THE UNIVERSITY OF WESTERN AUSTRALIA

Data Warehousing

Lecture 5 Practical Concerns of Dimensional Modelling

CITS3401
CITS5504

Zeyi Wen

Computer Science and Software Engineering

School of Maths, Physics and Computing

Acknowledgement: The lecture slides are based on online sources.

Project 1

- **An example of dimension with few levels of concept**
  - "Gender Type"→ All→
- **You need to make decisions by yourself.**
  - How many dimension tables to use?
  - Whether I should delete xxx?
  - There is no unique solution. Everything is trade-off.

- **Some students have completed and submitted Project 1.**
  - The quality of the submission looks good.
  - Well done.

- **No new lab sheet for Week 6; we'll work on Project 1.**

A quick overview of the past topics

THE UNIVERSITY OF WESTERN AUSTRALIA

- Multidimensional Cubes
- Schema (Star, Snowflake, Fact Constellation)
- StarNet; Business Query Footprints
- Concept Hierarchies
- OLAP Operations; MOLAP vs. ROLAP
- group by cube() vs. group by rollup()
- Fact and Dimension Tables; ETL

Lecture Outline

- **Dimension Topics**
  - How many dimensions?
  - Date/Time Dimensions
  - Surrogate Keys
- **Fact Topics**
  - Semi-additive facts
  - “Factless” fact tables
- **Slowly Changing Dimensions**
  - Overwrite history
  - Preserve history
  - Hybrid schemes
- **More dimension topics**
  - Dimension roles
  - Junk dimension
- **More fact topics**
  - Multiple currencies
  - Master/Detail facts and fact allocation
  - Accumulating Snapshot fact tables

Data Warehouse Design Template (Lecture 2)

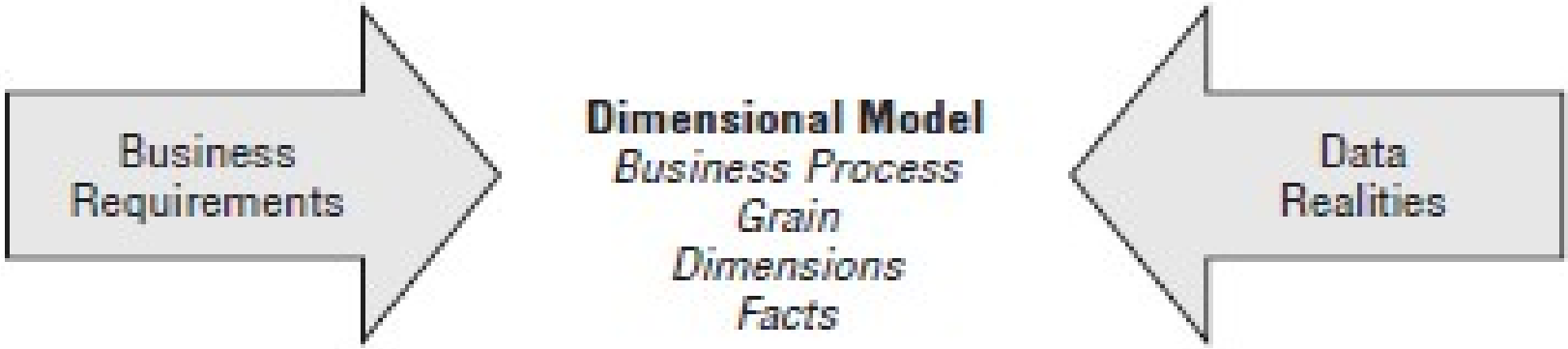
THE UNIVERSITY OF WESTERN AUSTRALIA

Kimball's four steps

- Identify a business process to model
  - E.g. orders, invoices, shipments, sales …
- Determine the grain of the business process
  - E.g. individual transactions, individual daily snapshots
- Choose the dimensions that apply to fact table rows
  - Example dimensions are time, item, customer, supplier, transaction type and status
- Identify the measure that populates fact table rows
  - Typical measures are numeric additive quantities like dollars_sold and units_sold

Kimball's four steps dimensional modeling

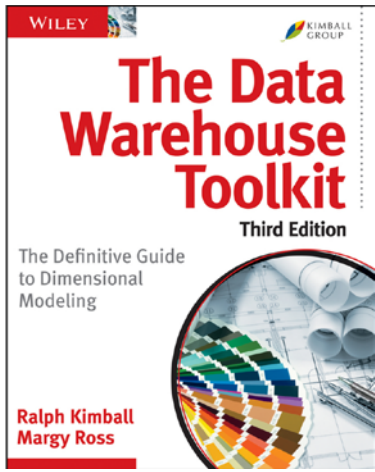
- Step 1: Identify the process being modelled.
- Step 2: Determine the grain at which facts will be stored.
- Step 3: Choose the dimensions.
- Step 4: Identify the numeric measures for the facts.



