

1. Data Warehousing conceptsa. Introduction to Data Warehousing concepts.

What is DWH? Not same as Database

DWH usage      DB platform



Data comes from elsewhere

→ Possibly a dozen  
of them.

4 Rules of DWH Data is copied.

DWH → Integrated environment

Subject oriented

Time variant (Historical data)

Non volatile (DWH remains stable as it is refreshed)

- Improvement in data quality.

b) Reasons to build DWH

Data driven decisions

Past

Present

Future

The Unknown

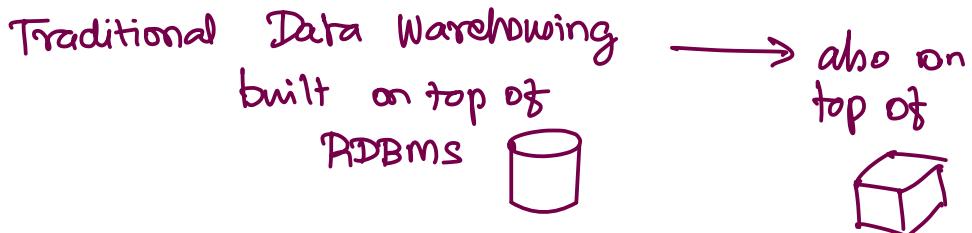
BI  
Business Intelligence

One stop shopping

Before DWH  
we need to get  
or explore more  
sources.

$$BI + DWH = \text{Best Value}$$

### ③ Compare DWH and Data Lake



Data Lake Built on top of  
Big Data

The 3 V's

<u>Differences</u>	Volume	(amount of data)
	Velocity	(takes new and changed data rapidly)
	Variety	(structured and unstructured) Semi-structured.

DWH → DL      Some companies have both DWH & DL

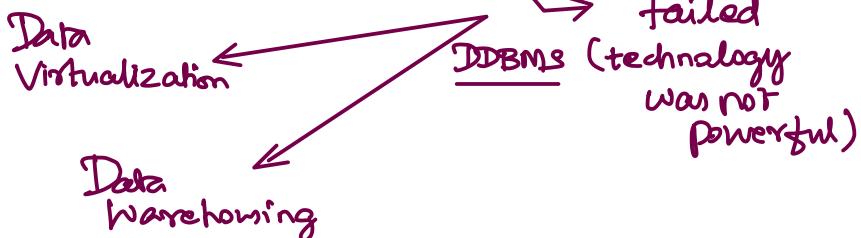
Blurring Differences

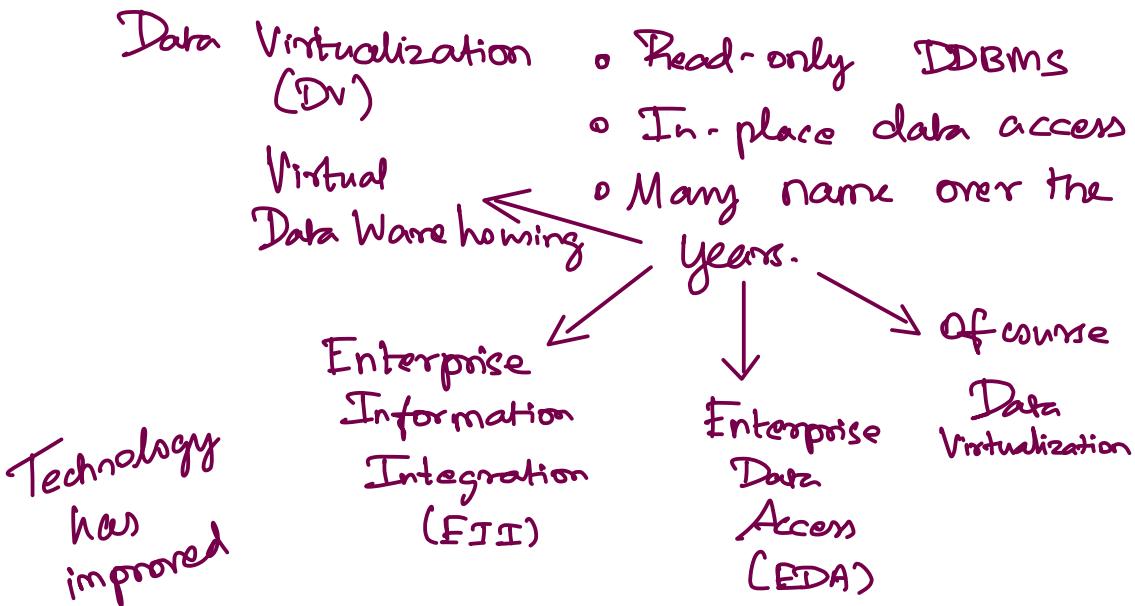
BI + DWH/DL = Business Value

### ④ DWH and Data Virtualization

DWH History      1980's era extract files.

late 1980's Distributed DBMS



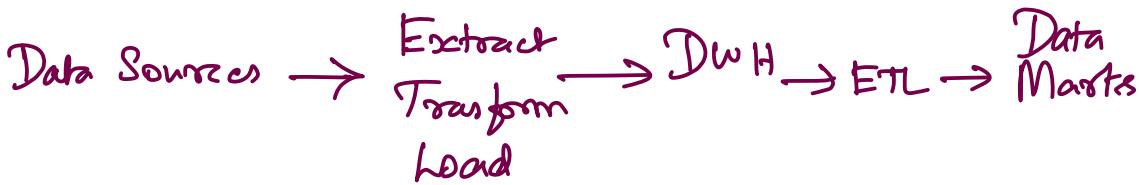


- DV use cases
- Simple transformations
  - Smaller number of data sources
  - Relaxed response time

BI + DWH / DL / DV = Best Value.

## ② Simple End to End DWH Environment

Typical DWH



Eg Suppliers → Wholesalers → Retailers

## 2. Data Warehousing Architecture

### a) Build a Centralized Data Warehouse

Single DWH Environment

Single database

One stop shop

Historical Challenges

- Technologies (solved)
- Work processes (solved)
- Org. and human factors

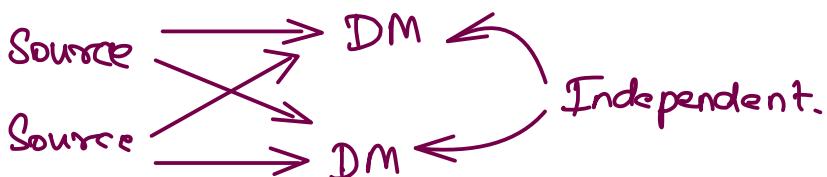
Today's challenges

### b) Compare a DWH to Data Mart (DM)

Data Sources → ETL → DWH → ETL → DM

Dependent DM → Existence depends on existence  
of DWH

Independent DM



Dependent DM (Vs)

- Sourced from DWH
- Uniform data across marts
- Architecturally straight forward.

Independent DM

- Sourced directly from applications and systems
- Little or no uniformity
- Spaghetti architecture

DWH

Many sources  
(dozens or even 100s)

ETL from sources

Probably large data volumes

Dimensionally organized data

vs.

Independent DM

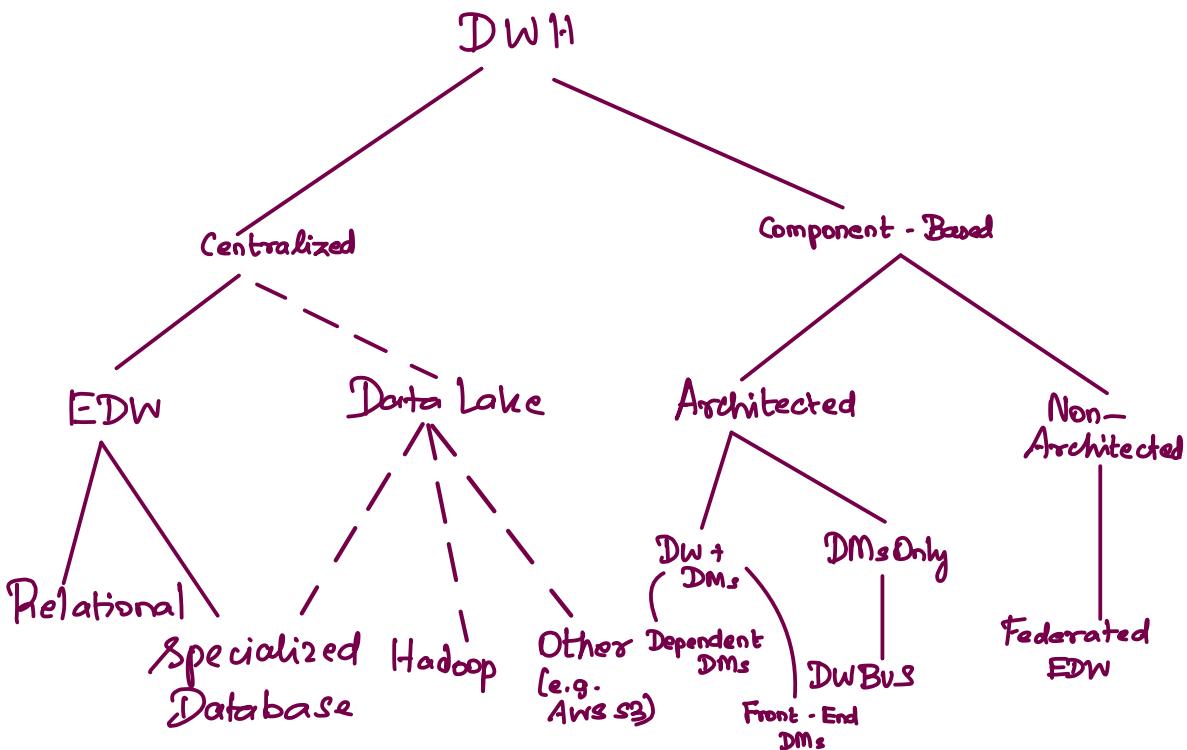
One or more sources  
(no specific number typically < 10)

ETL from sources

Possibly large data volumes

Dimensionally organized data

## ③ Your DWH Architectural Options



Cons

- Centralized → Pros
- Default option
  - One stop shopping
  - Modern Technology

### Requires

- High cross org cooperation
- High data governance
- Ripple effects

Cons

- Component Based Pros
- Decomposition ↘
  - Mix and Match tech.
  - "Bolt" together components
  - Overcome org challenges

Often inconsistent data  
Difficult to cross integrate  
Component Based

Centralized  
EDW or Data enterprise data仓库 lakes

(refers the tree in previous page)

### C.B Architected

DMs Only



DW Bns

### Confirmed Dimensions

### C.B Non-architected

Federated EDW

"Let's agree to disagree"

Collection of DMs

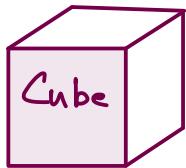
→ "Only as the last resort"

### Front-end DM

Instead of having DM behind a DWH, we have DM in front of the DWH. The DM is the place where business analytics takes place. We can have some data from the DM flowing into the DWH.

Better to use DWH & DL

## d) Including cubes in your DW/H environment.



Cube = Multi Dimensional DB

Not a RDBMS

Specialized "dimensionally aware" database

heading alternative for first gen. DW/H

Today = Best for smaller-scale DWs, DMs

### Advantages

Fast query response time

"Modest" data volumes

### Dis-advantages

Less flexible data structures than RDBMS

More vendor variation than RDBMSs

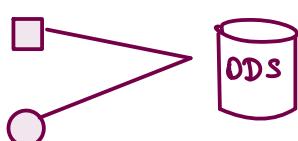
### Today

RDBMS + MDBMS = BEST VALUE

## e) Include Operational Data Stores in your DW/H Environment

### What is Operational Data Store ODS

Multiple Data Sources



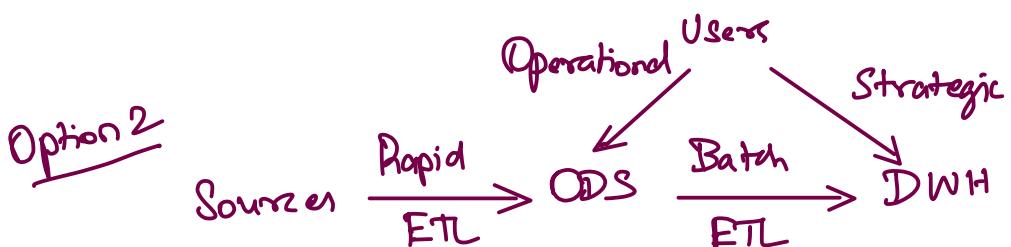
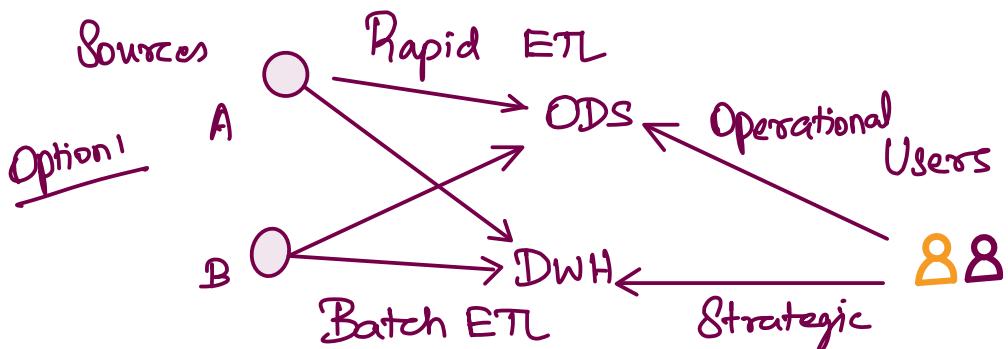
Integrates from multiple sources

Often real time  
Source → ODS data  
(not batch) feeds



IMP: Emphasis on current operational data

ODS ← Tell me what's happening right now!  
Popular late 1990s / early 2000s.



ODS → less popular

Faster and more current DWs

Superceded by Big Data

ODS components within Data Lake

Traditional ODSs still used in mission critical situations.

