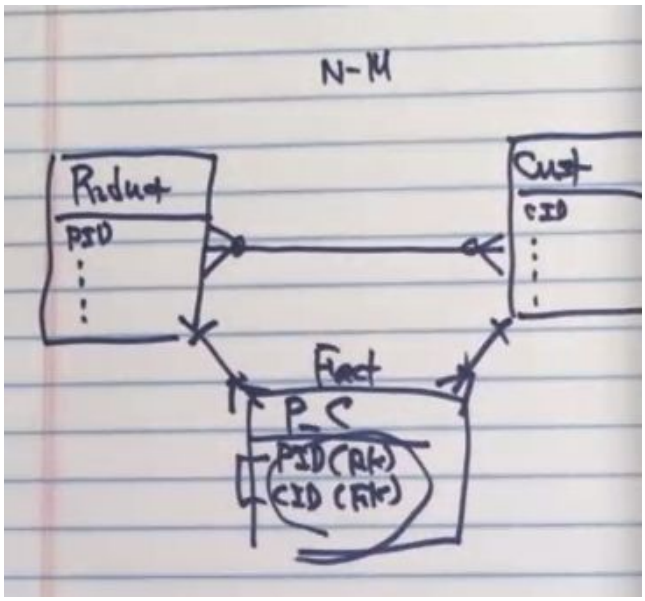
Day 0
Just syllabus

Day 1
S subject oriented - particular segment of a business / company
I integrated - multiple data sources integrated
N non-volatile - data is stable; added but not removed
T time-variant - all data is for a particular, identified point in time

Day 2
Why do we create warehouses? More user friendly and efficient / fast

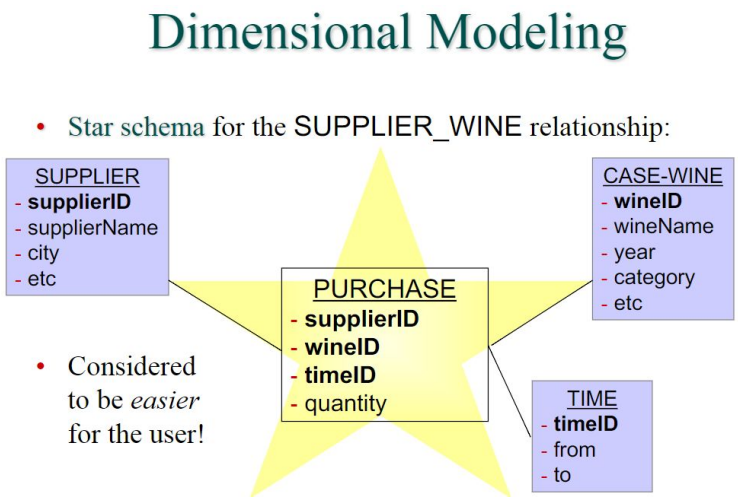
How to implement any to many - put a composite (fact) table in between using an associative identity



Look for any to many relationships as they will be a candidate for a fact table

Any data map MUST have time or date mapped (the T in SINT - time dimension)

Star schema is then created: fact table in middle with dimensional tables on the ends with one to many relationships



Day 3

Why do we need data warehousing?

- Accessible
- Roll-up / summary / details which can then be sliced, diced to specific data (ex. Getting data just from the state of NY)
- Easy, user friendly, fast, efficient
- Can show just what is important

Possible problems wh

- Duplicate data
- Incorrect or missing data
- Data with differing names (ex. 'Student Id' vs 'Id Number')

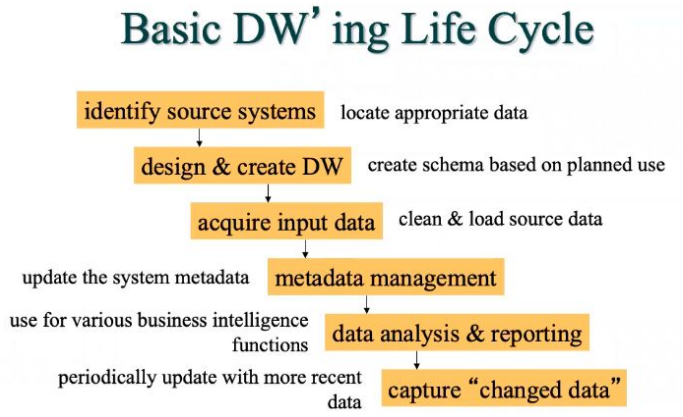
Operational systems = computing systems that provide the information necessary to run day to day business activities