Running Example – Retail Sales

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Grocery store chain recording POS retail sales**
  - Same example used in “The Data Warehouse Toolkit", Chapter 3
  - POS = Point of sale
    - Data collected by bar-code scanners at cash register
  - 100 grocery stores in 5 states
  - ~60,000 product SKUs
    - SKU = stock keeping unit
    - Represents an individual product
    - Some have UPCs (Universal Product Codes) assigned by manufacturer
    - Others don’t (for example, produce, bakery, meat, floral)
  - Goal: understand impact of pricing & promotions on sales, profits
    - Promotions = coupons, discounts, advertisements, etc.



Step 1: Retail Sales Questions

- **What is the lift due to a promotion?**
  - Lift = gain in sales in a product because it's being promoted
  - Requires estimated baseline sales value
    - Could be calculated based on historical sales figures
- **Detect time shifting**
  - Customers stock up on the product that's on sale
  - Then they don't buy more of it for a long time
- **Detect cannibalisation**
  - Customers buy the promoted product instead of competing products
  - Promoting Brand A reduces sales of Brand B
- **Detect cross-sell of complementary products**
  - Promoting charcoal increases sales of lighter fluid
  - Promoting hamburger meat increases sales of hamburger buns
- **What is the profitability of a promotion?**
  - Considering promotional costs, discounts, lift, time shifting, cannibalisation, and cross-sell

Step 2: Grain of a fact table

THE UNIVERSITY OF WESTERN AUSTRALIA

- Grain of a fact table = the meaning of one fact table row
- Determines the maximum level of detail of the warehouse
- Example grain statements: (*one fact row represents a…*)
  - Line item from a cash register receipt
  - Boarding pass to get on a flight
  - Daily snapshot of inventory level for a product in a warehouse
  - Sensor reading per minute for a sensor
  - Student enrolled in a course
- Finer-grained fact tables:
  - are more expressive
  - have more rows
- Trade-off between performance and expressiveness
  - Rule of thumb: Errors in favor of expressiveness
  - Pre-computed aggregates can solve performance problems

A sample cash register receipt

THE UNIVERSITY OF WESTERN AUSTRALIA

Allstar Grocery
123 Loon Street
Green Prairie, MN 55555
(952) 555-1212

Store: 0022
Cashier: 00245409/Alan

0030503347 Baked Well Multigrain Muffins	2.50
2120201195 Diet Cola 12-pack	4.99
Saved $.50 off $5.49	
0070806048 Sparkly Toothpaste	1.99
Coupon $.30 off $2.29	
2840201912 SoySoy Milk Quart	3.19
TOTAL	12.67
AMOUNT TENDERED	
CASH	12.67
ITEM COUNT: 4	

Transaction: 649 4/15/2013 10:56 AM

Thank you for shopping at Allstar

00649002204152013002454O9

Step 3: Choosing dimensions

THE UNIVERSITY OF WESTERN AUSTRALIA

- Determine a candidate key based on the grain statement.
  - Example 1: a student enrolled in a course
    - (Course, Student, Term) is a candidate key
  - Example 2: line item from cash register receipt
    - (Transaction ID, Product SKU) is a candidate key

- Add other relevant dimensions that are functionally determined by the candidate key.
  - Example 1: Instructor and Classroom
    - Assuming each course has a single instructor!
  - Example 2: Store, Date, and Promotion

Step 4: Some numeric measures

- Quantity sold
- Total dollar sales
- Unit price
- Percentages (% discount)
- …

Date Dimension

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Nearly every data warehouse will have one**
  - Most data marts are time series
  - Allows us to find historical/temporal trends
- **Typical grain: each row = 1 day**
  - For finer-grained time measurements, use separate date and time-of-day dimensions
- **Properties of a date**
  - Holiday/non-holiday, weekday/weekend, day of week
  - Selling season (Back-to-school, Christmas, etc.)
  - Fiscal calendar (fiscal quarter/year)
  - Day-number-in-year, Day-number-in-epoch
    - Allows for easy date arithmetic
- **Why use a date dimension instead of a SQL timestamp?**
  - Capture interesting date properties (e.g. holiday/non-holiday)
  - Avoid relying on special date-handling SQL functions
  - Make use of indexes

