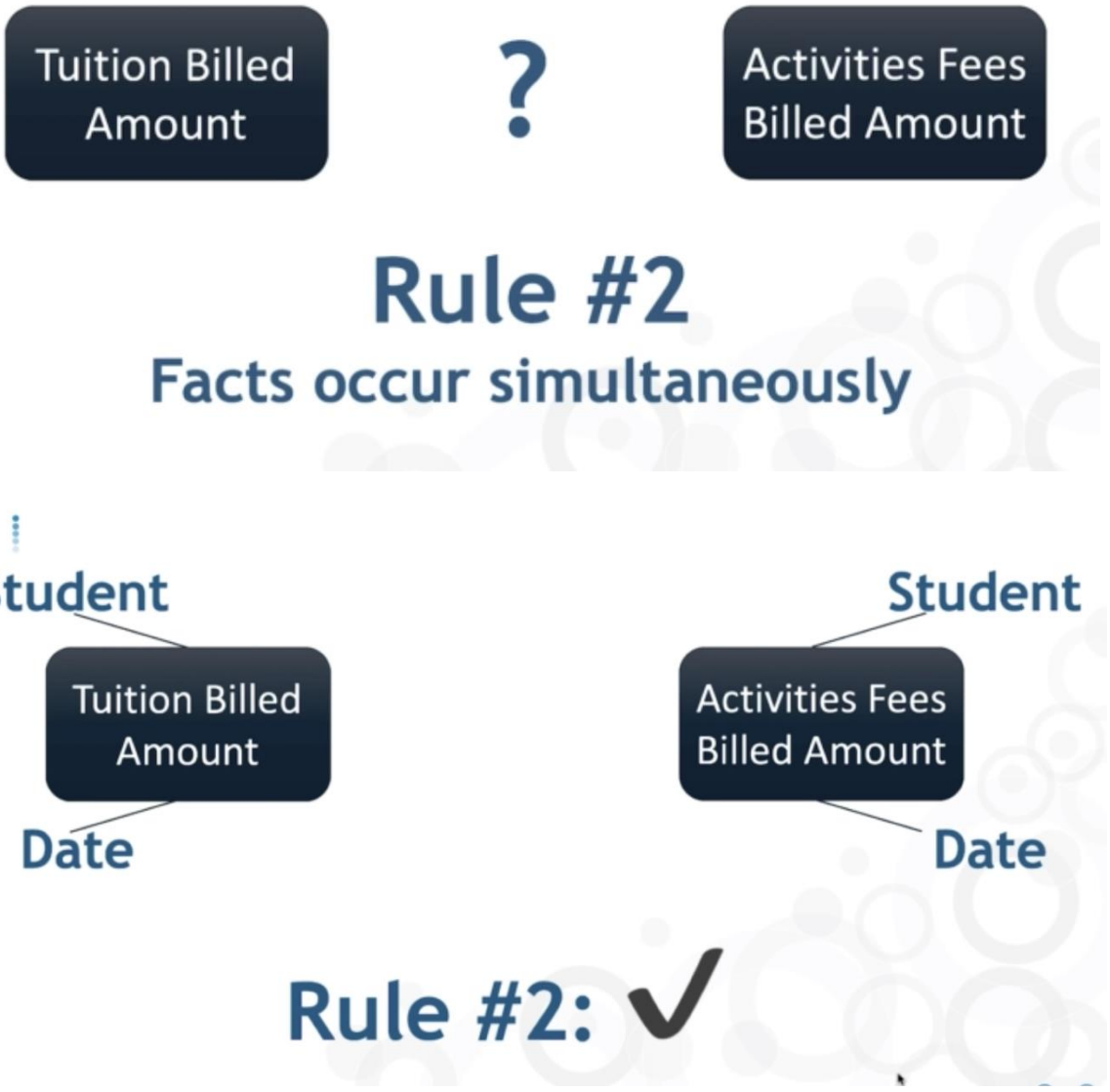


Workaround for Rule 2 violation

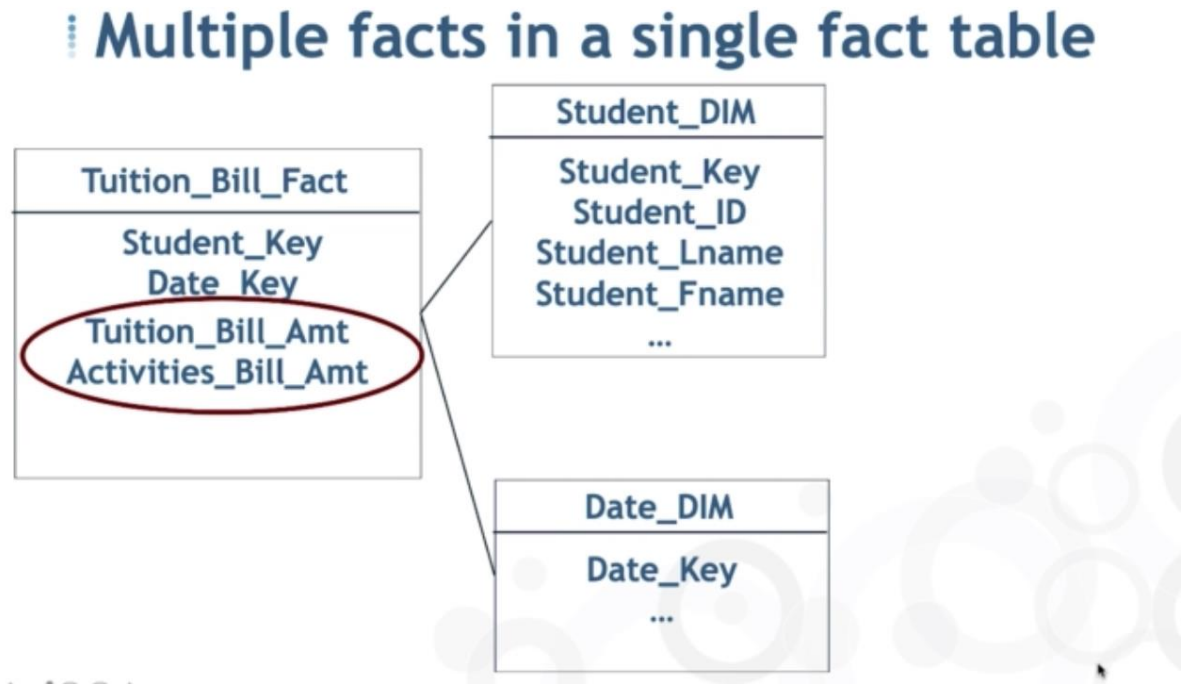
Keep the facts in separate tables. Business processes belong in their own separate fact tables.



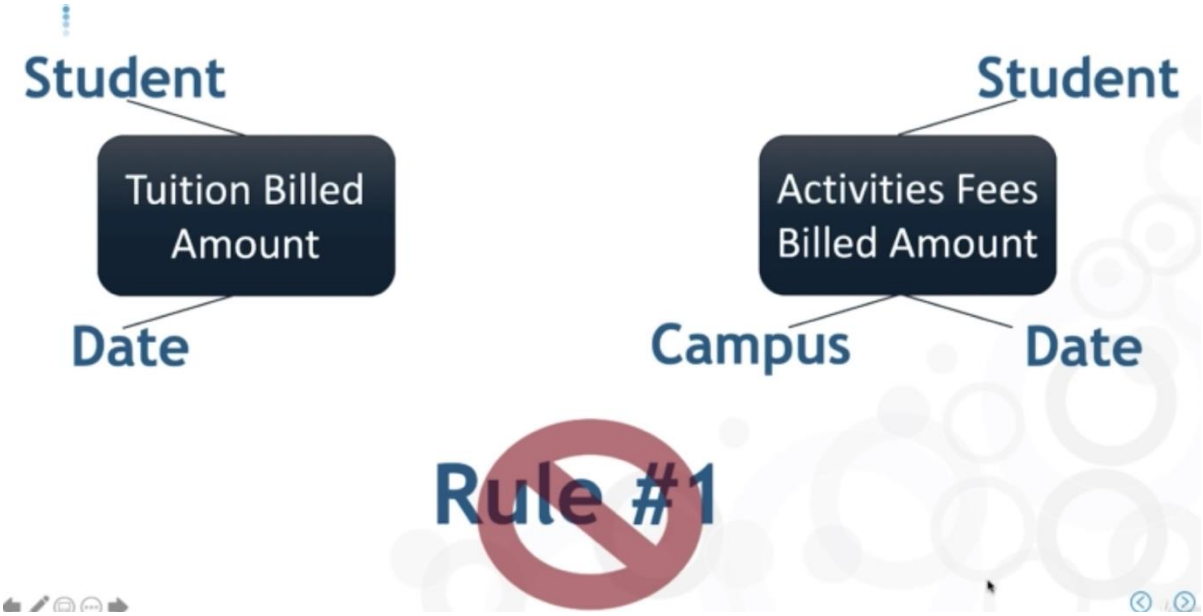
What facts can be stored together ?



Here, if Tuition and Activity fee bill amount comes in same bill, then it is occurring simultaneously.



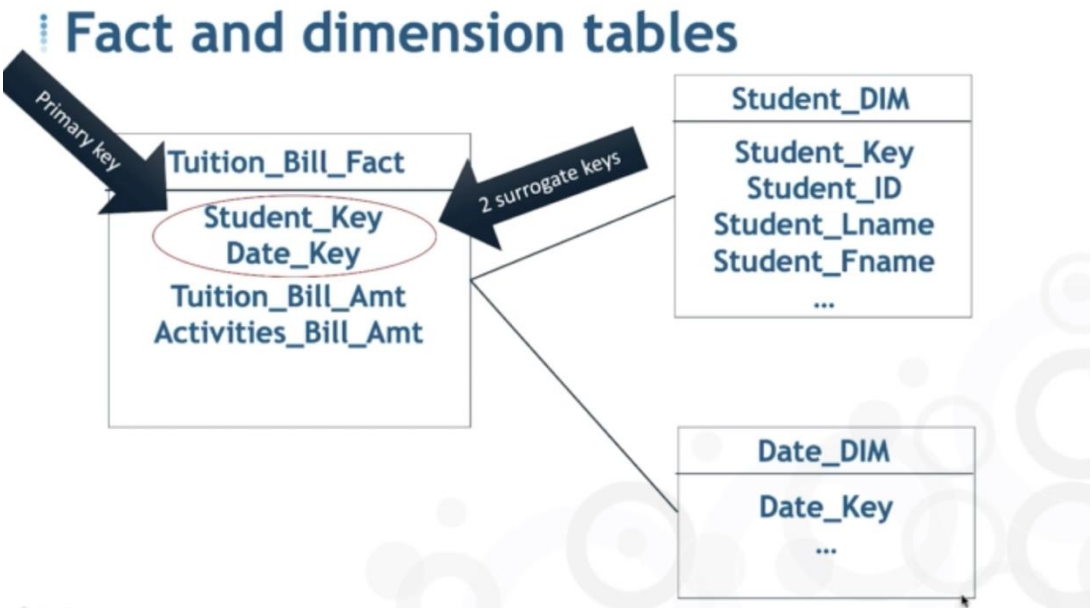
Example of facts at different grains



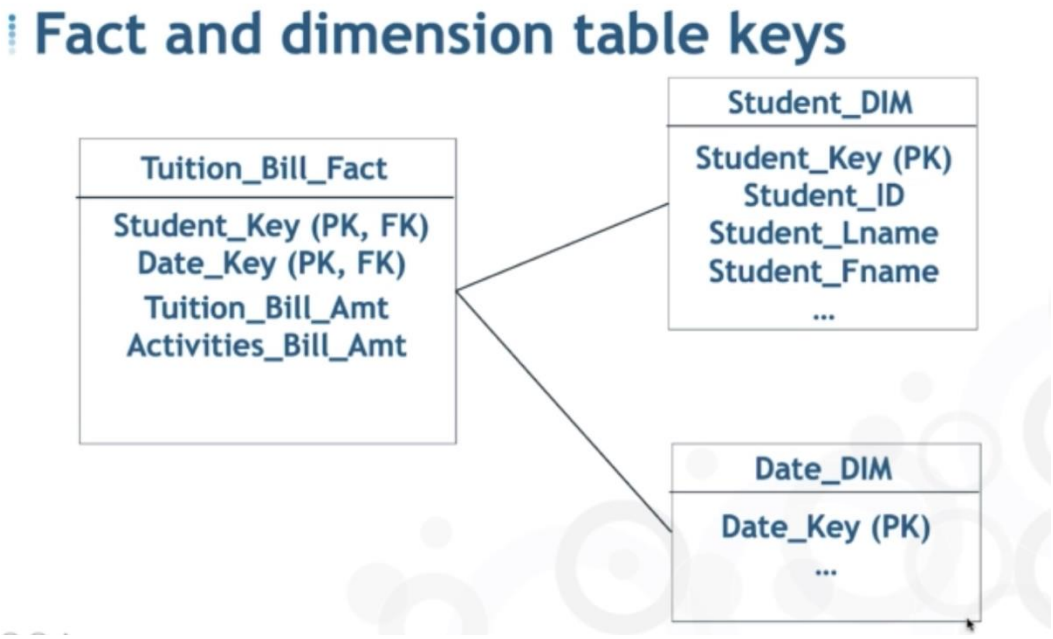
Here, activities fees differ based on campus.

Primary and Foreign Keys for Fact Tables

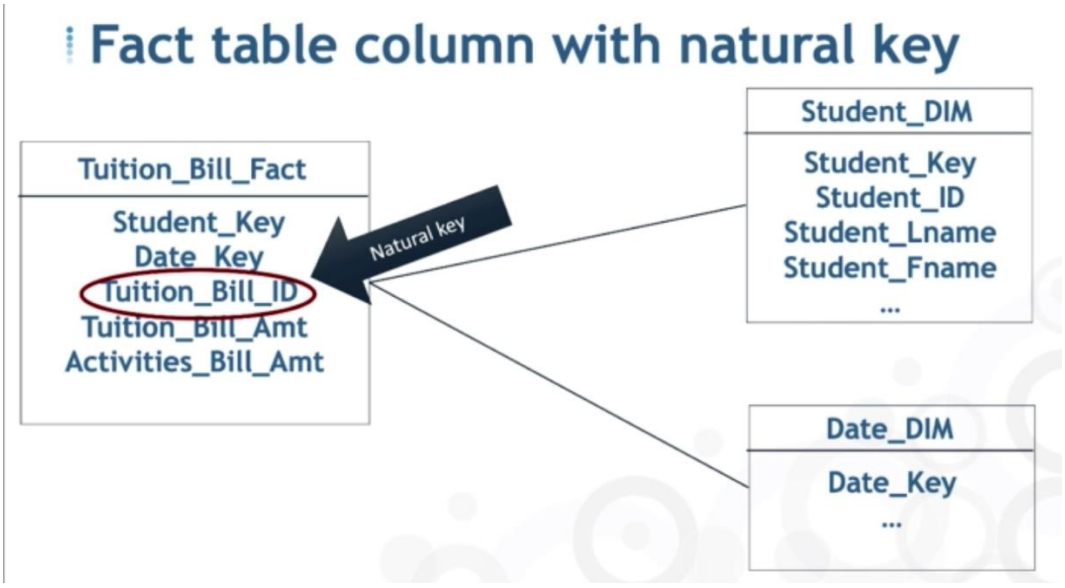
Primary Keys

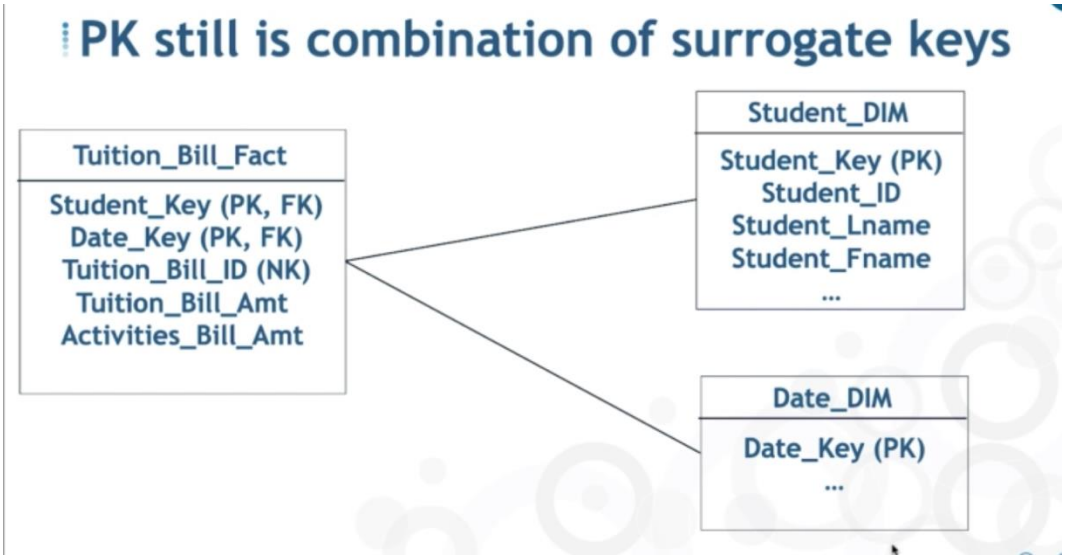


Also those 2 surrogates keys also a foreign keys.

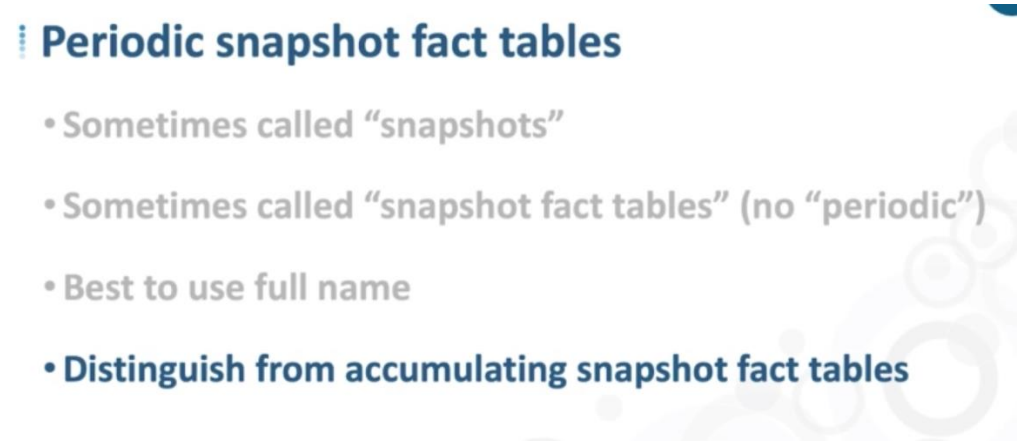


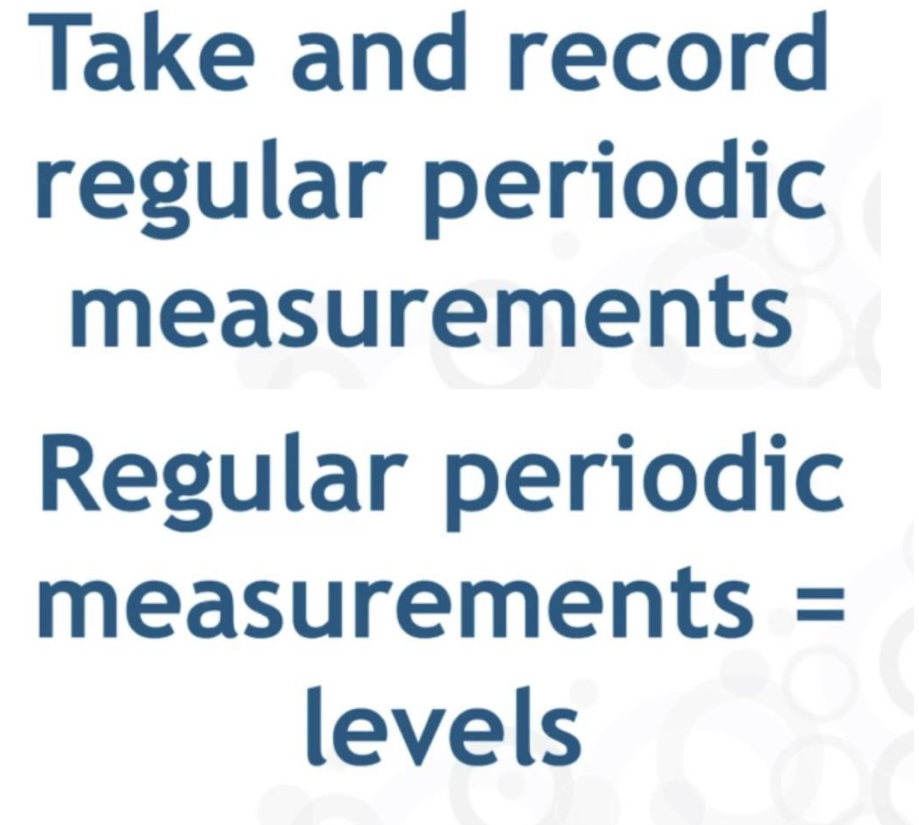
Combination of Student_Key and Date_Key are primary key for Tuition_Bill_Fact table.



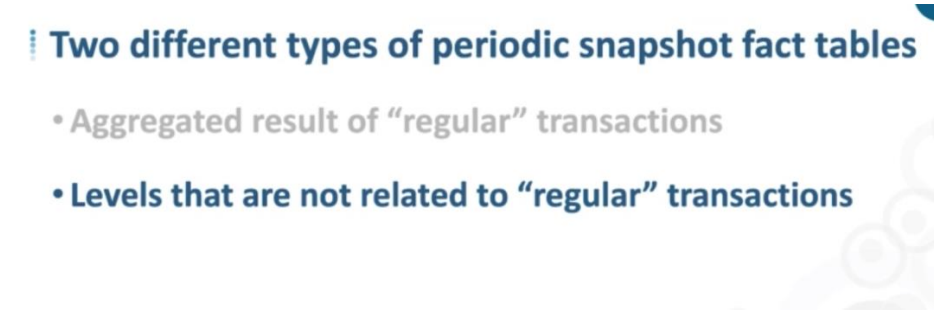


The Role of Periodic Snapshot Fact Table

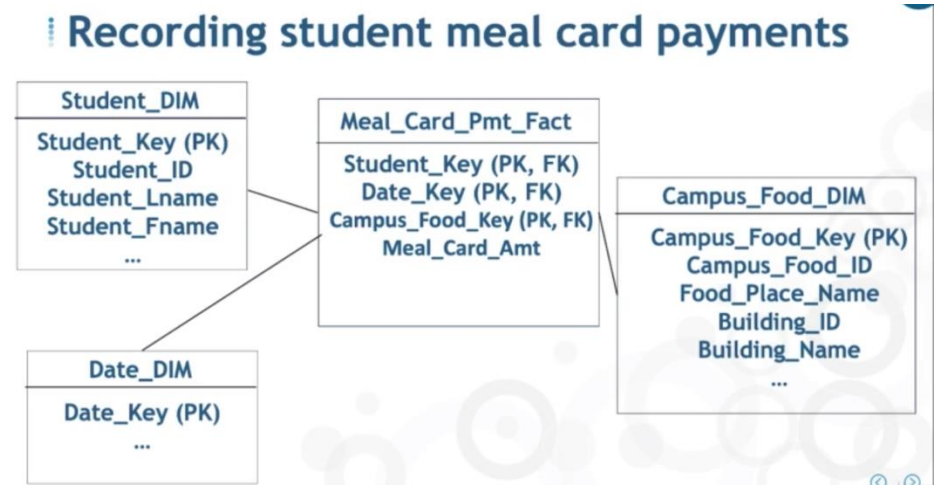


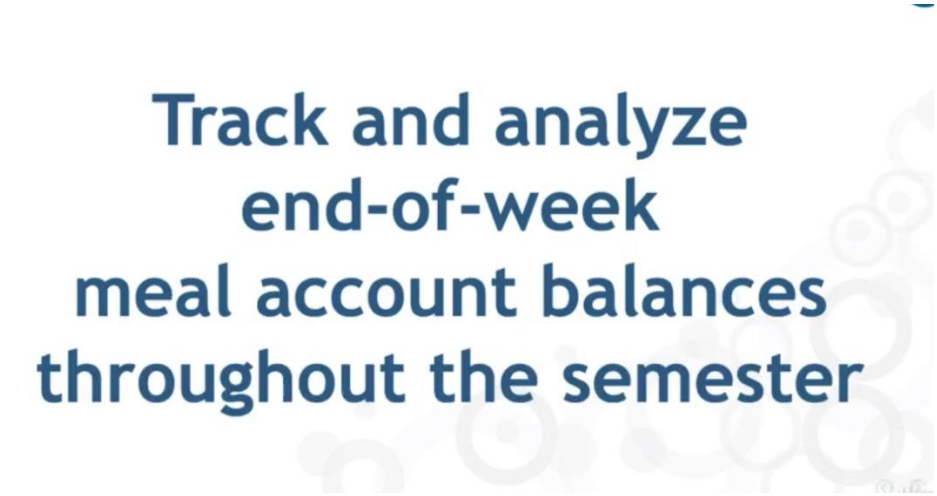


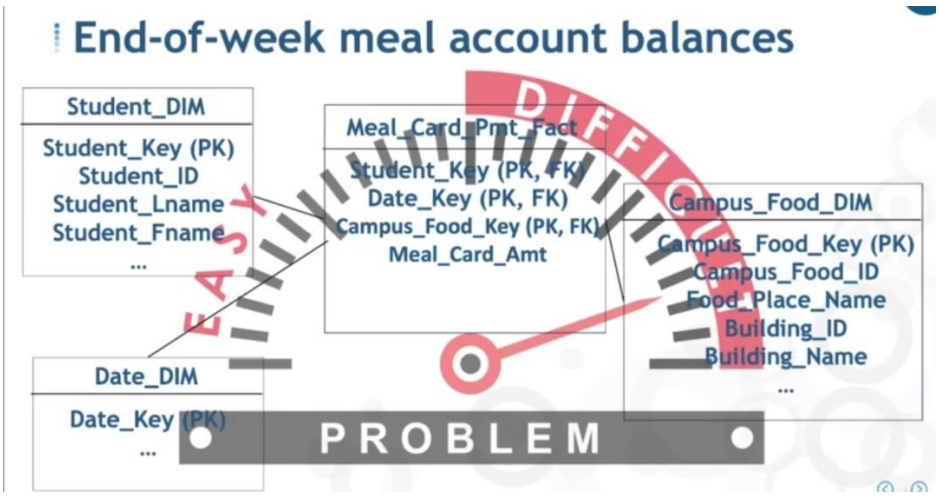
Two different types of periodic snapshot fact tables



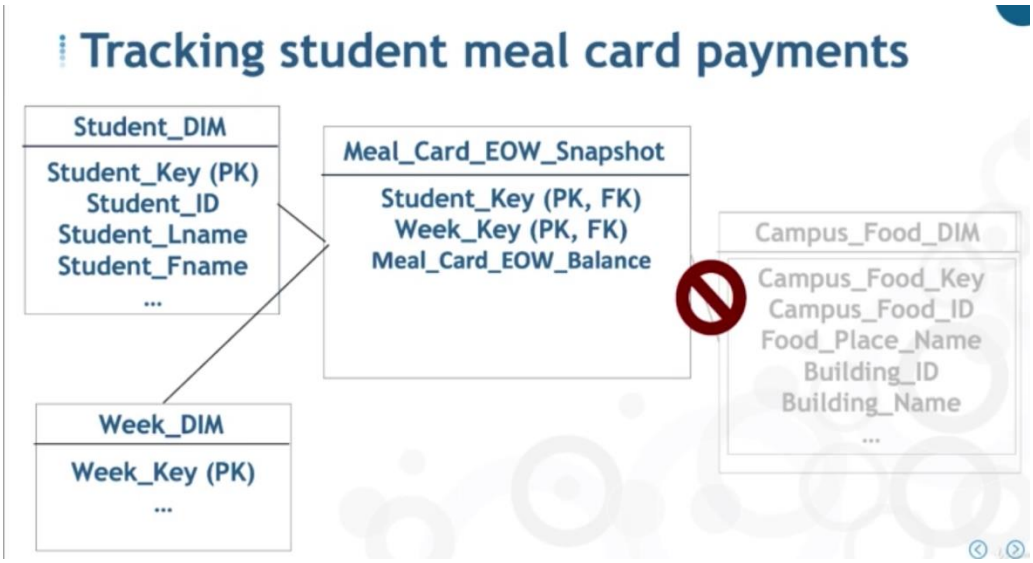
Recording student meal card payments







Keeping the record in regular transaction grain fact table is do-able. But it is not configured to support these kind of analysis. So, instead we can go with periodic snapshot fact table.



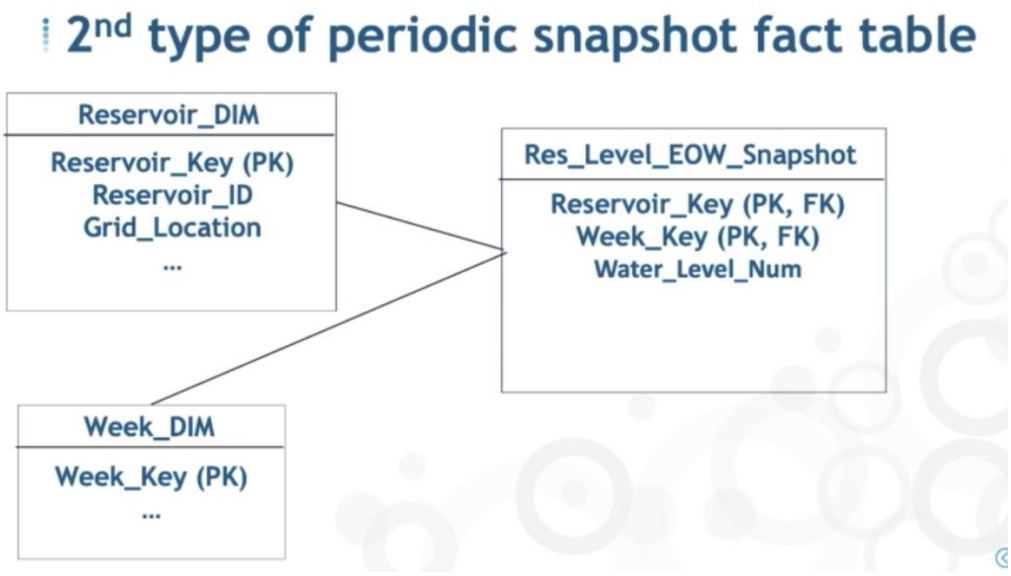
Sample snapshot fact table data

Meal_Card_EOW_Snapshot

Student_Key	Week_Key	Meal_Card_EOW_Balance
732017235	2234	$455.12
732017235	2781	$388.65
732017235	2865	$388.65
732017235	2567	$319.76
481011832	2234	$561.23
481011832	2781	$499.87
481011832	2865	$412.52
481011832	2567	$412.52
881838281	2234	$215.63
881838281	2781	$176.84
...		