Decision support systems = computing systems that provide vital strategic information for effective decision making

- Good strategic info:
  - Integrated: a single, enterprise-wide viewpoint
  - Data integrity: data is correct and accurately represents business rules
  - Credible: single source values
  - Accessible: ease of access, flexible for intuitive and investigational analysis
  - Timely: up to date ino is available when needed

Data warehousing definition - the activities needed to create, maintain, and utilize a data warehouse or data mart

- Creating
- Populating
- Querying (user accessing)



<- Handling dimensional change is important!

Ex: someone buy tv in NY state, person later moves to CA, so we need to handle that they lived in NY when it was bought and not CA - "Slowly changing dimension(al table)"

General Guidelines:

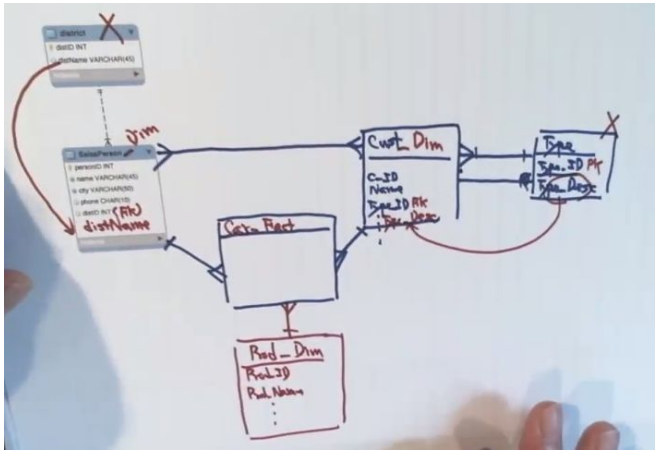
- Build from a clear set of business objectives and user requirements
- Define the architectural framework in advance
- Document all assumptions
- Use the right tools for the job
- Understand the life cycle
- Expect data problems
- Learn from mistakes

Any data map MUST have a time table (remember SINT)

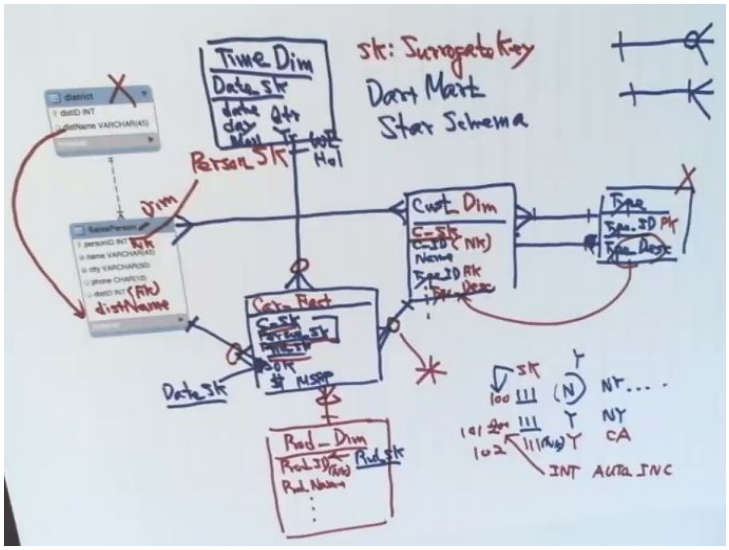
Day 4

Denormalizing a many to many relationship with an associative identity to a Data Mart (star schema):

1. Bring district name into sales person to remove the District table
2. Bring type description into customer to remove the Type table
3. Customer, sales person, and rod are all connected and represented by individual dimensional tables, connected by the “car” fact table