BI +

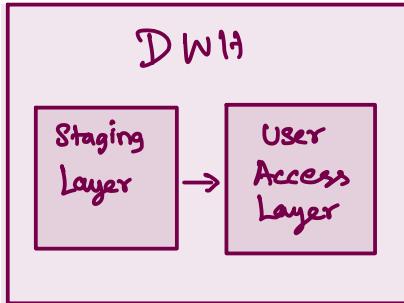
DWH  
and/or

DL  
and/or

DV  
and/or  
ODS,

= Best  
Value

## ④ Explore the role of Staging layer



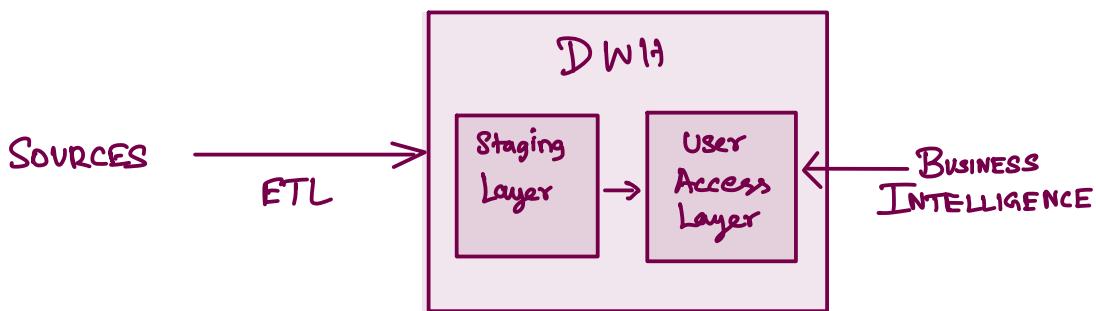
### Staging Layer

- Landing zone
- "E" within ETL
- 2 Variations

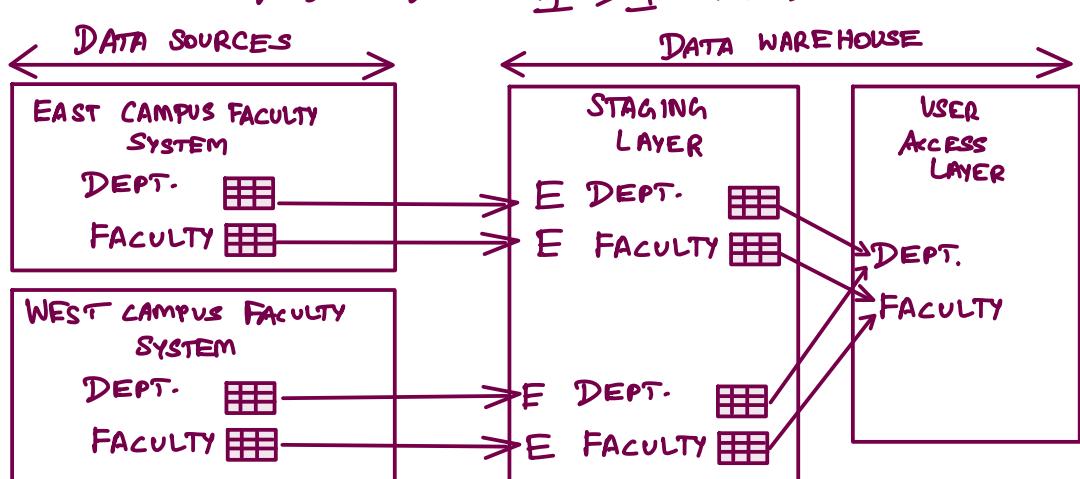
### User Access Layer

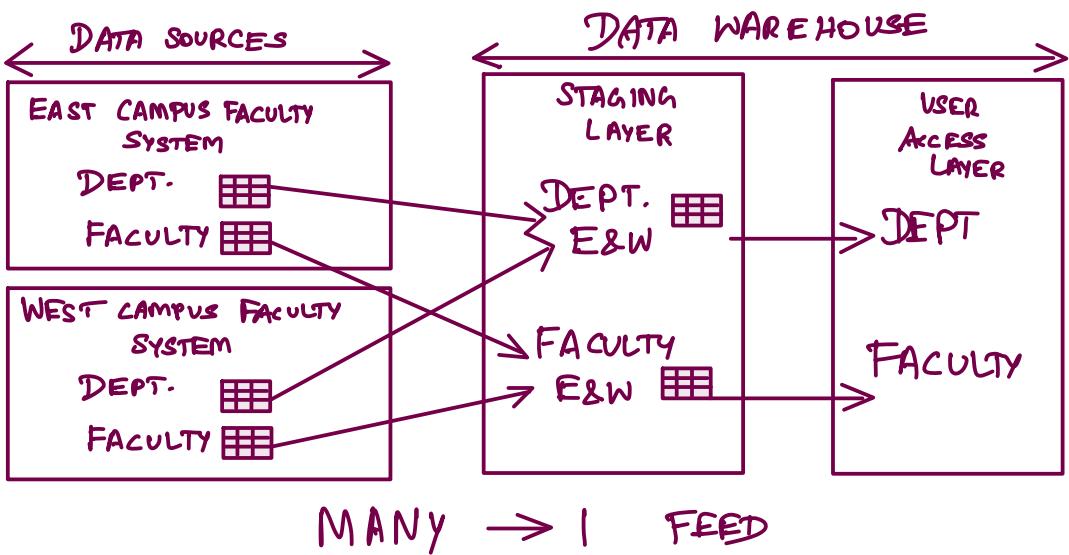
- Where users go
- Dimensional Data

"Pull the data in a non-intensive way"



### Inside Staging layer





Many to 1

Multiple instances of identical source applications  
(Mostly uniform across data marts)

1 to 1

Multiple customized instances of source applications  
Different vendor apps for the same function

Focus on  
"E" Extract

MINIMAL  
"T" Transform

## Q) Comparing Two Types of Staging layers

### Two types

#### Non-Persistent

Prior to each update  
it is empty.

Once data is moved to  
DW/H the NPSL is cleared

SL EMPTY

Source → SL

SL → UAL

EMPTY SL

#### Persistent

SL has the data  
We don't empty  
out  
Data is persisted.

SL has data

Source → SL

SL → UAL

SL has data

### NPSL Advantages

- Less storage space
- Data already moved to user access layer

### Disadvantages

- Need to go back to source to rebuild user access layer
- Data QA also requires source systems.

### PSL

#### Advantages

Rebuild user layer  
without source systems

Data QA: compare staging  
layer with user layer

#### Disadvantages

More storage space  
Risk of "ungoverned"  
access.

### 3. Bring Data into your Data Warehouse

#### @ Compare ETL vs ELT

Extract      Transform      Load      Extract      Load      Transform

Quickly pull data

Batches

Raw data ... with errors  
and all ...

Load data in ~~staging layer~~

#### Transform

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

#### Load

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

#### Challenges around ETL

Significant business analysis } Before loading  
Significant data modeling } the data.

Change the order

ELT instead of ETL

- Blot data into big data environment.
- Raw data in Hadoop HDFS, AWS S3, etc.
- Using big data environment computing power to transform when needed.

"Schema on read" vs. "Schema on write"

ETL

You need the schema before you read the data from source

ELT

schema and the associated analytics preparatory work can happen just before write.

For traditional DWH → ETL

For DL or DWH · DL Hybrid → ELT

⑥ Designing the initial load ETL

Source → ETL → DWH

Initial ETL and Incremental ETL

'first time'

'then on'

## Initial ETL

Normally one time only.

Right before the DWH goes live.

All relevant data necessary for BI and analytics.

Redo it data warehouse blows up.

DON'T BRING IN ALL POSSIBLE DATA  
BRING ONLY WHAT IS  
'RELEVANT'

What to bring into DWH ?

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

Once Ready to go!

from then on "Incremental ETL"

② Compare different models for incremental ETL

2 variations of ETL

Initial ETL

Incremental ETL

## Incremental ETL

- Incrementally refreshes the DWH
- New data
- Modified data
- Special handling for deleted data: We keep data for historical purposes but mark it as deleted or inactive.

## BRING THE DWH UP TO DATE ✓

### 4 MAJOR INCREMENTAL ETL PATTERNS

- Append - append new info on top of existing data
- In-place update - update data in place
- Complete replacement - total overwrite of the data
- Rolling append - You maintain only certain history. e.g. You maintain only 36 months of data. You append data and the time window keeps rolling.

## ETL Today

Append

In-place update.

## d) Explore the role of data transformation

Data transformation overarching goals

Uniformity

Apple  $\leftrightarrow$  Apple

Restructuring

### COMMON TRANSFORMATION MODELS

- part of d Refers to e
- 1. Data value unification
  - 2. Data type and size unification
  - 3. De-duplication
  - 4. Dropping Columns (vertical slicing)
  - 5. Value-based row filtering (horizontal slicing)
  - 6. Correcting known errors
- Uniformity      Restructuring

#### 1. Data Value Unification

e.g Faculty Rank

System 1

Professor  
Asst. Professor  
Lecturer

System 2

Rank

P

AP

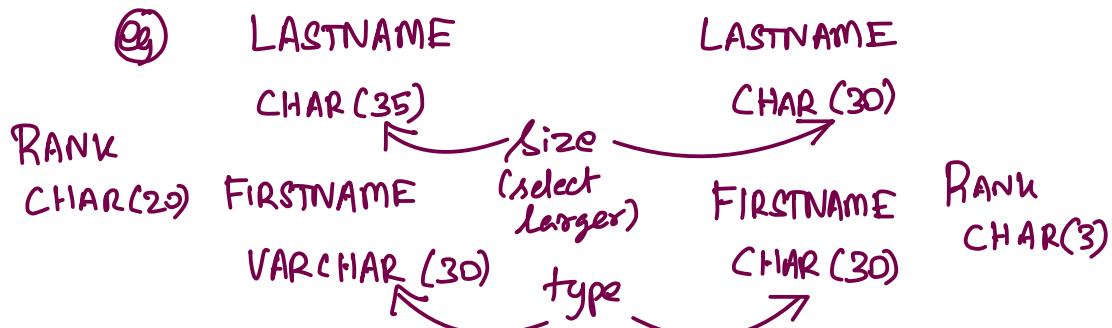
L

Unify the data values

e.g Faculty Master Dimension

lets say we chose Rank column values  
as P, AP, L

## 2. Data type and size unification



We need to unify the types and values

### Faculty Master Dimension

LASTNAME      CHAR(35)  
FIRSTNAME      CHAR(30)

If we decide to abbreviate the rank

RANK      CHAR(3)

## 3. De-duplication

Say a student goes 2 campuses so the student appears in the student table of both campuses. Solution: De-duplication

## ② More data transformations within

You ETL

4. 5. 6. are discussed in this section

## 4. Dropping Columns

e.g. Some columns may not get used by business analytics. So we just go ahead and ignore these columns as part of ETL process.

FACULTY MASTER DIMENSION only has required columns → used for analytics.

## 5. Value based row filtering (horizontal slicing)

### Students

FN	LN	YEAR	MAJOR
----	----	------	-------

Business  
Engineering  
Literature

Assume we build a data mart for "College of Business"  
The users might be interested in students who are having their major as "business"

## 6. Correcting Known Errors

Fix all the known data errors so that reports are accurate!

## f) Implementing Mix and Match Incremental ETL

Among ETL feeds ... frequency might differ  
Daily, Hourly, Weekly  
... approaches of handling data  
Complete, Append, Replace, Rolling append  
Replacement  
Mix and match within an ETL feed from various  
source systems

Design and build what  
makes sense for your environment

Context is the key  
as in architecture "It depends"

## 4. Data Warehouse Design : Building Blocks

### ① How will your data warehouse be used?

Making data-driven decisions

Past      Present      Future      Unknowns

BI + DWH = Best Value ← Remember?

BI Category drives data model

Basic Reporting

Dimensional

Online Analytical

Dimensional

Processing (OLAP)

Predictive Analytics

Data Mining / Specialized

Exploratory Analytics

Data Mining / Specialized

### ② The basic principles of dimensionality

We will focus on first 2

• One or more measurement

• Dimensional context for each measurement

Eg. 5312 ← some measurement

some F sign on the front

F 5312

what does this amount mean

Some more context

F 5312 payment

Now we know it is a payment for something, some service, we don't know much.

Absence of context is the problem!

Providing dimensional context

by using the words 'By' 'For'

e.g. Typical dimensional insights

- What is the average annual faculty salary by rank, by department, by academic year.
- What is the student loan balance by major, by class year, by campus?

MEASUREMENT

DIMENSIONAL CONTEXT

Some times the questions may be specific dimension out of all dimensional contexts

e.g. for assistant professors

for engineering majors

## WORDING

## USAGE (almost always)

By

For

- "Sliced and grouped" by values of the entire dimension.
- One or more specific values from the entire dimension.

## Implied wording

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

## Using "by" instead of "for" for grammatical reasons

What is the total number of vacation days taken

Last academic year by administration employees?

→ implied "for"

→ sounds more natural than  
"for administration employees"

## The critical lesson

MEASUREMENT + CONTEXT = INSIGHTS FOR

DATA-DRIVEN DECISIONS

- (C) Compare facts, fact tables, dimension, dimension tables

- Measurements → Facts
- Dimensional context → Dimensions

Facts Numeric Quantifiable, Think "measurement"

- Eg • Salary      • Number of credits  
• Dollar amount      • Number of years

DWHT fact is not the same as a logical fact

---

If we use RDBMS store facts in a fact table

Fact  $\neq$  Fact Table

We will look at

- Deciding which facts can go into the same fact table
- Different types of fact tables
- How to connect fact tables and dimension tables.

(Eg) Of dimensions

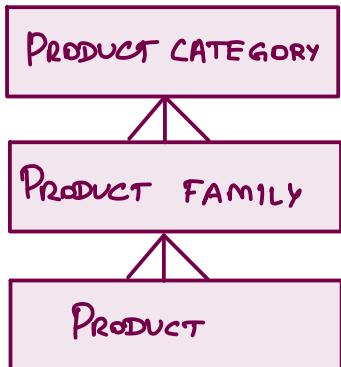
Academic Department, Major, Student, Employee

We store dimensions in dimensions table

Sometimes we interchange the phrases  
dimension and dimension table

## (Q) The product dimension

Hierarchy



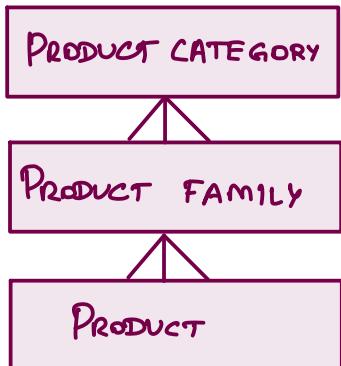
3

levels of dimensions

Why the confusion?

Difference is between  
Star vs. Snowflake  
schema

## Star Schema

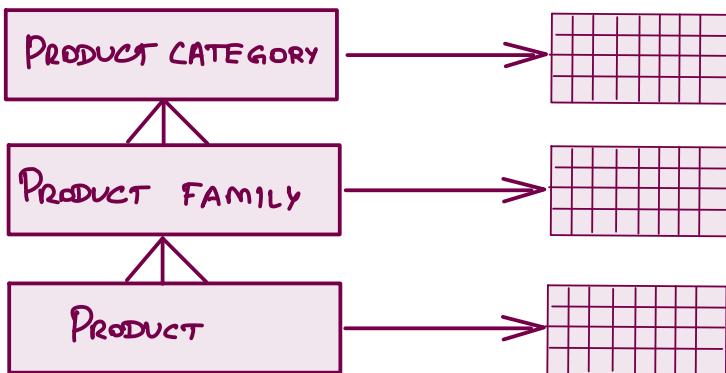


- 1 Dimension table known as product dimension(table)


- Actually 3 dimensions (from same hierarchy) in 1 table

# Snowflake Schema

3 dimensions      3 dimension  
tables



(d) Compare different forms of additivity

in facts

A data warehouse fact can be

### Additive

An additive fact can be added under all circumstances

(e) A faculty member salary

Say add all faculty member salaries

(f) Add salary of a faculty salary over the years

### Non additive

Cannot be added!  
Grade Point

Average  
GPA  
You cannot add average or ratios.

NOTE: Additivity means ability to add.  
Typical non additive facts  
Margins, Ratios, Percentages, Calculated average

### Semi-additive

Sometimes you can add, but other times you cannot.