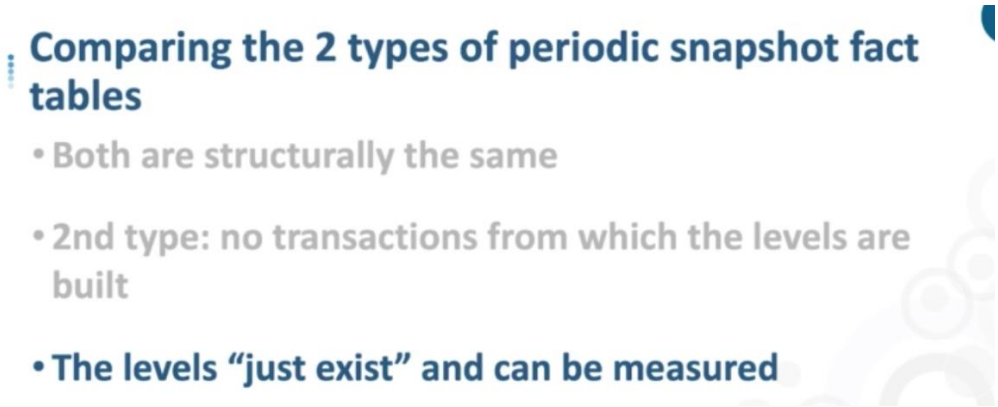
Much easier, more direct access for certain types of business questions

Second type of periodic snapshot fact table

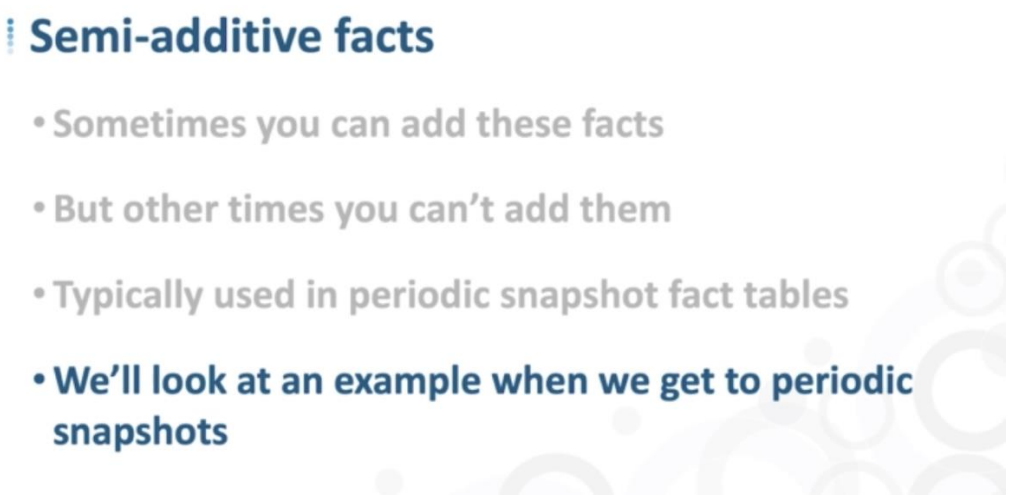


Here, we having only one fact which 'Water_Level_Num'. Other than this, there won't be any other facts.

Comparing the two 2 types of periodic snapshot fact tables



Periodic Snapshot Fact Tables and Semi Additive Facts



What was the ending meal card balance for Sally Jackson on any given week?

Sample snapshot fact table data

Meal_Card_EOW_Snapshot

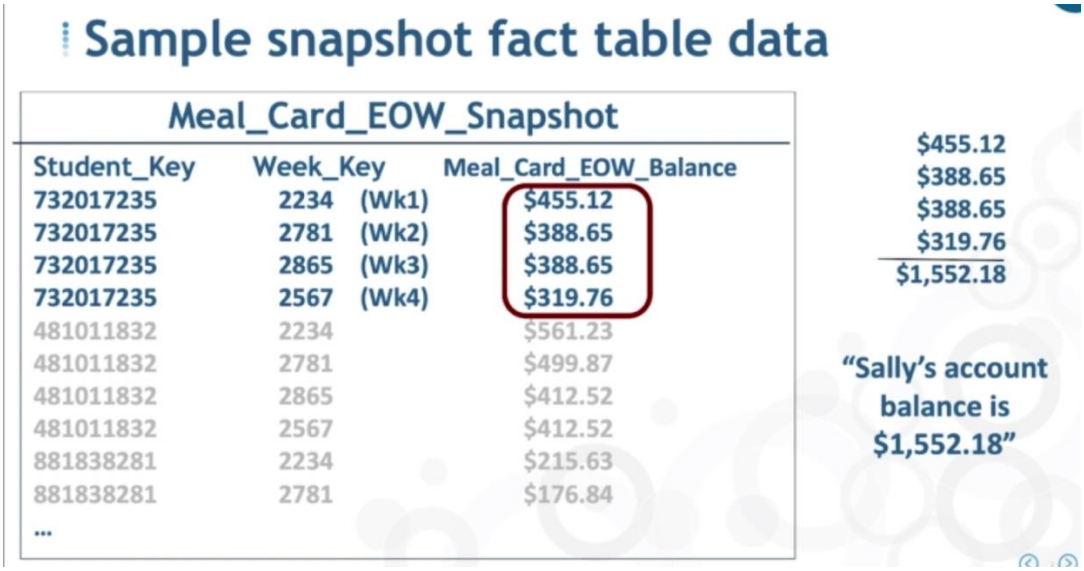
Student_Key	Week_Key	Meal_Card_EOW_Balance
732017235	2234	$455.12
732017235	2781	$388.65
732017235	2865	$388.65
732017235	2567	$319.76
481011832	2234	$561.23
481011832	2781	
481011832	2865	
481011832	2567	
881838281	2234	
881838281	2781	
...		

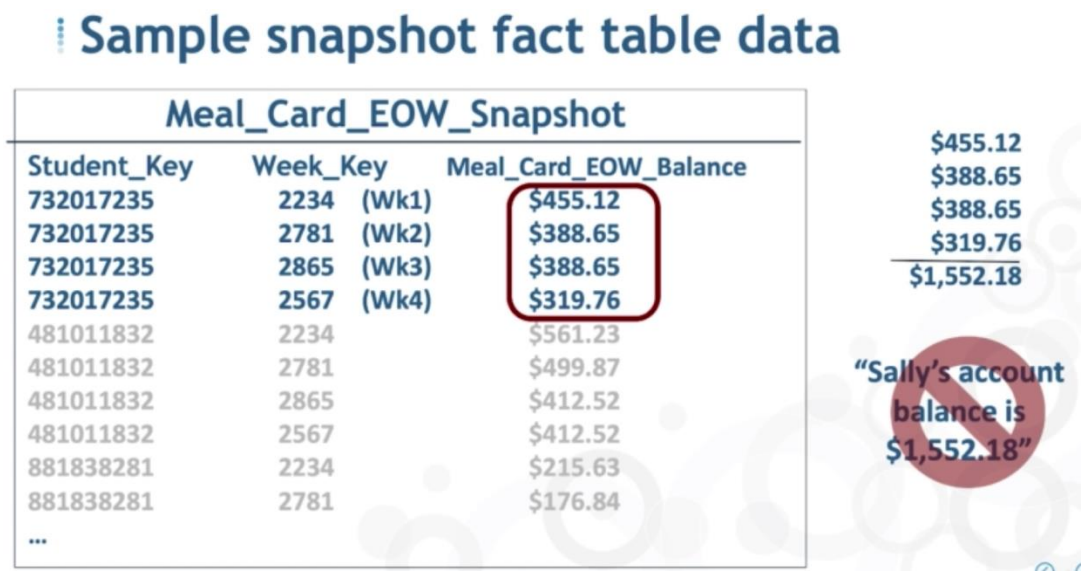
Student_DIM

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

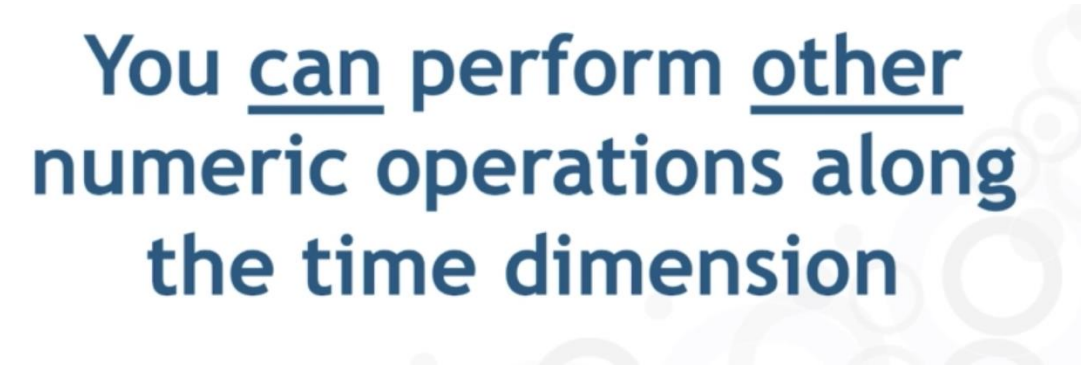
Meal_Card_EOW_Balance

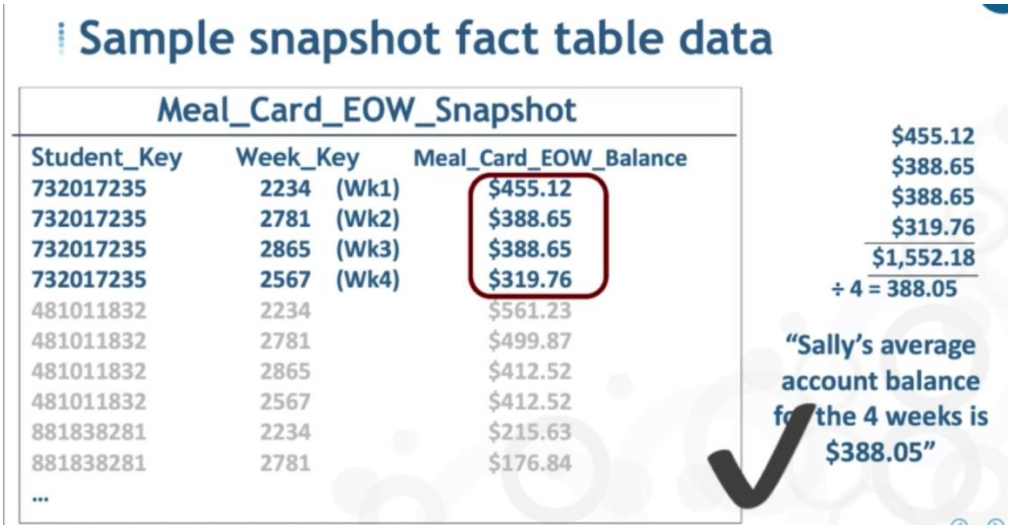
- Definitely a fact (measurement)
- Unlike a fact in a transaction-grained fact table...
- **Can't add it along the time dimension**

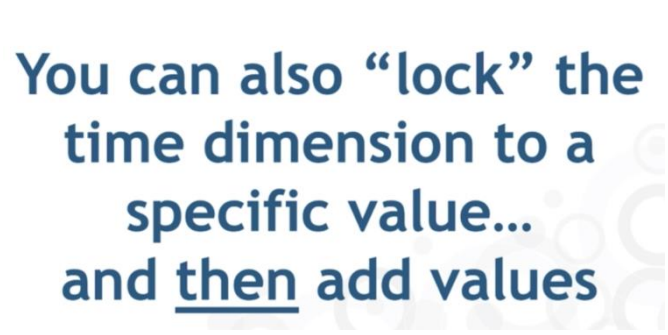


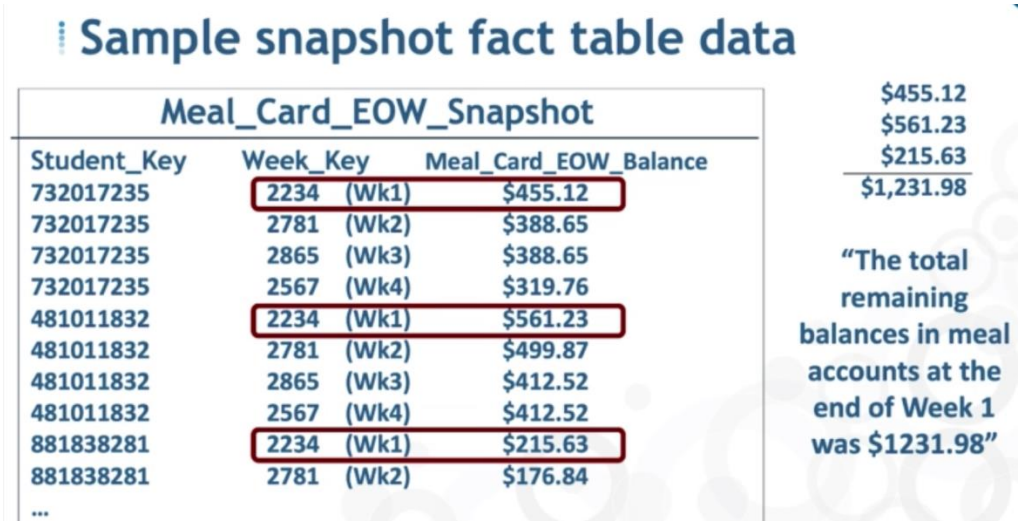


Here, we are trying to add all four balance. But that is wrong. This called semi additive.









This is perfectly fine. Here, we are locking up the week 1 time.

Assignments

Questions for this assignment

1. Design a transaction-grained fact table and the associated dimension tables to meet the following requirement:

   1) The university has many different locations around campus that serve meals to students using their meal cards

   2) We want to be able to analyze meal purchases by student, by date, and by location

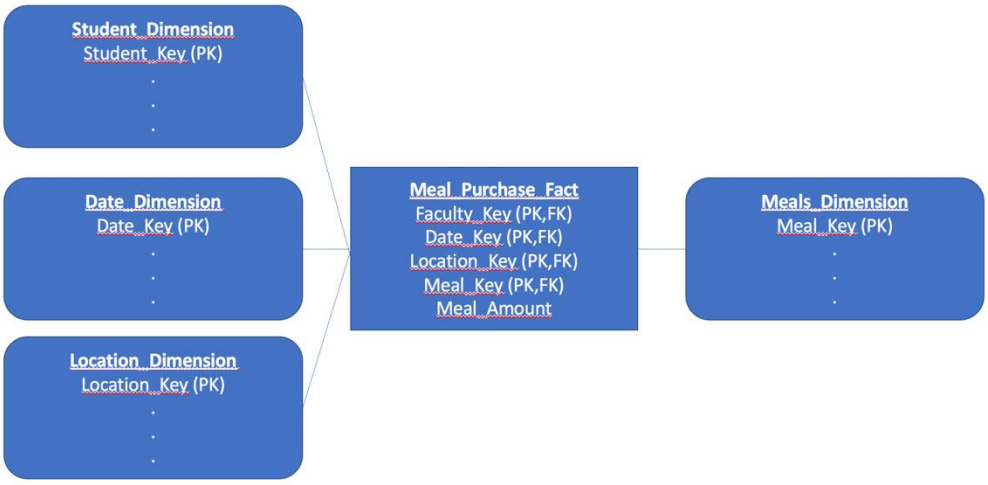
2. Design a periodic snapshot fact table and the associated dimension tables to meet the following requirement:

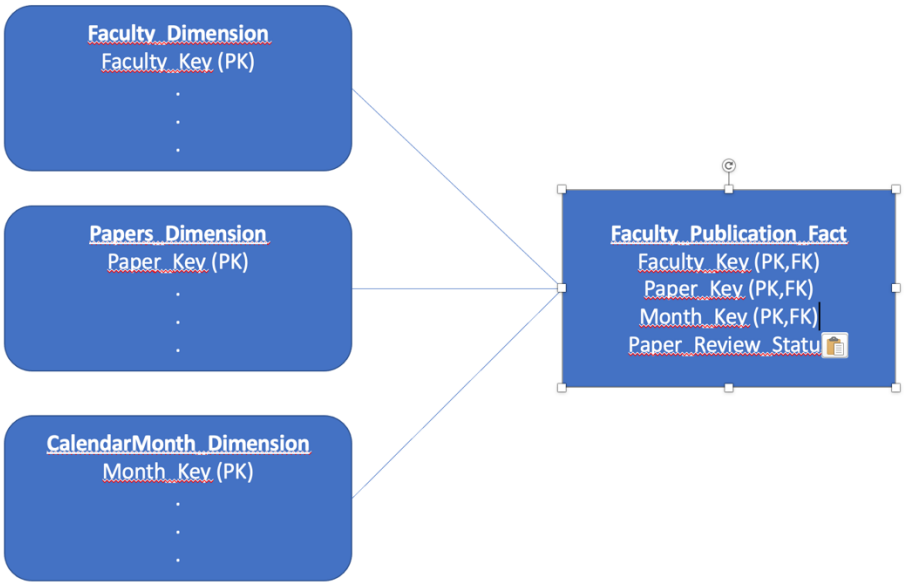
   1) An important part of a faculty member's position is publishing papers in academic journals

   2) A faculty member will have, at any given point in time, some papers going through initial review; some papers going through final review; other papers that have been approved for publication but not published yet; and other papers that have been published

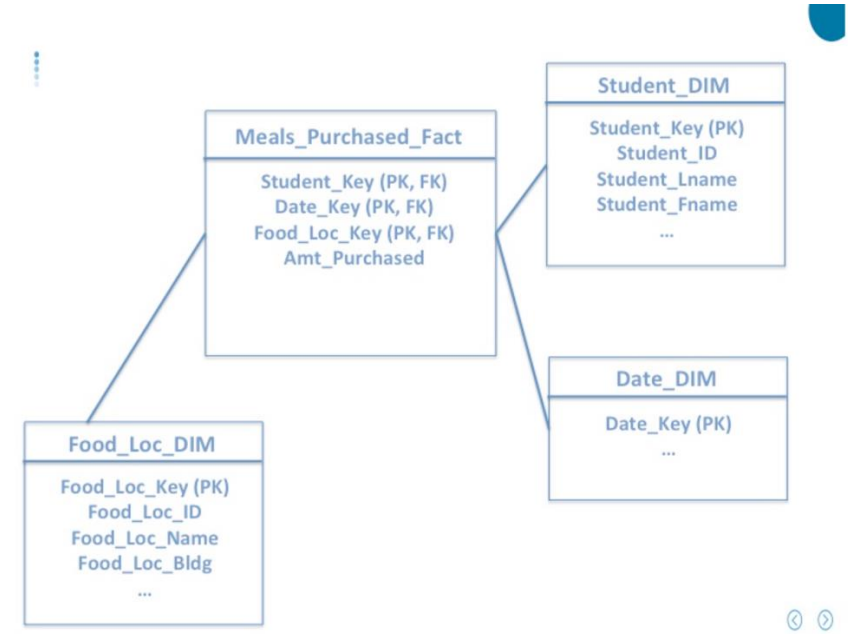
   3) The university administration office conducts analysis at the end of each calendar month about faculty publications...specifically, how many papers each faculty member has at various stages at each "snapshot point in time"

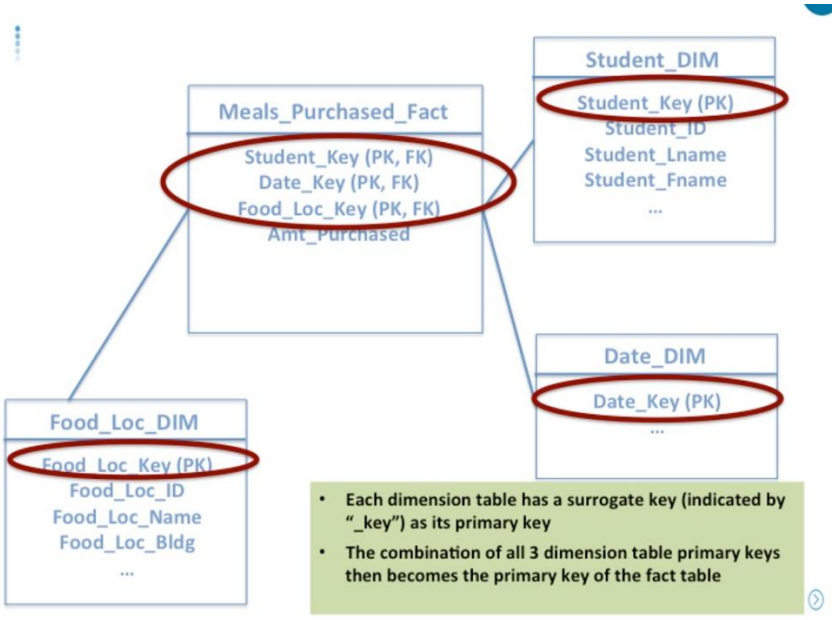
Submitted:

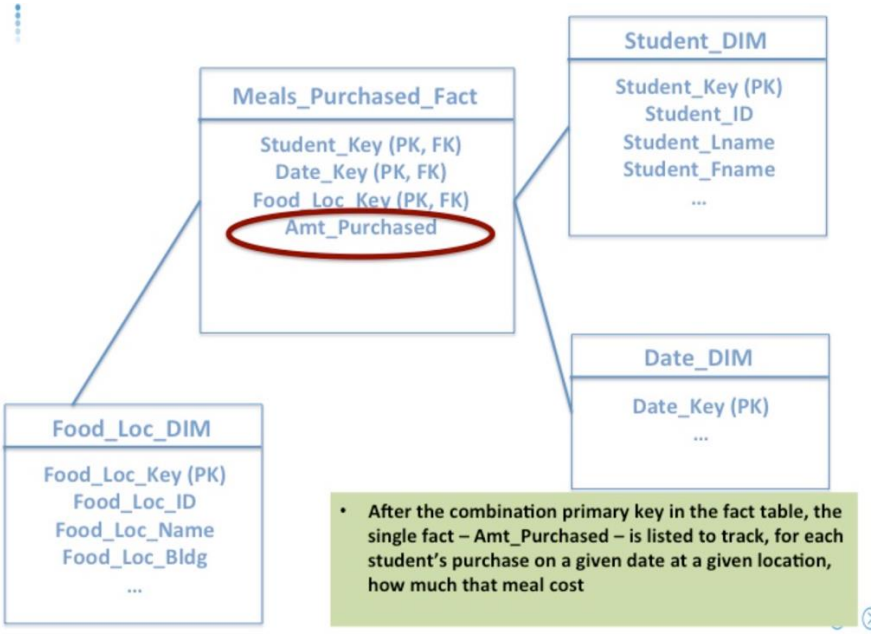




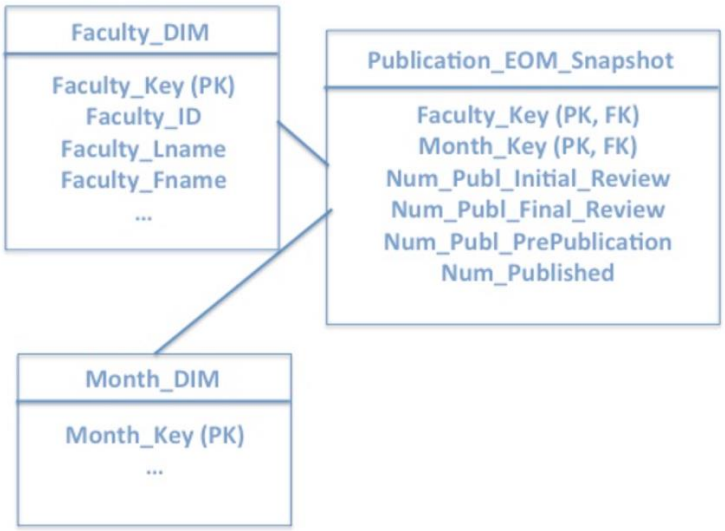
Authors Solution:

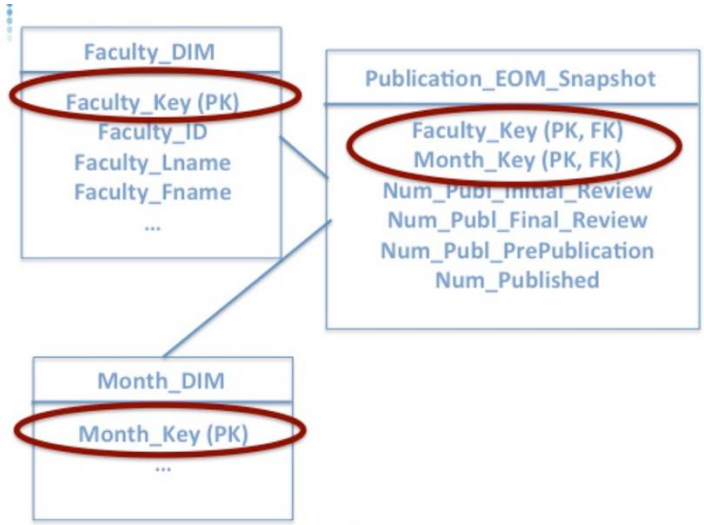




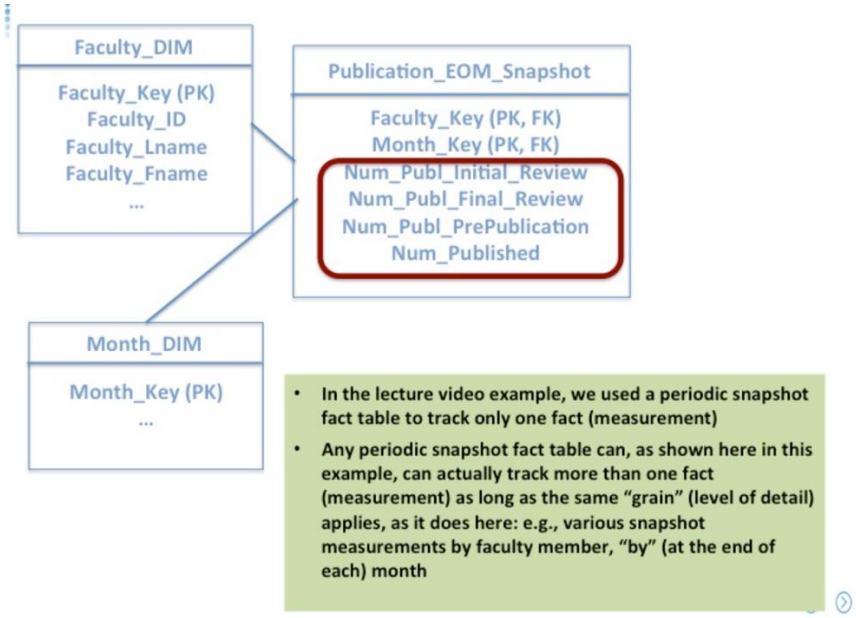


- **After the combination primary key in the fact table, the single fact – Amt_Purchased – is listed to track, for each student's purchase on a given date at a given location, how much that meal cost**

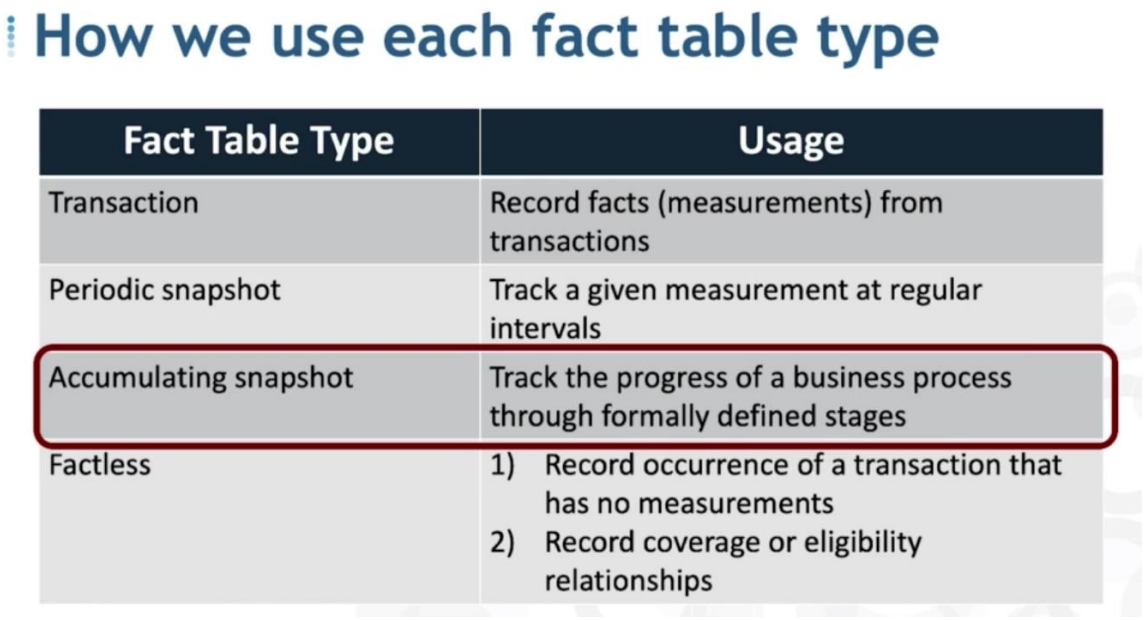




- **Each dimension table has a surrogate key (indicated by "_key") as its primary key**
- **The combination of both dimension table primary keys then becomes the primary key of the fact table**