4. Surrogate key represented by ‘_SK’ (which are also the primary keys), all added to the car fact table, auto increments when information changes
5. We NEED a time/date table now (always!)
6. Add the circles to create ANY to one instead of many to one
7. For the original primary keys, we represent them as ‘_NK’, natural keys now



Tips for exercise 1:

Put Type_Desc into customer and get rid of Customer_Type table; this creates 'Customer_DIM' table

'Customer_ID' must now be natural key -> add a surrogate key 'Customer_SK'

Move Cat_Name into Product and get rid of Category table

Customer_ID and Product_ID in Sales table becomes _SK

Create Date dimensional table using the Date from the Sales table, with 'Date_SK' as primary/surrogate key

Change 'many to one' to 'any to one' with open circle on the many side

Bring a hard copy to class!

Major components of DW architecture:

- Data Acquisition
  - Extracts data from legacy systems & external sources, consolidates & summarizes the data, and loads it into the Data Storage
- Data Storage
  - Contains the integrated data, metadata, and associated software
- Information Delivery
  - Allows users to access and analyze data in the warehouse

Data Staging

- An area used to receive data from operational sources and prepare it to be placed into a data warehouse
- Extracting (E): read and understand the source data, and copy the parts that are needed to the data staging area for further work
- Transforming (T): Once the data is extracted, there are many possible transformation steps, including cleaning, purging, combining, creating **surrogate keys**, & building aggregates
- Loading (L): Initial load moves very large volumes of data, & the business conditions determine the refresh cycles

Day 5

In person class going over Exercise 1

Day 6

Data Warehouse vs. Data Marts

Data Warehouse	Data Mart
Enterprise-wide scope	Departmental scope
Union of all data marts	Focused on a single business process
Organized on E-R modeling (3rd Normal form)	Organized on Dimensional Model (Star Schema: Facts & Dimensions)
Structured for a corporate-wide view of the data	Structured to suit the departmental view of the data

Design process:

1. Select business process to model
2. Determine the grain (lowest level of detail) of the business process
3. Chose the dimensions that apply to each fact table row
4. Identify the numeric facts that will populate each fact table row

Day 7

Grain: granularity of the data at the detail level (row) for the measurements in a fact table

Grain selection - Kimball: "How do you describe a single row in a fact table?" What information does the user need

Ex. Line item on a receipt/bill, boarding pass for flight, monthly snapshot of bank transactions

Important: Keep the fact table at the lowest grain

If the data mart is not at the lowest, then you will need to make another dimensional table

Universe of discourse: whenever you create a database, you have to describe what the database is for (one sentence) so user can understand what the database (mart in this case) is all about

Fact Table Characteristics

- The table key is concatenated
- The grain of the data is identified
- Fully additive measures
- Semi-additive measures
- Large numbers of records
- Only a few attributes
- Sparse data
- Degenerate dimension

Dimension Table Characteristics

- Has a primary key attribute
- Many attributes/columns
- Typically mostly text attributes
- Attributes are not directly related
- "Flattened out" – i.e. not normalized
- Support drill down and roll up
- Contain multiple hierarchies
- Have fewer records than fact tables

Day 8
In person class going over Exercise 2.

Day 9
OLAP:

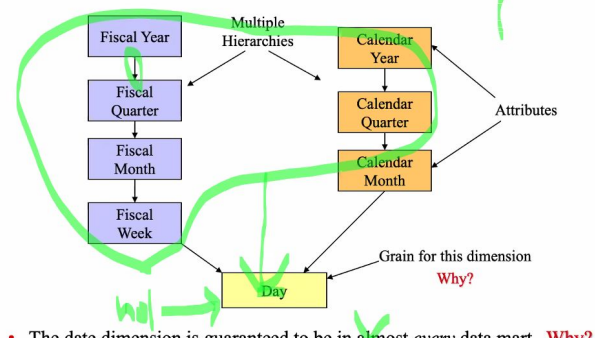
- Slice n dice
- Drill down
- Roll up

Hierarchy def: a cascaded series of many to one relationship across a series of descriptive entities
Ex. If hierarchy is by location (country - state - city) roll up goes from city up to state, roll down goes from state down to city

Date dimension will have two hierarchies - fiscal and calendar
Why must we have a date dimension? Data mart must have the time variant!