Typically used in "periodic snapshot fact tables"

## 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.

## ② Compare Star Schema to a snowflake schema

OLAP = Dimensional Analysis

supported by

Underlying dimensional data

Facts, Dimensions,

Fact tables and  
Dimension Tables

Star Schema

- All dimensions along a given hierarchy in one dimension table
- Only one level away from fact table along each hierarchy
- With one fact table visually resembles a star
- "Denormalized" dimension table data ↳ means more repetitive data

Snowflake schema

Each dimension / dimension level in its own table

One or more levels away from fact table along each hierarchy

With one fact table visually resembles a snowflake.  
"Normalized" dimension table data

## Star Schema

Department - Dim

Expense - Category - Dim

Budget - Fact

Time - Dim

## Snowflake Schema

College - Dim

Department - Dim

Budget - Fact

SAME DIMENSIONS  
DIFFERENT TABLE  
REPRESENTATION

Academic - Year - Dim

Semester - Dim

Expense - Category - Dim

Expense - Category - Grouping - Dim

OPEX - CAPEX - Dim

# F) Database Keys for data warehousing

## Fundamentals

- Star or Snowflake schemas implemented in RDBMSs
- RDBMSs use logical relationships to relate data across tables.
- Technically any data can be related to any other data
- "Official" relationships handled through keys

## RDBMS Keys

Primary vs. Foreign keys

Natural vs. Surrogate keys

### PRIMARY KEYS

A unique identifier for each row in a database.

Single column or more than one column

Faculty Master Dimension e.g. FacultyID

### FOREIGN KEY

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

Faculty Master

FacultyID

Department Master

Dept-ID ----- Dept-ID

## NATURAL KEY

## Might be cryptic

eg

Faculty-ID  
 32132 }  
 48789 }  
 83156 }



(Or)

## Might be understandable

### Dept-ID

ACCT1  
 MKT1  
 MAT1  
 COMP1  
 FIN1

"Natural keys "travel" from source systems with the rest of the data"

"Best practice is to use surrogate keys to relate data across tables"

## Surrogate Keys

- No business meaning
- Different than a cryptic natural key
- Generated by the database itself or a supplemental "key management" system.

### Faculty Master

Faculty-Key
12498
12349
41323

Faculty-ID
123892
564321
789321

This is used in the database to relate dimensions.  
 eg. Faculty to Dept.

# Key DWH design decision about keys

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

## 5. Design facts, fact tables, dimensions and dimension tables

### @ Design dimension tables for star schemas and snowflake schemas

Recall Dimension  $\rightarrow$  context for measurements (facts)

Dimension  $\neq$  Dimension tables.

Table  $\rightarrow$  RDBMS

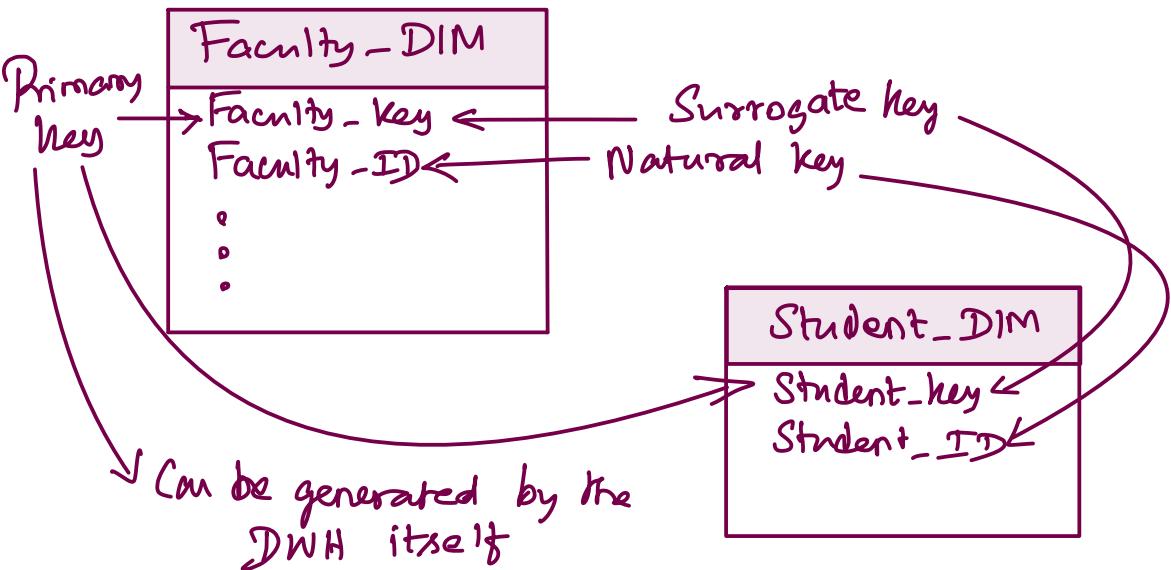
RDBMS table requires Primary 

DWH Primary  = Surrogate 

Representative Dimensions

Faculty

Students



## Dimension tables

Key DWH subject areas

Provide context to measurement

All or most meaningful information

"One stop shop" for dimensional subjects.

Hierarchical vs Flat dimensions

College

Department

Faculty

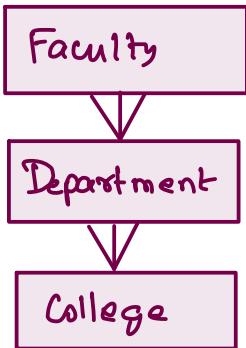
Student

Belongs to a hierarchy

Does not belong to a hierarchy.

## Star Schema

"Hierarchy reversed to show how columns appear in the dimension table"



## Primary Key

Faculty_DIM
Faculty - Key
Faculty - ID
Faculty - Name
:
Dept - ID
Dept - Name
:
College - ID
College - Name
:

Surrogate key  
Natural keys

## Snowflake Schema



Non terminal Dimension

Terminal Dimension  
(highest level)

College - DIM
College - key (PK)
College - ID
College - Name
...

Highest level  
so no foreign key

Surrogate keys  
(PK) and (FK)

Faculty - DIM
Faculty - key (PK)
Faculty - ID
Faculty - Name
...
Dept - key (FK)

IMPORTANT

↙ Snowflake Schema Pk-FK rules

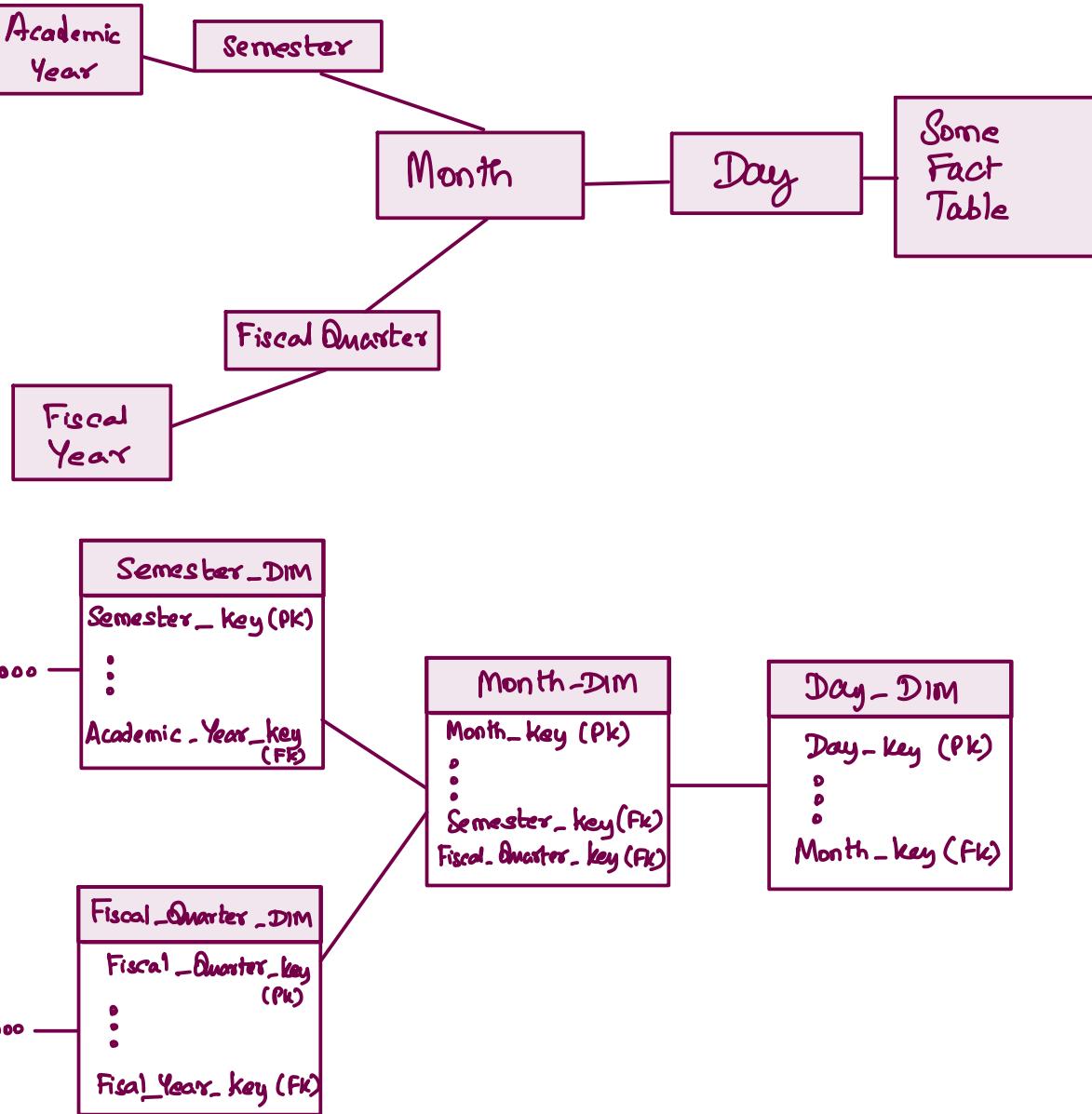
1 table for each level of a hierarchy

Every non-terminal dimension has: Primary / Surrogate key  
The next highest level's primary / surrogate as foreign key



Every terminal dimension has primary / Surrogate key  
No hierarchy based Fk (because no higher level)

## Snowflake hierarchy with branching



## (b) The four main types of DWH fact tables

Recap We store our facts in a fact table.

Fact  $\neq$  Fact tables

### 4 types of fact tables

#### Transaction Fact Tables

- Record facts (measurements) from transactions.

#### Periodic Snapshot Fact Tables

- Track a given measurement at regular intervals

#### Accumulating Snapshot Fact Tables

- Track the progress of a business process through formally defined stages

#### Factless Fact Tables

- Record occurrence of a transaction that has no measurements
- Record coverage or eligibility relationships

## (c) The role of transaction fact tables

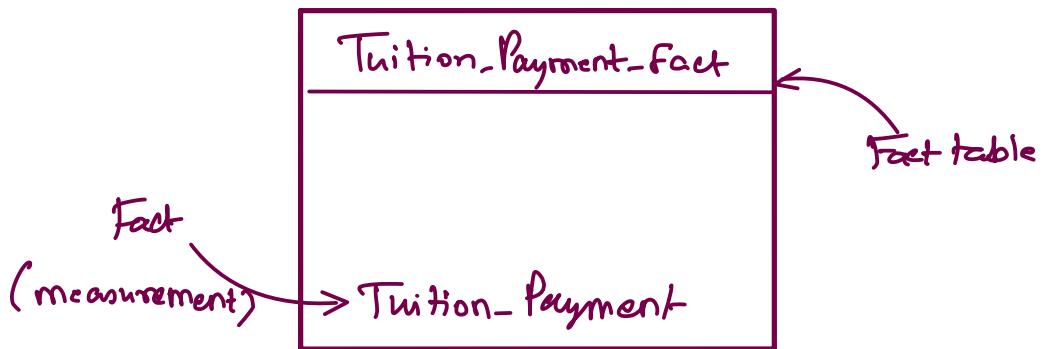
### Transaction fact table

Formally - transaction - grained fact table  
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.



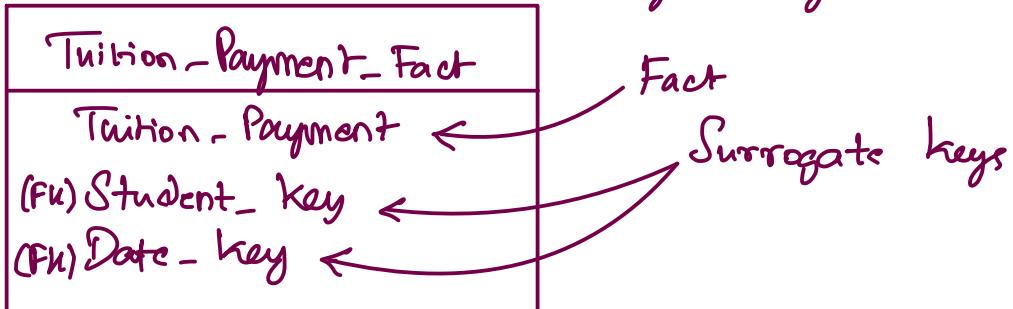
Fact + Context = Decision

What's missing      Student      Payment Date

We won't add the Student\_ID  
and payment\_Date

Remember:

We need to use the surrogate keys



Student - DIM
Student - Key (PK)
Student - ID
...

Date - DIM
Date - Key (PK)
...

Context

## ⑦ The rules governing transaction fact tables

What are the rules?

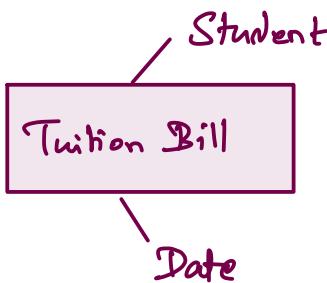
Rule #1

Facts available at same grain (level of detail)

Rule #2

Facts occur simultaneously

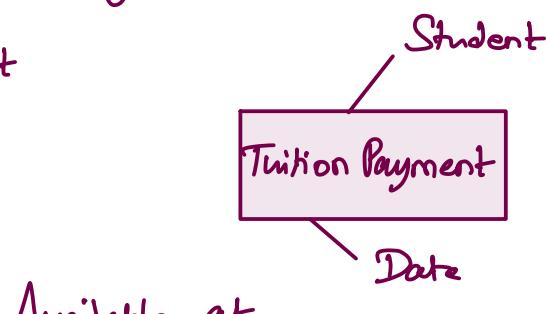
Eg



To check

Rule #1

look at dimensions



Available at  
same grain?

YES

Rule #2 Do they occur simultaneously? NO

Student receives a bill and then they pay the bill.