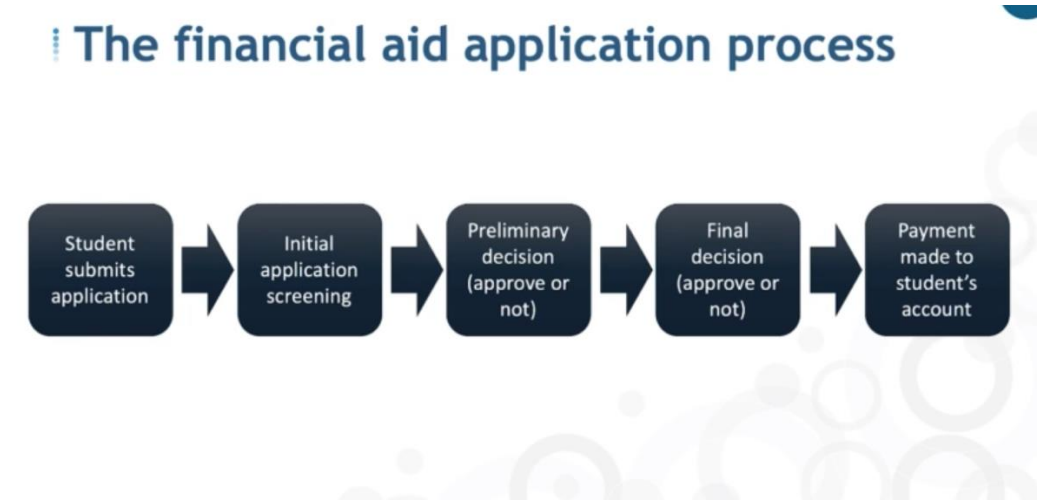


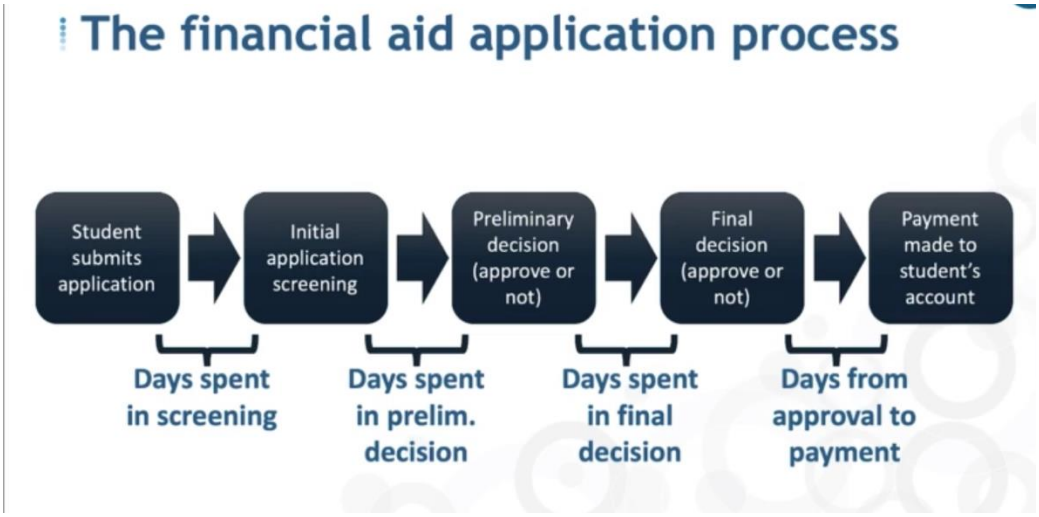
The Role of Accumulating Snapshot Fact Table

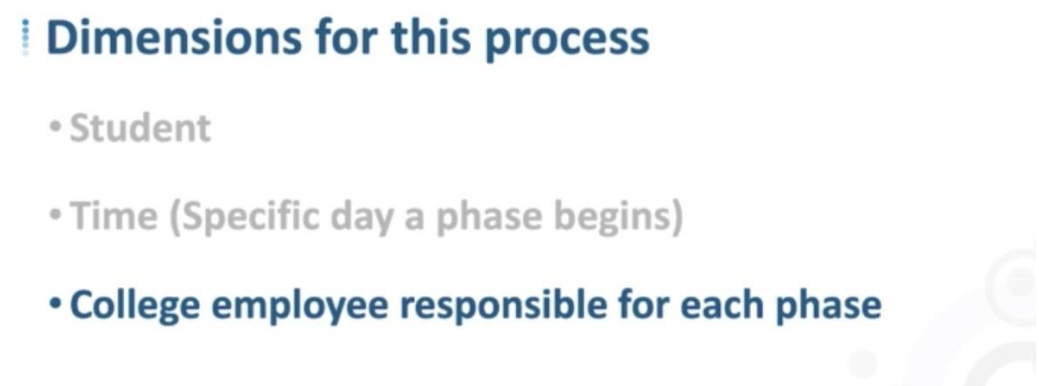


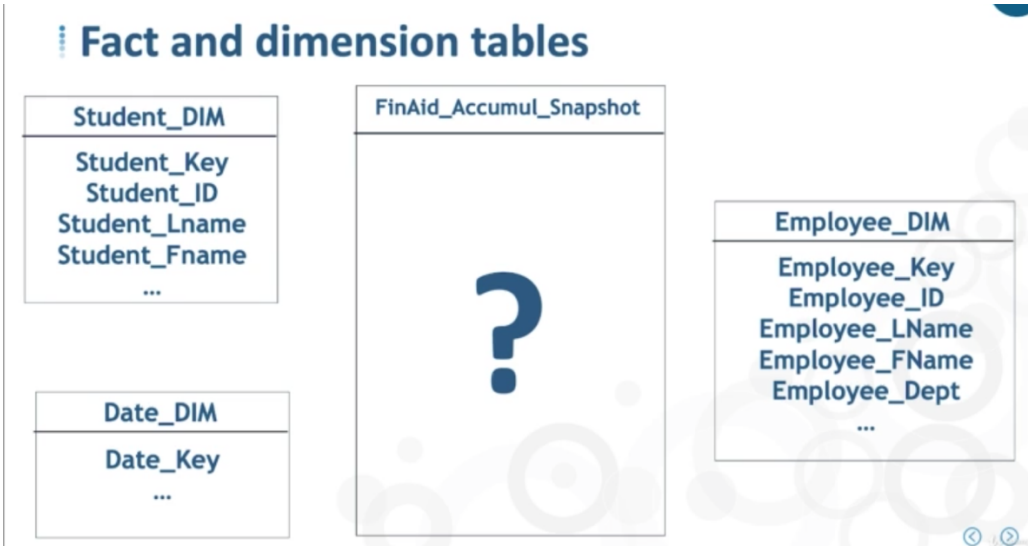
Student financial aid application process

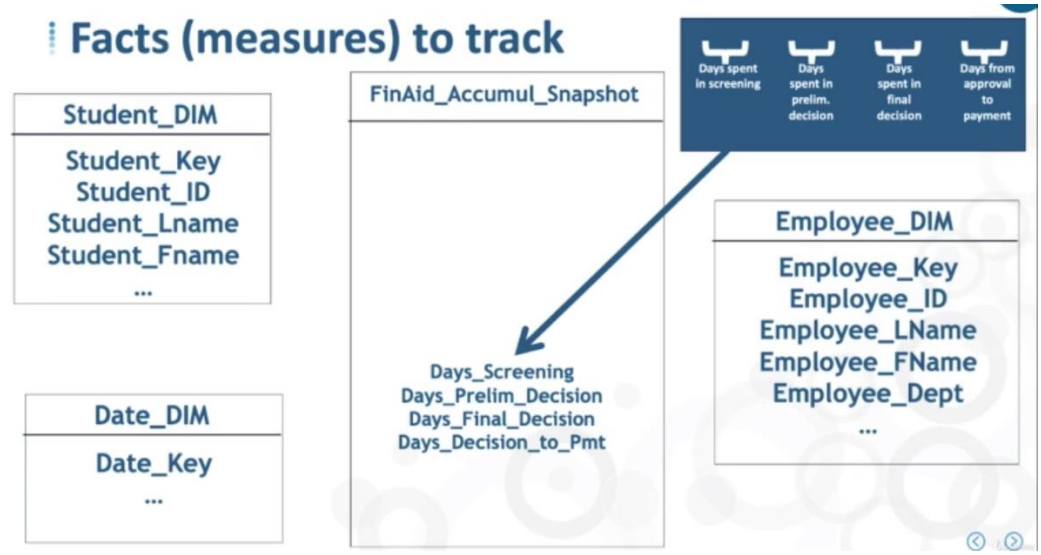
The financial aid application process

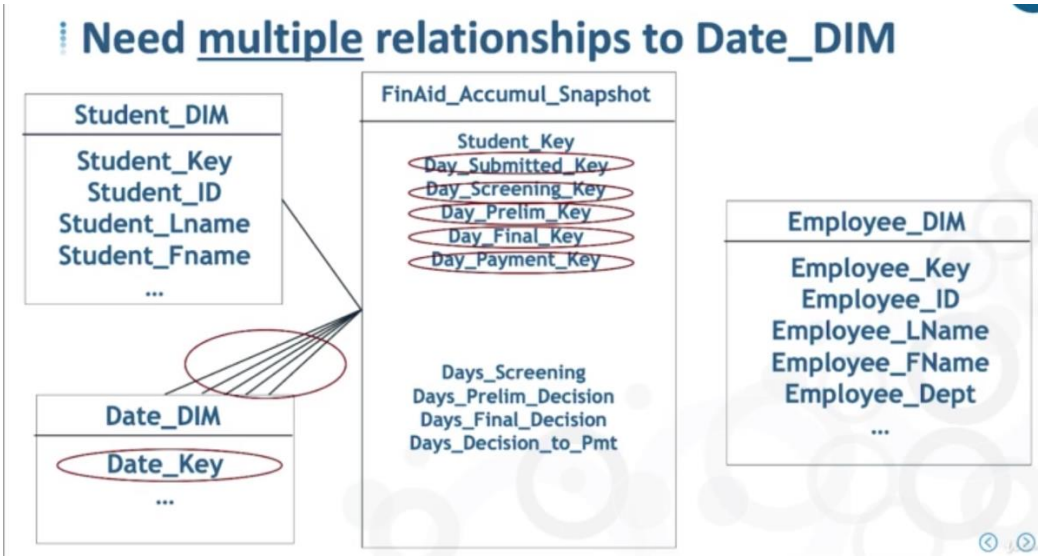








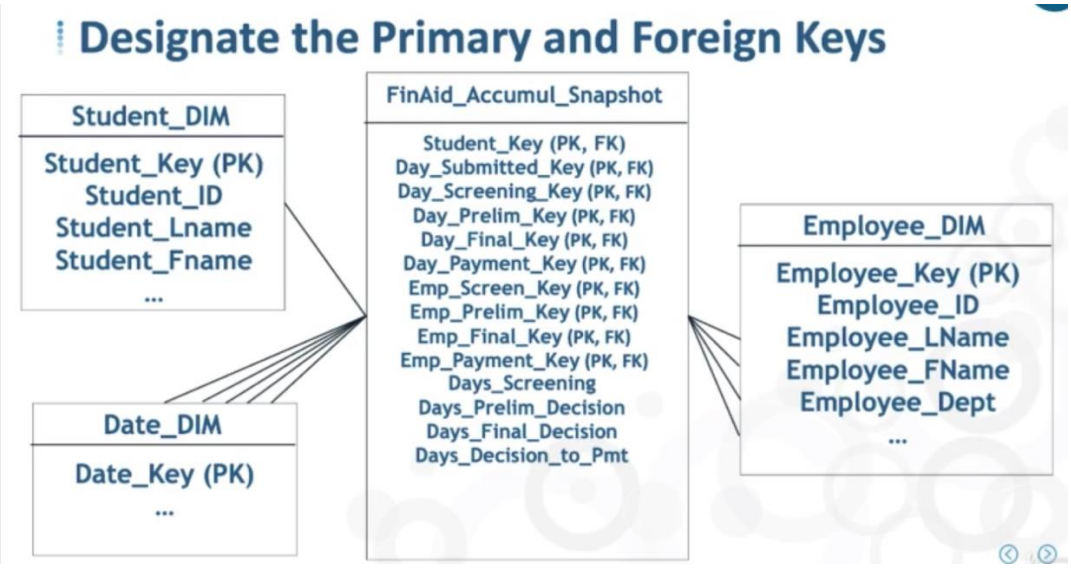




Here, we are getting multiple relationship from fact table back to single dimension table.

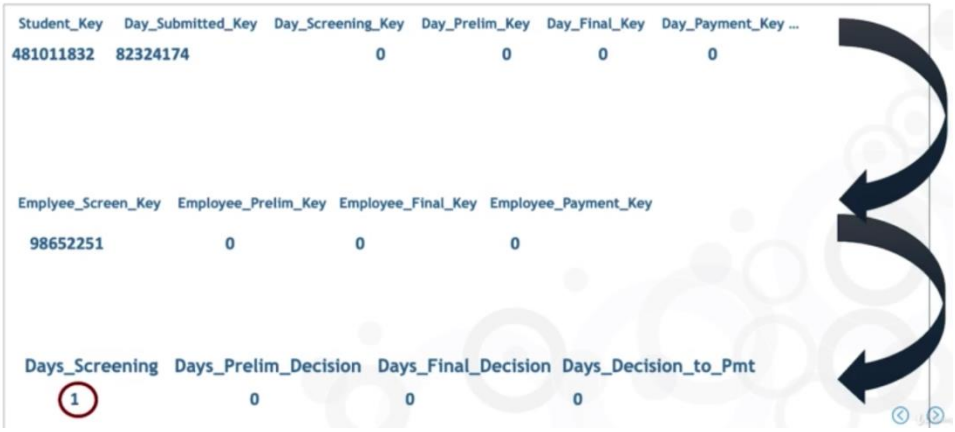
Because, we need to track the date elapsed on different business processes like submitted, screening, prelim, final and payment.

Designate the Primary and Foreign Keys

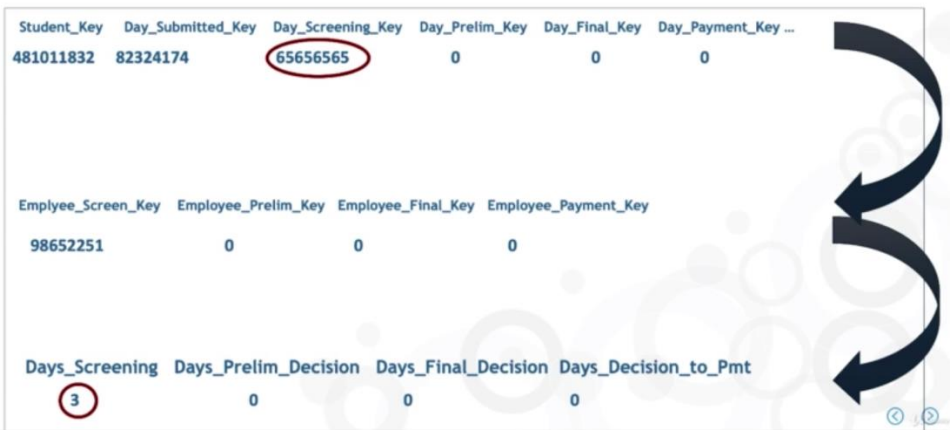


Richard Thompson applies for $8,000 in financial aid on 8/11/2020

Day 1: New application



Day 3: Screening completed

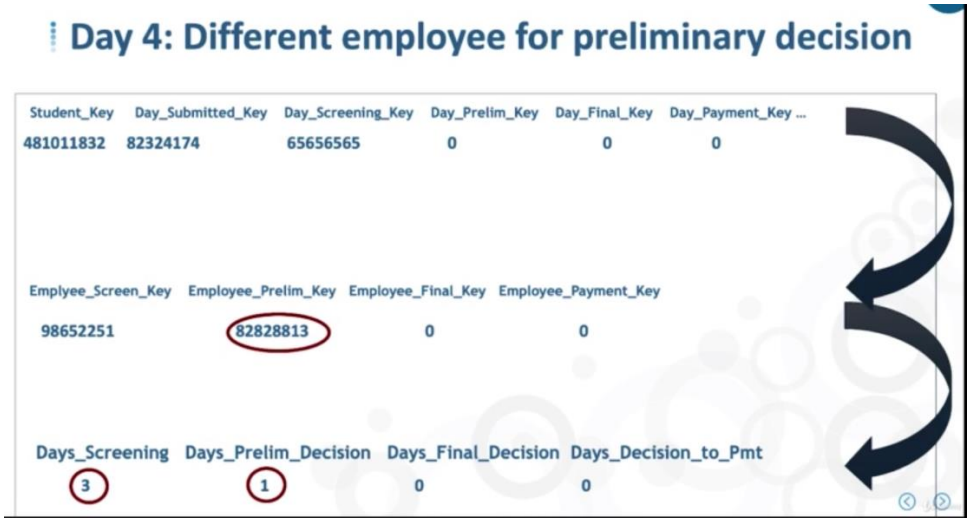


Day 4: Different employee for preliminary decision

Student_Key	Day_Submitted_Key	Day_Screening_Key	Day_Prelim_Key	Day_Final_Key	Day_Payment_Key ...
481011832	82324174	65656565	0	0	0

Emplyee_Screen_Key	Employee_Prelim_Key	Employee_Final_Key	Employee_Payment_Key
98652251	82828813	0	0

Days_Screening	Days_Prelim_Decision	Days_Final_Decision	Days_Decision_to_Pmt
3	1	0	0

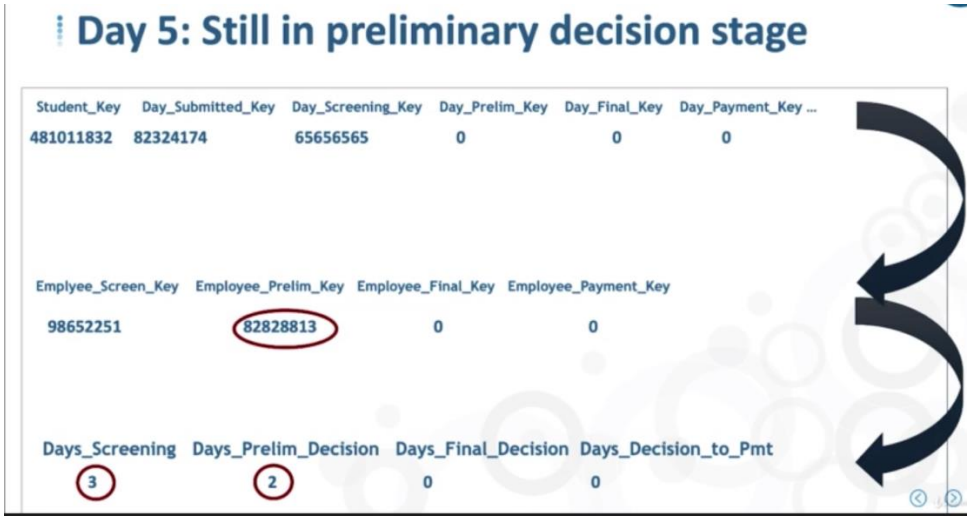


Day 5: Still in preliminary decision stage

Student_Key	Day_Submitted_Key	Day_Screening_Key	Day_Prelim_Key	Day_Final_Key	Day_Payment_Key ...
481011832	82324174	65656565	0	0	0

Emplyee_Screen_Key	Employee_Prelim_Key	Employee_Final_Key	Employee_Payment_Key
98652251	82828813	0	0

Days_Screening	Days_Prelim_Decision	Days_Final_Decision	Days_Decision_to_Pmt
3	2	0	0



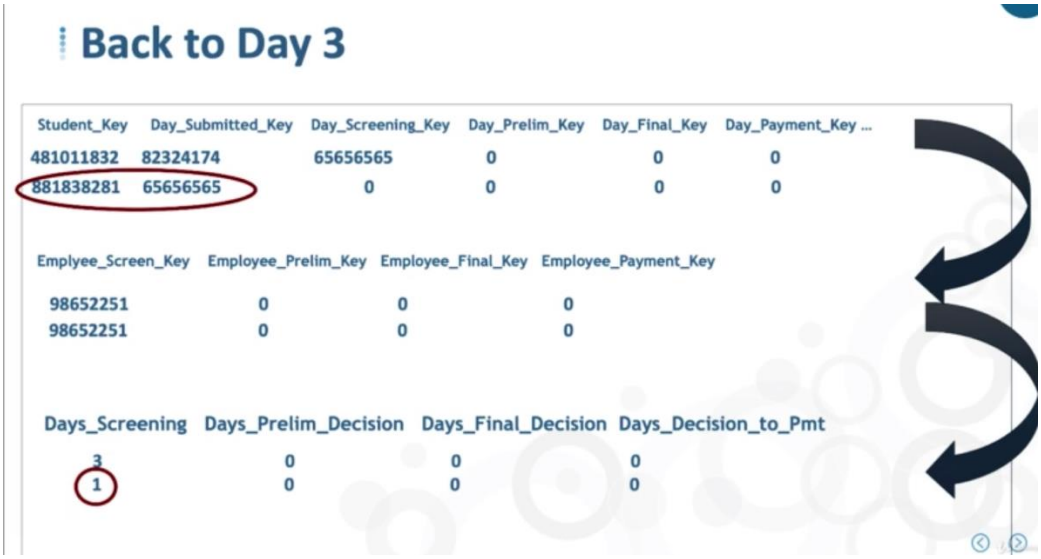
Financial aid application approved

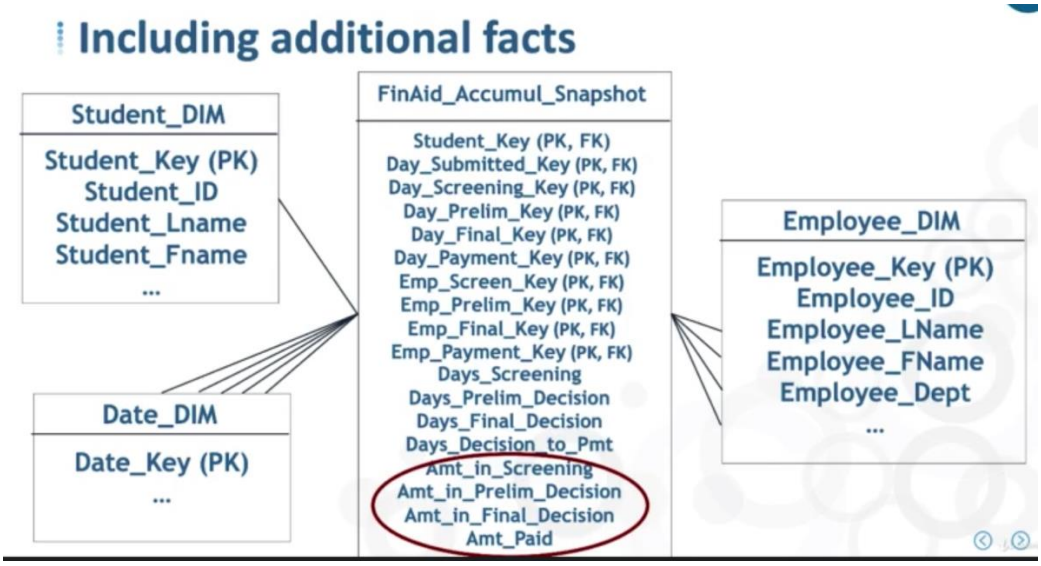
Student_Key	Day_Submitted_Key	Day_Screening_Key	Day_Prelim_Key	Day_Final_Key	Day_Payment_Key ...
481011832	82324174	65656565	75711144	88913621	43422255

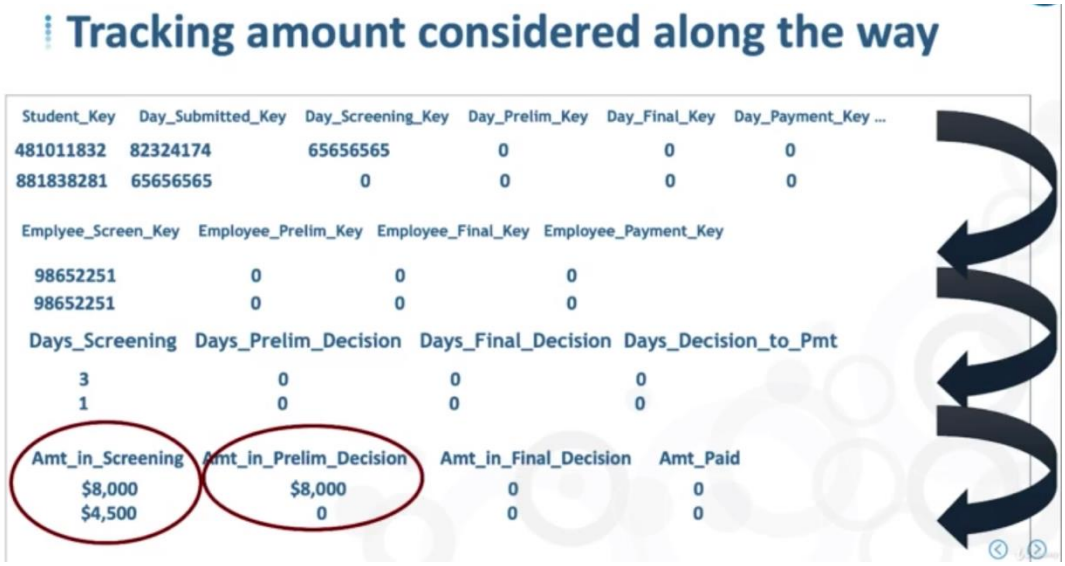
Emplyee_Screen_Key	Employee_Prelim_Key	Employee_Final_Key	Employee_Payment_Key
98652251	82828813	762224242	1282176

Days_Screening	Days_Prelim_Decision	Days_Final_Decision	Days_Decision_to_Pmt
3	4	3	6

Another student applied for a financial aid.



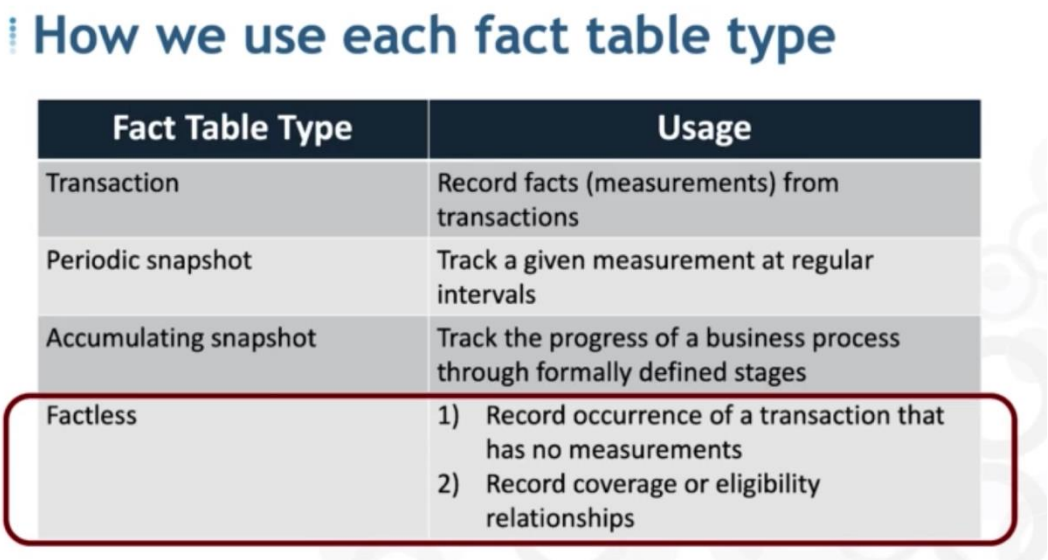




From this example, We can see

- Screen, Prelim, Final are Payment are business process.
- We are measuring and tracking the date, employee and amount.
- Business process is periodic snapshot.