Date dimension and sample rows

THE UNIVERSITY OF WESTERN AUSTRALIA

Date Dimension
Date Key (PK)
Date
Full Date Description
Day of Week
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Year
Fiscal Week
Fiscal Week Number in Year
Fiscal Month
Fiscal Month Number in Year
Fiscal Year-Month
Fiscal Quarter
Fiscal Year-Quarter
Fiscal Half Year
Fiscal Year
Holiday Indicator
Weekday Indicator
SQL Date Stamp
...

Date Key	Date	Full Date Description	Day of Week	Calendar Month	Calendar Quarter	Calendar Year	Fiscal Year-Month	Holiday Indicator	Weekday Indicator
20130101	01/01/2013	January 1, 2013	Tuesday	January	Q1	2013	F2013-01	Holiday	Weekday
20130102	01/02/2013	January 2, 2013	Wednesday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130103	01/03/2013	January 3, 2013	Thursday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130104	01/04/2013	January 4, 2013	Friday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130105	01/05/2013	January 5, 2013	Saturday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130106	01/06/2013	January 6, 2013	Sunday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130107	01/07/2013	January 7, 2013	Monday	January	Q1	2013	F2013-01	Non-Holiday	Weekday
20130108	01/08/2013	January 8, 2013	Tuesday	January	Q1	2013	F2013-01	Non-Holiday	Weekday

- Each column in the date dimension table is defined by the particular day that the row represents.
- The day-of-week column contains the day's name, such as Monday.
- The day number in calendar month column starts with 1 at the beginning of each month and runs to 28, 29, 30, or 31 depending on the month.
- Similarly, you could have a month number in year (1, . . ., 12).
- Are these redundant? Should they be computed using stored procedure in SQL?

Flags and Indicators as Text

- Use meaningful values for indicators, because dimension table attributes serve as report labels and values appear in drill-down query filter lists.

- For example, a holiday indicator
  - Y/N, 1/0, or True/False or
  - Holiday or Non-holiday

Monthly Sales

Period: June 2013

Product Baked Well Sourdough

Holiday Indicator	Extended Sales Dollar Amount
N	1,009
Y	6,298

OR

Monthly Sales

Period: June 2013

Product Baked Well Sourdough

Holiday Indicator	Extended Sales Dollar Amount
Holiday	6,298
Non-holiday	1,009

Figure 3-6: Sample reports with cryptic versus textual indicators.

Time-of-Day as a Dimension or Fact?

Date Dimension
Date Key (PK)
Date
Full Date Description
Day of Week
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Year
Fiscal Week
Fiscal Week Number in Year
Fiscal Month
Fiscal Month Number in Year
Fiscal Year-Month
Fiscal Quarter
Fiscal Year-Quarter
Fiscal Half Year
Fiscal Year
Holiday Indicator
Weekday Indicator
SQL Date Stamp
...

Hour
Minutes
Second

Date Dimension
Date Key (PK)
Date
Full Date Description
Day of Week
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Year
Fiscal Week
Fiscal Week Number in Year
Fiscal Month
Fiscal Month Number in Year
Fiscal Year-Month
Fiscal Quarter
Fiscal Year-Quarter
Fiscal Half Year
Fiscal Year
Holiday Indicator
Weekday Indicator
SQL Date Stamp
...

Fact table
… Hour Minutes Second …

Time-of-Day as a Dimension or Fact?

- Separate time-of-day from the date dimension to avoid a row count explosion in the date dimension.
  - A date dimension with 20 years of history contains approximately 7,300 rows.
  - Changing the grain to one row per minute in a day, over 10 million rows are needed to accommodate the 1,440 minutes per day.
  - If you tracked time to the second, you'd have more than 31 million rows per year!
- Use a time-of-day dimension table with one row per discrete time period, to filter or roll up time periods based on summarised day part groupings, for example
  - activity during 15-minute intervals, hours, shifts, lunch hour, or prime time,
  - one row per minute in a 24-hour period resulting in a dimension with 1,440 rows.
- If there's no need to roll up or filter on time-of-day groupings, time-of-day should be handled as a simple date/time fact in the fact table.
  - Business users are often more interested in time lags, such as the transaction's duration, rather than discrete start and stop times.
  - Time lags can easily be computed by taking the difference between date/time stamps.