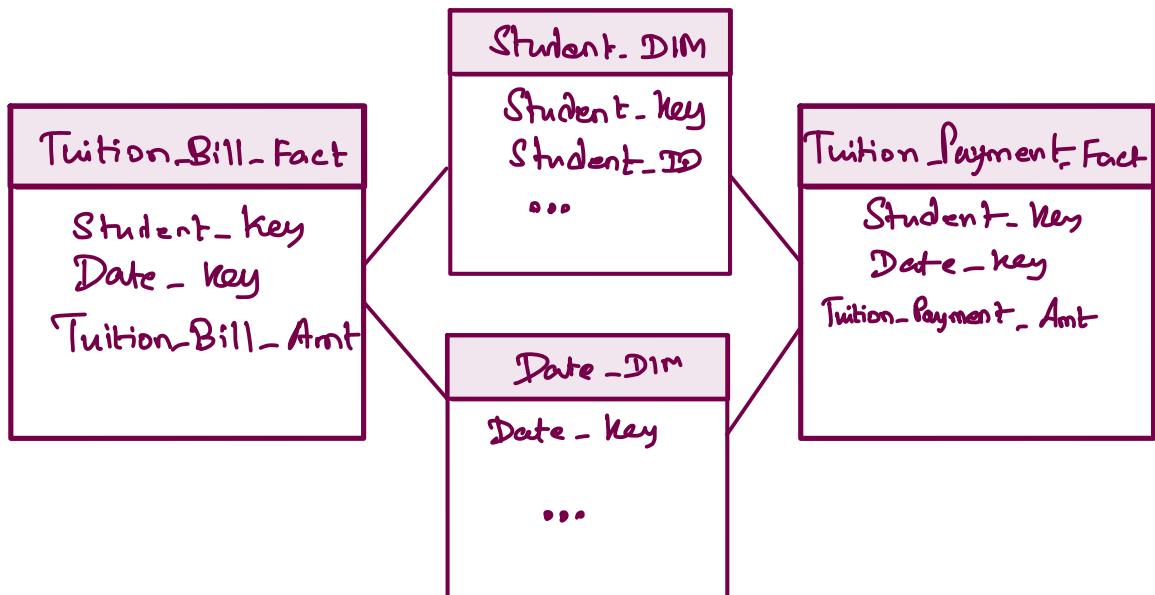
Why not violate the two rules

Complicates data analysis. Requires SQL workarounds

Billing and Payments are 2 different business processes

Business processes belong in their own  
separate fact tables.

# How to represent?

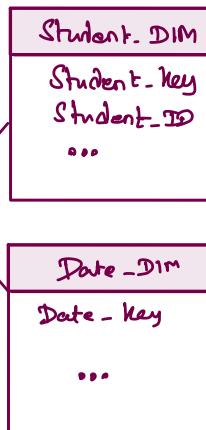
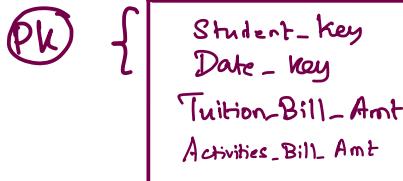


What facts can be stored together?



Rule 1 ✓  
YES

Rule 2 ✓  
Yes



## Example of facts at different grains

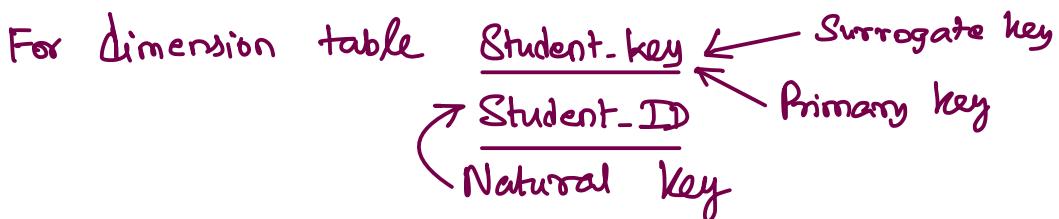
Assume if activities fees also depends on the campus dimension



Rule #1 is not satisfied

## ② Primary and Foreign Keys for Fact Tables

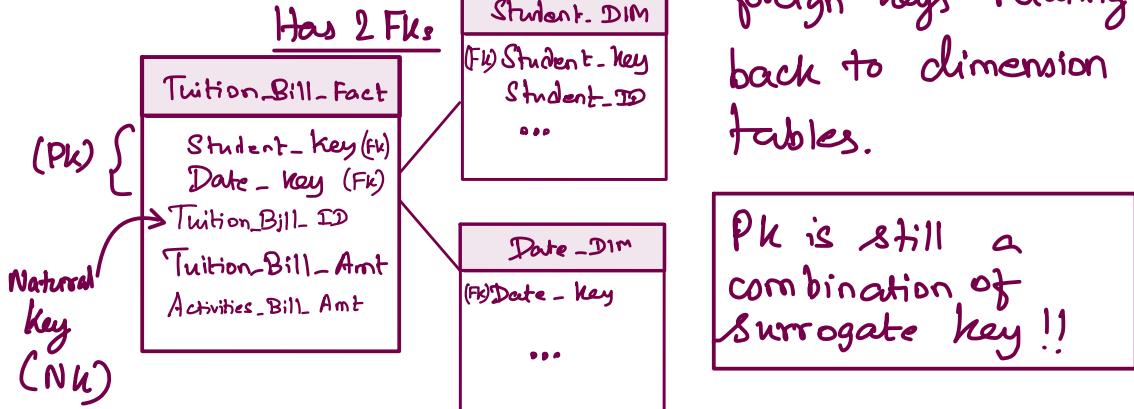
Recap Pk - Primary key Fk - Foreign key



Fact tables handled differently !

## PK of fact tables

Combination of all foreign keys relating back to dimension tables.



## f) The role of periodic snapshot fact tables

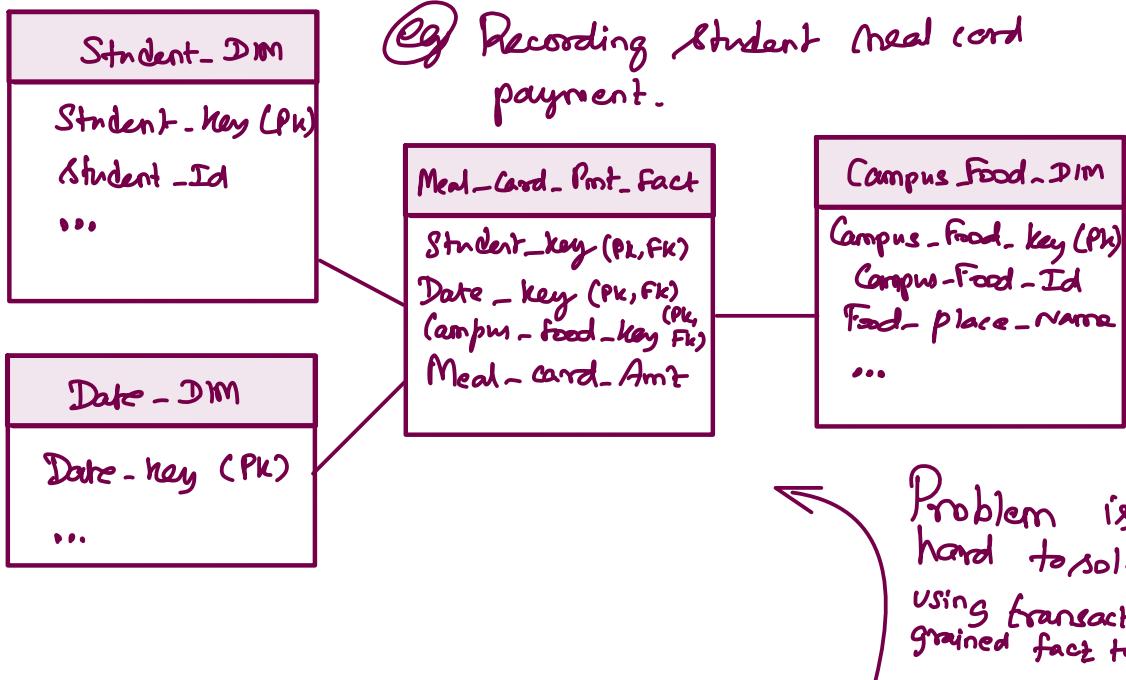
Sometimes called snapshots → Better use full name

Take and record regular periodic measurement.

2 types of periodic snapshot fact tables

Aggregate result of "regular" transactions

Levels that are not related to "regular" transactions



② Track and analyze the end-of-week meal account balances throughout the semester.

We create a periodic snapshot fact table along side regular transaction grained fact table

Student - DM
Student - key (Pk)
Student - Id
...

Date - DM
Date - key (Pk)
...

Meal_Card_EOW_Snapshot
Student - key (Pk, Fk)
Week_key (Pk, Fk)
Meal_card - EOW - Balance

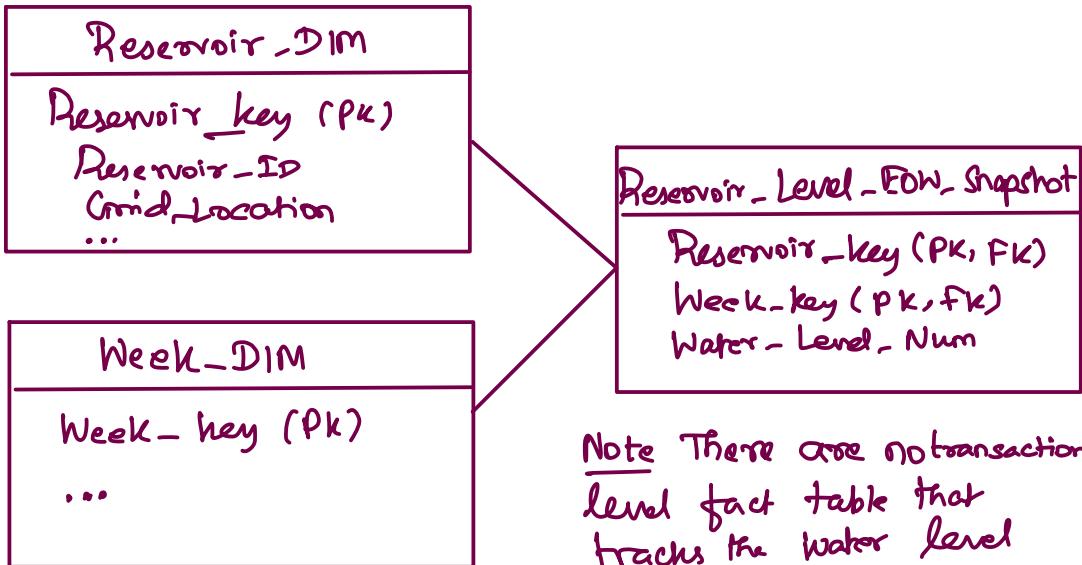
(3)

### Meal\_Card\_EOW\_Snapshot

Student - key	Week - key	Meal - Card - EOW - Balance
43232	34	\$ 210.12
43232	35	\$ 200.12
43232	38	\$ 200.12
43232	39	\$ 150.12
81768	34	\$ 430.34
81768	35	\$ 400.34
81768	38	\$ 400.34
81768	39	...
...	...	

Much easier, more direct access for certain type of business questions.

## 2<sup>nd</sup> type of periodic snapshot fact table



Note There are no transaction level fact table that tracks the water level in a reservoir.

### Comparing the 2 types of periodic snapshot fact tables

- Both are structurally the same.
- 2<sup>nd</sup> type : no transactions from which levels are built
- The levels "just exist" and can be measured.

### COMPLICATION ALERT

#### Nest - Semi-additive facts

### ⑨ Periodic Snapshot Fact Tables and Semi-Additive facts

Semi-additive facts Recap Additive, Non-additive and

semi-additive

\* Some times you can add

\* But other times you cannot add

\* Typically used in periodic snapshot fact tables

(Q) What was the ending meal card balance for a student on any given week?

Meal-Card- EOW-Snapshot		
Student-key	Week-key	Meal-Card-EOW-Balance
14323	4313	\$ 455.12
14323	4317	\$ 388.65
14323	4318	\$ 388.65
14323	4320	\$ 319.76

Measurement

We cannot add the EOW-Balance along the time dimension!

Student-DIM		
Student-key	Student ID	Name
14323	SOACK32	Sally J

→ Total \$1552.18 ← doesn't make sense.

We are doing a snapshot each week.

Non additive      But you can perform other numeric operations along the time dimension.

Avg. of EOW Balance for 4 weeks  $\frac{\$1552.18}{4} = \$388.05$

Also you can lock time dimension to one specific value ... and then add values.

Meal - Card - EOW - Snapshot		
Student_key	Week_key	Meal_Card_EOW_Balance
14323	4313	\$ 455.12
14323	4317	\$ 388.65
14323	4318	\$ 388.65
14323	4320	\$ 319.76
13531	4313	\$ 561.23
14381	4313	\$ 215.63

Measurement

We lock the time dimension  
Week 4313

Then we can sum up to get

The total remaining balances in meal accounts at the end of week 4313 was \$1231.98

Then find average remaining account balance at the end of week 4313 was \$410.66.

**NOTE:** Periodic snapshot fact tables can play a valuable role in your data warehouse.

## ⑤ The role of accumulating snapshot fact tables

Accumulating snapshot → Track the progress of a business process through formally defined stages

## Tracking the progress of a business process

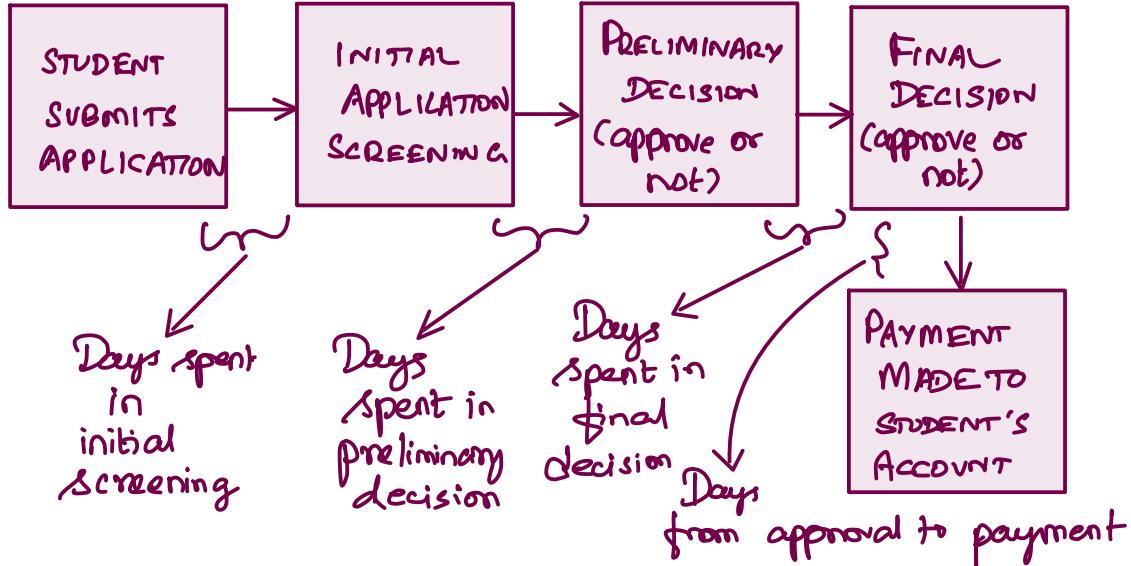
Elapsed time spent in each phase

Include both completed and in-progress phases.

Can also track other measures as a process proceeds

Introduces concept of multiple relationships from fact table to a single dimension table. We have seen from multiple dimensions to a single fact table.