Why a Factless Fact Table is not a Contradiction in Terms



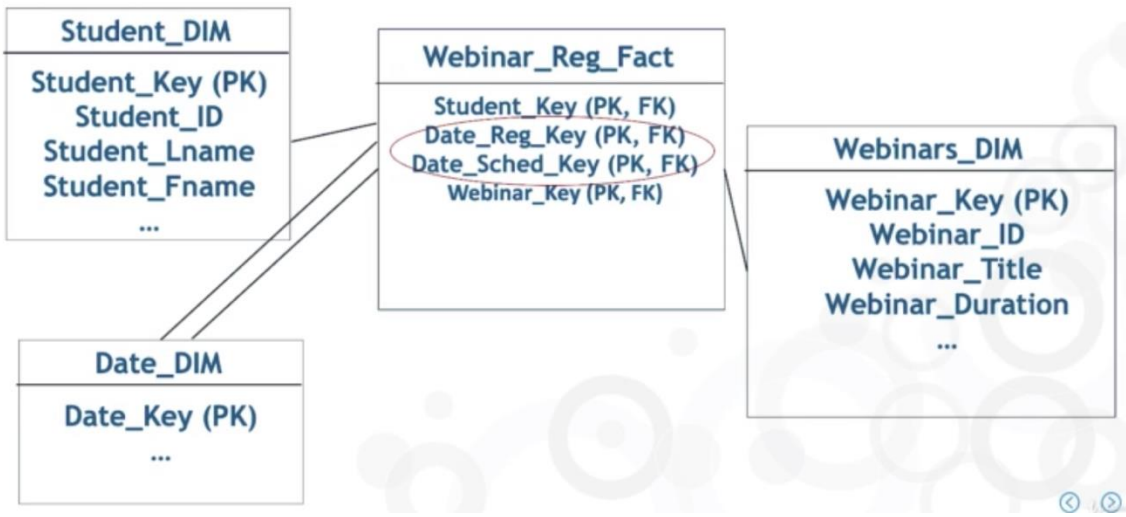
Factless Fact Table

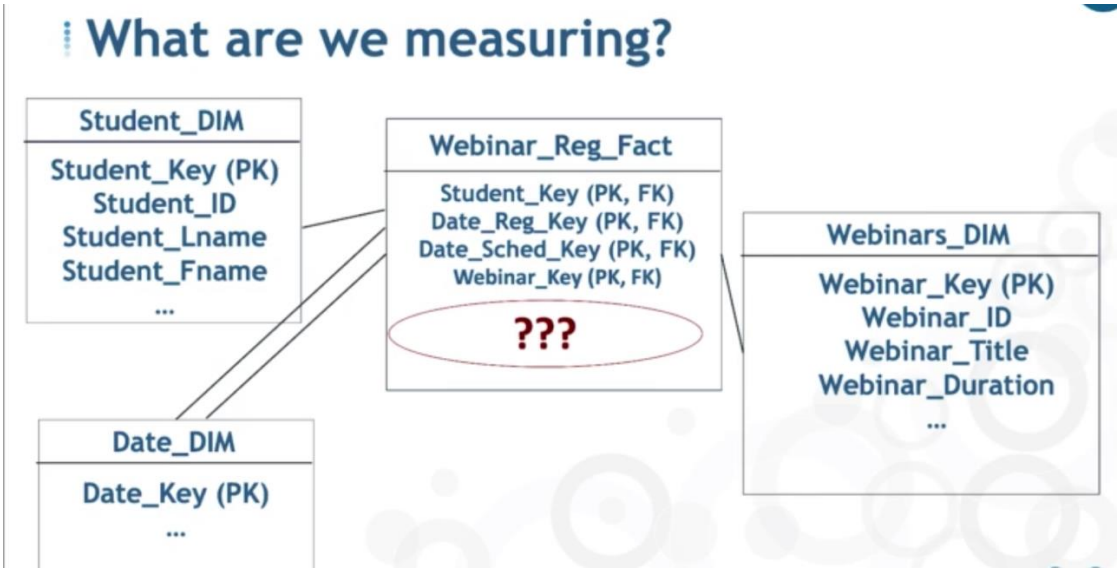
Students can register for online webinars throughout the semester

We want to track

- Which students register
- For which webinars
- The date each student registers for a given webinar
- **The scheduled date for each webinar**

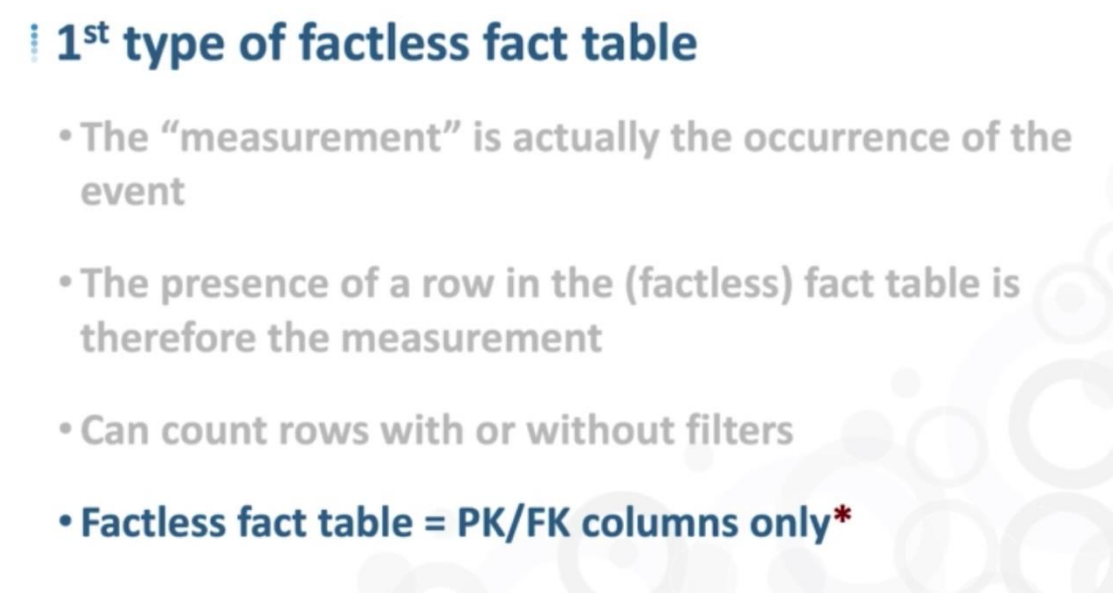
Recording registrations in a fact table

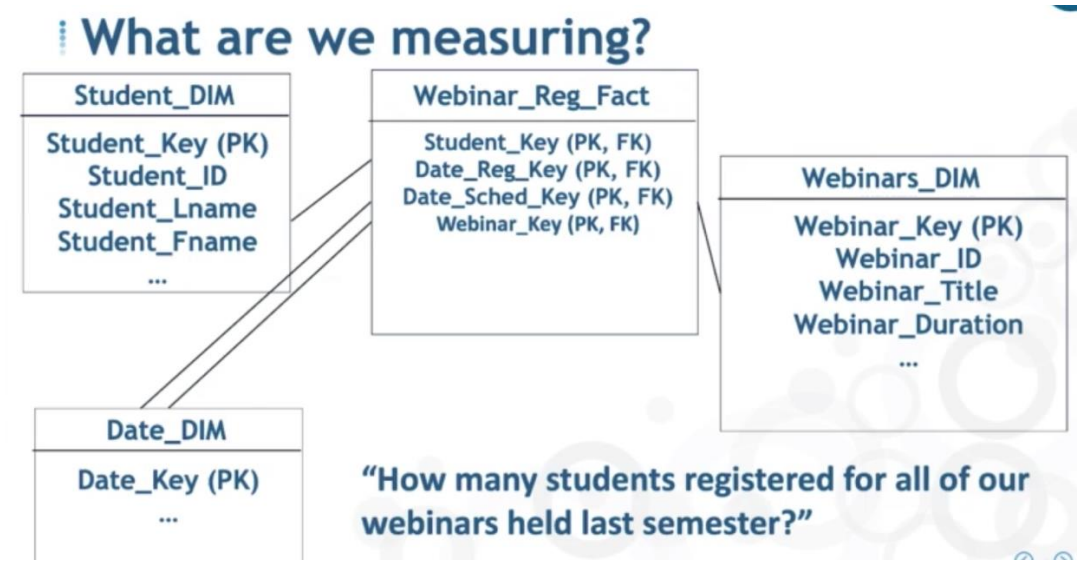


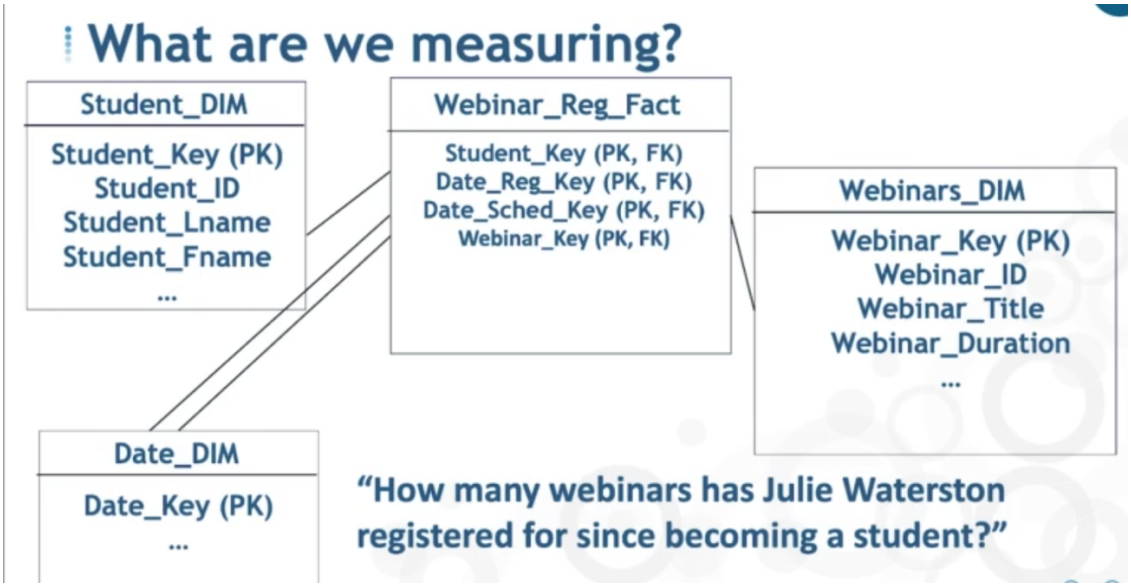


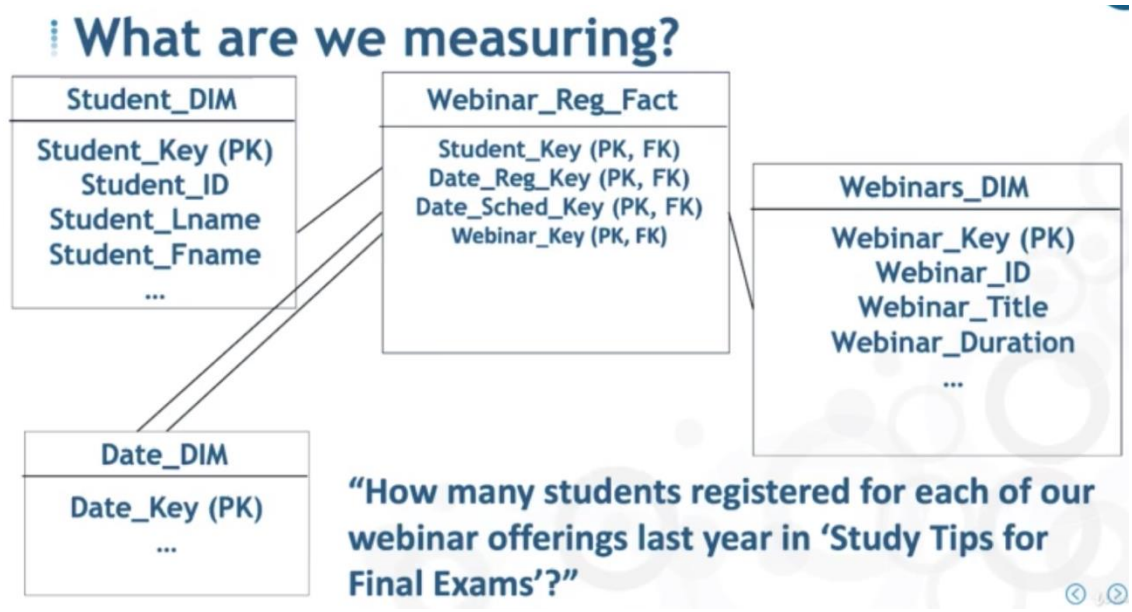
Here we don't have an any measurement, but it tracks the event. This is the first type of factless fact table.

We are keeping track of two dates in our fact table. The One when webinar is registered. And the other when webinar is held.

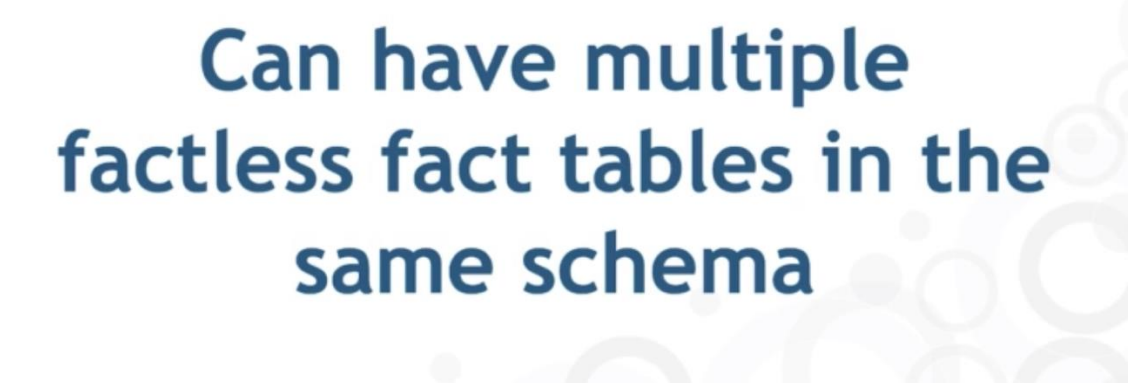




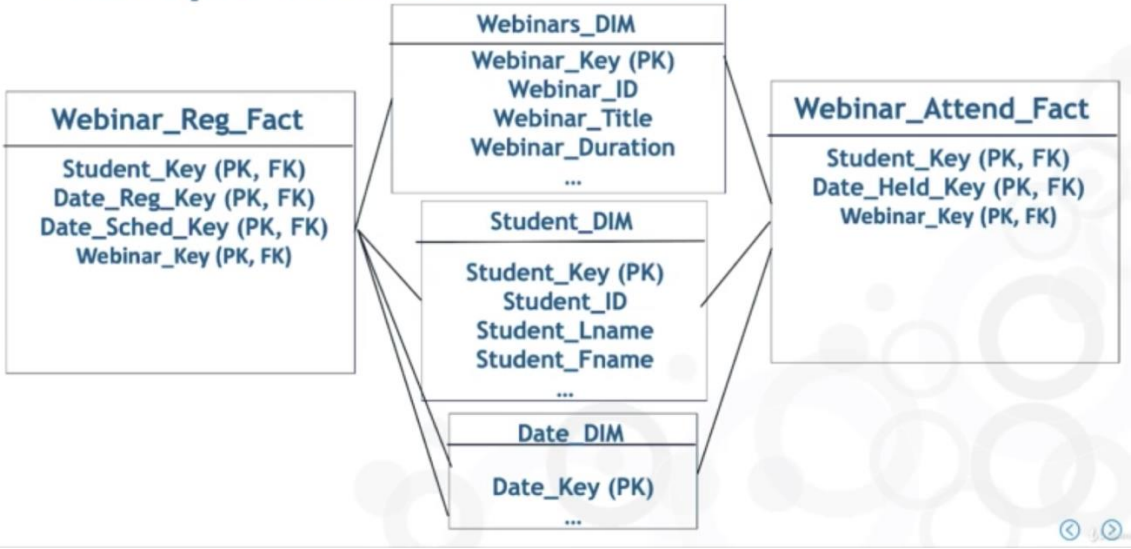




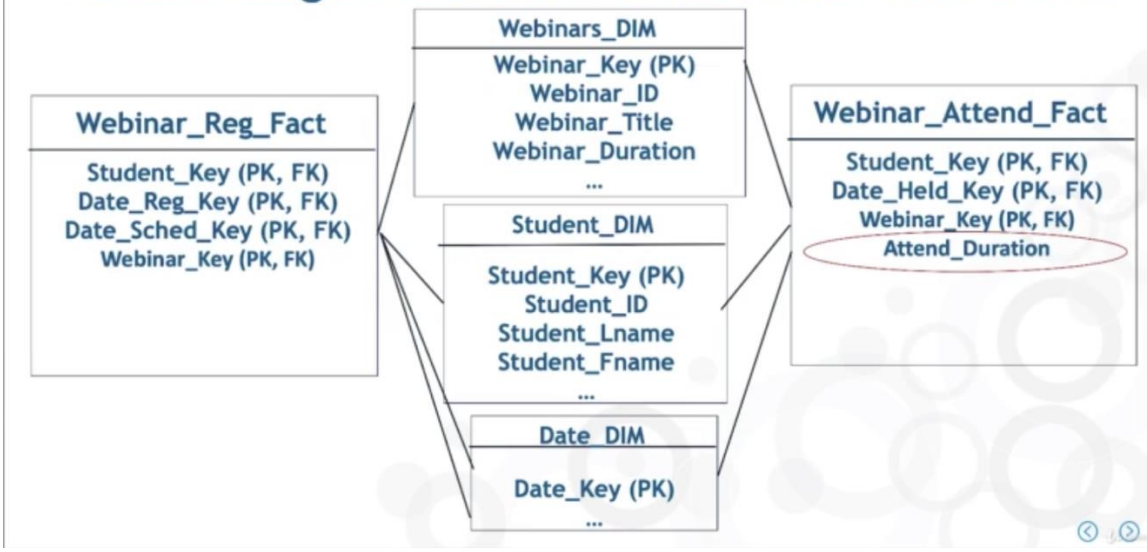
We are getting a measurement by applying the filter and counting the rows.



Multiple factless fact tables



Combining factless and transaction fact tables



Sometimes use a “tracking fact”

"Tracking fact"

- Value always = 1
- Allows you to use SQL SUM() rather than COUNT()
- More "readable and understandable" SQL...
- **Might be less efficient execution**

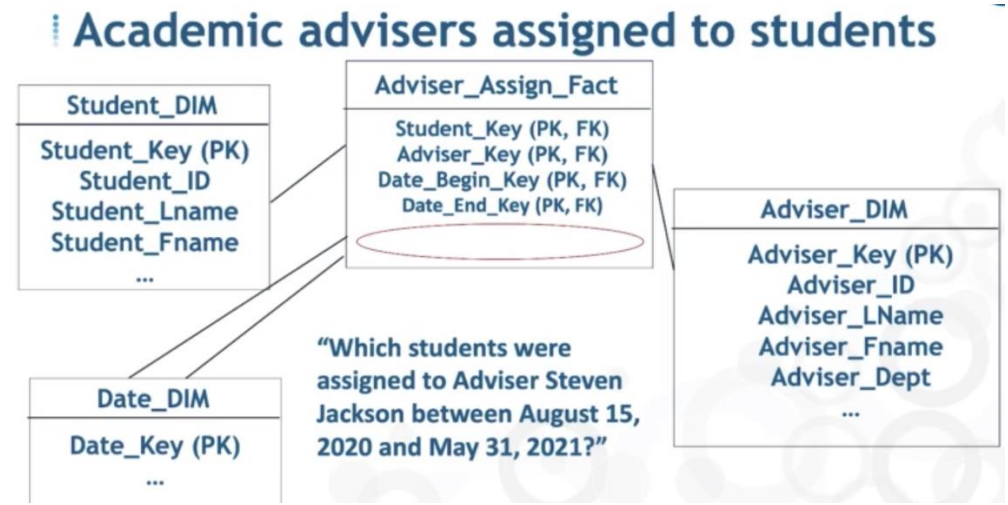
Second Type of Factless fact table

2nd type of factless fact table

- Recording a particular relationship or association among multiple parties
- Even if no transactions actually occur
- **Typically (but not always) between a starting and ending date or time**

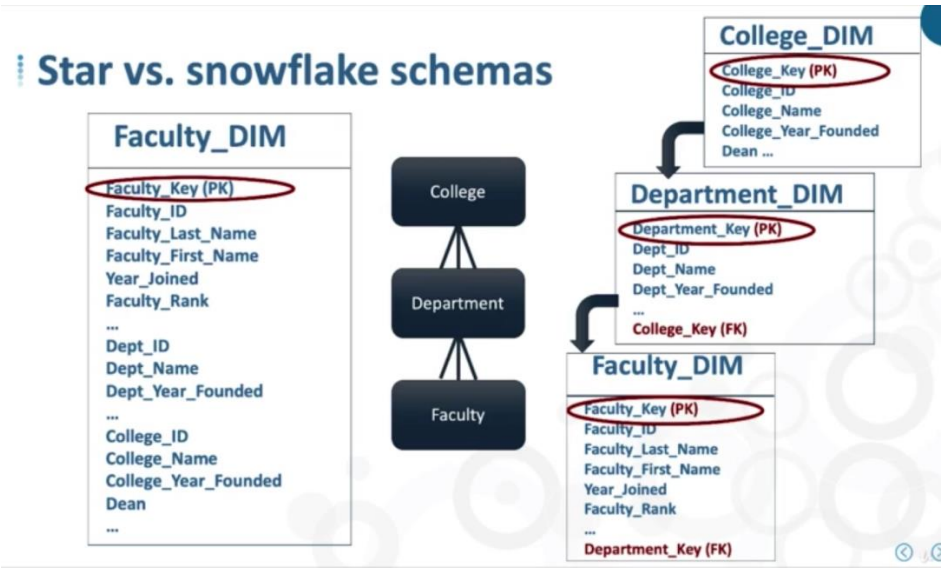
Advisers assigned to students

- If a student visits an adviser, we could use our first type of factless fact table to record that occurrence
- If we record duration of visit, could use a regular transaction-grained fact table
- What if a student never visits his/her adviser?
- **Still want record of the assignment**