Surrogate Keys

- **Primary keys of dimension tables should be surrogate keys, not natural keys**
  - Natural key: A key that is meaningful to users
  - Surrogate key: A meaningless integer key that is assigned by the data warehouse
  - Avoid using natural keys or codes generated by operational systems.
    - E.g. Account number, UPC code, Social Security Number
  - Syntactic vs. semantic

Benefits of Surrogate Keys

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Data warehouse insulated from changes to operational systems**
- **Easy to integrate data from multiple systems**
  - What if there's a merger or acquisition?
- **Narrow dimension keys (e.g. 8 bytes using natural key, only 4 bytes using surrogate key)**
  - Thinner fact table → Better performance
  - This can actually make a big performance difference.
- **Better handling of exceptional cases**
  - For example, what if the value is unknown or TBD?
  - Using NULL is a poor option
    - Three-valued logic is not intuitive to users
    - Users may get their queries wrong
    - Join performance will suffer
  - Better: Explicit dimension rows for "Unknown", "TBD", "N/A", etc.
- **Avoids tempting query writers to assume implicit semantics**
  - Example: WHERE date_key < '01/01/2020'
  - Will facts with unknown date be included?

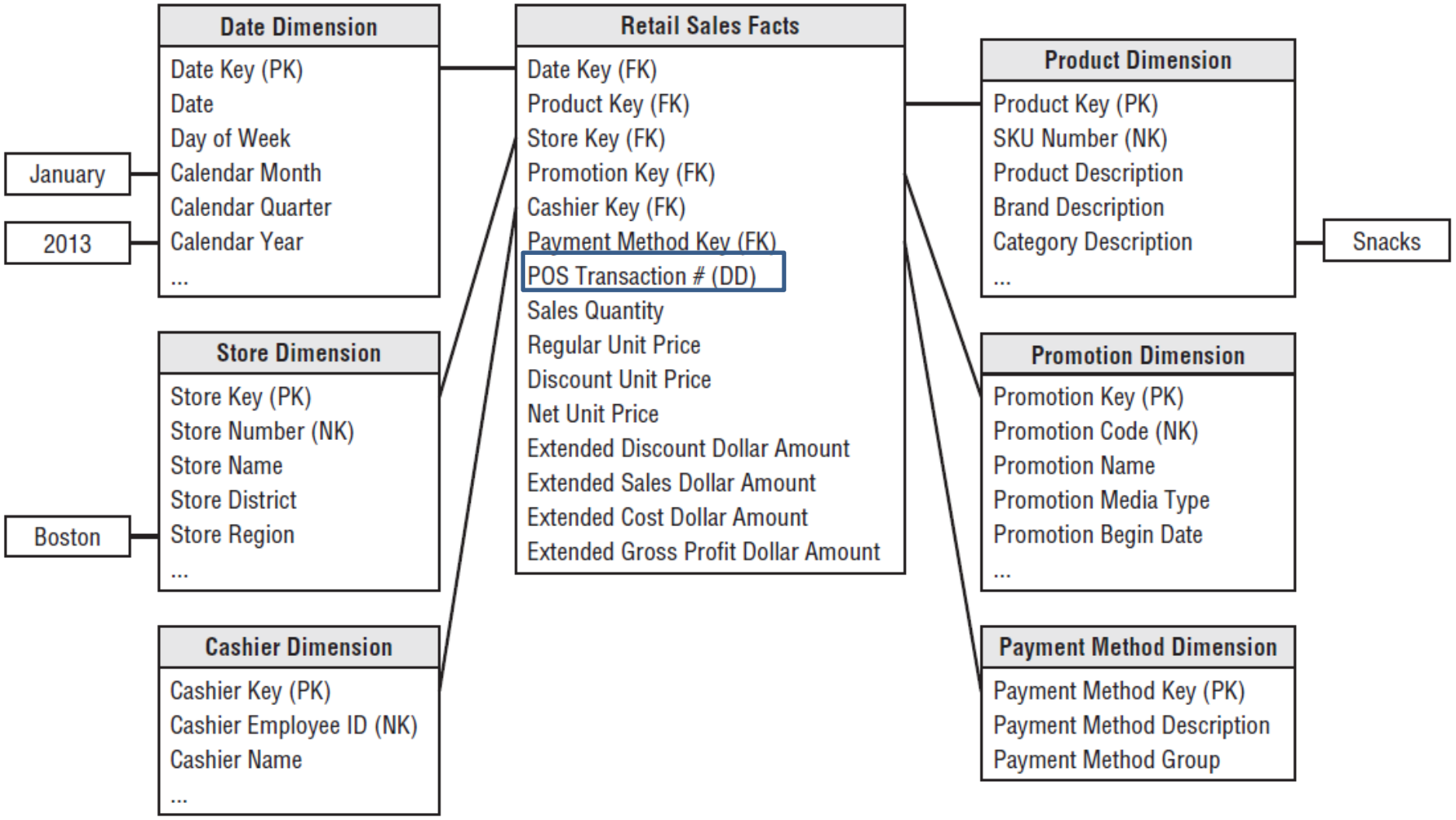
Degenerate dimension

- The degenerate dimension is a dimension key without a corresponding dimension table.

- Occasionally a dimension is merely an identifier, without any interesting attributes
  - "Transaction ID" is a unique identifier, and serves to group together products that were bought in the same shopping cart