## (e) Student financial aid application process

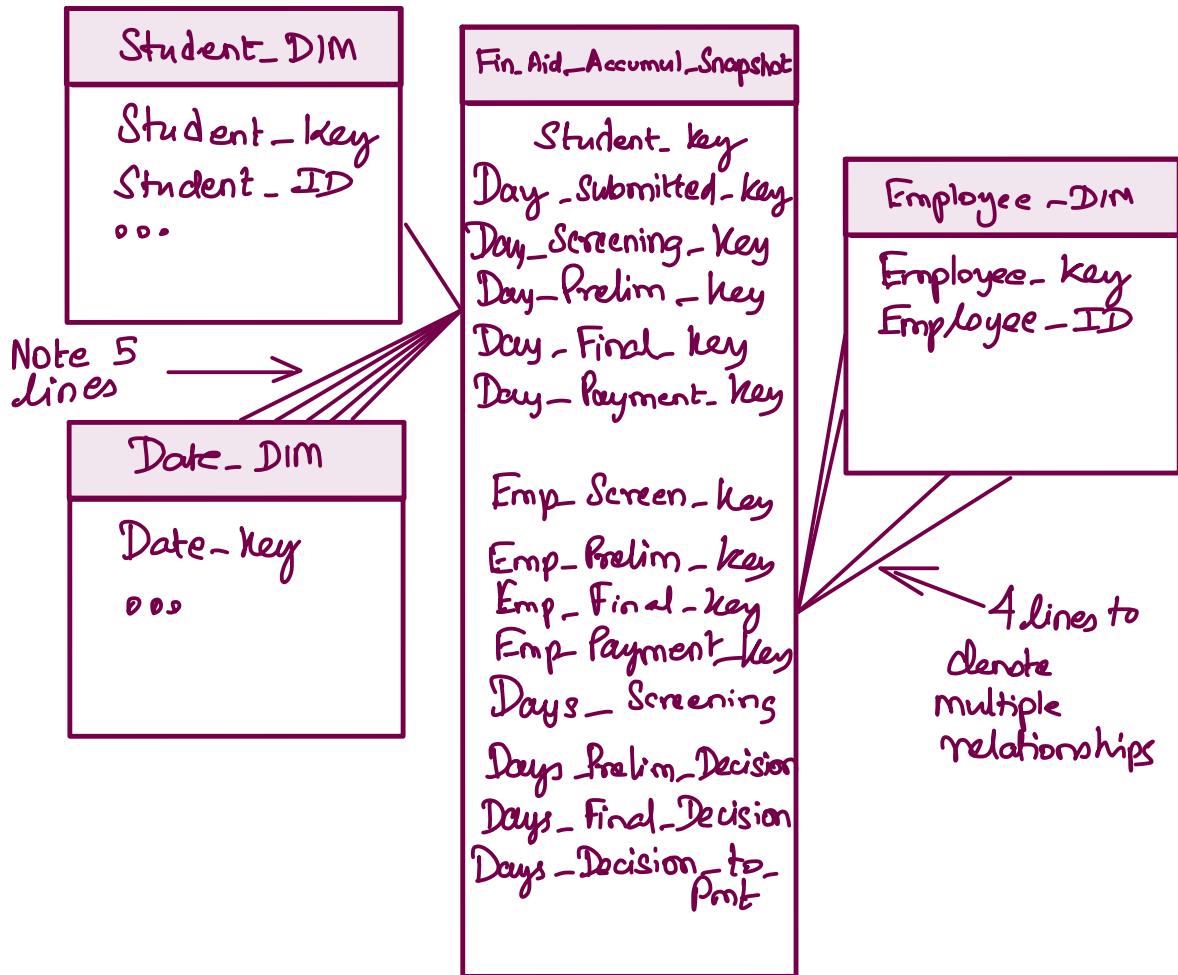


## Dimensions for this process

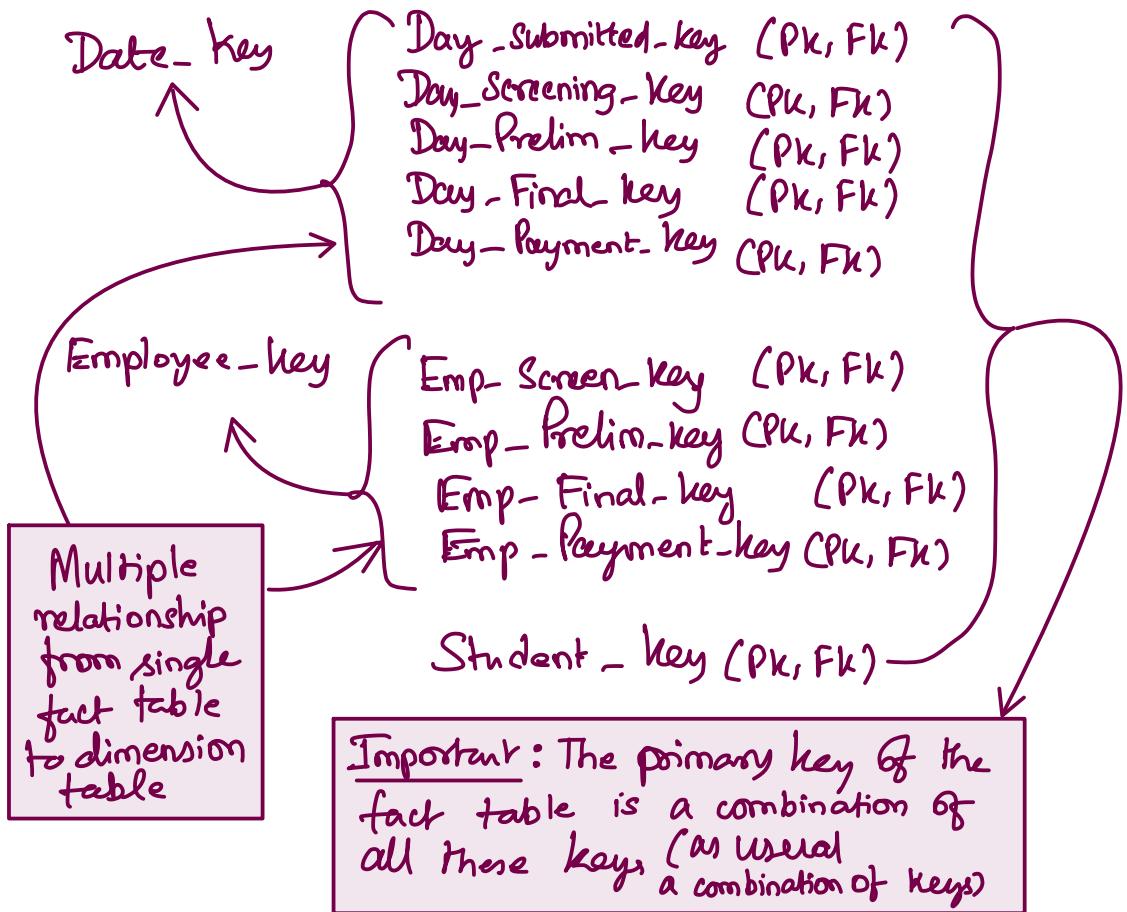
Student

Time (Specific day a phase begins)

College employee responsible for each phase

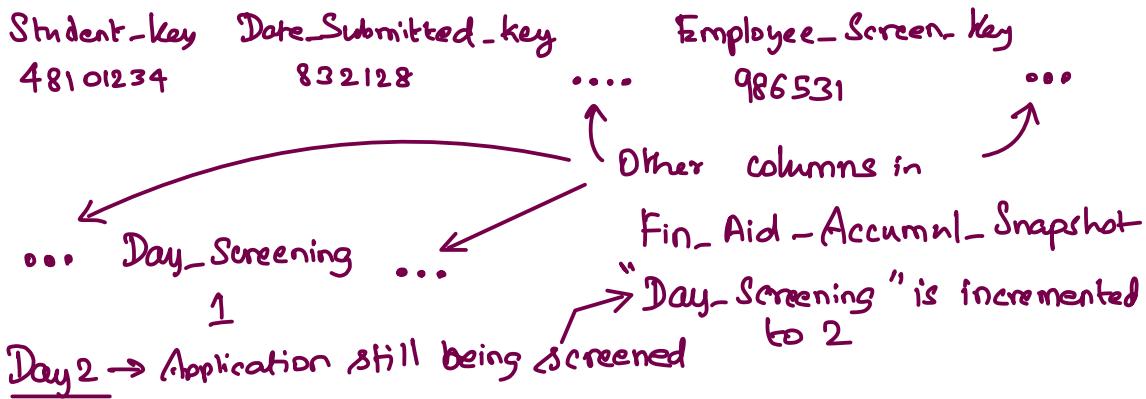


The name of key in accumulating snapshot fact table is not the same as the name of the primary key dimension table.



## ① Accumulating Snapshot fact table example

Richard applies for \$8000 on 8/11/2020



Day 3 → Screening completed

... Day\_Screening\_Key ... Days\_Screening  
638321 3

Day 4 ... Employee\_Prelim\_key ... Days\_Screening

Some other employee has handled the stage.  
→ 532128 ... Days\_Prelim ...  
-Decision 1

Day 5 Still in processing ... Days\_Prelim\_Decision...  
2

and Eventually the entire record is filled in!

Fin_Aid_Accumul_Snapshot
Student_key
Day_Submitted_key
Day_Screening_Key
Day_Prelim_key
Day_Final_key
Day_Payment_key
Emp_Screen_key
Emp_Prelim_key
Emp_Final_key
Emp_Payment_key
Days_Screening
Days_Prelim_Decision
Days_Final_Decision
Days_Decision_to_Pmt

Data

48101234
832128
638321
739432
861542
976852
986531
532128
638714
639813
3
4
3
6

Meanwhile other students are also applying for financial aid.

Can track other facts along the business process.

Assume Richard is not paid the whole aid amount and gets paid \$5000. We can track these additional facts.

Much more powerful and easier than using a transaction grained fact table

Fin_Aid_Accum_Snapshot
Student_key
Day_Submitted_key
Day_Screening_Key
Day_Prelim_key
Day_Final_key
Day_Payment_key
Emp_Screen_key
Emp_Prelim_key
Emp_Final_key
Emp_Payment_key
Days_Screening
Days_Prelim_Decision
Days_Final_Decision
Days_Decision_to_Pmt
Amt_in_Screening
Amt_in_prelim_decision
Amt_in_final_decision
Amt_paid

We can build additional reports like

- ✓ Average amount in screening
- ✓ Total amount in screening
- ✓ Average amount in final decision
- ✓ Total amount paid.

etc.

Additional facts  
Sample values

\$ 8000  
\$ 8000  
\$ 5000  
\$ 5000

## j) Why a factless fact table isn't a contradiction in terms

Recap Facts → Measurements

Fact ≠ Fact table

### Factless

1. Record occurrence of a transaction that has no measurements
2. Record coverage or eligibility relationships.

### Factless fact table

An event occurs that we want to track ...

(es) Students can register for online webinars throughout the semester.

### Track

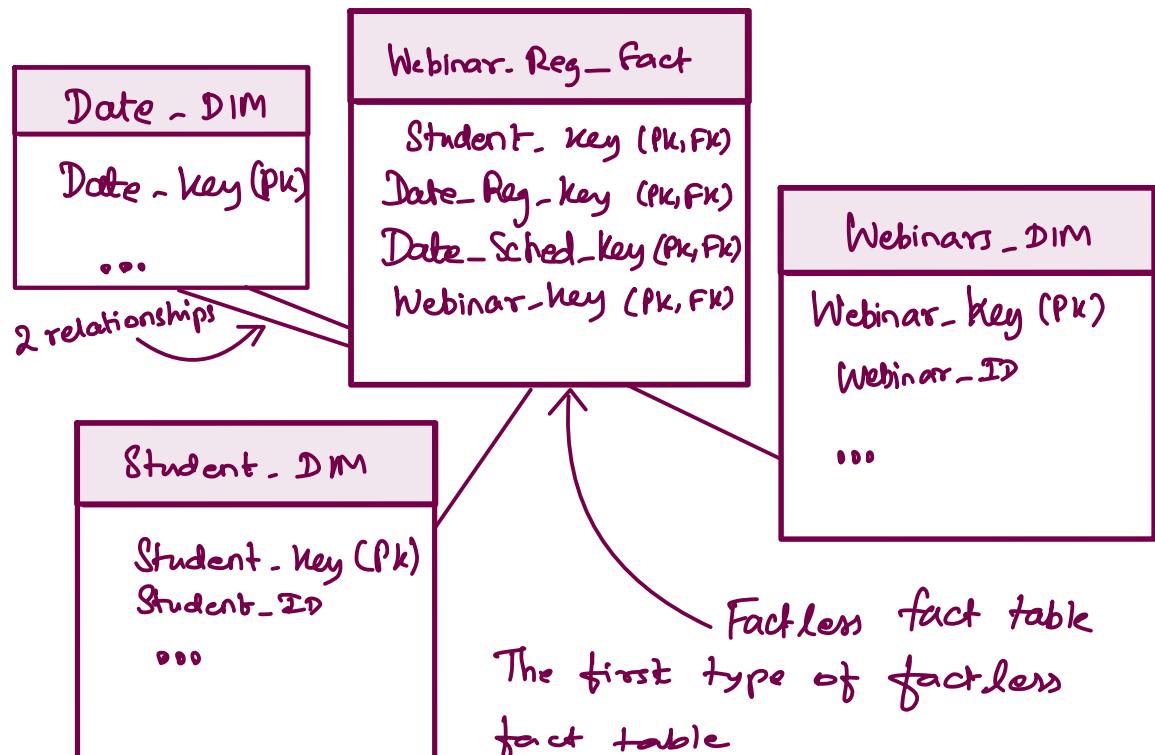
Which students register

which webinar

The date each student registers for a given webinar

The scheduled date for each webinar

# Recording registrations in a fact table



The "measurement" is actually the occurrence of the event.

presence of a row is therefore the measurement

Can count rows with or without filters

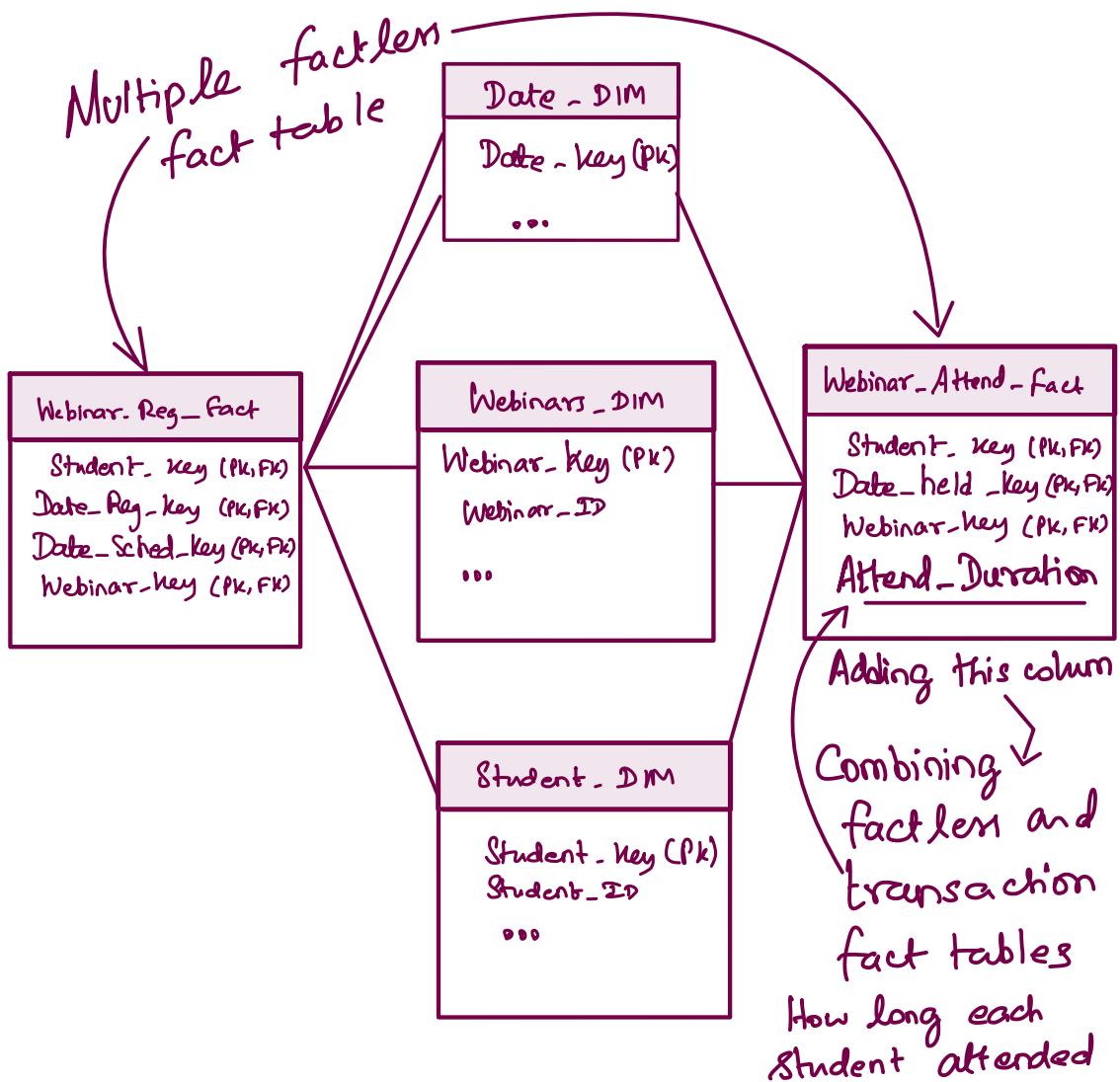
Factless fact table = Pk / Fk columns only \*

Directions How many webinars were held last semester?