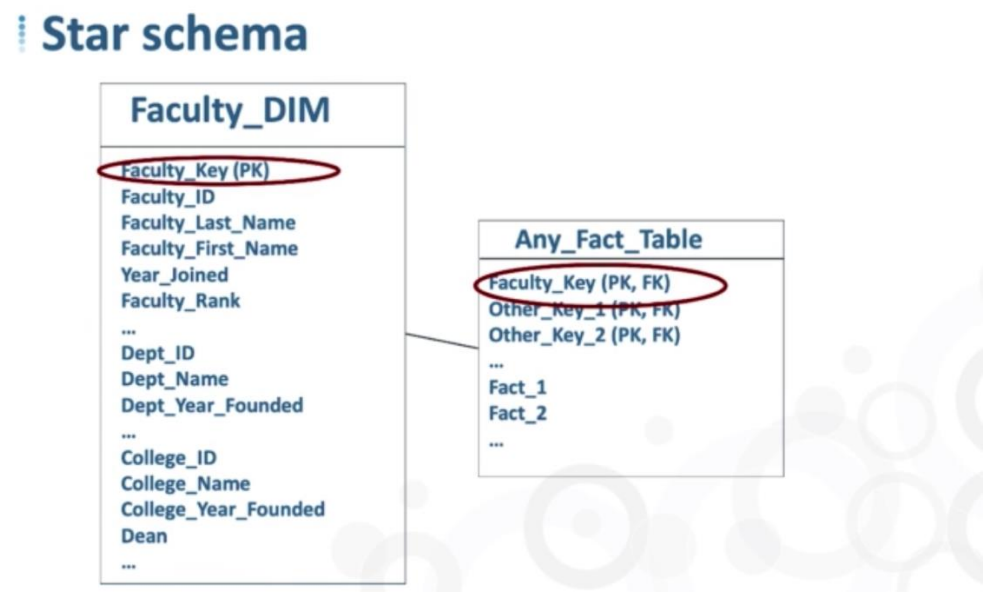


Compare the structure of Fact Tables in Star Schemas vs Snowflake Schemas

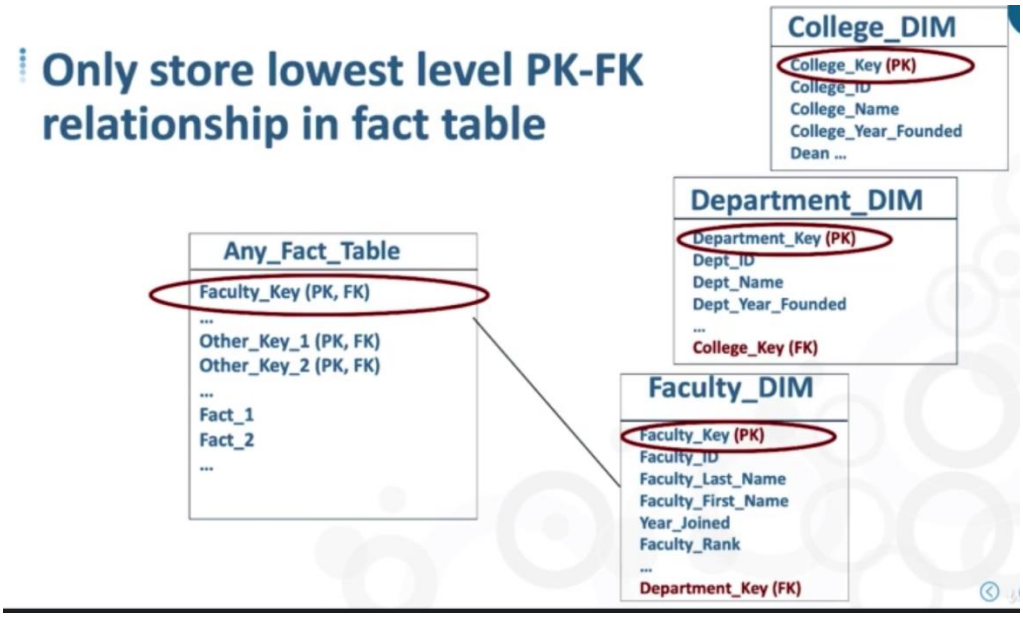
Recalling Dimension Table Comparison

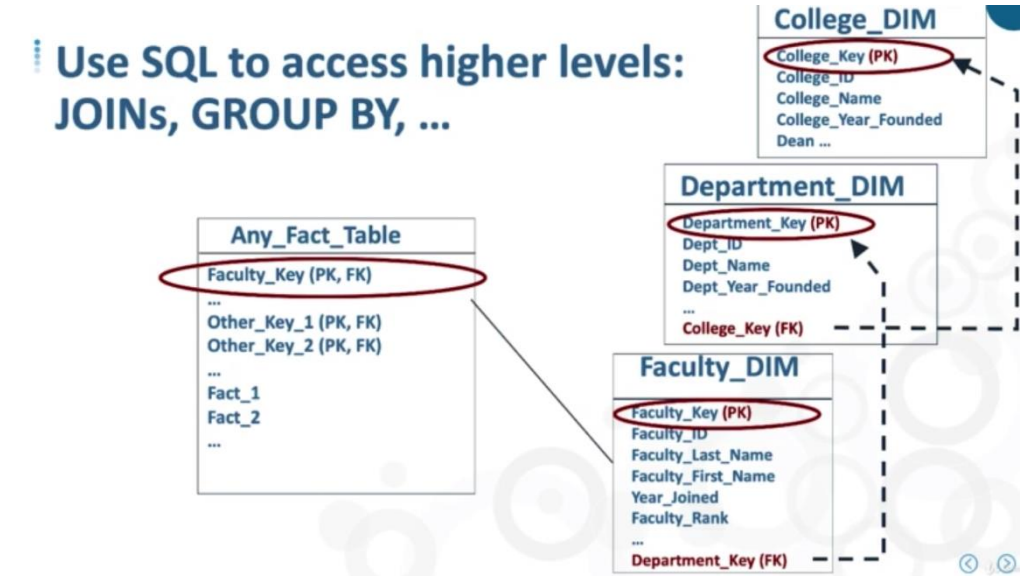


Fact Table in Star Schema



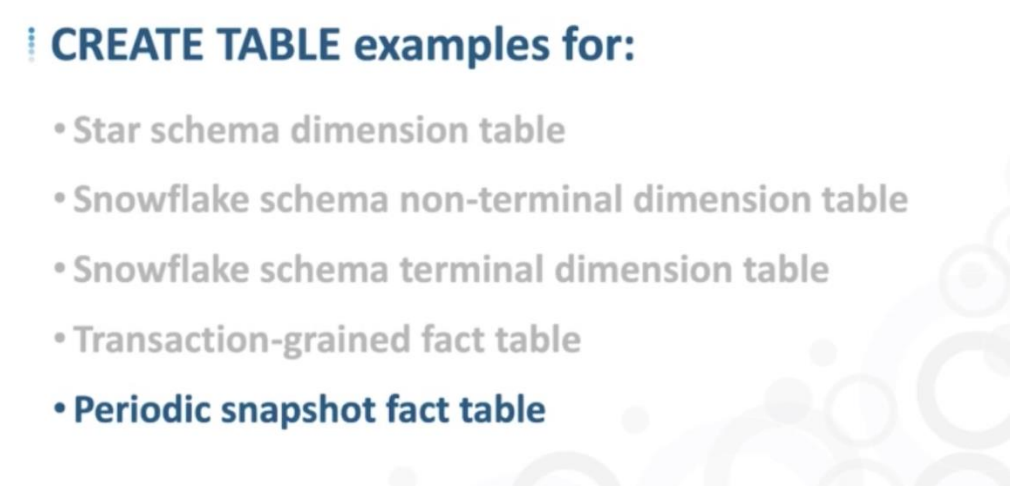
Fact Table in Snowflake Schema



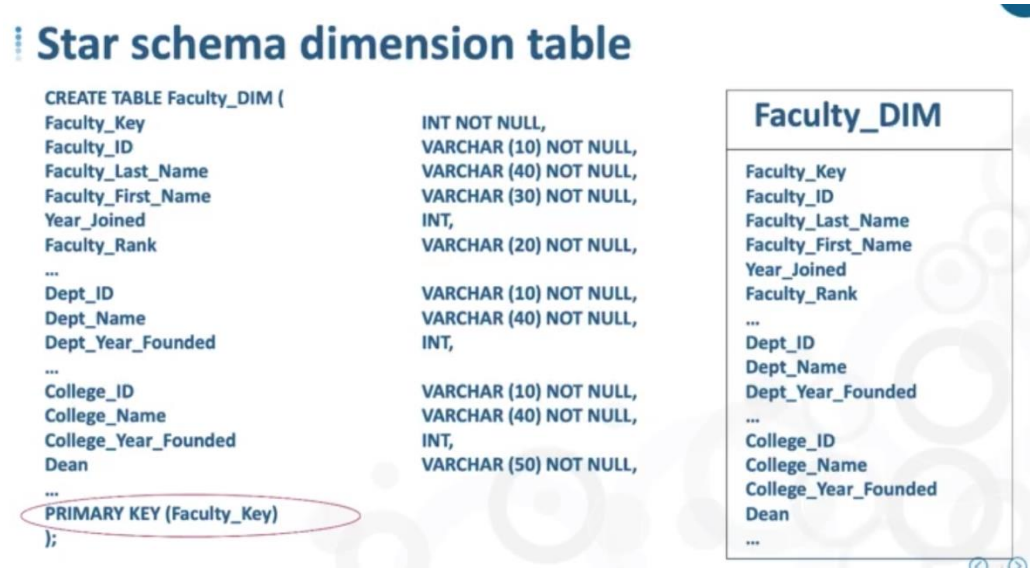


SQL for Dimension and Fact Tables

Create Table Examples

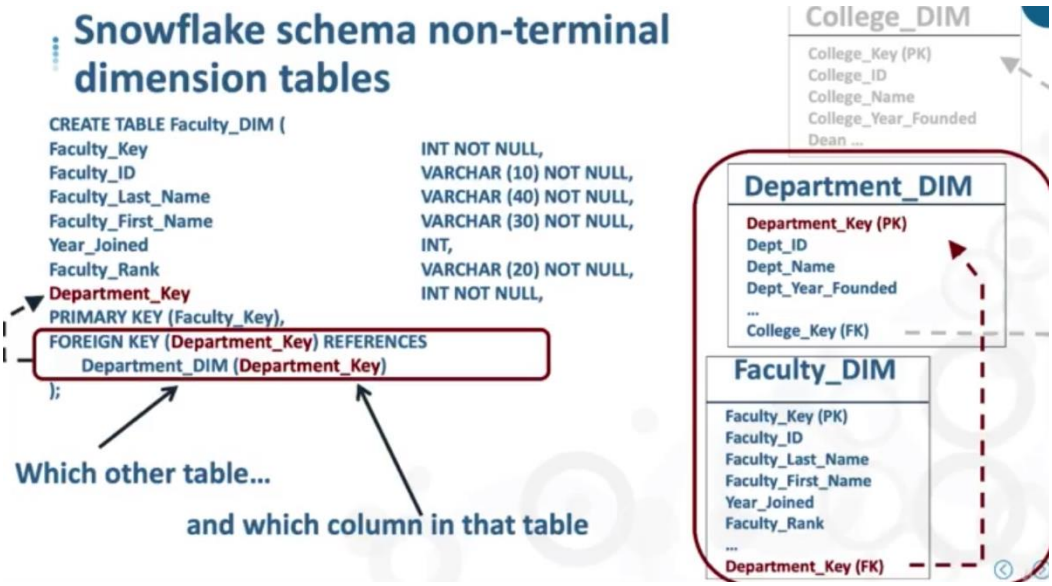


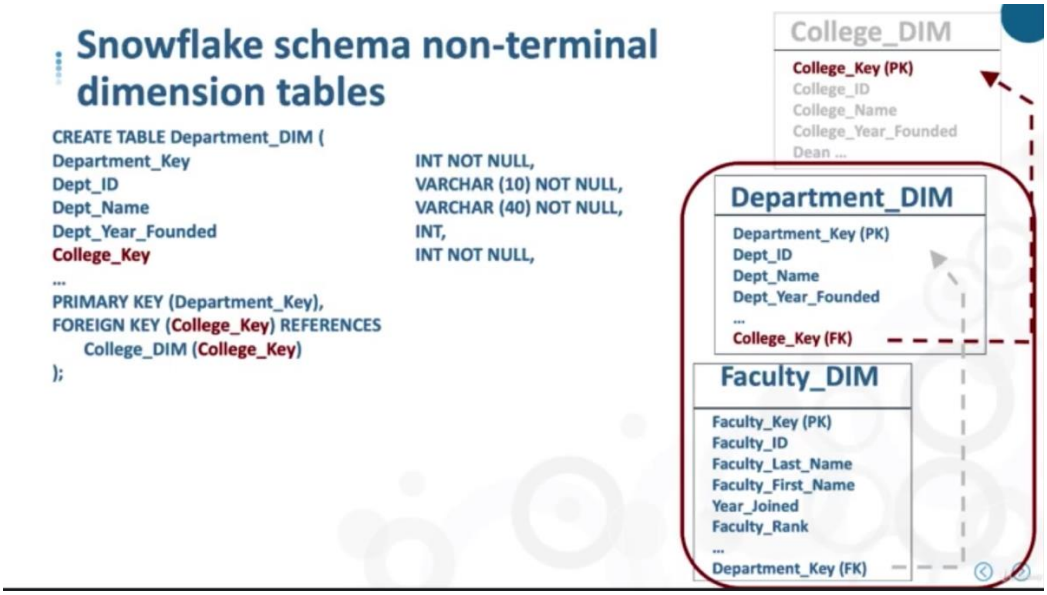
Star Schema Dimension Table



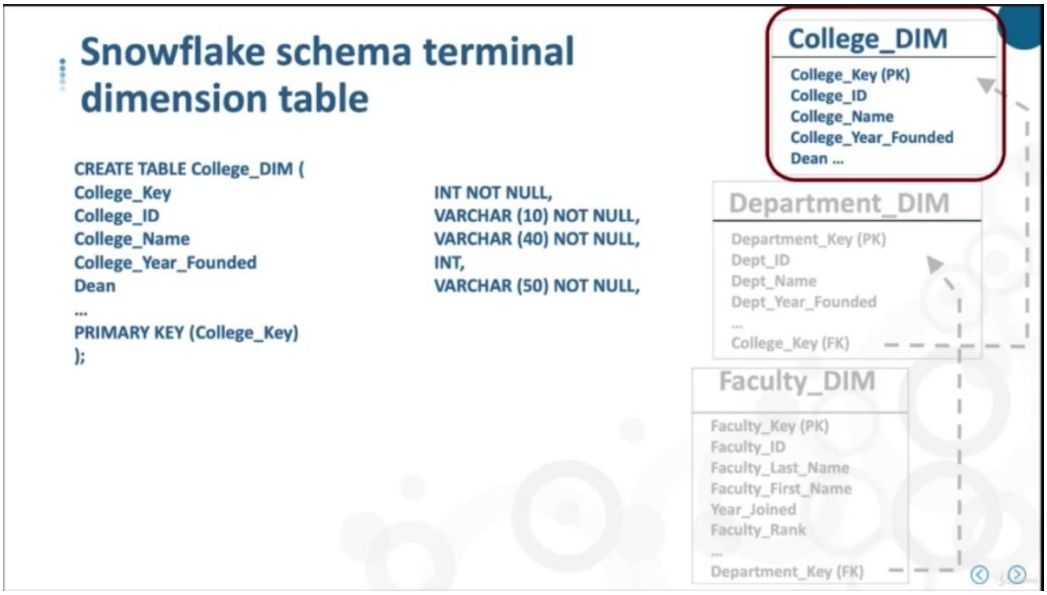
- All dimension kept in single dimension table.
- We have only one primary key for start schema dimension tables.

Snowflake Schema Non-Terminal Dimension Tables

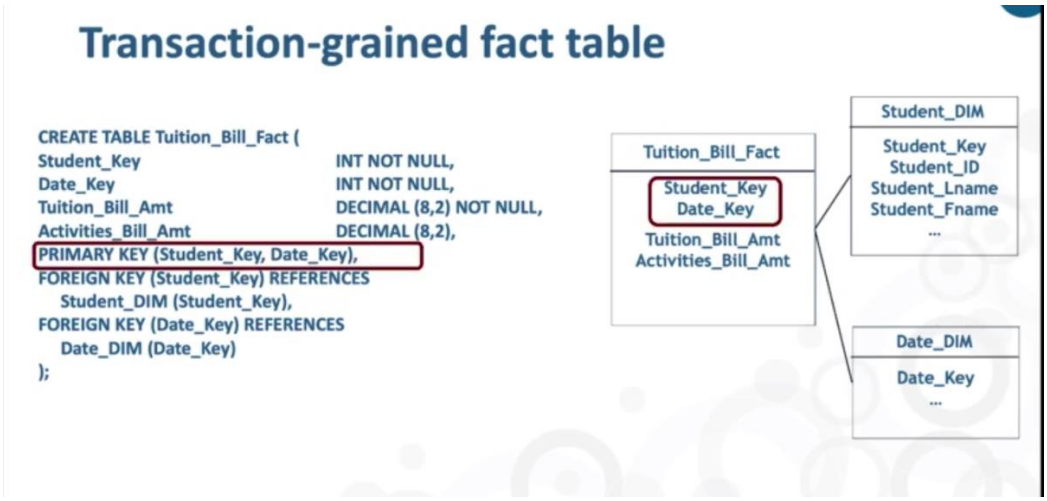




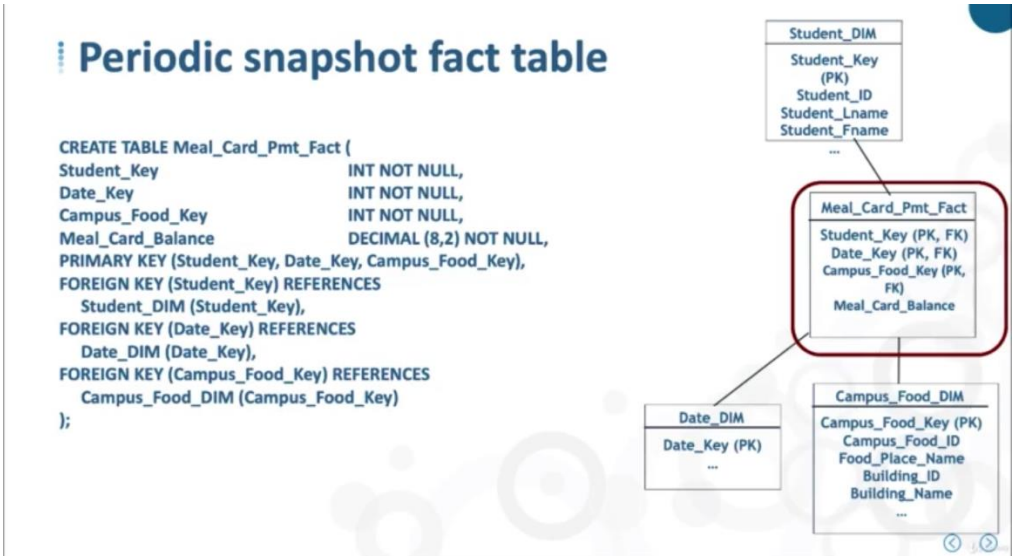
Snowflake Schema Terminal Dimension Tables



Transaction-grained Fact Table



Periodic Snapshot Fact Table



Assignment

Questions for this assignment

1. Design a factless fact table and the associated dimension tables to meet the following requirement:

   1) The university annually holds a three-day orientation program for incoming freshmen

   2) Seniors are encouraged to participate in the orientation program, along with the university administration personnel

   3) Each senior can participate in part or all of the orientation program… e.g., all three days, or just the 1st day, or just the 2nd and 3rd days, etc.

   4) To help plan activities, each senior signs up in an online survey system for which day(s) he/she will be participating

   5) The online survey system sends this information into the university data warehouse, where it is stored for analysis in a factless fact table

2. Design an accumulating snapshot fact table and the associated dimension tables to meet the following requirement:

   1) An important part of a faculty member's position is pursuing grants from various sources

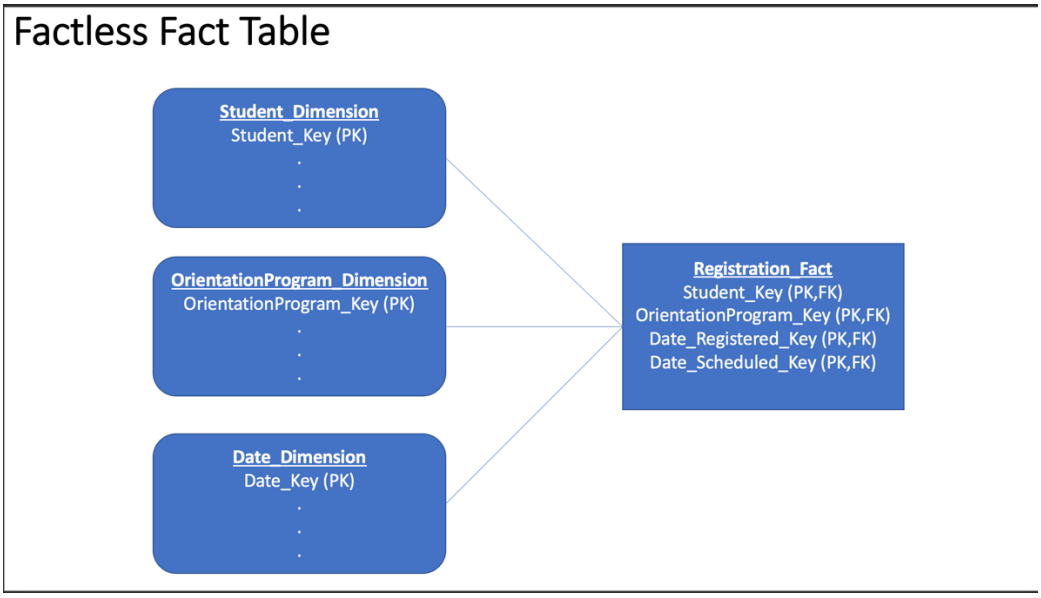
   2) For purposes of this exercise, assume that every grant goes through exactly the same process, regardless of source

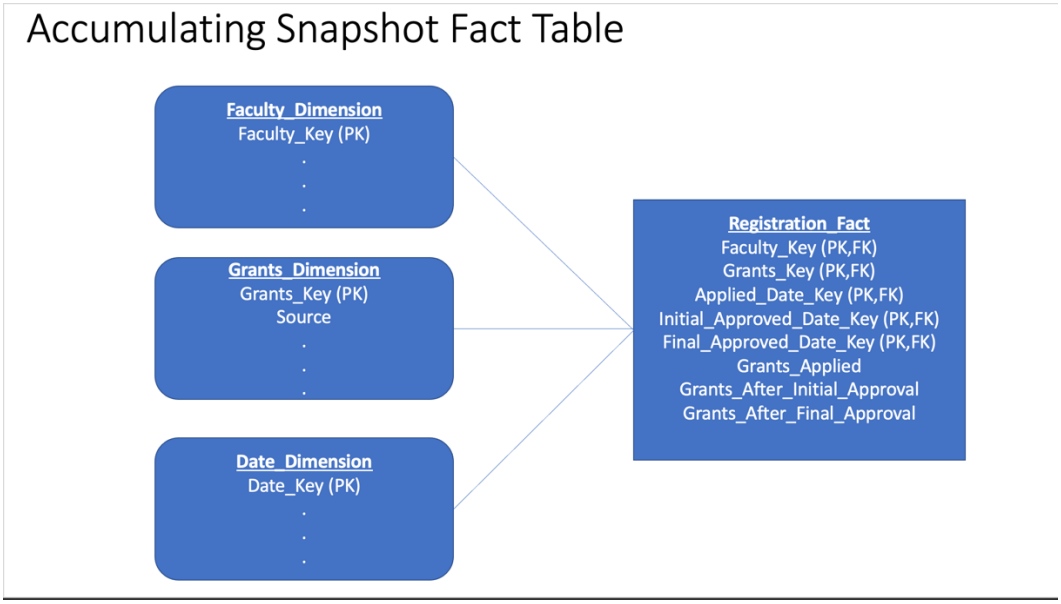
   3) A faculty member applies for a grant, for a specific amount of money

   4) Each grant goes through an initial approval step; if approved, the amount of money might be reduced from the original request

   5) Each grant then goes through a final approval step; if approved, the amount of money might be reduced from the amount that entered the final approval step

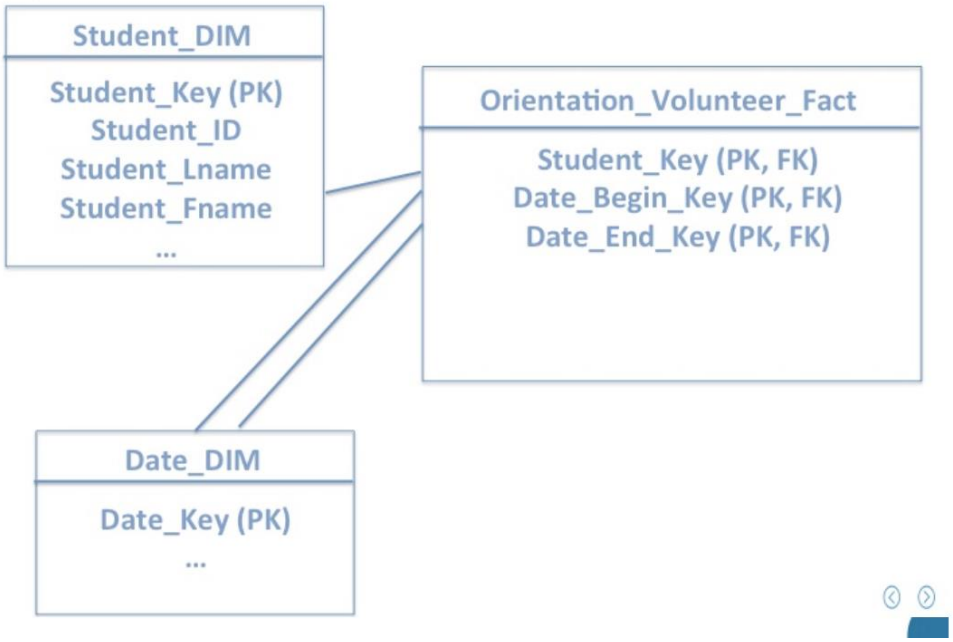
   6) The grant money for an approved grant is eventually paid to the faculty member

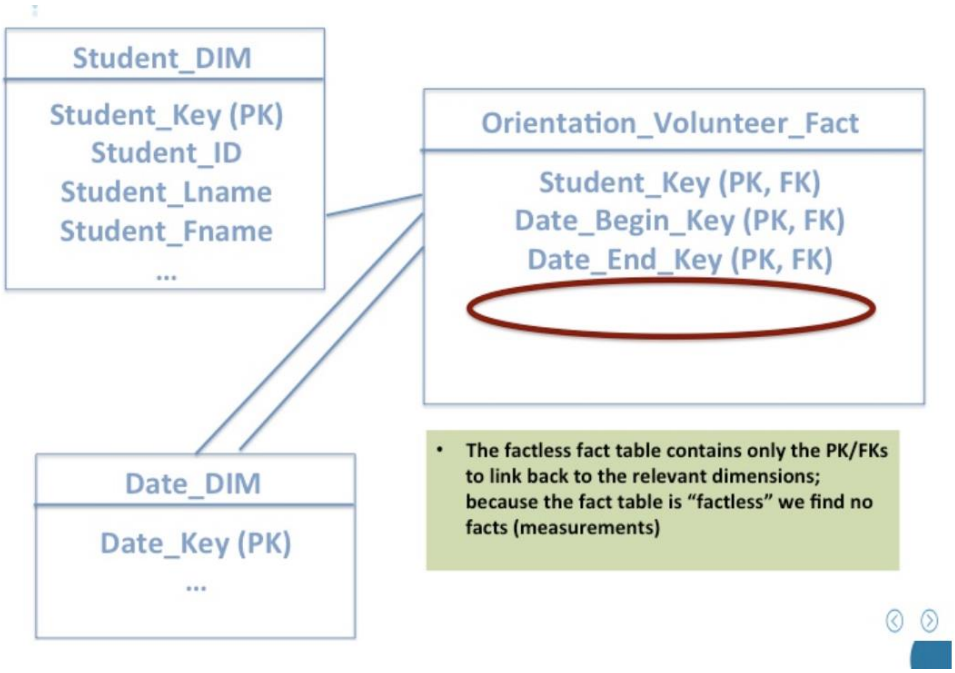


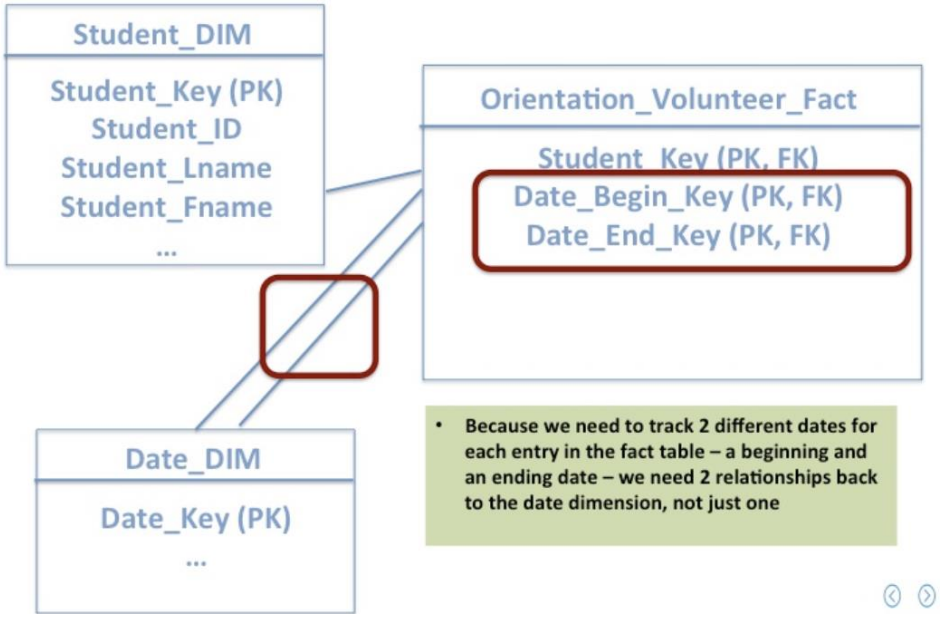


Author Solution

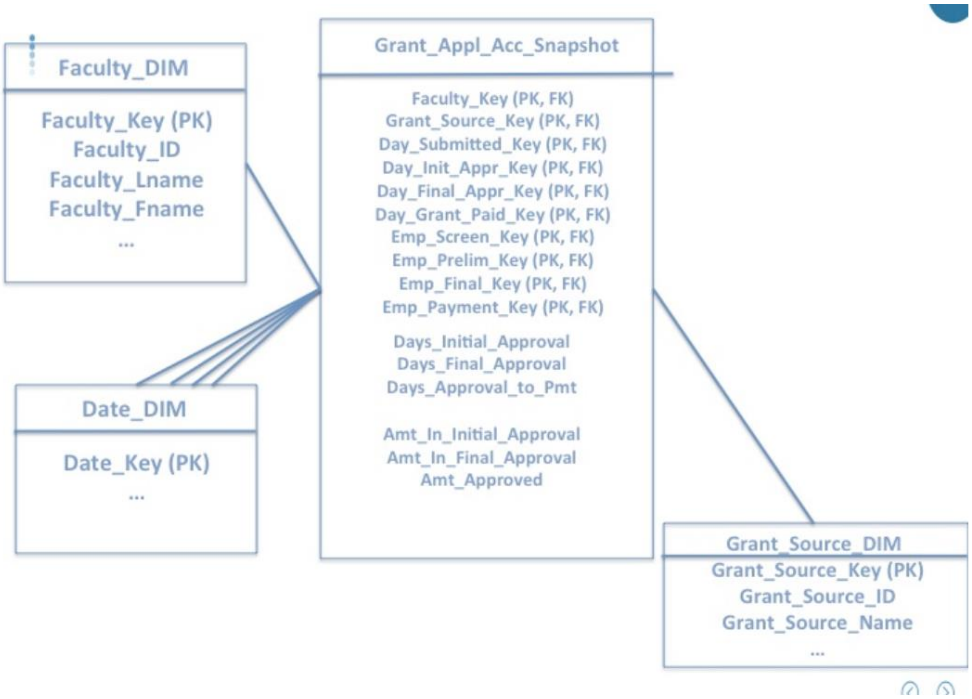
1. Factless Fact Table

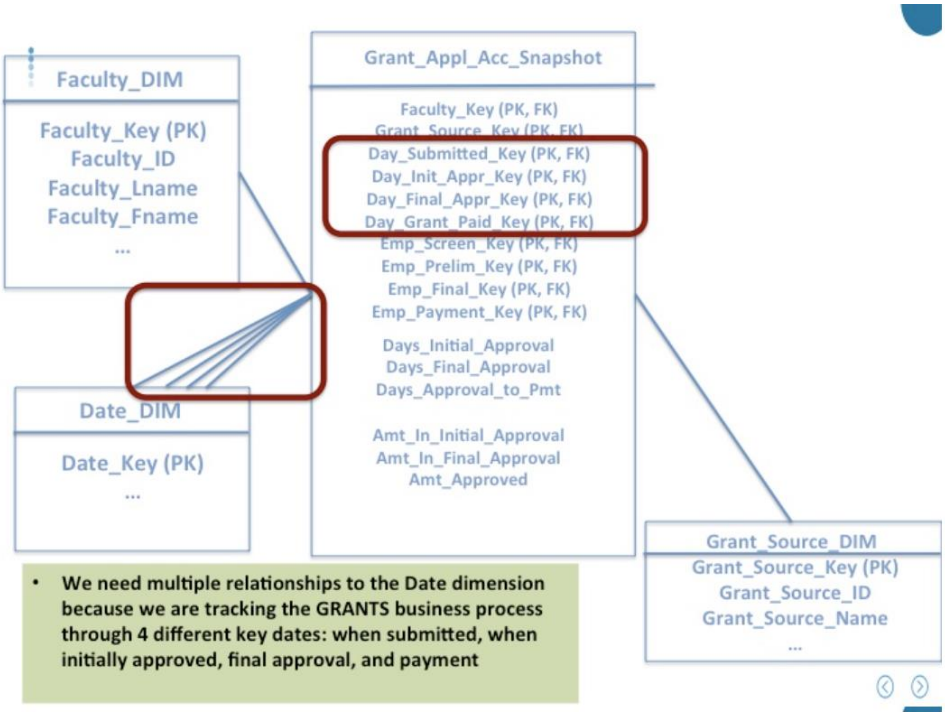




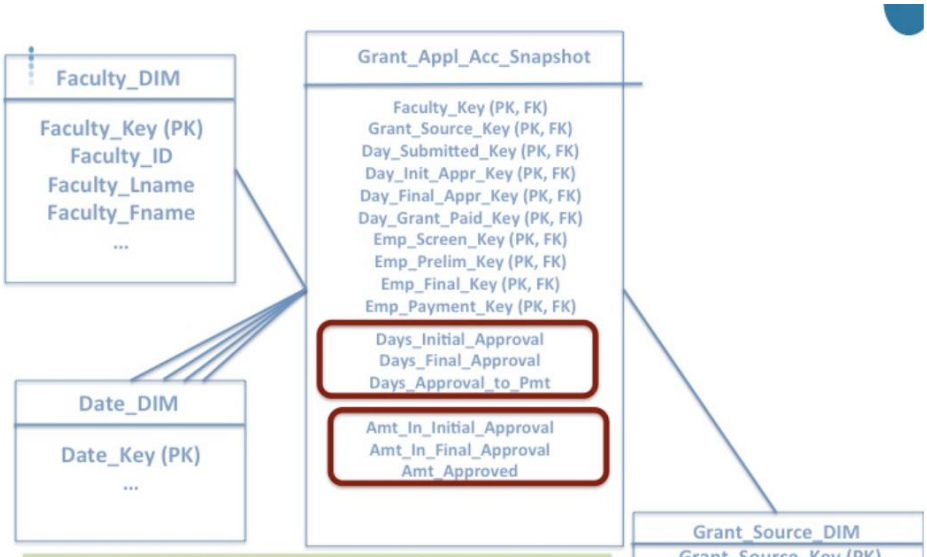


2. Accumulation snapshot fact table.





- **We need multiple relationships to the Date dimension because we are tracking the GRANTS business process through 4 different key dates: when submitted, when initially approved, final approval, and payment**

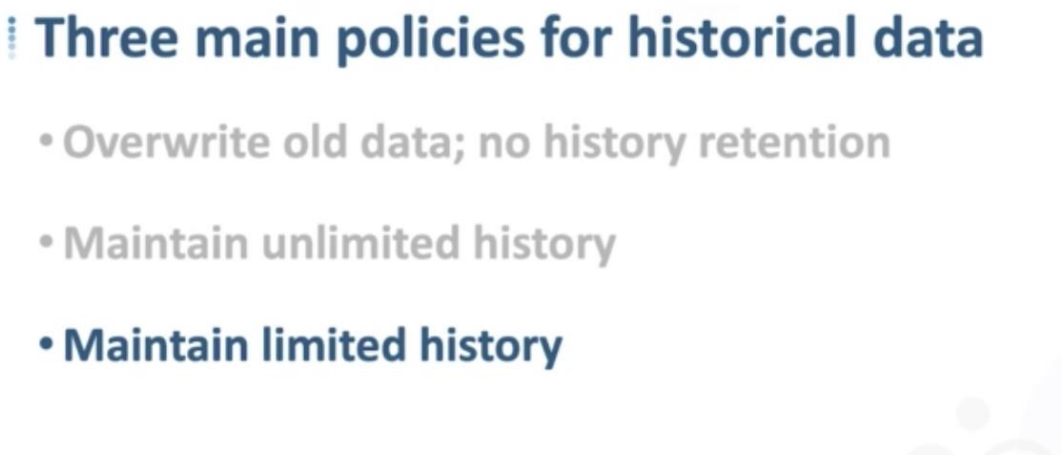


- In this example, we are accumulating two different "sets" of facts (measurements): 1) time, in the forms of days spent in each stage; and 2) the amounts at each stage, which could be different than the previous stage, as specified in the instructions

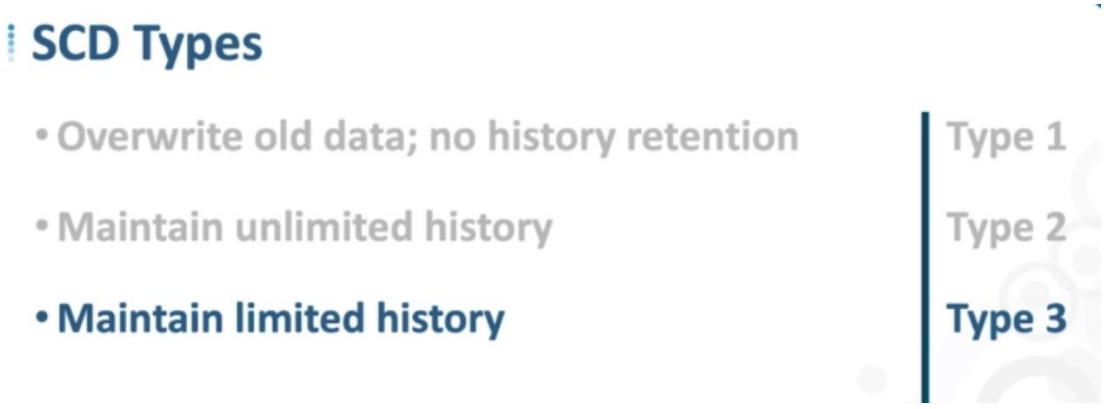
Managing Data Warehouse History Through Slowly Changing Dimensions

Slowly Changing Dimension (SCDs) and Data Warehouse History

Three main policies for historical data



SCD Types



Comparing 3 Main SCD Types

SCD Type	Technique	Implications
Type 1	"In-Place Update" ETL Pattern	Simplest…but no history maintained
Type 2	Create new dimension table row for each new "version" of history	Most architecturally complex…but robust representation of history
Type 3	Small number of dimension table columns for multiple "versions" of history	Easily switch back and forth between "as-is" and "as-was" reporting…but limited use cases