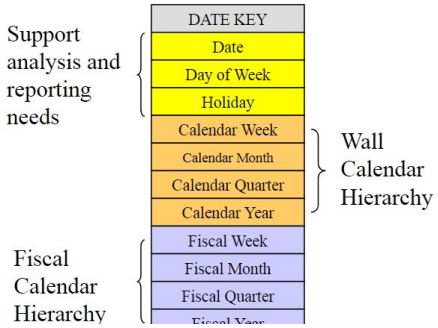
USE THIS FORMAT **NOT THIS**

Multiple Hierarchies Example: the Date Dimension



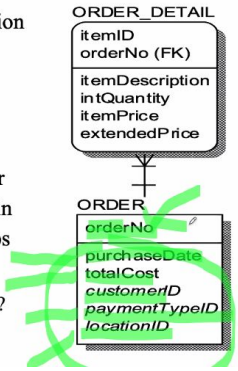
- The date dimension is guaranteed to be in almost *every* data mart. Why?

Multiple Hierarchies Example: Date Dimension Table



Degenerate Dimensions …

- After you identify attributes for the dimension and the fact tables, there may be some operational data items left.
  - Important related characteristics have been placed in a dimension.
- Example: order number, invoice number, or ticket number are the old parent keys used in parent-child transaction system relationships
  - What do we do with the order number?
  - Or the invoice number or the ticket number?
- Make a separate dimension table? Or not?



Degenerate Dimensions …

- Def. a degenerate dimension = a "leftover" dimension that is related to a key dimension in a model but is not at the *same* grain level as the important process transaction(s) that are being modeled.
  - a dimension that is left without attributes after the relevant dimensions have been defined.
- Often occur when a fact table's grain represents the lowest transactional-level data for a process.
- The decision to use degenerate dimensions is often based on the desire to provide a direct reference back to a parent entity in a transactional system without the overhead of maintaining a separate dimension table.
  - order numbers, invoice numbers, bill-of-lading numbers, etc

In the case above, instead of creating empty dimensional table (besides the surrogate key), you will put the SK (orderNo) into the fact table

Why don't we want to create another table? We need to keep the grain at the lowest level and the same for all tables

Day 10

Role Playing Dimensions

- Def. role-playing = a situation in which a single dimension appears several times in the *same* fact table.
- Example: date
  - frequently appears repeatedly
    - order date, ship date, Federal Express "staged" delivery dates
  - Each value of the date domain should be a foreign key in the fact table
- But, can we join *two or more* FKs to the *same* dimension table?
- Implementation options:
  1) Use database views to make one single physical table appear as multiple logical tables
  2) Create as multiple physical tables copies

  Which one is preferred?