  - (Transaction ID, Product) was the candidate key.

- Two options:
  - Discard the dimension
    - A good option if the dimension isn't needed for analysis
  - Use a "degenerate dimension"
    - Store the dimension identifier directly in the fact table
    - Don't create a separate dimension table
    - Used for transaction ID, invoice number, etc.

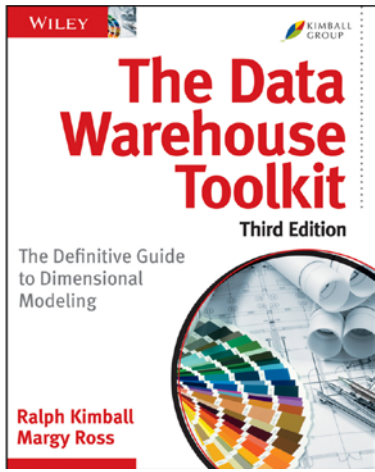
Which is the degenerated dimension?



Running Example – Retail Sales (again)

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Grocery store chain recording POS retail sales**
  - Same example used in “The Data Warehouse Toolkit", Chapter 3
  - POS = Point of sale
    - Data collected by bar-code scanners at cash register
  - 100 grocery stores in 5 states
  - ~60,000 product SKUs
    - SKU = stock keeping unit
    - Represents an individual product
    - Some have UPCs (Universal Product Codes) assigned by manufacturer
    - Others don’t (for example, produce, bakery, meat, floral)
  - Goal: understand impact of pricing & promotions on sales, profits
    - Promotions = coupons, discounts, advertisements, etc.



How many dimensions?

- Should two concepts be modeled as separate dimensions or two attributes of the same dimension?

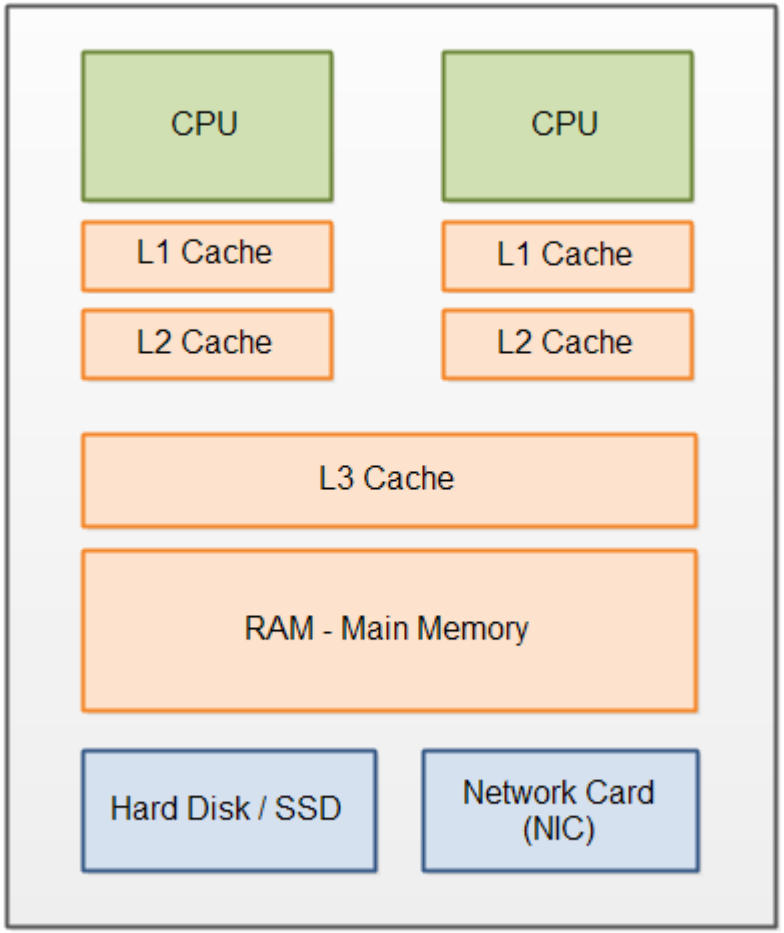
- Example: Different types of promotions
  - Ads, discounts, coupons, end-of-aisle displays
  - Option A: 4 dimensions
    - Separate dimension for each type of promotion
  - Option B: 1 dimension
    - Each dimension row captures a combination of ad, discount, coupon, and end-of-aisle display