Additional SCD types for special cases

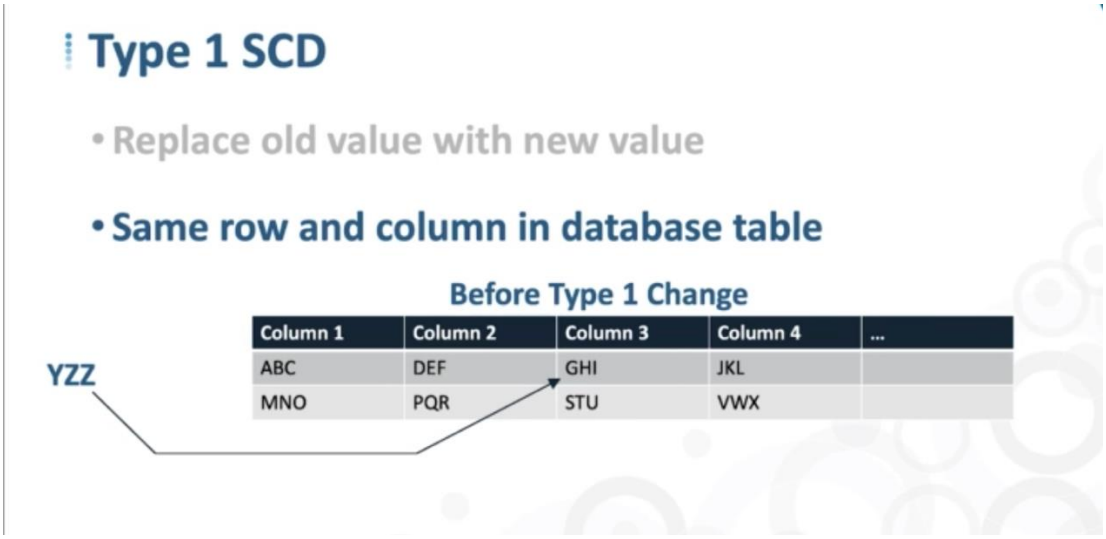
- Type 0 (always retain original value)
- Type 4 (min-dimension)
- **Types 5, 6, and 7 (various hybrid techniques)**

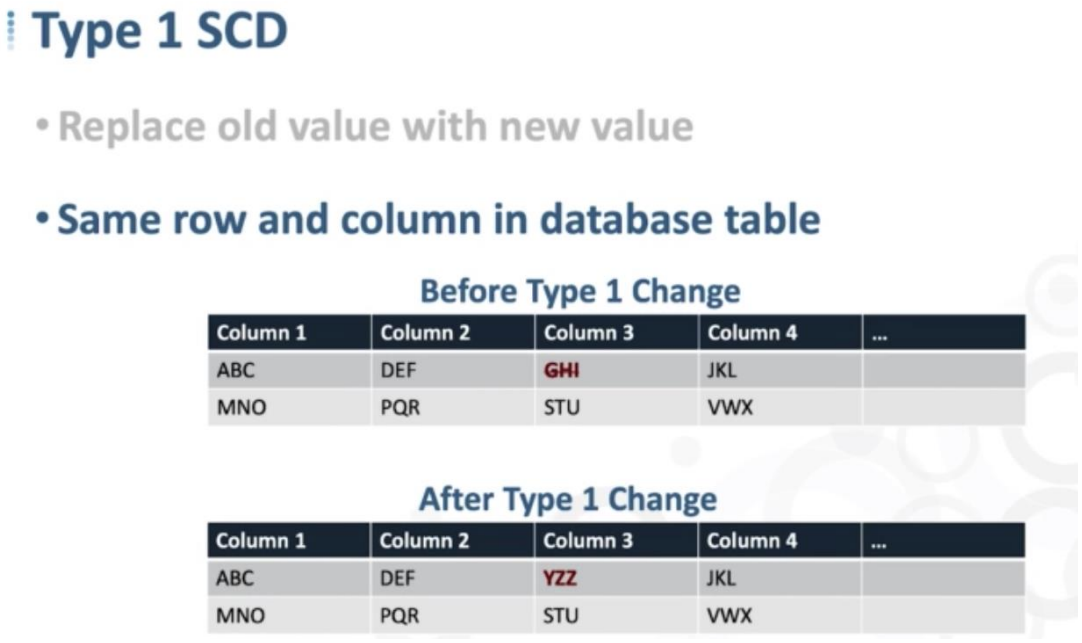
Our focus

- **Type 0 (always retain original value)**
- **Type 1 (overwrite)**
- **Type 2 (new row)**
- **Type 3 (new column)**
- **Type 4 (min-dimension)**
- **Types 5, 6, and 7 (various hybrid techniques)**

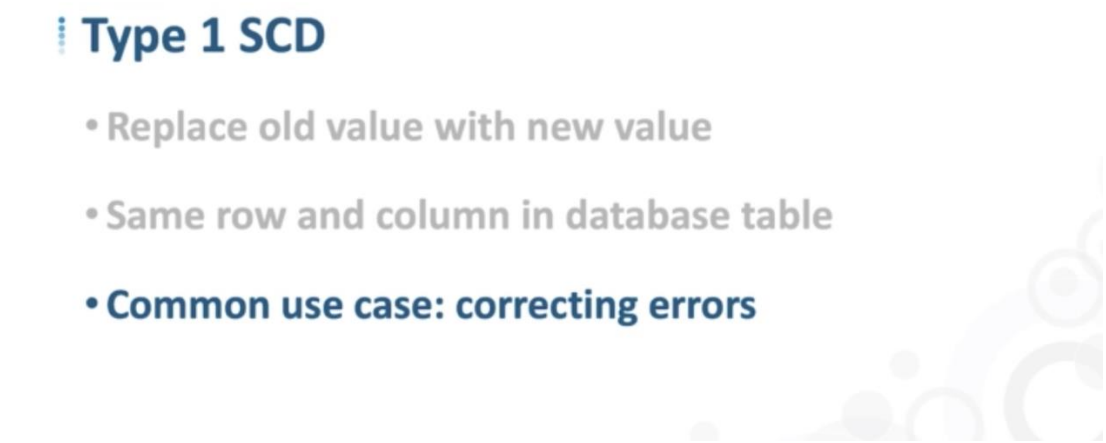
Type 1 Slowly Changing Dimension (SCD)

Replace old value with new value





Correcting Errors



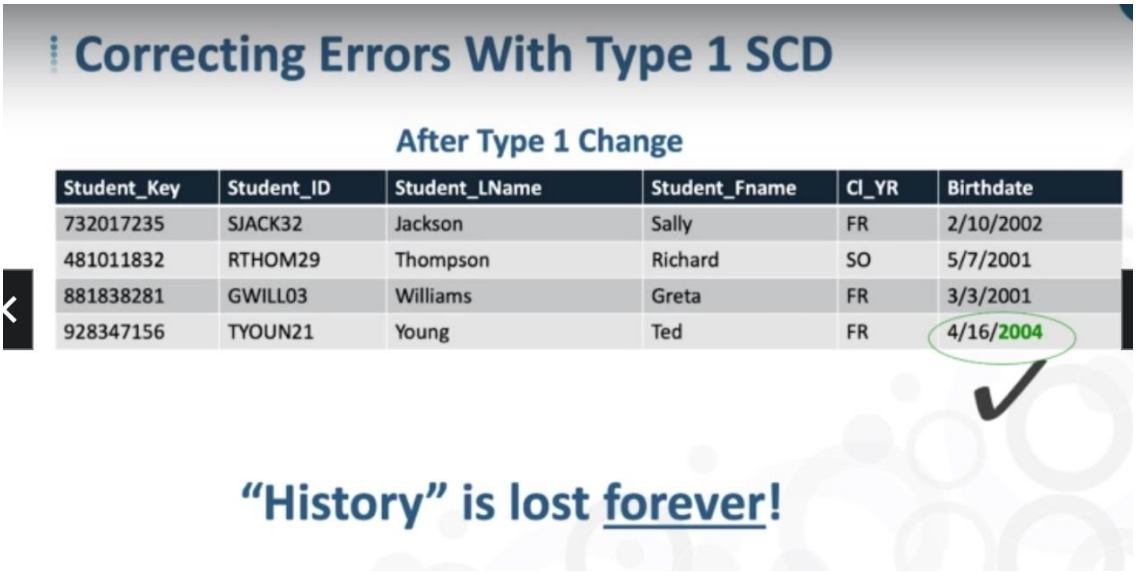
Correcting Errors With Type 1 SCD

Before Type 1 Change

Student_Key	Student_ID	Student_LName	Student_Fname	Cl_YR	Birthdate
732017235	SJACK32	Jackson	Sally	FR	2/10/2002
481011832	RTHOM29	Thompson	Richard	SO	5/7/2001
881838281	GWILL03	Williams	Greta	FR	3/3/2001
928347156	TYOUN21	Young	Ted	FR	4/16/2014

Should be 2004, not 2014

We will be doing the overwrite.



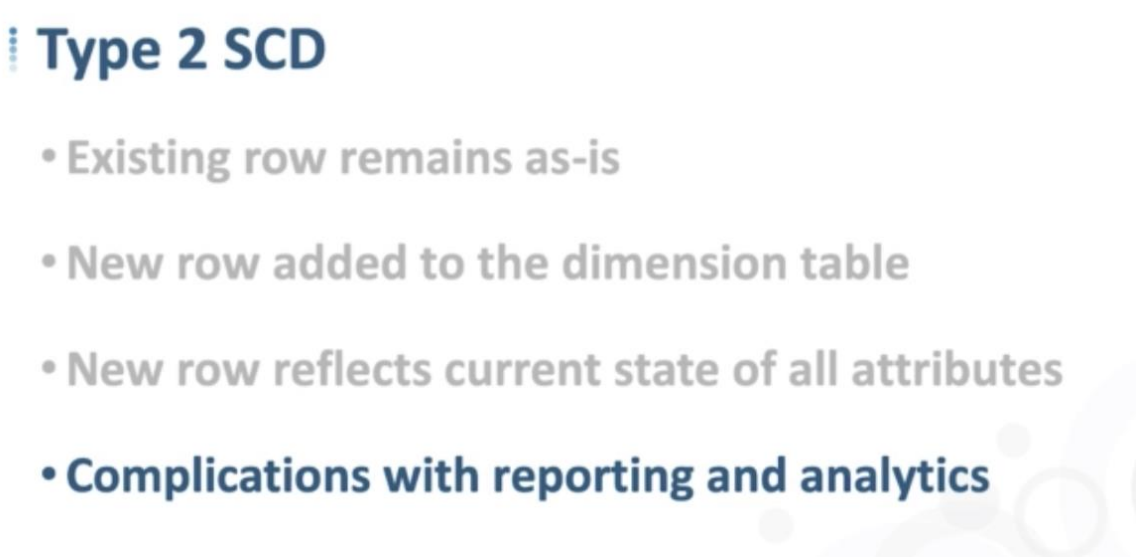
SCD Type 1 Trade-offs

SCD Type 1 Tradeoffs

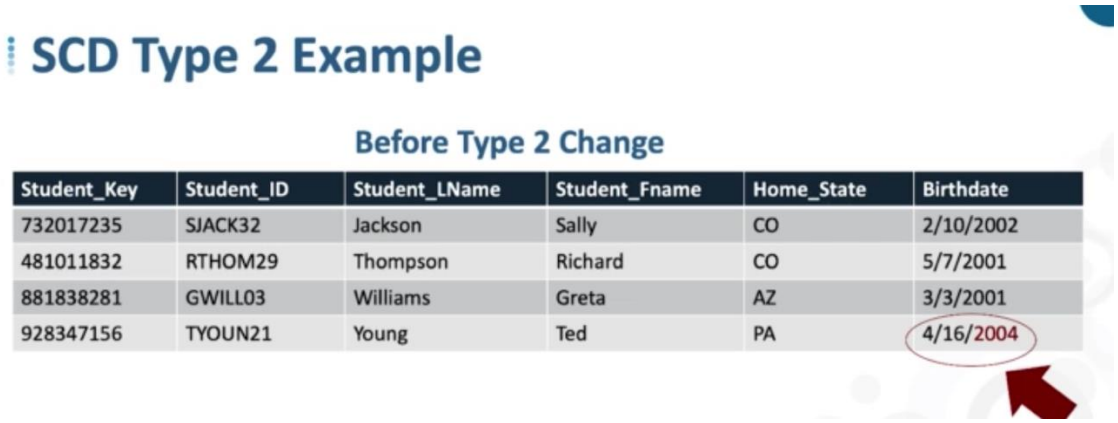
Advantages	Disadvantages
Simplest and most straightforward	Might still want history of errors for auditing purposes
Data warehouse content errors are purged forever	Reporting before and after Type 1 change could vary
Best for error correction and any other "don't need any history" situations	Tendency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

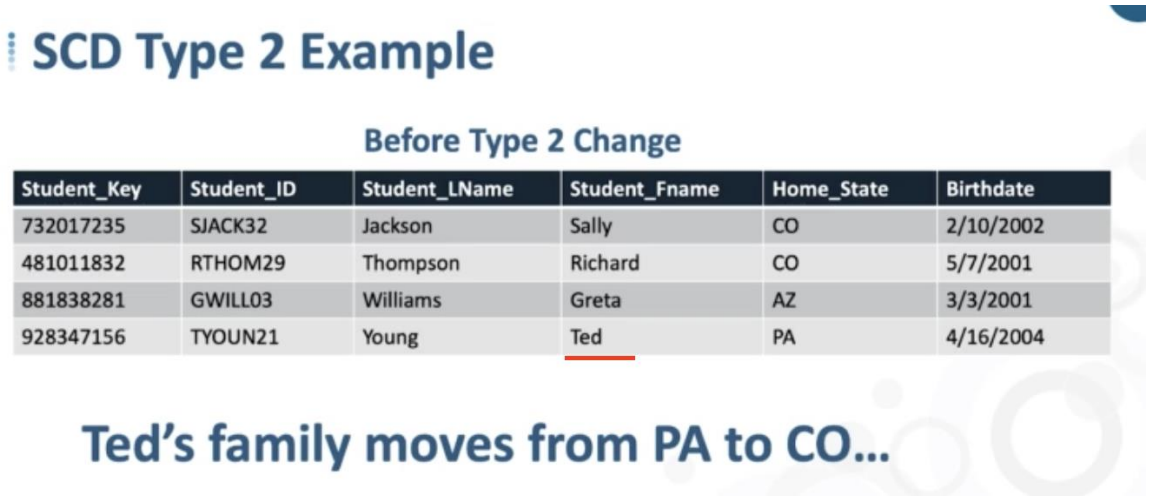
Type 2 Slowly Changing Dimension (SCD)

Type 2 SCD



SCD Type 2 Example





SCD Type 2 Example

Before 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

We need to maintain home state history for reporting and analytics

SCD Type 2 Example

Before 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

A Type 1 change for "Home_State" won't allow us to maintain that history!

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
	TYOUN21	Young	Ted	CO	4/16/2004

Add an entirely new row to dimension table with new/updated attribute value

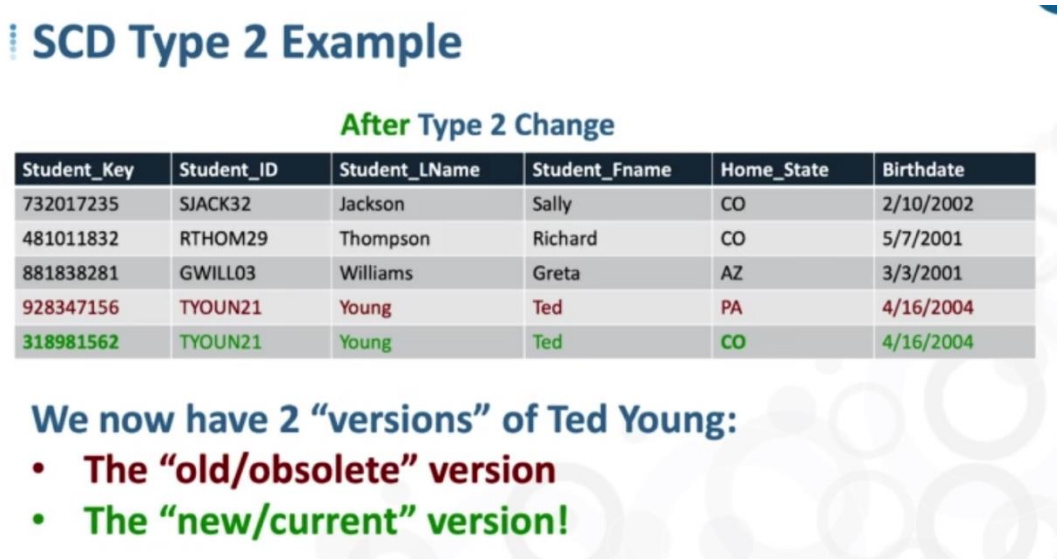
SCD Type 2 Example

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
???	TYOUN21	Young	Ted	CO	4/16/2004

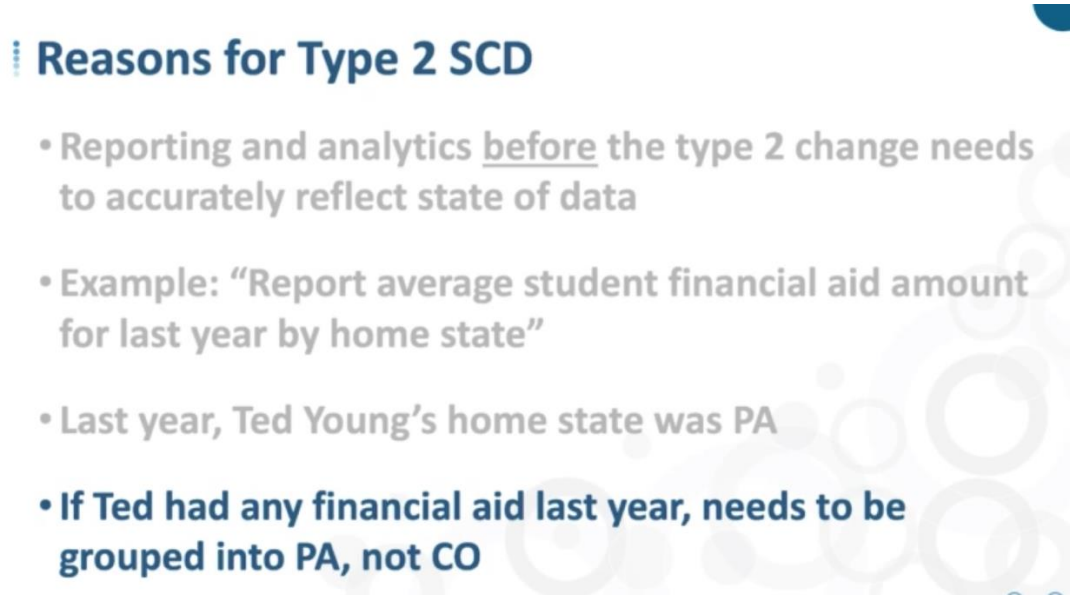
What about the Student_Key ?

- Student_Key is surrogate key which will be generated by data warehouse.
- For the new row created, Student_Key will be generated.

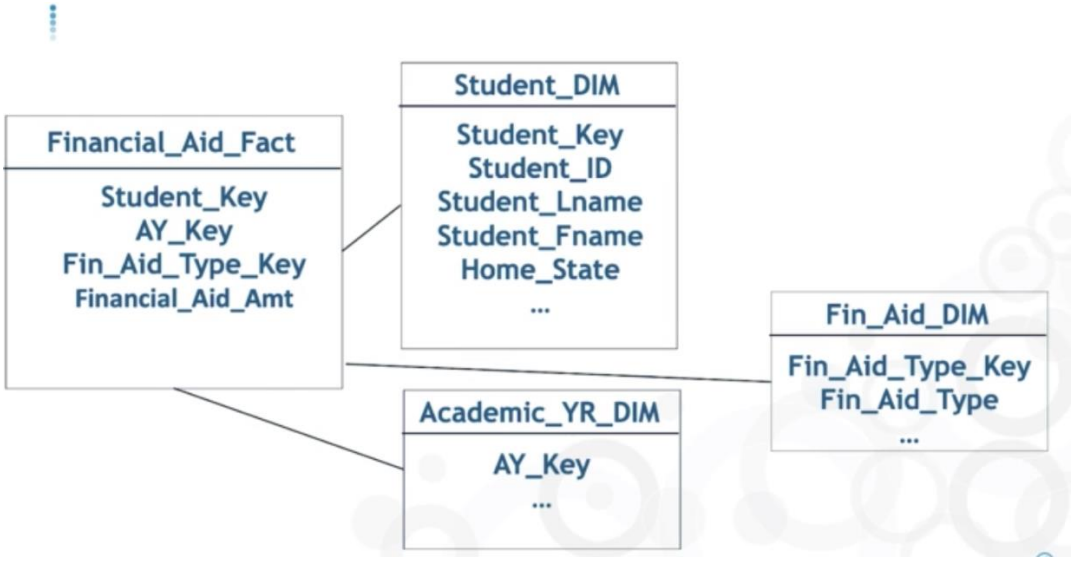


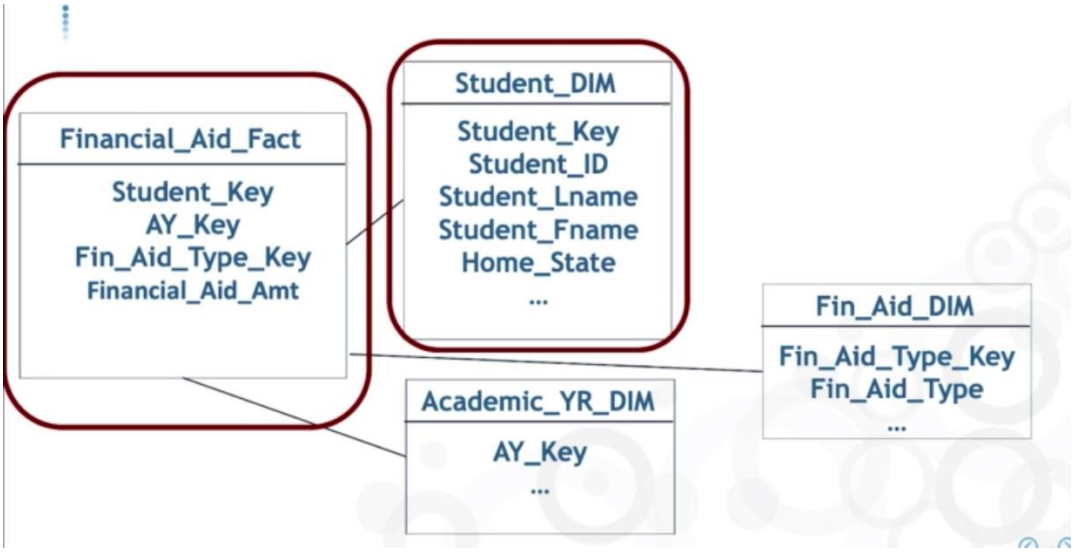
Why do we need a two version of data ?

Reason for Type 2 SCD



Fact Table Complications with Type 2 SCDs





SCD Type 2 and Fact Tables

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

Student_Key	AY_Key	Fin_Aid_Type_Key	Financial_Aid_Amt
732017235	22221113	11118888	$7,500.00
481011832	22221113	11118888	$4,000.00
881838281	87729999	11118888	$7,500.00
928347156	22221113	11118888	$7,500.00
318981562	87729999	11118888	$2,000.00

This example shows that same student received a financial aid for different states.

Find out all the financial aid that Ted Young received in fact table is challenging. So we are including the natural keys in fact table.

Correct Reporting with Type 2 SCDs

- Could include natural keys also in fact tables
- **Supports queries such as "show all financial aid for Ted Young"**

Student_Key	Student_ID	AY_Key	Fin_Aid_Type_Key	Financial_Aid_Amt
732017235	SJACK32	22221113	11118888	$7,500.00
481011832	RTHOM29	22221113	11118888	$4,000.00
881838281	GWILL03	87729999	11118888	$7,500.00
928347156	TYOUN21	22221113	11118888	$7,500.00
318981562	TYOUN21	87729999	11118888	$2,000.00

Correct Reporting with Type 2 SCDs

- Could include natural keys also in fact tables
- Supports queries such as "show all financial aid for Ted Young"
- **Unwieldy and takes significant storage for fact tables with many dimensions and billions of rows**

SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	NK	SK	...

If we are going by this approach, we will end up including the natural keys for serval surrogate keys. Then, if we are dealing with large data, fact table becomes more wider and difficult to handle.

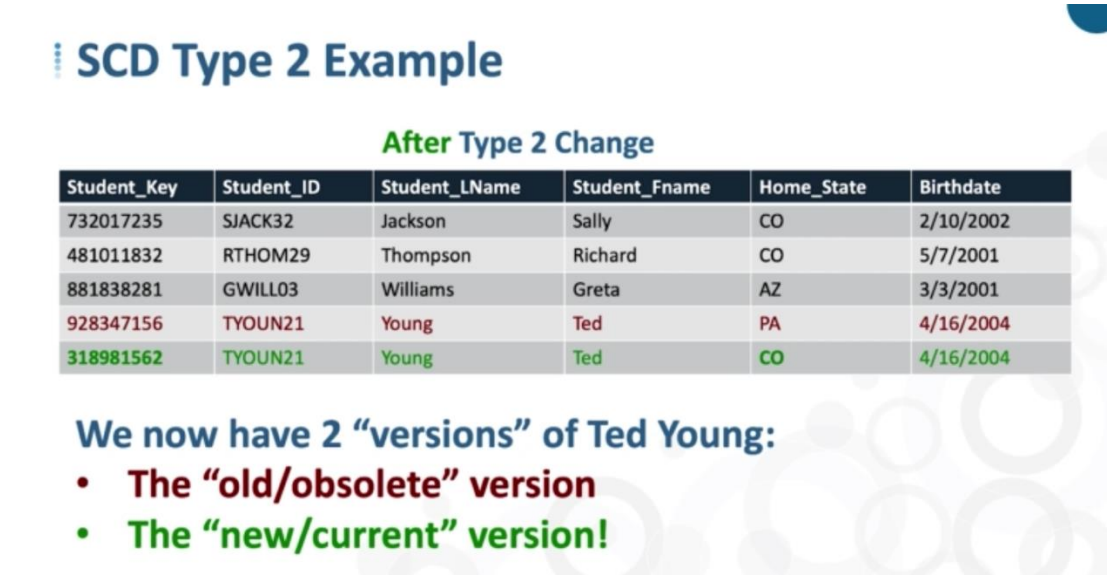
To overcome this problem, we can use multi step SQL to connect between fact and dimension tables.

Correct Reporting with Type 2 SCDs

- Best handled through careful "multi-step" SQL between dimension tables and fact tables
- First: find all SKs for desired student
- **Next: retrieve all fact table rows for all of the relevant SKs**

Still another complication…

Maintain Correct Data Order with Type 2 SCDs



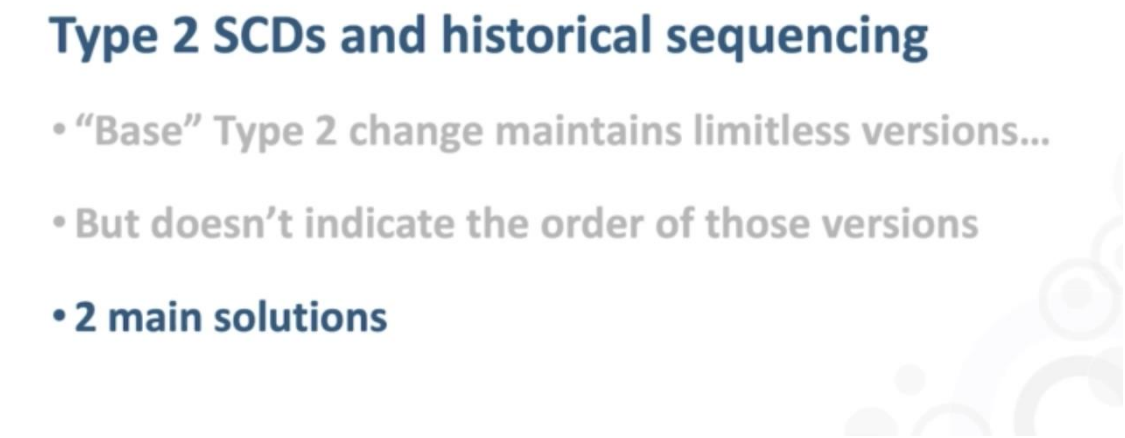
SCD Type 2 Example

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!

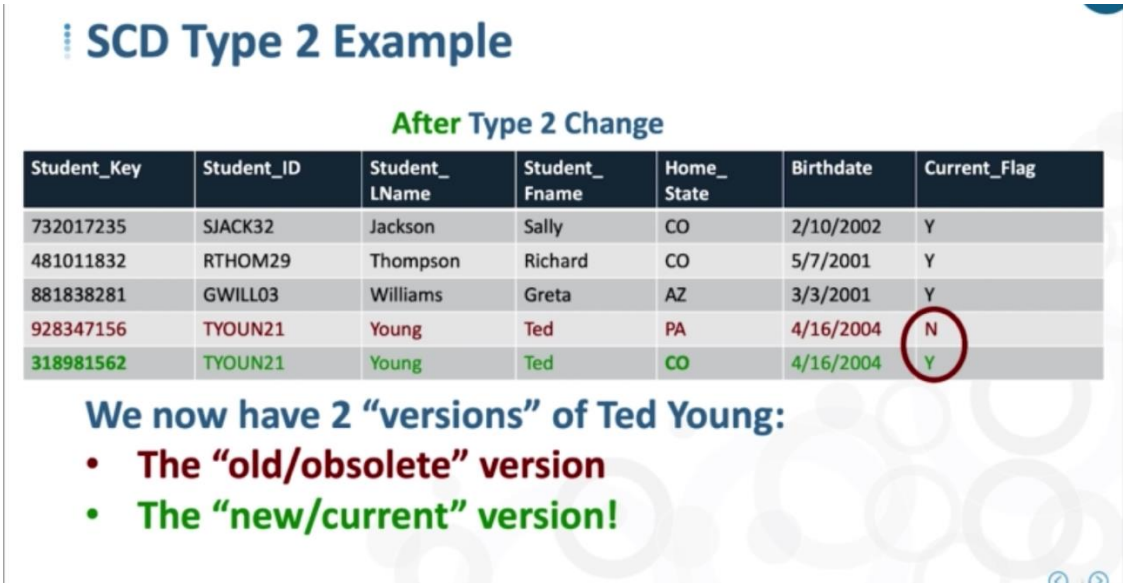
We are not maintain any data to represent which is new and old.