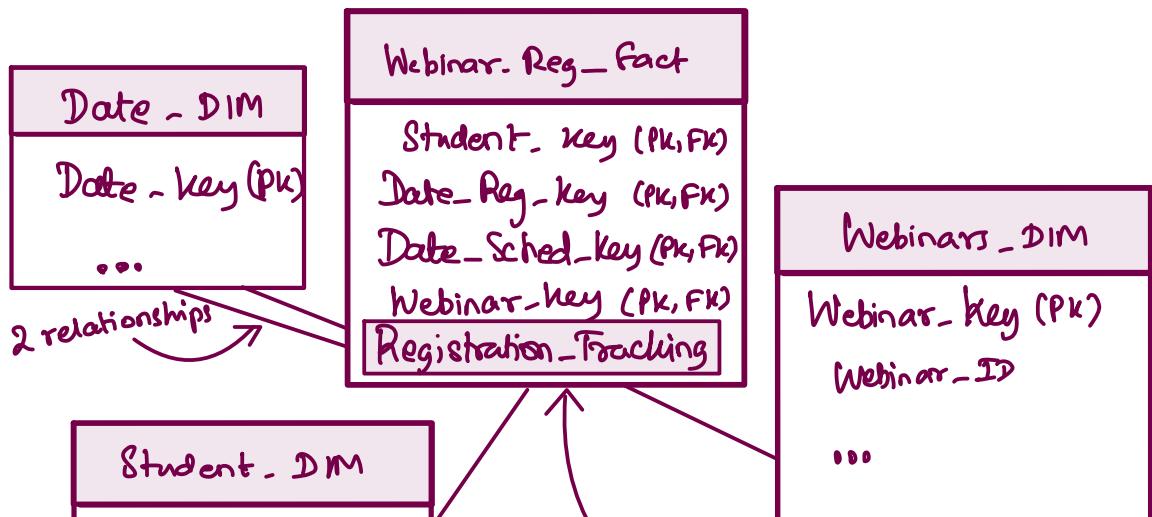
\* See p.52  
How many webinars has Rob registered for since becoming a student.

You can essentially query by a dimension and count the records in the factless fact table.

Can have multiple factless fact table in the same schema.



Sometimes use a "tracking fact"



A tracking fact

column

Way to have a quasi fact  
Some folks are not comfortable  
with a fact table without facts.

"Tracking fact"

Value is always = 1

Allows you to use SQL SUMC)  
function rather than COUNT() function.

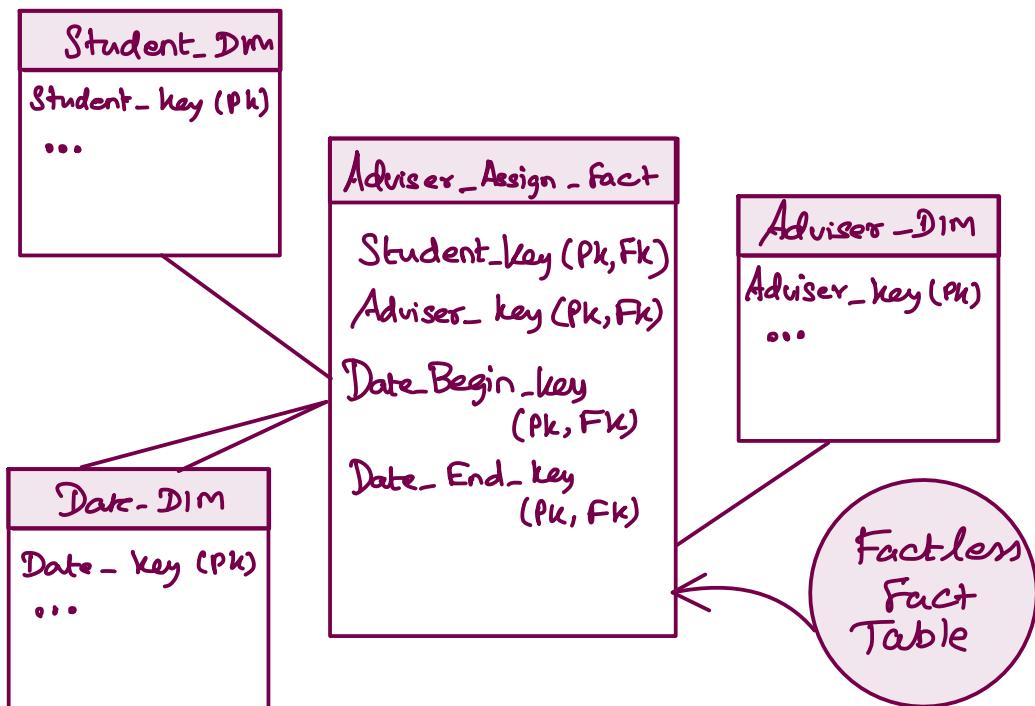
More readable and understandable but it is  
less efficient in execution.

## 2<sup>nd</sup> type of factless fact table

Allows

- Record coverage or eligibility relationships among multiple parties
- Even if no transactions occurred.
- Typically (but not always) between a starting and ending date or time.

Academic Advisers assigned to Students



If student visited advisor we can use the 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 visited his/her adviser?

Still we want to record the assignment.

Which students were assigned to an adviser from <from-date> to <to-date>

(K) Compare structure of fact tables in star vs. snowflake schema

Star vs Snowflake Schema

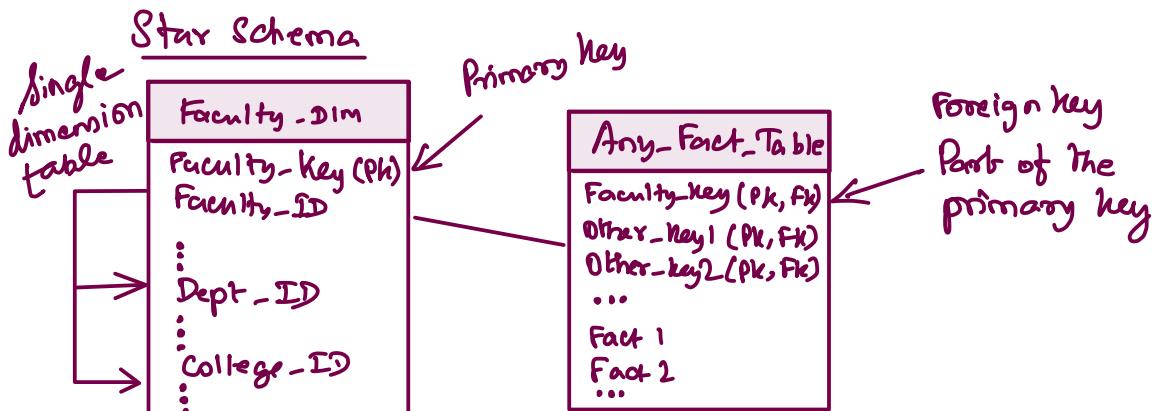
Recap Star vs Snowflake schemas for dimension tables.

Star schema

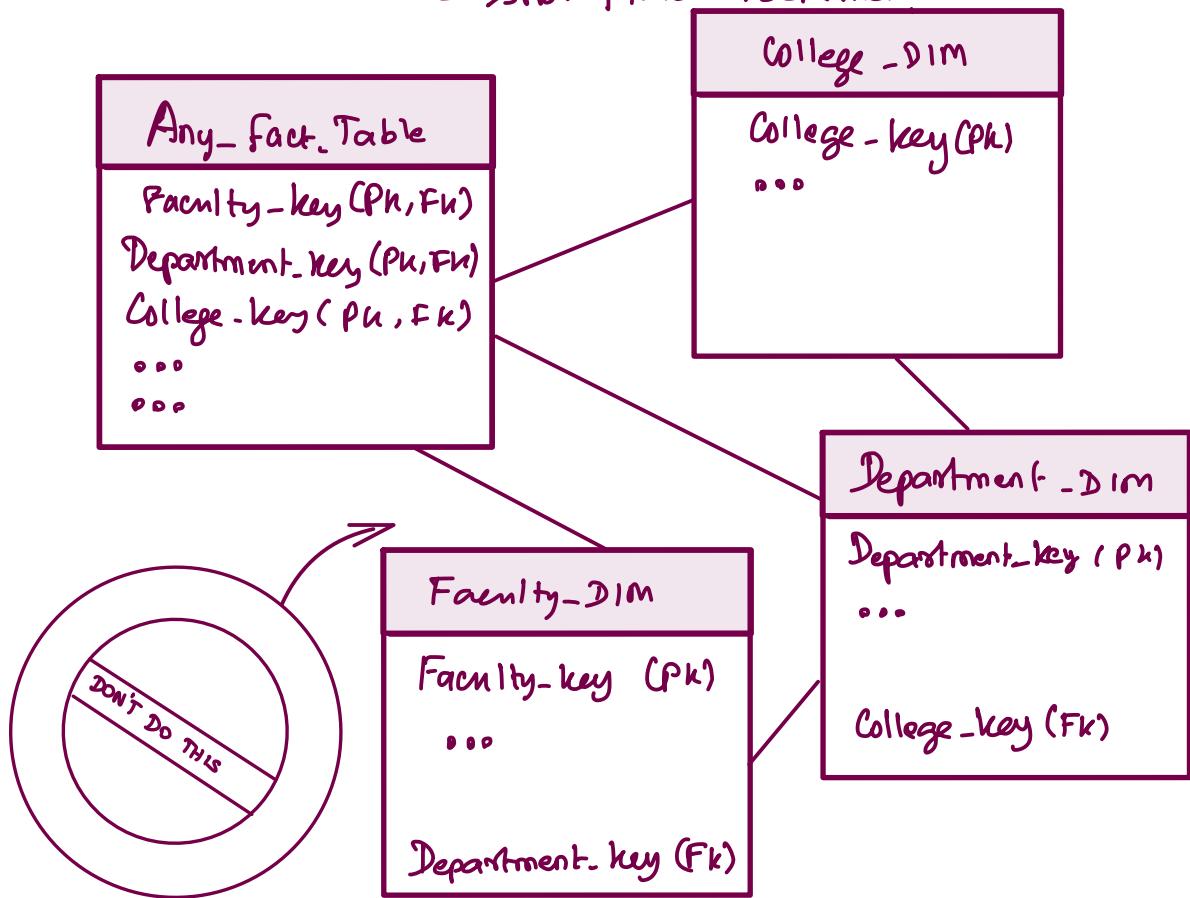
one primary key

Snowflake schema

primary key and foreign key



What about snowflake schema?

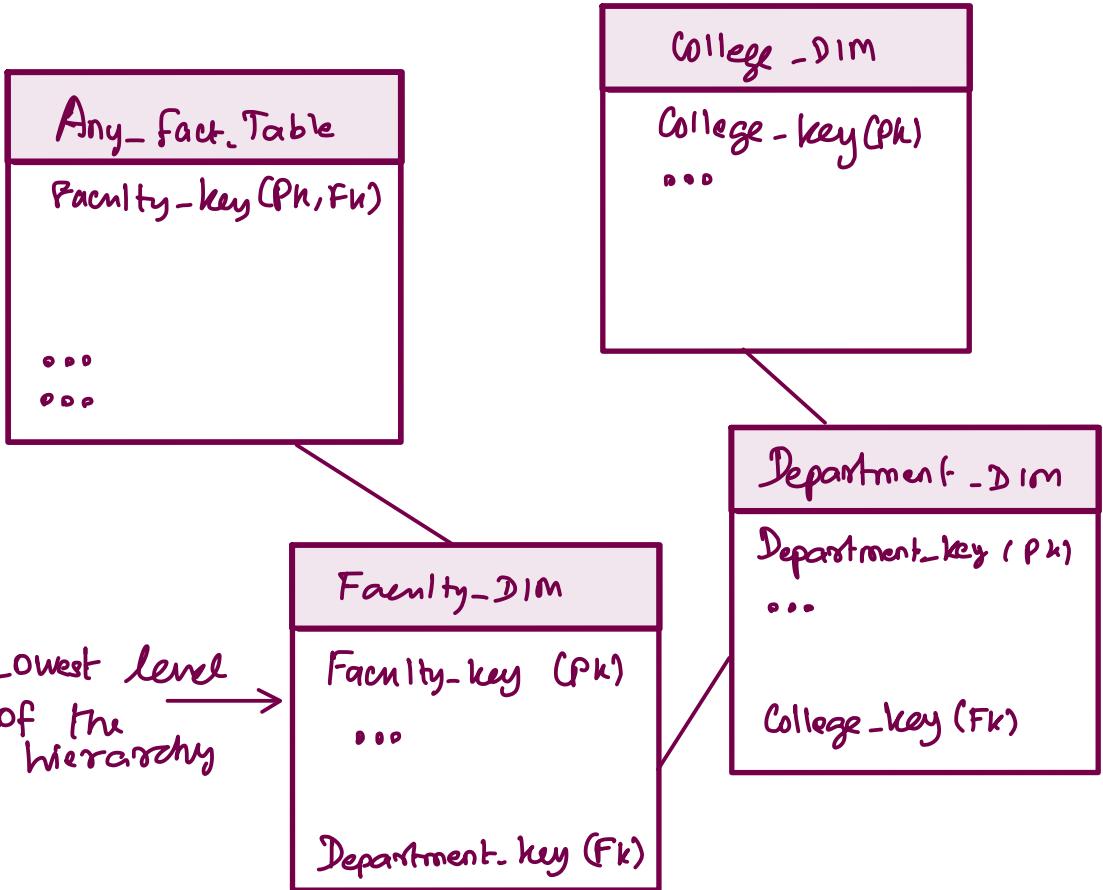


Pk-Fk links to every level of every hierarchy

- Unwieldy and complicated } Not the most elegant
- "Centipede" fact tables } Solution.