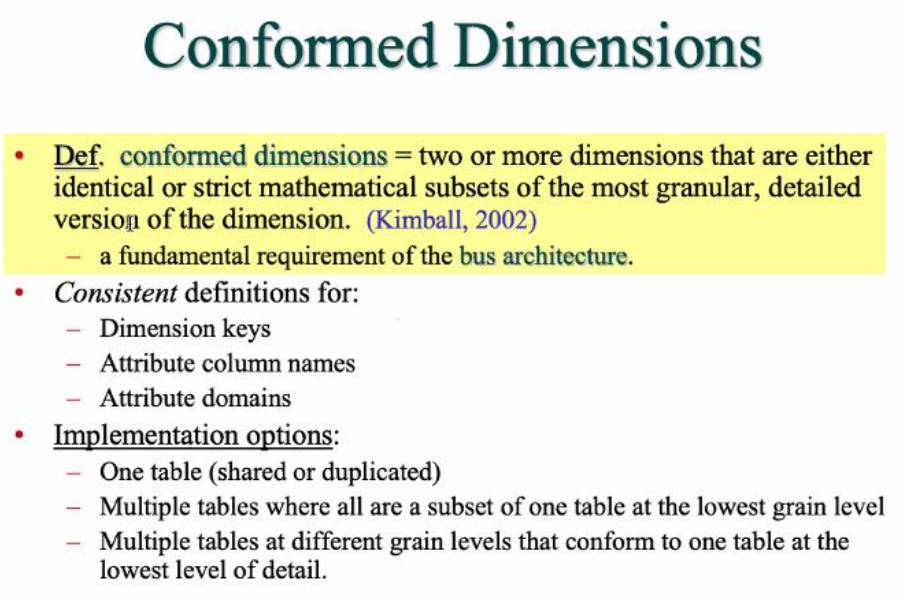
Chains and Circles …

- Many business processes have a logical flow that has a beginning and an end.
  - An order, or a product, or a customer may "evolve" over time through a serious of steps.
- Usually, it makes sense to make a *separate* fact table for each step.
- In the manufacturing world, this is called a **supply chain**:

- Raw material production
- Ingredient purchasing
- Ingredient delivery
- Ingredient inventory
- Bill of materials
- Manufacturing process control
- Manufacturing costs
- Packaging
- Trans-shipping to warehouse
- Finished goods inventory

Day 11
Remember the 3 types of dimensions: degenerate, role playing, and hierarchies

Important: Conformed dimensions - when user accesses more than 1 fact table / data marts, dimensions will be conformed (keys and attribute names are same):



Day 12
Going over report 1 and how to do exercise 3.

Day 13
Different types of fact tables?
Transaction (most of them bc of lowest level/detail), [periodic] snapshot (only defined at end of each defined period), aggregate

Different facts within the fact table?
Fully additive, semi-additive, and fact-less

Why does a fact table rarely change?
When there are updates to make for a fact table / fact, that comes in through a different batch
Ex. if sales was logged as 4000 but was actually 5000, log the extra 1000 with the next days sales

Slowly Changing Dimensions: characteristics of a dimension can change slowly over time
Type 0 - Retain Original. Dimensional attribute value never changes
Ex. Most attributes in a date dimension
Type 1 - Overwrite. Usually a data error where a value must be completely overwritten and old value is not important (it will be lost forever), easiest to implement
Ex. Department changes from an Education dep to a Strategy dep

Type 2 - Add New Row. Non error change occurs and history must be preserved, most used

Ex. Same as example for 1, but we keep Education dep with a new surrogate key

– Create a new product dimension row (new surrogate key) to reflect the new department attribute value:

(SK) ProdKey	ProdDesc	Dept	SKU# (NK)
12345	BrainBang 2.0	**Education**	ABC123
29876	BrainBang 2.0	**Strategy**	ABC123

New attributes should also be included:

Current flag - allows selecting the most current version

Effective/expiration dates - reflect time period for when a version of change was in effect

Therefore, 3 columns total are added.

New SK is first due to new value, then current flag to flag the most recent version (integer data type called version number), then the expiration date to tell when the change was in effect

Important:

When we update fact tables, if we cannot find the natural key which means that we cannot insert into the fact table, how do we handle this? Every dimensional table when implemented, the first thing to do is create a record with all null values called a 'null record'.

Ex. entry where surrogate key is '0' will be the null record

Type 3 - Add New Attribute. Adding a new column for the new value while preserving old values

Ex. Same as example for 1 and 2, but we create a new column called 'PriorDept' and we just update the original dept

(SK) ProdKey	ProdDesc	Dept	PriorDept	SKU#(NK)
12345	BrainBang 2.0	**Strategy**	**Education**	ABC123

Day 14

In class going over PE03.