Type 2 SCDs and historical sequencing

- "Base" Type 2 change maintains limitless versions…
- But doesn't indicate the order of those versions
- 2 main solutions

First Solution: Current_Flag



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!

What if a row changes more than once ?

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	N
229988772	TYOUN21	Young	Ted	NM	4/16/2004	Y

We now have 3 "versions" of Ted Young:

- Two "old/obsolete" versions
- One "new/current" version

Second Solution – Exp_Date

To overcome this problem we maintain, Eff_Date and Exp_Date instead of Current_Flag.

For the rows which are currently active, we will updating the Exp_Date column with highest future date.

Second solution: effective/expiration dates

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

Third Solution – Combination of Effective Date, Expiry Date and Current Flag

A third solution (Kimball, DW Toolkit)

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

Comparing 3 Main SCD Types

SCD Type	Technique	Implications
Type 1	"In-Place Update" ETL Pattern	Simplest…but no history maintained
Type 2	Create new dimension table row for each new "version" of history	Most architecturally complex…but robust representation of history
Type 3	Small number of dimension table columns for multiple "versions" of history	Easily switch back and forth between "as-is" and "as-was" reporting…but limited use cases

Type 3 SCD

- Add new column rather than new row to reflect changes
- "Old value" column and "new value" column
- Supports back-and-forth switching for flexible reporting and analytics
- **Best used for all data in a dimension being "reorganized"**

Type 3 SCD Example

Type 3 SCD example

- Textbook publisher's sales divisions being reorganized
- Formerly EAST and WEST divisions
- Now EAST, CENTRAL, and WEST
- All sales reps reassigned to 1 of the 3 new divisions
- **Reorganization effective January 1, 2021**

We want to report:

- 2020 (and earlier) sales data "sliced" by old NORTH and SOUTH divisions
- **2021 (and forward) sales data "sliced" by new EAST, CENTRAL, and WEST divisions**

We want to report:

- 2020 (and earlier) sales data "sliced" by new EAST, CENTRAL, and WEST divisions
- 2021 (and forward) sales data "sliced" by old NORTH and SOUTH divisions
- **Old and new sales data together, "sliced" by either old or new division structures**

We could use Type 2 changes, But queries very complex.

A better solution

Type 3 SCD example

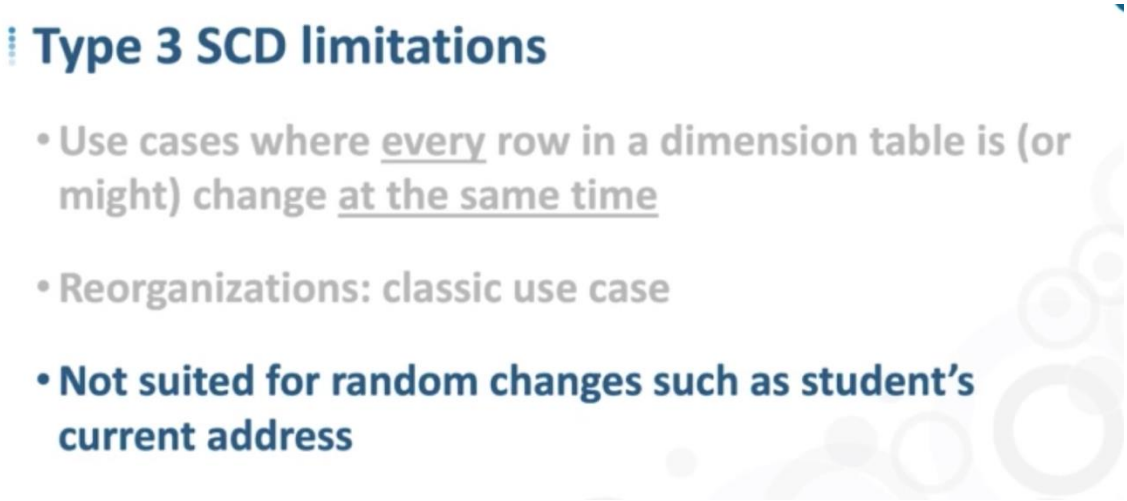
Sales_Rep_DIM

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

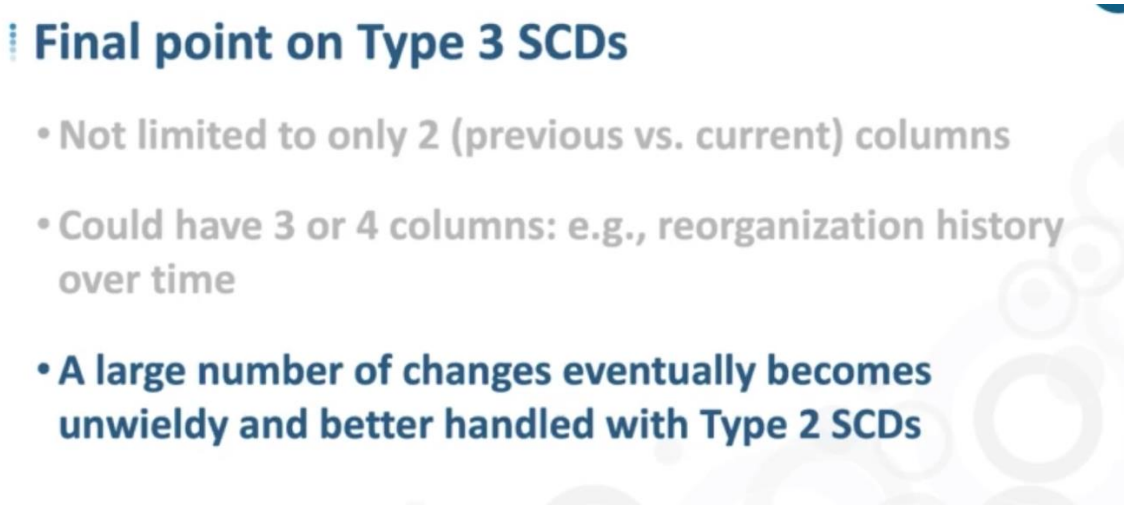
SCD Type 3 "pair of attributes"

Report/Analytic	Column to Use
2020 (and earlier) sales data "sliced" by old NORTH and SOUTH divisions	PREVIOUS_DIV
2021 (and forward) sales data "sliced" by new EAST, CENTRAL, and WEST divisions	CURRENT_DIV
2020 (and earlier) sales data "sliced" by new EAST, CENTRAL, and WEST divisions	CURRENT_DIV
2021 (and forward) sales data "sliced" by old NORTH and SOUTH divisions	PREVIOUS_DIV
Old and new sales data together, "sliced" by either old or new division structures	CURRENT_DIV or PREVIOUS_DIV

Type 3 SCD Limitations



Final Point on Type 3 SCDs



Assignment

The **basic setup** for this assessment is as follows:

In our data warehouse, the FACULTY_DIM dimension table serves as the "master" list of key information about our faculty members

In addition to general, basic identifying and demographic columns (fields) such as Last_Name, First_Name, Address, Age, etc., we have 2 columns specific to being a university faculty member:

Rank (e.g., Assistant_Professor, Associate_Professor, Professor, Lecturer, etc.)

Highest_Degree (e.g., PhD, MD, JD, DBA, MS, MA, MBA, etc.)

The **business rules** regarding faculty rank include:

Not all faculty members have doctoral degrees (e.g., PhD, DBA, JD, MD, etc.)

Some faculty members who are Lecturers or Senior Lecturers have a Master's Degree as their highest degree

However, some Lecturers and Senior Lecturers either 1) already have a doctoral degree, or 2) go on to earn a doctoral degree while teaching at our university

For our **data warehouse and managing history**:

Regarding these two faculty-specific database columns:

"Rank" must be handled as a Type 2 change because we want to maintain a history of all ranks a given faculty member has held

"Highest_Degree" will be handled as a Type 1 change because, for purposes of this assessment, we decide that we only our data warehouse to keep track of the highest degree any faculty member has earned rather than a complete history

The following row of data from FACULTY_DIM for Mary Johnson is what you will work with for this assessment.

FAC_KEY	FAC_ID	FAC_LNAME	FAC_FNAME	.... RANK	HIGH_DEGREE
..... (other rows of faculty data)					
4721465	JOHNSM1	Johnson	Mary	LECTURER	M.A.
..... (other rows of faculty data)					

Questions for this assignment

1. ***PART 1***

   Mary Johnson has been enrolled in the university's PhD program while teaching, and has now earned her PhD

   Mary is now being considered for promotion next academic year but continues to teach as a Lecturer

   What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

   PART 2

   Mary Johnson is approved for promotion to Assistant Professor because she now has a PhD.

   What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

2. **FOR QUESTION 2, DISREGARD THE WORK YOU DID IN QUESTION 1 AND START OVER FROM THE BEGINNING ACCORDING TO THE FOLLOWING:**

   PART 1

   Mary Johnson has been enrolled in the university's PhD program while teaching and (unlike Question 1) is still working on her PhD

   Mary is promoted from Lecturer to Senior Lecturer (again, while still working on her PhD)

   What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

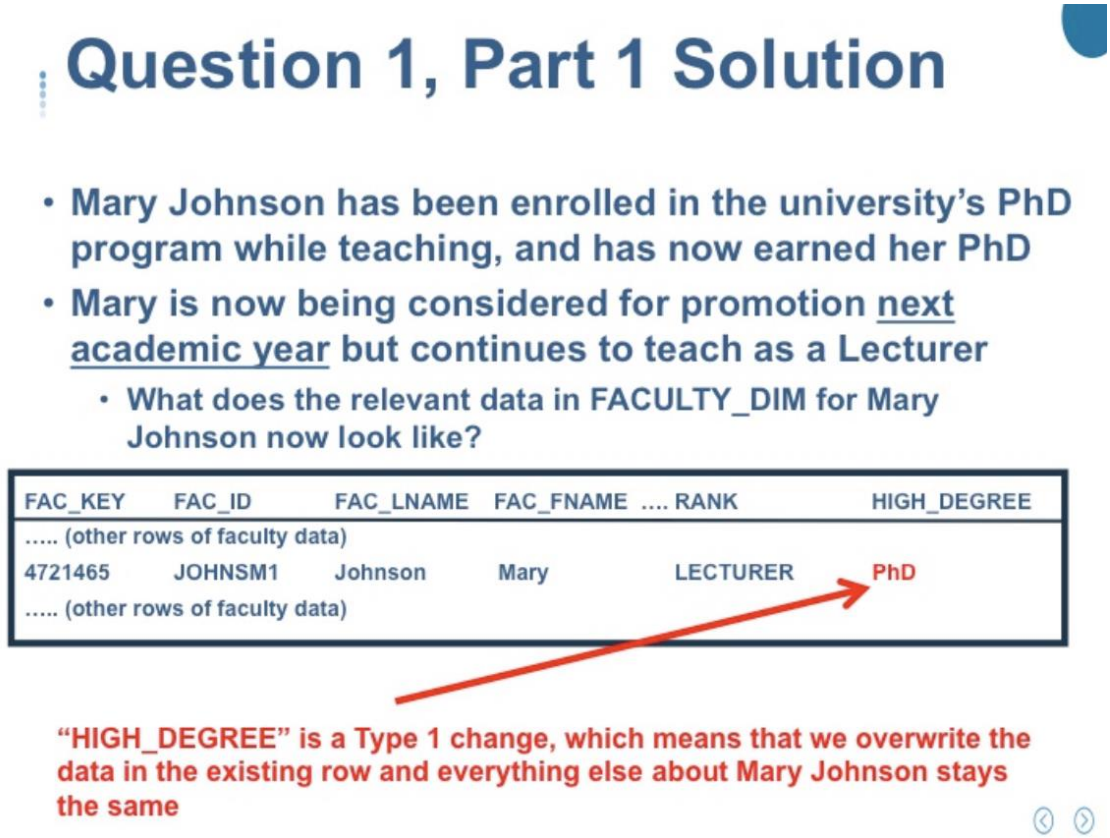
   PART 2

   Mary Johnson now completes her doctoral studies and earns her PhD…but is still teaching as a Senior Lecturer

   What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

1									
Part 1	**FAC_KEY**	**FAC_ID**	**FAC_LNAME**	**FAC_FNAME**	**RANK**	**HIGH_DEGREE**	**EFF_DATE**	**EXP_DATE**	**CURRENT_FLAG**
	4721465	JOHNSM1	Johnson	Mary	LECTURER	M.A.	01/01/2020	31/12/2199	YES
Part 2	**FAC_KEY**	**FAC_ID**	**FAC_LNAME**	**FAC_FNAME**	**RANK**	**HIGH_DEGREE**	**EFF_DATE**	**EXP_DATE**	**CURRENT_FLAG**
	4721465	JOHNSM1	Johnson	Mary	LECTURER	Ph.D	01/01/2020	01/01/2022	NO
	4721466	JOHNSM1	Johnson	Mary	Assistant Professor	Ph.D	01/01/2022	31/12/2199	YES
2									
Part 1	**FAC_KEY**	**FAC_ID**	**FAC_LNAME**	**FAC_FNAME**	**RANK**	**HIGH_DEGREE**	**EFF_DATE**	**EXP_DATE**	**CURRENT_FLAG**
	4721465	JOHNSM1	Johnson	Mary	LECTURER	M.A.	01/01/2020	01/01/2022	NO
	4721466	JOHNSM1	Johnson	Mary	Senior Lecturer	M.A.	01/01/2022	31/12/2199	YES
Part 2	**FAC_KEY**	**FAC_ID**	**FAC_LNAME**	**FAC_FNAME**	**RANK**	**HIGH_DEGREE**	**EFF_DATE**	**EXP_DATE**	**CURRENT_FLAG**
	4721465	JOHNSM1	Johnson	Mary	LECTURER	Ph.D	01/01/2020	01/01/2022	NO
	4721466	JOHNSM1	Johnson	Mary	Senior Lecturer	Ph.D	01/01/2022	31/12/2199	YES

Author Solution



Question 1, Part 2 Solution

- Mary Johnson is approved for promotion to Assistant Professor because she now has a PhD.
  - What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

FAC_KEY	FAC_ID	FAC_LNAME	FAC_FNAME	.... RANK	HIGH_DEGREE
..... (other rows of faculty data)					
4721465	JOHNSM1	Johnson	Mary	LECTURER	PhD
..... (other rows of faculty data)					
8722752	JOHNSM1	Johnson	Mary	ASST_PROF	PhD

An entirely new row of data for Mary Johnson – basically, a new "version" of Mary – is created with a randomly generated surrogate key as the primary key for this new row

The natural key for this row (JOHNSM1) remains the same

Mary's new Rank goes into the new row of data but her previous rank remains as-is with the "old version of Mary"

The previous Type 1 change of Mary's HIGH_DEGREE to PhD remains

Question 2, Part 1 Solution

- Mary is promoted from Lecturer to Senior Lecturer (again, while still working on her PhD)
  - What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

FAC_KEY	FAC_ID	FAC_LNAME	FAC_FNAME	.... RANK	HIGH_DEGREE
..... (other rows of faculty data)					
4721465	JOHNSM1	Johnson	Mary	LECTURER	M.A.
..... (other rows of faculty data)					
8722752	JOHNSM1	Johnson	Mary	SR_LECTURER	M.A.

WITH A TYPE 2 CHANGE, an entirely new row of data for Mary Johnson – basically, a new "version" of Mary – is created with a randomly generated surrogate key as the primary key for this new row

The natural key for this row (JOHNSM1) remains the same

Mary's new Rank goes into the new row of data but her previous rank remains as-is with the "old version of Mary"

Question 2, Part 2 Solution

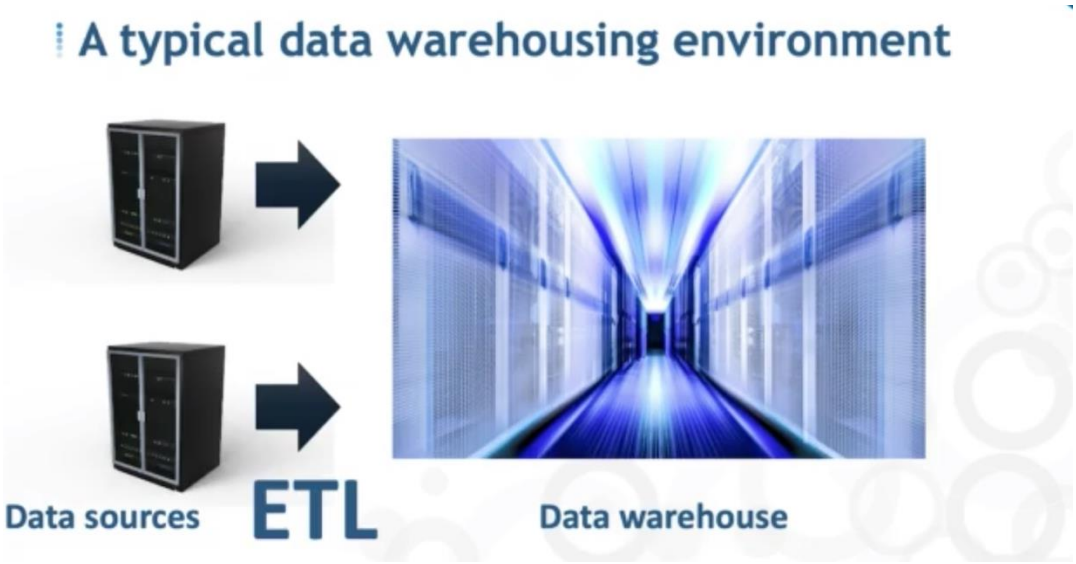
- Mary Johnson now completes her doctoral studies and earns her PhD…but is still teaching as a Senior Lecturer
  - What does the relevant data in FACULTY_DIM for Mary Johnson now look like?

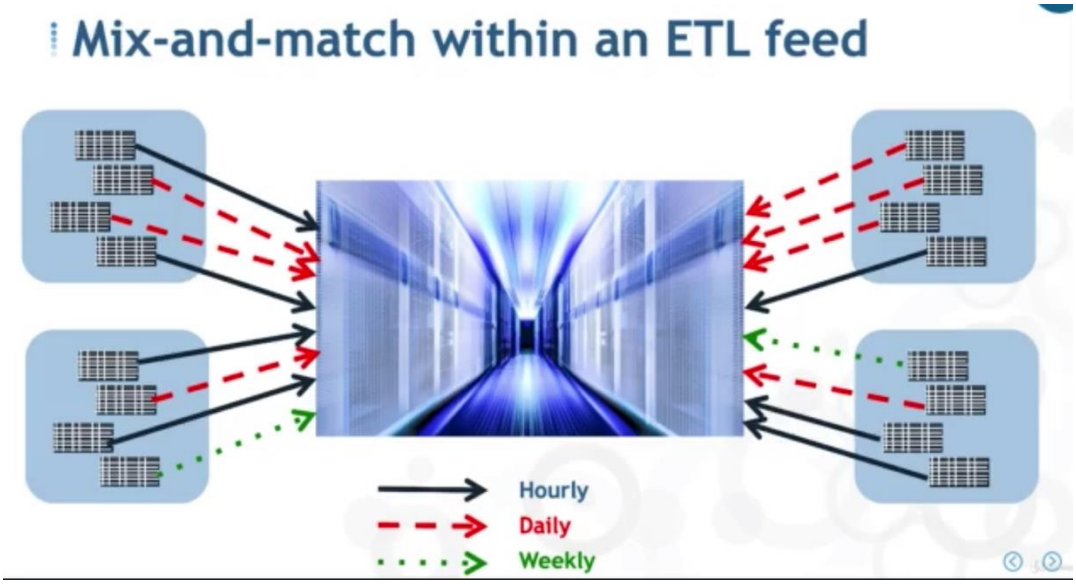
FAC_KEY	FAC_ID	FAC_LNAME	FAC_FNAME	…. RANK	HIGH_DEGREE
….. (other rows of faculty data)					
4721465	JOHNSM1	Johnson	Mary	LECTURER	PhD
….. (other rows of faculty data)					
8722752	JOHNSM1	Johnson	Mary	SR_LECTURER	PhD

Even though HIGH_DEGREE is a Type 1 change, because a Type 2 change had previously occurred, the overwrite must occur in 2 different rows (basically, both "versions" of Mary Johnson)

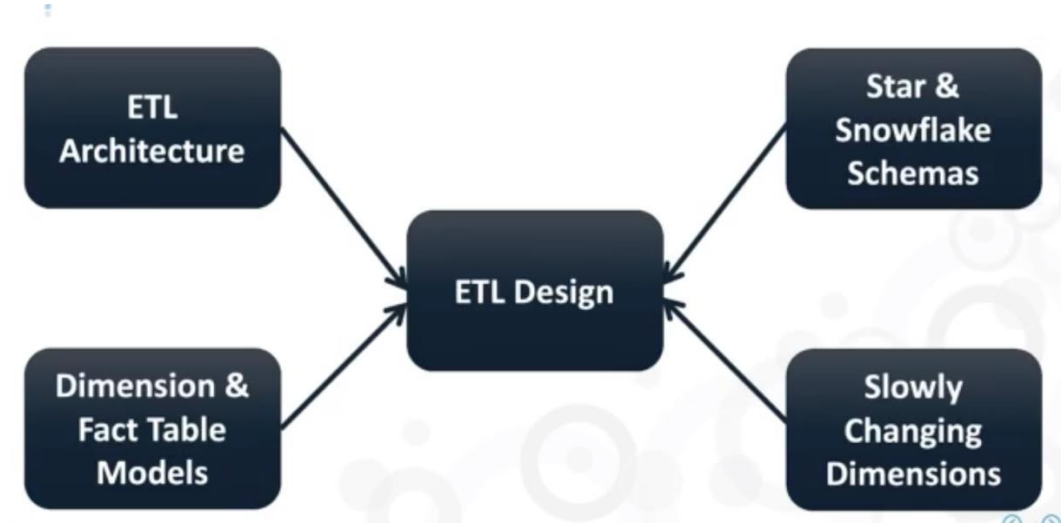
Designing Your ETL

Build your ETL Design from your ETL Architecture

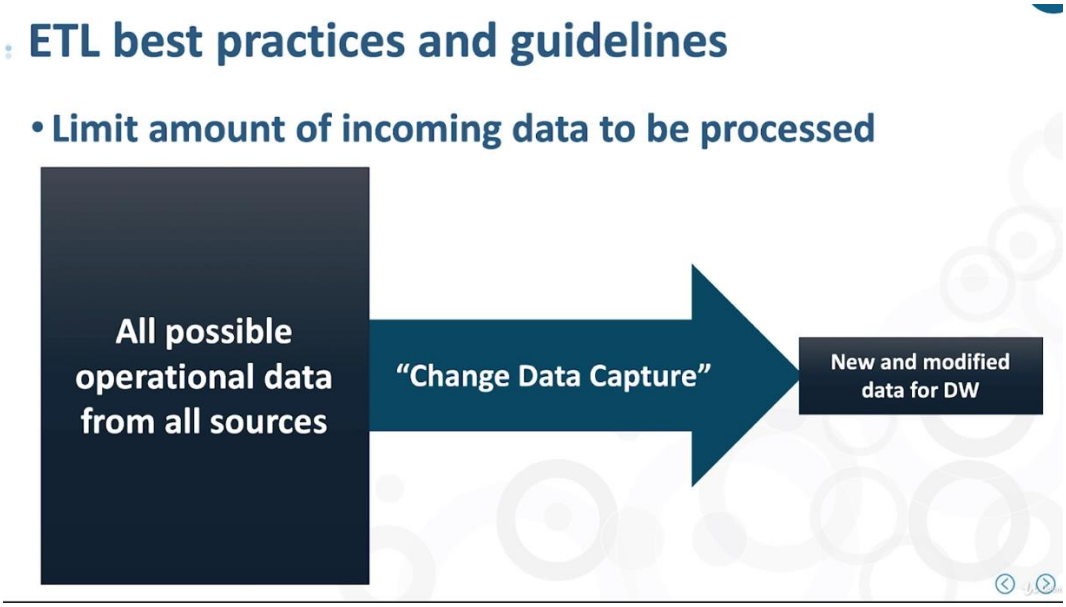


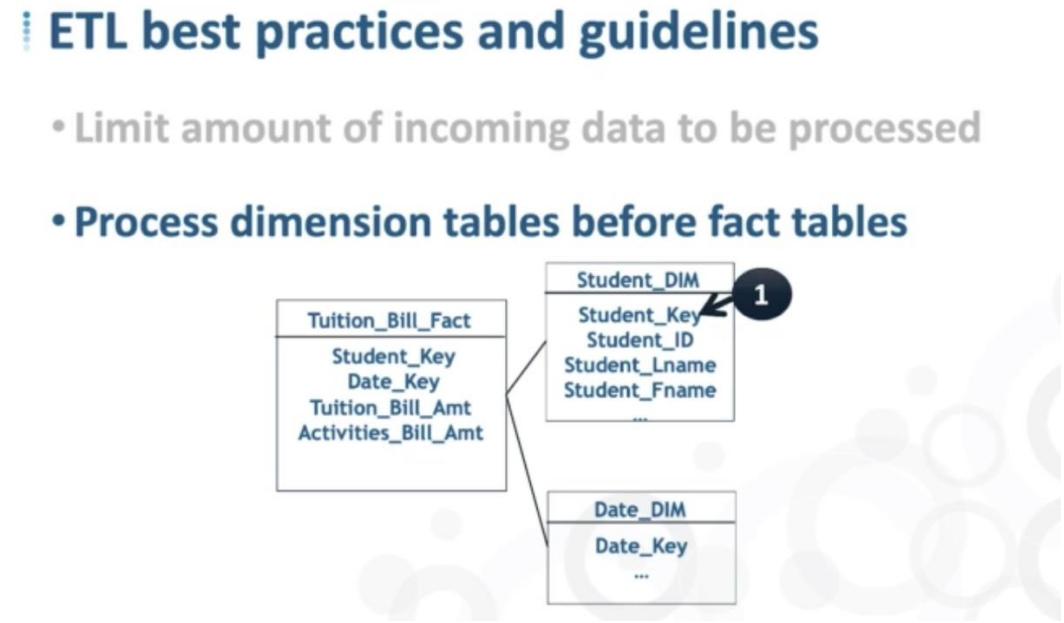


ETL Design Depends On



ETL Best Practices and Guidelines



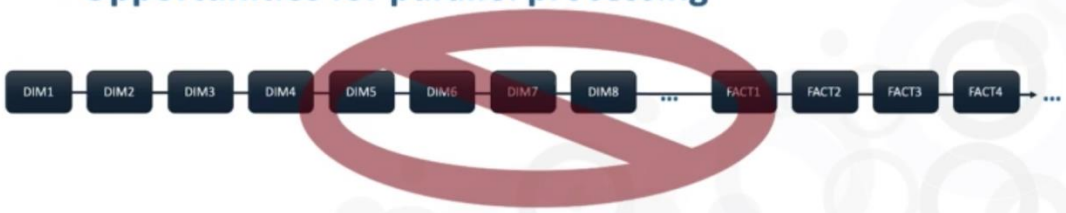


ETL best practices and guidelines

- Limit amount of incoming data to be processed
- Process dimension tables before fact tables
- **Opportunities for parallel processing**

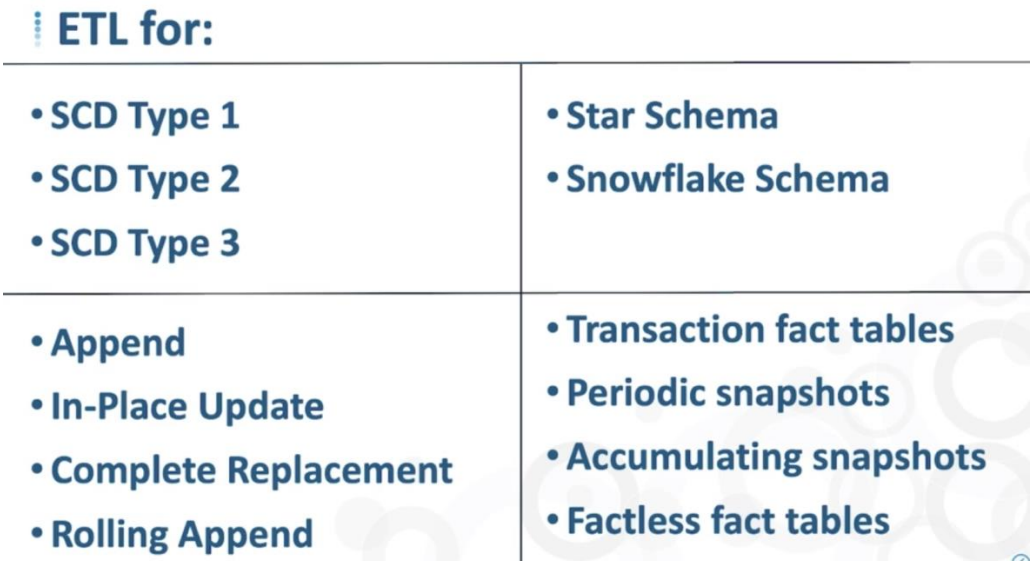
ETL best practices and guidelines

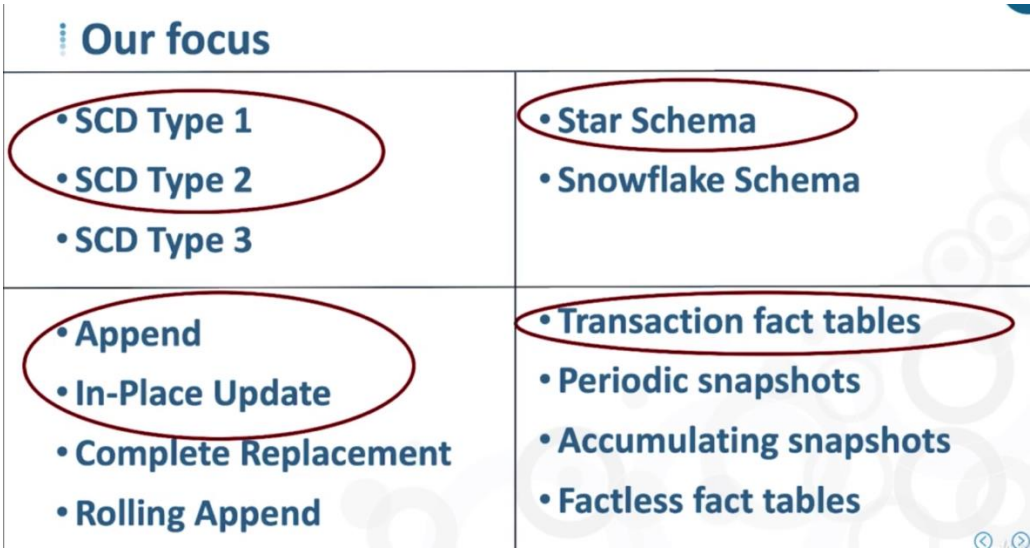
- Limit amount of incoming data to be processed
- Process dimension tables before fact tables
- **Opportunities for parallel processing**



ETL best practices and guidelines

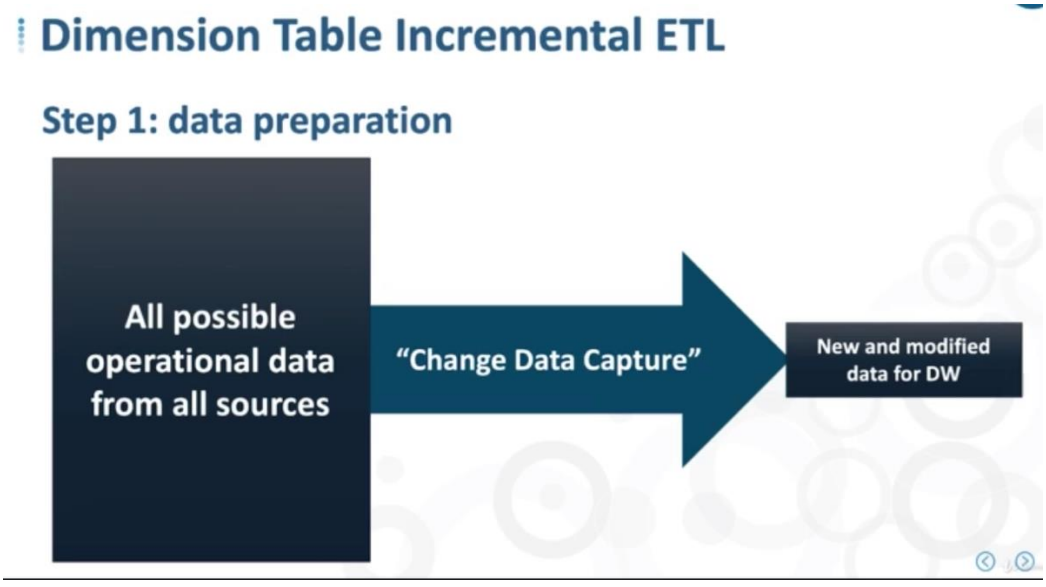
- Limit amount of incoming data to be processed
- Process dimension tables before fact tables
- **Opportunities for parallel processing**





Dimension Table ETL

Star Schema ETL



Subset of all possible data.