Just have the Pk and Fk between the fact table and the related dimension table

In this case Between Fact table and faculty dimension table.



## ① SQL for dimension and fact tables.

CREATE TABLE examples

- Star schema dimension table.
- Snowflake schema non-terminal dimension table.
- Snowflake schema terminal dimension table.
- Transaction-grained fact table.
- Periodic snapshot fact table.

## • Star Schema Dimension Table

```
CREATE TABLE Faculty-DIM (
    Faculty-Key      INT NOT NULL,
    Faculty-ID       VARCHAR(10) NOT NULL,
    Faculty-Last-Name VARCHAR(40) NOT NULL,
    Faculty-First-Name VARCHAR(20) NOT NULL,
    Year-Joined      INT,
    Faculty-Rank     VARCHAR(20) NOT NULL,
    ...
    Dept-ID          VARCHAR(10) NOT NULL,
    Dept-Name        VARCHAR(40) NOT NULL,
    Dept-Year-Founded INT,
    ...
    College-ID       VARCHAR(10) NOT NULL,
    College-Name     VARCHAR(40) NOT NULL,
    College-Year-Founded INT,
    Dean              VARCHAR(50) NOT NULL,
    ...
    PRIMARY KEY (Faculty-Key)
),
```

Single key only in a star schema dimension.

## • Snowflake schema non-terminal dimension tables

```
CREATE TABLE Faculty-DIM (
    Faculty-Key      INT NOT NULL,
    Faculty-ID       VARCHAR(10) NOT NULL,
    ...
    Department-Key   INT NOT NULL,
    PRIMARY KEY (Faculty-Key),
    FOREIGN KEY (Department-Key) REFERENCES
    Department-DIM (Department-Key)
);
```

Non terminal dimension table has PK, FK relationship.

Similarly for Department table

```
CREATE TABLE Department-DIM(
    Department-key      INT NOT NULL,
    ...
    ...
    PRIMARY KEY (Department-key),
    FOREIGN KEY (College-key) REFERENCES
    College-DIM (college-key)
);
```

- Snowflake schema for terminal dimension tables

```
CREATE TABLE College-DIM(
    College-key      INT NOT NULL,
    ...
    ...
    PRIMARY KEY (College-key)
);
```

No FK as  
it is the  
terminal  
dimension.

- Transaction grained fact table

```
CREATE TABLE Tuition-Bill-Fact(
    Student-key      INT NOT NULL,
    Date-key         INT NOT NULL,
    Tuition-Bill-Amt DECIMAL (8,2) NOT NULL,
    Activities-Bill-Amt DECIMAL (8,2),
    ...
    ...
    ...
);
```

PK is combination of Fks

→ PRIMARY KEY (Student-key, Date-key),  
FOREIGN-KEY (Student-key) REFERENCES  
Student-DIM (Student-key),  
FOREIGN-KEY (Date-key) REFERENCES  
Date-DIM (Date-key)

FK for each dimension

## • Periodic Snapshot fact table

### ⑨ Meal card balance

```
CREATE TABLE Meal-Card-Payment-fact (
    Student-key          INT NOT NULL,
    Date-key              INT NOT NULL,
    Campus-key            INT NOT NULL,
    Meal-Card-Balance     DECIMAL (8,2) NOT NULL,
    PRIMARY KEY (Student-key, Date-key, Campus-key),
    FOREIGN KEY (Student-key) REFERENCES
        Student-DIM (Student-key),
    FOREIGN KEY (Date-key) REFERENCES
        Date-DIM (Date-key),
    FOREIGN KEY (Campus-key) REFERENCES
        Campus-DIM (Campus-key)
);
```

Primary key is Combination of FKs

Foreign key clauses for each dimension

If you have more than one reference to same dimension you will have appropriate number of FOREIGN KEY clauses

## ⑥ Managing DW History through slowly changing dimensions

SCD → Slowly Changing Dimensions.

### ② SCDs and DW History

- Techniques to manage history within DW.
- Multiple techniques based on various historical data policies
- Enables DW to appropriately manage history regardless of policies in transactional applications.

### 3 main policies

Overwrite old data : no history retention

Type 1

Maintain unlimited history

Type 2

Maintain limited history

Type 3

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.

Even more SCD types

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

Our Focus	Type 1	Overwrite
	Type 2	New row
	Type 3	New column

## b) Designing a type 1 SCD.

Replace old value with new value



Old-value  
lost ✓

(a) Correcting errors with  
type 1 SCD

e.g. Fixing student's birthdate that was  
entered incorrectly

### Advantages

Simplest and most  
straightforward

DW content errors are  
purged forever

Best for error correction  
and any other "don't need  
any history" situations.

### Disadvantages

Might still want history of  
errors for auditing purposes

Reposting before and after  
type 1 change could vary

Tendency to overuse type 1  
changes as it is easier than  
type 2.

## C) Designing a type 2 SCD

### Type 2

existing row remains as-is.

new row added to dimension table

new row reflects current state of all attributes  
complications with reporting and analytics.

Before

Column

Old-value

② Student relocating  
from one state  
to another  
we need the history

After

Column

Old-value

New-value

} Both old and new

Note:

The Student-key (Dimension table's PK  
is auto generated. is a surrogate key.)

Student-key

456312

458643

Name

Ted

Ted

State

CA

CO

Old version

Current version

Why would we need both versions?

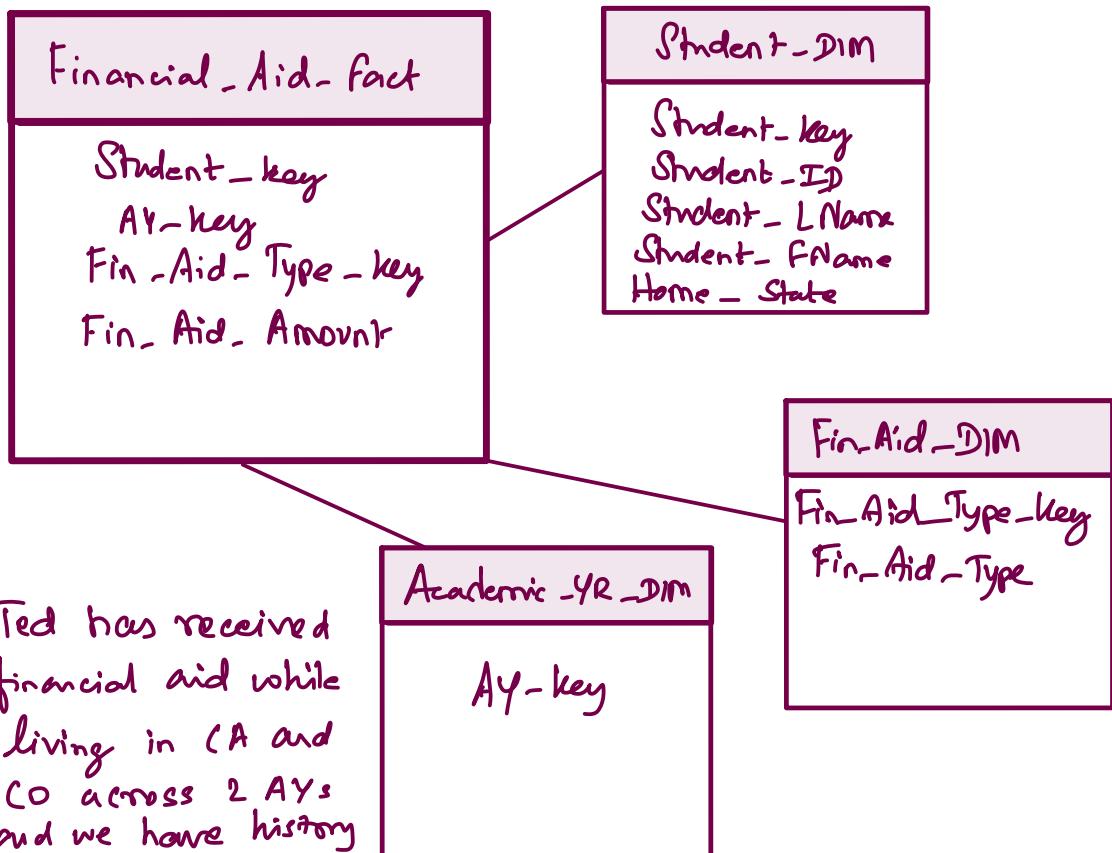
### Reasons for Type 2 SCD

Reporting and analytics before the type 2 change needs to accurately reflect data values.

Q Report average student financial aid amount for last year by home state.

Last year Ted's home state was CA. So we need history for these kind of reports.

Fact table complications with type 2 SCDs



Student-key    Student-ID    Student-LName    Student-FName    Home-state

985631	TYOUNG21	Young	Ted	CA
384081	TYOUNG21	Young	Ted	CO
Student-Key	AY-key	Fin-Aid-Type-key	Fin-Aid-Amount	
985631	221323	1181	\$7500.00	
384081	581233	1181	\$2500.00	

In the above example if you want to report all the aid that Ted received, we have a small problem. 2 records have different Student-keys.

### Correct reporting with type 2 SCDS

Could include natural keys also in fact tables

Student-key	Student-ID	AY-key	Trn_Aid_Typekey	Fin_Aid_Amount
985631	TYOUNG21	221323	1181	\$7500
384081	TYOUNG21	581233	1181	\$2500

Natural key →

WAIT!

Unwieldy and takes significant storage for fact tables with many dimensions and billions of rows!

Recommendation Do not include the natural keys also in fact tables.

Best handled through careful multi-step SQL between dimension tables and fact tables.

- First: find all surrogate keys for desired student.
- Next: retrieve all fact table rows for all of the relevant surrogate keys.

There is yet another challenge

"Maintain correct data order with type 2 SCDS"

## d) Maintain correct data order with type 2 SCDs

SCD type 2 change → We have before and after records - how do we know which is which?

### 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 : Current-Flag column

Dimension table

Col 1	Col 2	Col 3	Col 4	...	Current-Flag
—	—	—	—	—	Y
—	—	—	—	—	Y ← Indicates record is current.

Before  
change

Col 1	Col 2	Col 3	Col 4	...	Current-Flag
—	—	—	—	—	Y
—	—	—	—	—	N ← Indicates old record.
—	—	—	—	—	Y ← Indicates record is current.

After  
change

"What if a row changes  
more than once?"

What if Ted moves from CO (Colorado) to NM (New Mexico)  
 2 rows will have current flag set to 'N'

Home-State      Current-flag

CA	N	Which is first?
CO	N	
NM	Y	

Solution 2 : Instead of one flag we have 2 columns

Home-State      Effective-Date      Expiration-Date

CA	8/11/2017	8/14/2020	Obsolete
CO	8/15/2020	11/2/2020	
NM	11/3/2020	12/31/2199	

This date way into the future indicates it is current.

Third Solution

Combination of 1 and 2

Effective-Date      Expiration-Date      Current-flag

use this for use cases involving count!

② Designing a Type 3 SCD

Type 2 robust but architecturally complex.

- Add a 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.

### Type 3 SCD example

Textbook publisher's sales divisions being reorganized  
Formerly NORTH and SOUTH  
Now EAST, WEST, CENTRAL

All sales reps reassigned to 1 of the 3 new divisions  
Reorganization effective 01 01 2021

We want to report

2020 (and earlier) sales data "sliced" by the old NORTH and SOUTH divisions.

2021 (and forward) sales data "sliced" by the new EAST, WEST and CENTRAL divisions.

But also:

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 are very complex"

A "Better" Solution

Sales\_Rep-Dim

Sales_Rep-key	...	...	Current-Div	Previous-Div
4367			E	S
8316			E	N
3612			C	N
4319			W	S
Report / Analytic	Column to use			
2020 (and earlier) by N & S	Previous-Div			
2021 (and forward) by E, W & C	Current-Div			
2020 (and earlier) by E, W, & C	Current-Div			
2021 (and forward) by N & S	Previous-Div			
Old and New by either old or new	Current-Div or Previous-Div			

It seems easy? Because it is easy.

Type 3 SCD limitations

Use cases where every row in a dimension table is (or might) change at the same time

Reorganisations → classic use case

Not suited for random changes such as students' addresses.

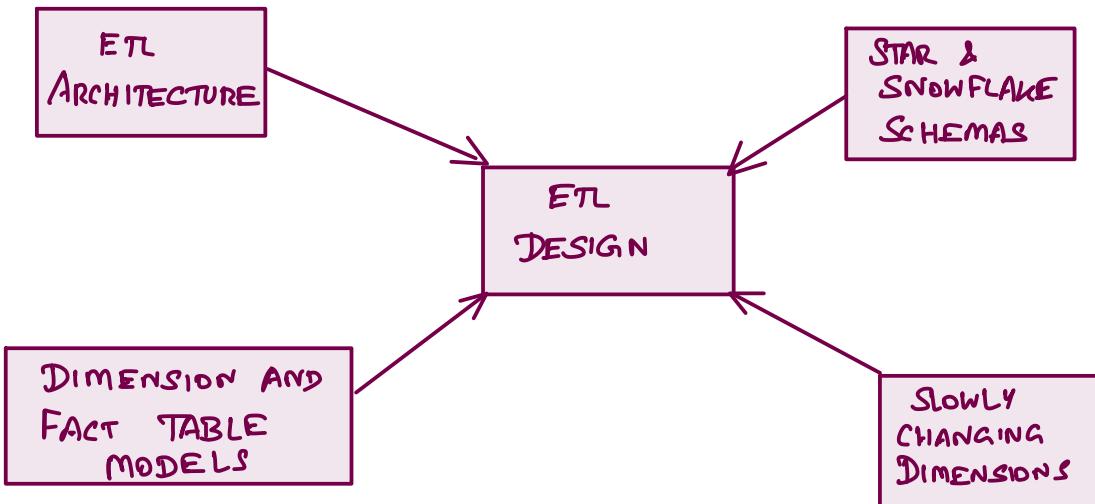
Final Points: Type 3 SCD

Not limited to only 2 columns (prev. vs. current). Could have 3 or 4 e.g. Reorganization history over time.