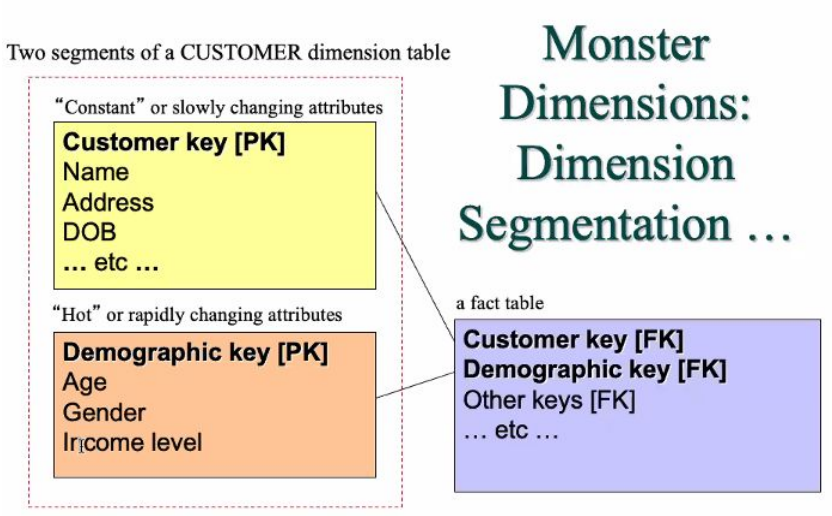
Day 15

Monster Dimensions

- Massively large dimension tables are a business reality.
  - Multi-million row tables are common: customer table for a telecommunication conglomerate
- Causes two (2) types of challenges:
  - Can take too long to join to a fact table for data "browsing"
  - SCD techniques for handling changes can add even more rows
    - SCD2 or hybrid approaches
- Solution:
  - Break off frequently analyzed or frequently changing attributes into a *separate* dimension.
    - a minidimension

Type 4 - Monster dimension: extremely large tables

- Will take a long time to find a particular record
- Create a mini dimension by creating a separate dimensional table and putting in attributes that frequently change / are used
- Can also put attributes into a range - like income is 20,000-24,999



Day 16

Type 5 - Add SCD 4 and SCD 1, the star schema starts going towards a snowflake schema

SCD Type 5: SCD4+SCD1

- Add Mini-Dimension and Type 1 Outrigger
- SCD 4 + SCD 1
- Join base dimension table and "current profile" mini-dimension
- The "current profile" mini-dimension is SCD Type 1

Type 6 - Add SCD 2, 3, and 1

SCD Type 6

- Add Type 1 attribute to Type 2 dimension
- A Type 2 row with a Type 3 column
- Type 3 column is overwritten as a Type 1
- Type 2+3+1

Type 7 -

SCD Type 7

- Fact table contains dual FK's for a dimension table
  - A SK linked to the dimension table where type 2 attributes are tracked
  - A DK (Durable Key) linked to present current attribute values
- Same as Type 6, but it's accomplished via DK instead of overwriting the current attributes

SCD Techniques Summary

DWT3 Fig 5-17

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change to attribute value	Facts associated with attribute's original value
Type 1	Overwrite attribute value	Facts associated with attribute's current value
Type 2	Add new dimension row for profile with new attribute value	Facts associated with attribute value in effect when fact occurred
Type 3	Add new column to preserve attribute's current and prior values	Facts associated with both current and prior attribute alternative values
Type 4	Add mini-dimension table containing rapidly changing attributes	Facts associated with rapidly changing attributes in effect when fact occurred
Type 5	Add type 4 mini-dimension, along with overwritten type 1 mini-dimension key in base dimension	Facts associated with rapidly changing attributes in effect when fact occurred, plus current rapidly changing attribute values
Type 6	Add type 1 overwritten attributes to type 2 dimension row, and overwrite all prior dimension rows	Facts associated with attribute value in effect when fact occurred, plus current values
Type 7	Add type 2 dimension row with new attribute value, plus view limited to current rows and/or attribute values	Facts associated with attribute value in effect when fact occurred, plus current values

Junk Dimensions:
Attributes from dimensions are “left over”