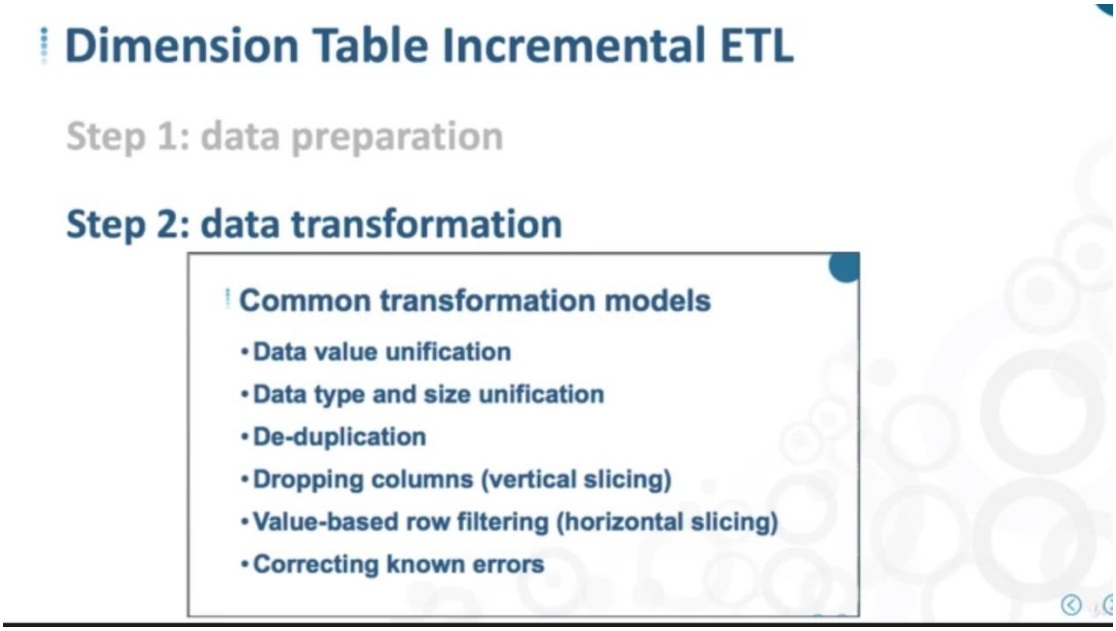
Change Data Capture

Transactional data timestamps – filtering the data based on timestamp.

Compare with Data logs – It is complex.

Compare with actual database itself.

Data Transformation



Process new dimension rows

Dimension Table Incremental ETL

Step 1: data preparation

Step 2: data transformation

Step 3: process new dimension rows

Step 4: process SCD type 1 changes

Step 5: process SCD type 2 changes

More accurately:

For each row:

If new: add to DIM table

If not new: process any Type 1 and Type 2 changes

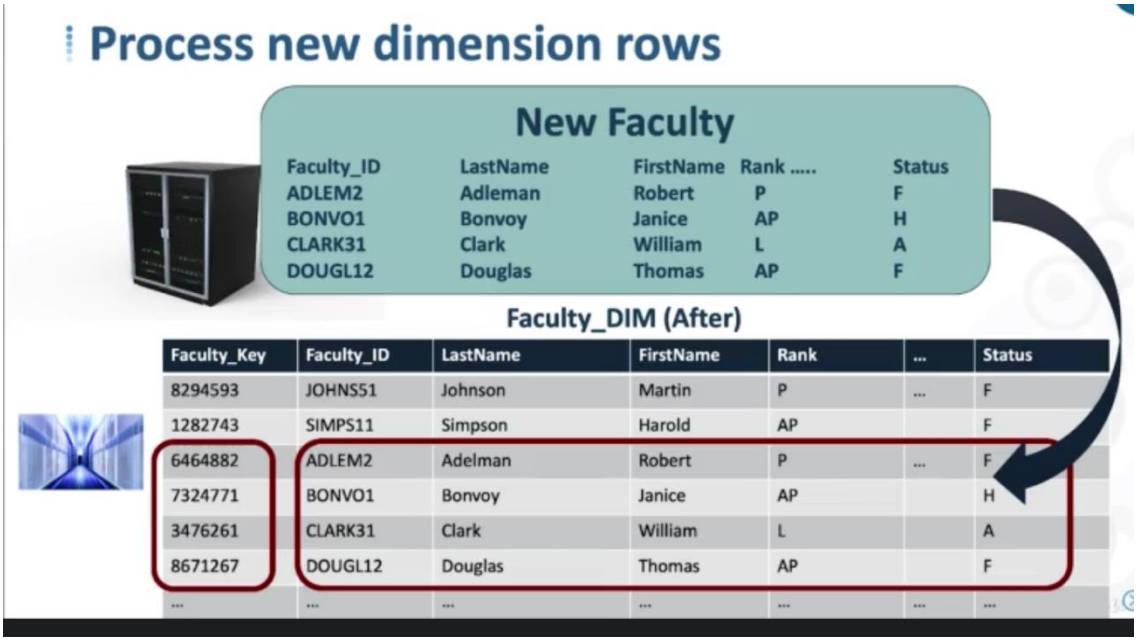
Process new dimension rows

New Faculty

Faculty_ID	LastName	FirstName	Rank	Status
ADLEM2	Adleman	Robert	P	F
BONVO1	Bonvoy	Janice	AP	H
CLARK31	Clark	William	L	A
DOUGL12	Douglas	Thomas	AP	F

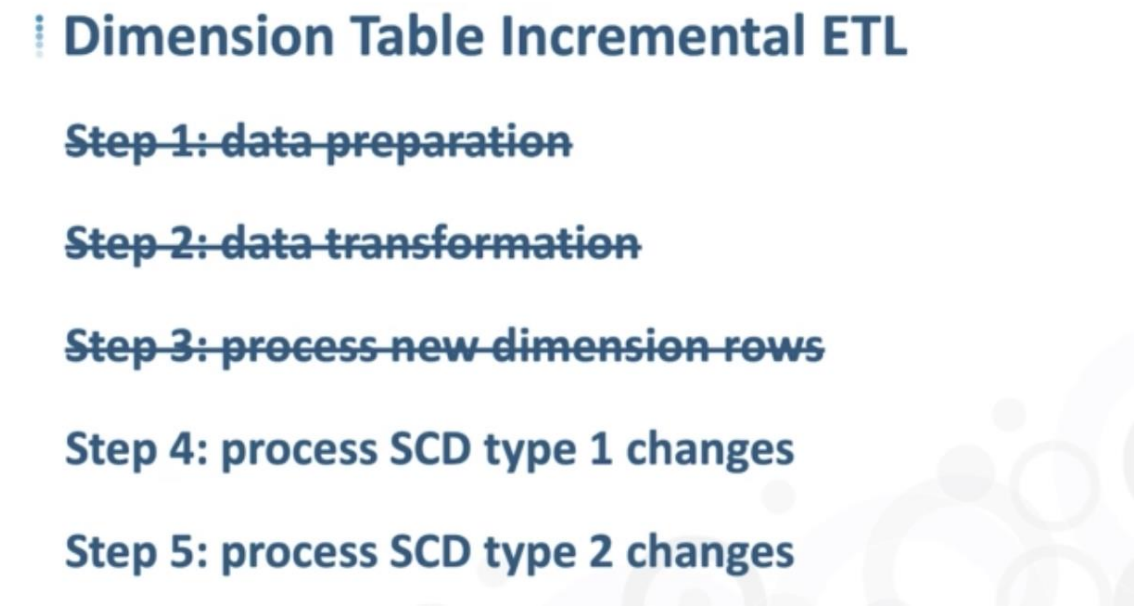
Faculty_DIM (Before)

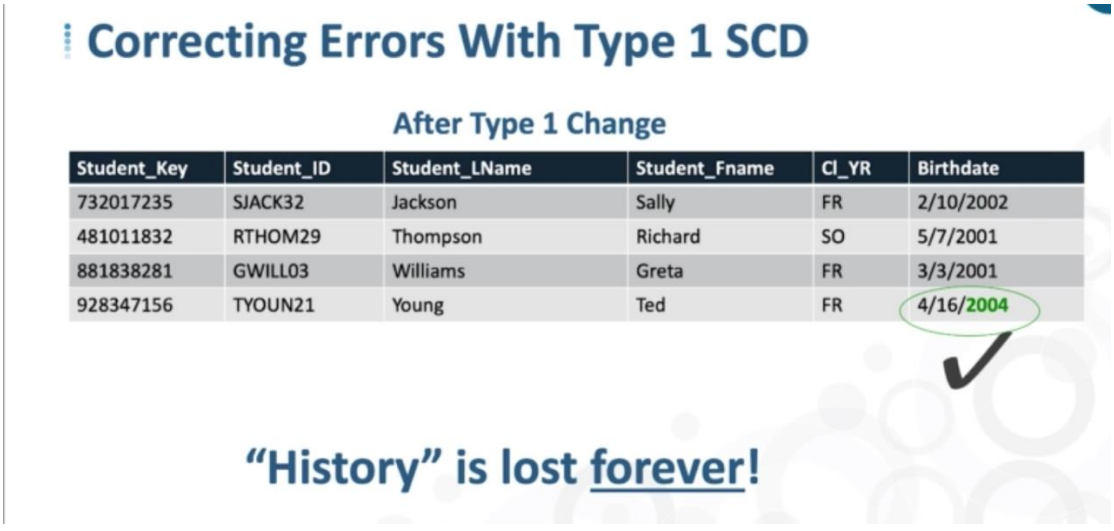
Faculty_Key	Faculty_ID	LastName	FirstName	Rank	...	Status
8294593	JOHNS51	Johnson	Martin	P	...	F
1282743	SIMPS11	Simpson	Harold	AP		F
...	...	...	...	...	...	...



Now we need to tackle, Type 1 and Type 2 SCDs

Process SCD Type 1 Changes to a Dimension Table





In place update pattern,

Remember the SCD Type 2

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	4/16/2014
318981562	TYOUN21	Young	Ted	CO	4/16/2014

We now have 2 "versions" of Ted Young:

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

Remember the SCD Type 2

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/**2014**
318981562	TYOUN21	Young	Ted	CO	4/16/**2014**

But now we have two rows of data with an incorrect birthdate!

Type 1 change to multiple rows

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**

"History" is lost forever!

Look for all dimension table rows for that natural key

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

Process SCD Type 2 Changes to a Dimension Table

Dimension Table Incremental ETL

~~Step 1: data preparation~~

~~Step 2: data transformation~~

~~Step 3: process new dimension rows~~

~~Step 4: process SCD type 1 changes~~

Step 5: process SCD type 2 changes

SCD Type 2 Example

Before 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

Basic Append… with new surrogate key

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!**

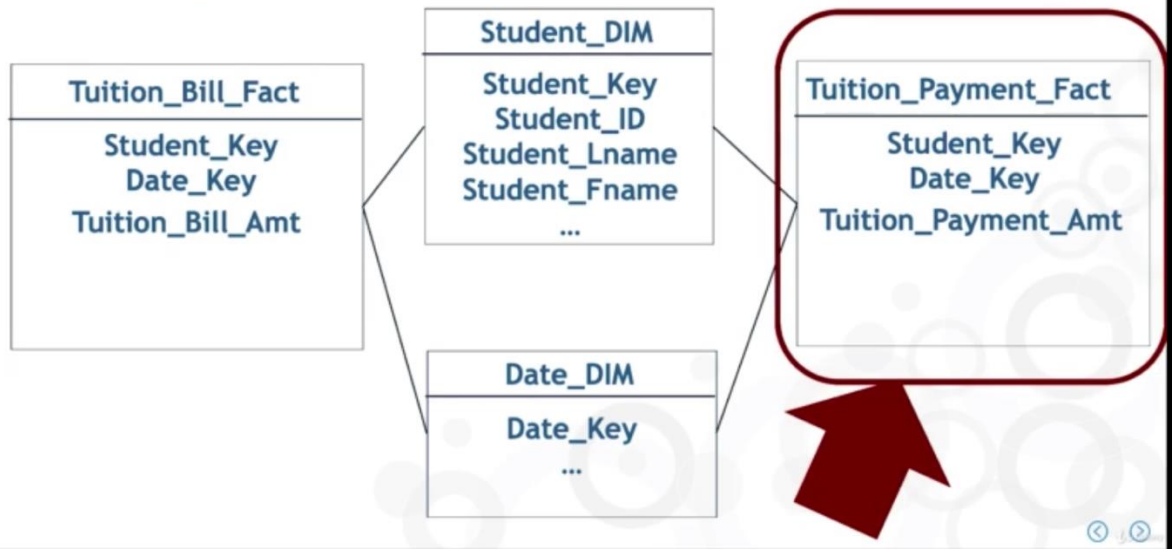
Design ETL for Fact Tables

ETL best practices and guidelines

- ~~Process dimension tables~~
- Process fact tables

Basic Append... but with a twist!

Multiple fact tables + shared dimension tables



Tuition_Payment_Fact before ETL

Student_Key	Date_Key	Tuition_Payment_Amt
775722727	838328681	$4,500.00
877292992	161632611	$4,250.00
737727273	727571666	$7,000.00
...	...	...

Included in next incremental ETL...

- Ted Young pays $5,500 in tuition
- Transactional systems provide natural keys
- **TYOUN21 – 8/12/2022 – $5,500**

But which Ted Young?

More accurately...
Which version of Ted Young?

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
229988772	TYOUN21	Young	Ted	NM	4/16/2004

There are three version for Ted Young

A third solution (Kimball, DW Toolkit)

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

We need natural keys and "relevant date"

Student_DIM

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

Student_DIM

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

Tuition_Payment_Fact

Student_Key	Date_Key	Tuition_Payment_Amt
775722727	838328681	$4,500.00
877292992	161632611	$4,250.00
737727273	727571666	$7,000.00
...	...	...
229988772	737177655	$5,500.00

If we were reloading historical data...

Student_DIM

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

TYOUN21 – 8/27/2020 – $6,000

(Current date: 12/15/2023)

We are considering the row where Date fall between Eff_Date and Exp_Date.

Student_DIM

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

Tuition_Payment_Fact

Student_Key	Date_Key	Tuition_Payment_Amt
...	...	...
229988772	737177655	$5,500.00
...	...	...
928347156	**437799915**	**$6,000.00**

Important lesson...

Watch out for complications with all dimension and fact table data during ETL!

Good job!
Correct - dimension table surrogate keys are needed for fact tables, and therefore must be processed first.

Question 1:

When designing your ETL, which types of tables do you need to process first as part of an incremental ETL run"?

- ○ Fact tables
- ◉ **Dimension tables**
- ○ Fact tables and dimension tables can be processed at the same time; it doesn't matter which is processed first.

Good job!
This is correct, based on what was explained in the lecture videos

Question 2:

One of your colleagues is helping you with ETL design, and is confused about the proper order of the steps to follow. You walk to the whiteboard and write down the order of steps. Which of the following answers is what you will write?

- ○ 1) Data transformation; 2) data preparation; 3) process and load Type 1 and 2 changes; 4) process and load new dimension table data; 5) process and load new fact table data
- ○ 1) Data preparation; 2) process and load new dimension table data; 3) process and load Type 1 and 2 changes; 4) process and load new fact table data; 5) data transformation
- ◉ **1) Data preparation; 2) data transformation; 3) process and load new dimension table data; 4) process and load Type 1 and 2 changes; 5) process and load new fact table data**

Good job!
This is correct, because now we will have "2 versions of the same dimensional data" based on the rules of the Type 2 change

Question 3:

When doing ETL processing for a Type 2 SCD (slowly changing dimension) change, you will:

- ○ Overwrite and replace an existing value in a dimension table row
- ○ Add a new dimension table row with the new data and a new surrogate key, and then change the surrogate key/primary key of the old dimension table row to be the same as the new row's surrogate key
- ◉ **Add a new dimension table row with the new data and a new surrogate key, and leave the old/existing dimension table row exactly as is...except for possibly changing the "Current" flag to "N" and/or updating the expiration date for that row**

Good job!
This is correct, as explained in the lecture video. You still need to carefully design which tables are able to be processed in parallel based on relative sizes and other factors

Question 4:

Which of the following statements is most true for when you are designing each incremental ETL job for your organization's new data warehouse?

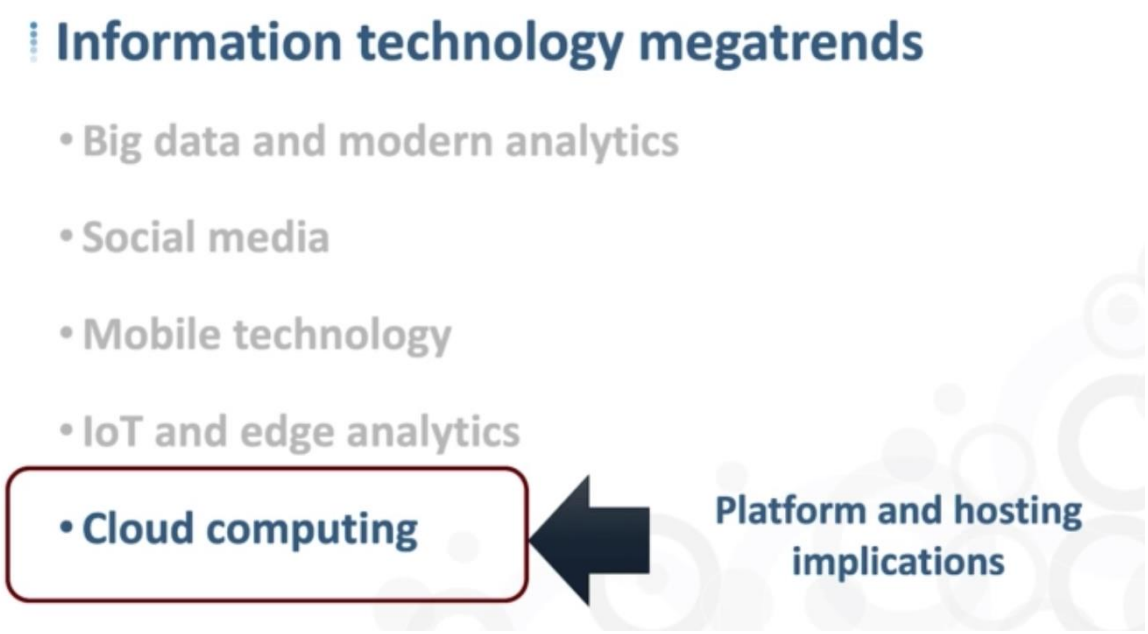
- ○ You can only process one dimension table at a time, but you can process multiple fact tables in parallel with each other
- ○ You can only process one fact table at a time, but you can process multiple dimension tables in parallel with each other
- ◉ **You can process dimension tables in parallel with each other, and after all of your dimension tables have been processed you can then process fact tables in parallel with each other**

Selecting Your Data Warehousing Environment

Cloud Computing

Decide Between Cloud and On-premises Settings for Your Data Warehouse

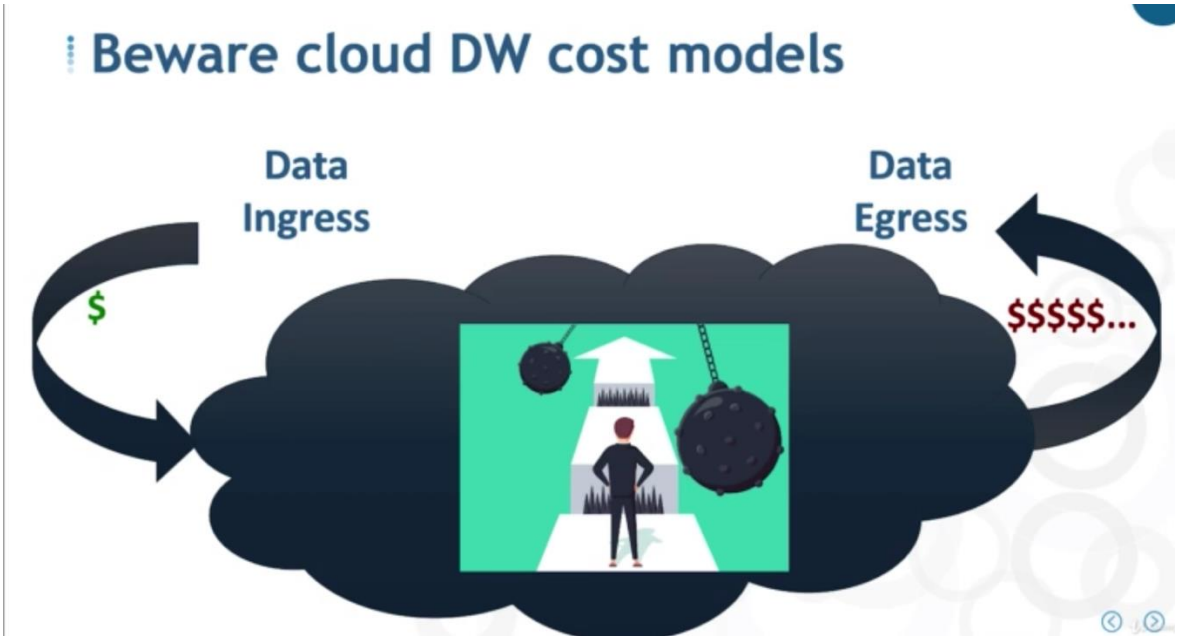
Information Technology Megatrends



Advantages to cloud-based data warehousing

Challenges to cloud based data warehousing

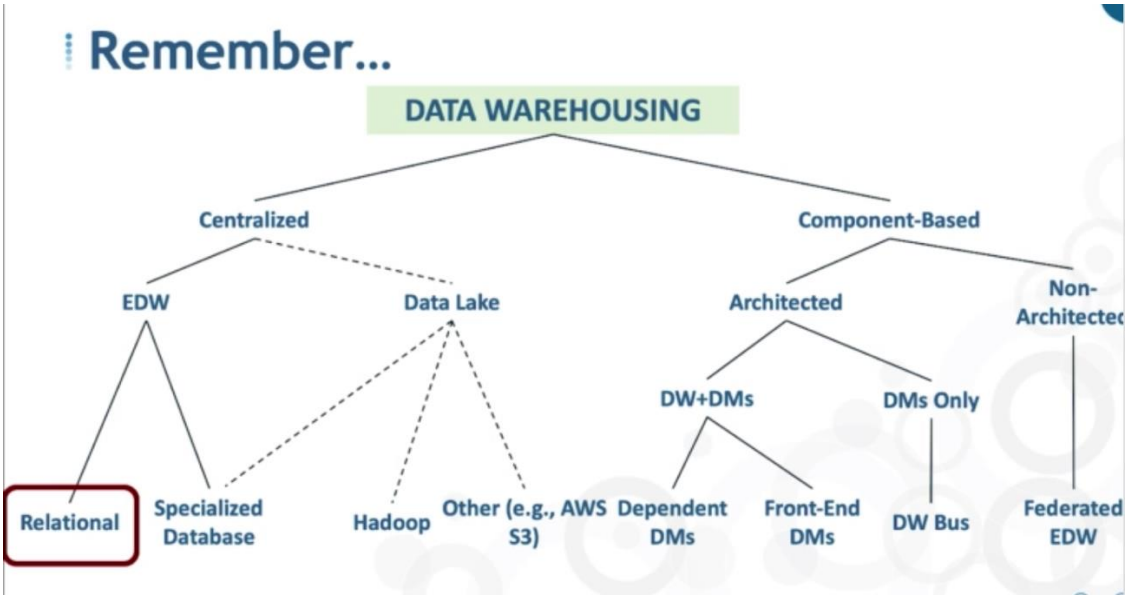
Cloud DW cost models



Data egress cost challenges

- Costs for large data volumes
- Not just operational data egress costs
- Development, testing, QA, problem chasing,…
- **Experimental or exploratory analytics**

Architecture and Design Implications for Your Selected Platform



Platform vs. user view and data model

Platform	User View*	Physical Data Model
On-Premises RDBMS	Dimensional	Star/snowflake schema
On-Premises Data Mart Cube	Dimensional	Multidimensional
On-Premises Specialized DB	Dimensional	Columnar, in-memory, etc.
Cloud Hadoop	Dimensional	HDFS (SQL access)
Cloud AWS S3	Dimensional	S3 buckets (SQL access)
Cloud RDBMS	Dimensional	Star/snowflake schema
Cloud Data Mart Cube	Dimensional	Multidimensional
Cloud Specialized DB	Dimensional	Columnar, in-memory, etc.

***For OLAP/Business Intelligence**

Trends

Trends and Trajectory

Platform	User View	Physical Data Model
On-Premises RDBMS	Dimensional	Star/snowflake schema
On-Premises Data Mart Cube	Dimensional	Multidimensional
On-Premises Specialized DB	Dimensional	Columnar, in-memory, etc.
Cloud Hadoop	Dimensional	HDFS (SQL access)
Cloud AWS S3	**Dimensional**	**S3 buckets (SQL access)**
Cloud RDBMS	**Dimensional**	**Star/snowflake schema**
Cloud Data Mart Cube	Dimensional	Multidimensional
Cloud Specialized DB	**Dimensional**	**Columnar, in-memory, etc.**

Trending upward as DW platform

Implications

- Data warehousing shifting to cloud
- Data lakes alongside or succeeding data warehouses
- RDBMSs still prevalent…
- **…but being supplemented or replaced even for OLAP/BI**

Good job!
This is true based on the lecture video

Question 1:

One of your company executives asks you "what's the deal with cloud computing and data warehousing?" Which of the following statements should be part of your answer?

- ○ Today, data lakes are typically built on cloud platforms, while data warehouses are almost always built in corporate data centers.
- ◉ **Today, data warehouses are found in both corporate data centers and in the cloud, but the numbers of DWs being built in the cloud is steadily increasing**
- ○ Today, cloud-based data warehouses are commonly being migrated from cloud platforms to corporate data centers

Good job!
This is true based on the lecture video

Question 2:

When building a data warehouse (or data lake) in a cloud platform:

- ○ Data "egress" (accessing data) is the lowest-cost component to your overall pricing and cost of ownership
- ◉ **You need to carefully plan for and watch the costs associated with your data "egress" (accessing data)**
- ○ You don't need to worry about data "ingress" or "egress" because the cloud hosting companies provide those services for free

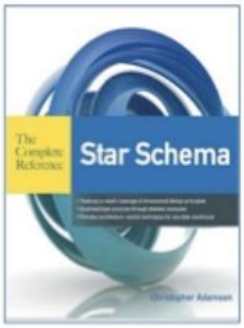
Good job!
This is true, based on the lecture video

Question 3:

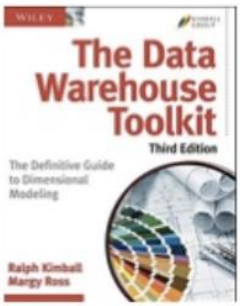
Suppose that one of your colleagues asks you to summarize the relationship between how many of your users will typically view the data from your data warehouse versus the underlying platform. Which of the following statements is most correct for your answer?

- ○ The user-facing view of a data center-hosted data warehouse is typically dimensional, but a cloud-hosted data warehouse is typically not dimensional.
- ◉ **Regardless of whether a data warehouse is built "in the cloud" or a corporate data center, users will typically still see the data dimensionally.**

Additional Resources for Advanced Data Warehousing Topics



Star Schema: The Complete Reference
Christopher Adamson
McGraw-Hill Education, 2010
ISBN-13: 978-0071744324



The Data Warehouse Toolkit (3rd edition)
Ralph Kimball
John Wiley & Sons, 2013
ISBN-13: 978-1118530801