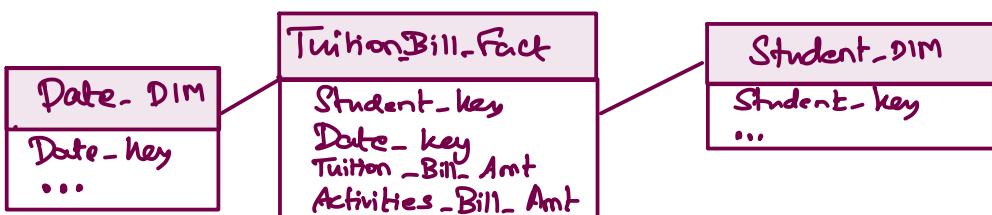
A large # of columns becomes unwieldy and better to use Type 2

## ⑦ Designing Your ETL

② Build your ETL design from your ETL Architecture

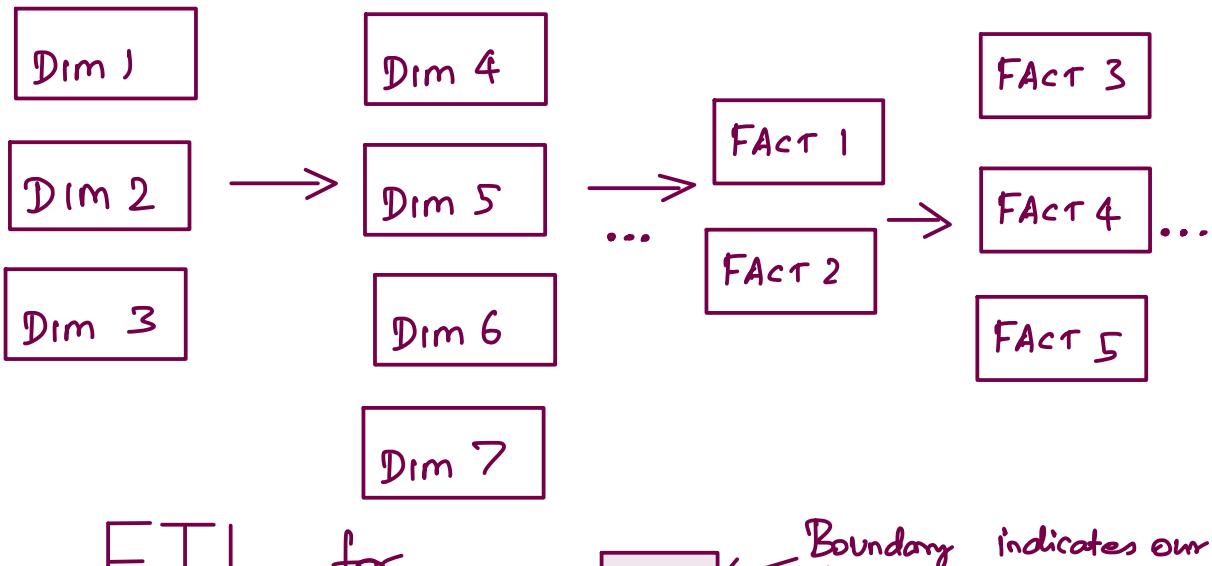


- Process dimension tables before fact tables.



We need to process Dim. before fact as we need the surrogate key (PK). Process all Dim tables and then create the fact table.

- Opportunities for parallel processing.



# ETL for

 Boundary indicates our focus areas.

SCD Type 1	
SCD Type 2	
SCD Type 3	
	Star Schema
	Snowflake schema
Append	Transaction grained fact table
In-place update	Periodic Snapshot fact table
Complete Replacement	Accumulating Snapshots
Rolling Append	Factless fact table

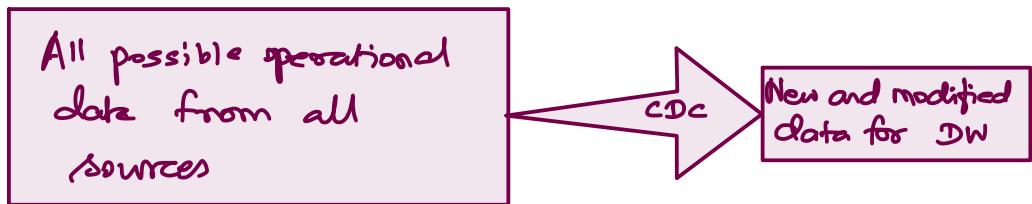
## (b) Dimension table ETL

Star Schema ETL  
(not Snowflake schema)

ETL for snowflake schema is complex given the hierarchy.

Dimension table incremental ETL

Step 1: Data preparation



"Change Data Capture" Techniques

- Transaction data timestamps
- Database logs (if not) ↗  
Compare with last ETL run's timestamp.
- Last resort: database scan and compare.

Step 2 Data transformation

Refer to common transformation models discussed earlier.

Data value mitigation

Data type and size mitigation

De-duplication

Dropping columns (vertical sizing)

Value based row filtering (horizontal slicing)

Correcting known errors.

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 ↴

Here every new row  
gets brand new  
surrogate key.

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

Faculty

Faculty-DIM

As new records are

added new surrogate keys  
get added to the  
dimension.

③ Process SCD Type 1 changes to a dimension table

Connecting errors with type 1 SCD. "History is lost forever."

Basic in-place update.

SCD Type 1 complication

Even though it might be a simple in-place update  
we may need to apply that to more than one  
row

Why? If we had a type 2 change that occurred  
before the type 1 change

The type 2 change leaves more than one row!

Apply type 1 change to all relevant rows.

How to handle this? **IMPORTANT**

Look for all dimension table rows for the natural key.

If you make a type 1 change you need to handle that for all possible cases.

(d) Process SCD Type 2 changes to dimension table.

Basic append with a new surrogate key

Use the natural key as a guide

After type 2 SCD

Student-Key	Student-ID	Student-LName	Student-Fname	Home-State	DOB
43G21	TYOUNG21	TED	YOUNG	CA	11/4/2004
53123	TYOUNG21	TED	YOUNG	CO	11/4/2004

→ New Record with new surrogate key.  
Old record

(e) Design ETL for Fact tables

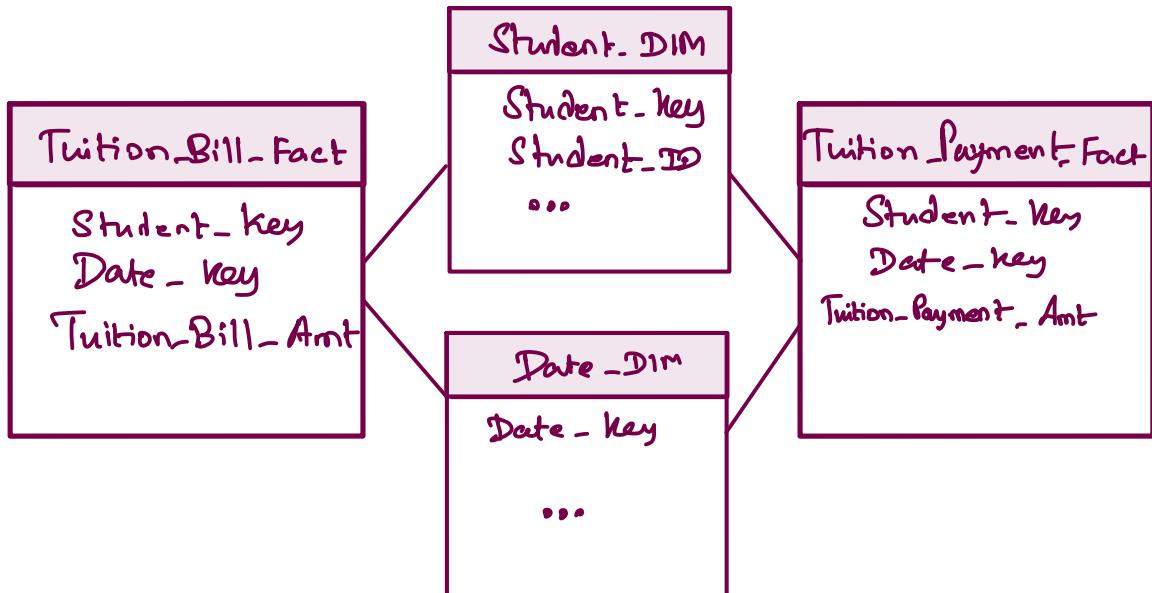
Process fact tables      Basic append but with a twist.

Refer to image in next page

2 fact tables.

2 dimension tables.

} Multiple fact tables  
and shared dimension  
tables



This example is going to use 'Tuition\_Payment\_fact'

### Tuition\_payment\_fact before ETL

Student_key	Date_key	Tuition_Payment_Amt
73273222	83853	\$4500.00
83291245	46467	\$3400.00
73678122	35356	\$7000.00

Included in the next incremental ETL run

Ted Young pays \$5000 in tuition

Transactional system provides natural keys

TYOUNG21 8/12/2021 \$5000

But which Ted young?

↳ Meaning which version of Ted Young?

As other type 2 changes in Student-DIM happened.

Student-DIM		Student-LName	Student-FName	Home-State	DOB
Student-key	Student-ID				
f36431	TYoung21	Young	Ted	CA	4/16/2004
S32215	TYoung21	Young	Ted	CO	4/16/2004
G1G7892	TYoung21	Young	Ted	NM	4/16/2004

We could assume the current version if we knew it.

Suppose if we are loading historical data?

Recollect the third solution (Kimball DW, toolkit)

Student-key	<other columns>	Effective-Date	Expiry-Date	Cur-Flag
436431		8/11/2017	8/15/2020	N
532215		8/16/2020	7/11/2021	N
6167892		7/12/2021	12/31/2199	Y

Handle versioning using Effective Date, Expiry Date, Cur-Flag

So for ETL "We need natural keys and relevant date"

Item for 8/12/2021 \$5000 paid by TYoung21

These dimension table record is the one with Student-key 6167892 as the payment date is between effective date and expiry date.

Take this key and add an entry into the fact table!

## Tuition\_payment\_fact after ETL

Student_key	Date_key	Tuition_Payment_Amt
73273222	83853	\$4500.00
83291245	46467	\$3400.00
73678122	35356	\$7000.00
→6167892	73864	\$5000.00

new fact

ETL for a type 2 change in Fact table requires us to go to the dimension table.

"If we were loading historical data..."

TYOUNG21 - 8/13/2018 - \$4000

Record in dimension table for this date is 436431

Add entry into fact table using this student-key

After loading historic data !

Student_key	Date_key	Tuition_Payment_Amt
73273222	83853	\$4500.00
83291245	46467	\$3400.00
73678122	35356	\$7000.00
6167892	73864	\$5000.00
→436431	89231	\$4000.00

|historic  
fact

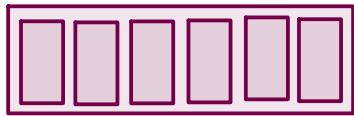
## IMPORTANT LESSON

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



### (8) Selecting your DW Environment

#### @ Decide between cloud and on premises settings for your Data Warehouse



IT megatrends

Big data and modern analytics

Social Media

Mobile technology

IoT and edge analytics

Cloud Computing

On premises

Platform and hashing implications

#### Advantages

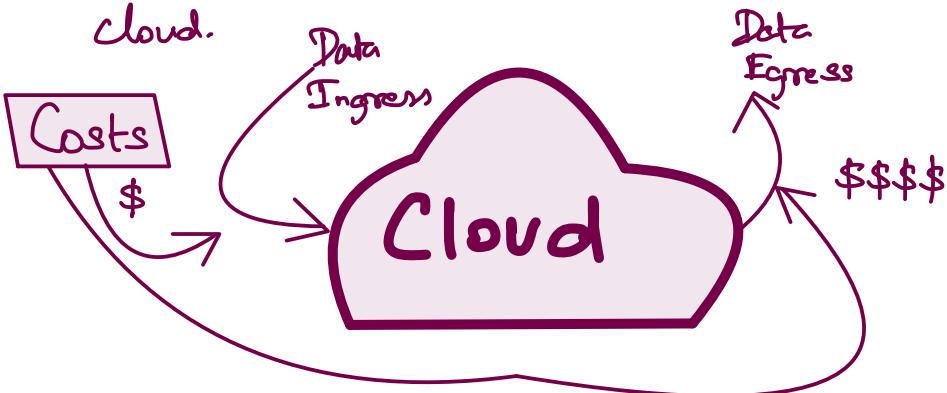
Offload routine system maintenance and upgrades  
(likely) lower initial platform investment.

Disaster Recovery

Data lake / big data synergies

## Challenges

- Security (2 edged sword) (some responsibility gets delegated to cloud providers)
- Migrating existing DWs from on-premises to cloud.
- cloud to cloud data transport  
e.g. OLTP (Azure) OLAP (AWS) ?
- Data transfer between corporate data center to cloud.



## Data Egress Cost challenges

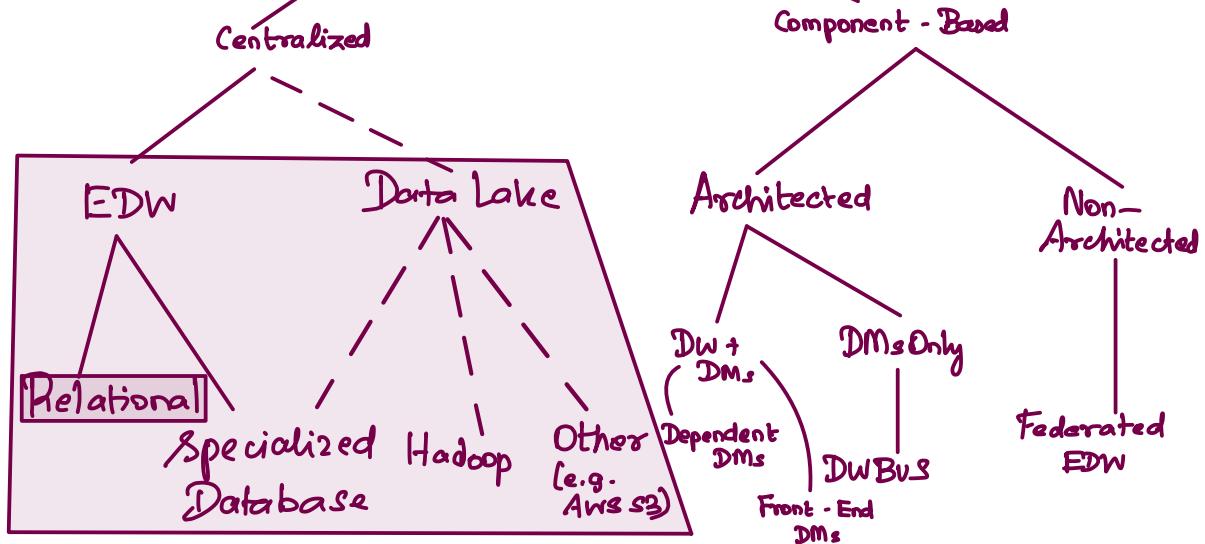
- Cost for large data volumes
- Not just operational data egress costs.
- Development, testing, QA, problem chasing...
- Experimental or exploratory analytics.

## (b) Architecture and Design Implications for your selected platform

The diagram below is from section 1.

Our focus was on Relational EDW

# DW11



Platform	User View (*)	Physical data model
On-prem RDBMS	Dimensional	Star / snowflake schema
On-prem DM Cube	Dimensional	Multidimensional
On-prem 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 DM Cube	Dimensional	Multidimensional
Cloud Specialized DB	Dimensional	Columnar, in memory, etc.

(\*) for OLAP / BI → Cloud based

Highlighted items trending upward as DW Platform

## Implications

- DW shifting to the cloud
- Data lakes alongside or succeeding data warehouses
- RDBMSs still prevalent
- ...but being supplemented or replaced even for OLAP / BI

Think Dimensional!  
(at least for user-facing BI)

Thank you for investing your time reading this. Thanks to Alan Simon for the wonderful course.

## Additional Resources

Star Schema : The Complete Reference  
The Data Warehouse Toolkit 3Ed.

THE  
END

SRIHARI  
SRIDHARAN