Junk Dimensions …

- Occasionally there are attributes from dimensions that are just “left over.”
  - Relics from older systems that are just not important now
  - Values that only have relevance for occasional use
- Options for handling these dimensions:
  - Exclude and discard them
    - May not want to do since could have useful information …
  - Include them in the fact table exactly as they were coded
    - May make the fact table unnecessary large
  - Setup a separate dimension table for *each* one
    - Likely to cause too many unnecessary, low-use dimensions
- Options for handling these dimensions (continued):
  - Keep those likely be the most useful in a special “junk” dimension table.
    - May be useful for constraining queries
  - Combine several of these dimension attributes into a single dimension using a derived attribute.
- How do you decide if a junk dimension is reasonable?
  - Do a quick calculation on the number of rows you’d need.
    - 5 attributes @ 3 values each = 3^5 = 243 rows ← OK
    - 5 attributes @ 100 values each = 100^5 = 100,000,000 rows ← Ouch!
      - Calculation assumes that the values are unrelated

Day 17
In class lab-ish discussion

Day 18

The ETL Process …

- Def. **ETL** = the process of obtaining data from various sources and preparing it for storage in a data warehouse so that it can be used to provide reliable strategic information.
- Acronym for data extraction, transformation and load.
  - extracting data from operational data stores or external data sources
  - transforming the data includes cleansing, aggregation, summarization, integration, as well as basic coding transformations
  - loading the data into some form of a data warehouse
    - ODS, enterprise data warehouse, data mart
- Goal = clean, consistent, integrated and – possibly – summarized data

Conforming solves an issue: when user is trying to retrieve info from multiple data marts (drill across), you cannot do a join

ETL Steps in Detail

1) Identify all target data. What's needed and in what format?
2) Determine all internal and external data sources.
3) Prepare mappings of source data to target data.
   ETL Toolkit Fig 3.1 Logical Data Map
4) Define the data extraction rules.
5) Determine the data cleansing and transformation rules.
6) Plan needed aggregate data tables.
7) Setup data staging area and test all tools.
8) Create procedures for loading the data.
9) Create – i.e. "ETL" – the dimension tables.
10) Create the fact tables.
11) Define user metadata.

Step 3 important to remember

Very Challenging!

- Data sources can be diverse and disparate. ETL "issues" include:
  - Multiple hardware platforms and operating systems.
  - Multiple versions of a source system with different architectures.
  - Data stored under older/obsolete technologies.
  - No historical information on data format changes.
  - Poor data quality
  - A data value may have different domains across systems.
    - Salary = weekly salary, monthly salary, bi-monthly salary
  - Avoid impacting the performance of operational systems.
- You may not be able to resolve data mismatches and inconsistencies across various source systems.

Day 19

Dirty data: data that is of poor quality due to inconsistencies, age, or incompleteness

Dimensions of Data Quality …

- Relevancy = the extent to which data is applicable and helpful for a specific task.
- Reputation = the extent to which data is highly regarded in terms of its source of content.
- Security = the extent to which access to data is restricted to appropriately maintain its security.
- Timeliness = the extent to which the data is sufficiently up-to-date for the task at hand.
- Understandability = the extent to which data is easily comprehended.

- Value-Added = the extent to which data is beneficial and provides advantages from its use.
- Cost-Effectiveness = the extent to which data costs the organization.

Data Cleansing

- Def. data cleansing = the transformation of data from its current state to a pre-defined, standardized format that is appropriate for information analysis.
  - Done with in-house code, vendor tools, and manually
- Data is cleansed by:
  - "Scrubbing" the data to eliminate items that are incorrect or incomplete, and
  - "Conforming" the data at the data integration step.

  Either:
  - *before* it is loaded into a data warehouse, or
  - As it is used *after* it is loaded ("clean as you go").

Day 20

In class going over PE04.

Day 21

Midterm exam highlights:

Conformed dimensions - definition and what do we mean by this?

Design process: One data mart can be shared by 2 or more processes -> choose grain (most detailed level) and why do we choose this?

Dimensions should have hierarchies otherwise it is not contributing

Why do we include degenerate dimension (DD)?
Types of fact tables (not just facts) - most are transactional as they involve a transaction
Junk dimension is not really junk (just a buzzword)
Every data warehouse must have a null record

Day 22
Going over Report 2 and SAS homework.

Day 23
Exam

Day 24
Explaining SAS - lab 2.

Day 25
SAS again