- Factors to consider
  - How do the users think about the data?
    - Are an ad and a coupon separate promotions or two aspects of the same promotion?
  - Fewer dimensions = fewer tables → good performance
    - Generally fewer tables = simpler design
  - Performance implications
    - See following slides…

How many dimensions – performance

- Most OLAP queries are “I/O bound”
  - Data-intensive not compute-intensive
  - Reading the data from disk is the bottleneck for “typical” queries, on “typical” hardware.
- Size of data on disk ≈ query performance
  - Keeping storage requirements small is important
- Dimensional modeling impacts storage requirements

Memory Hierarchy (aside)



Motherboard

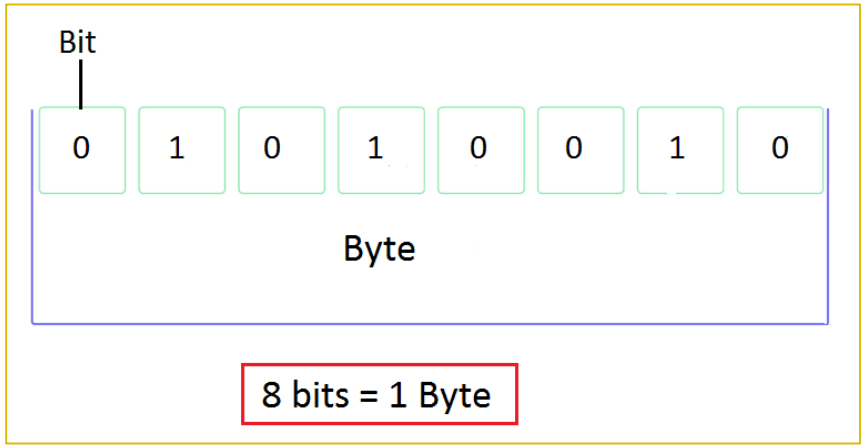
RAM—Main Memory

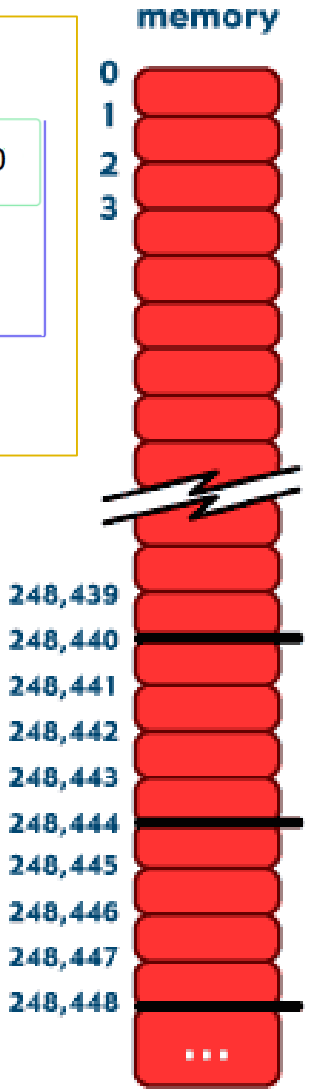
Basic Datatypes (aside)

THE UNIVERSITY OF WESTERN AUSTRALIA

Typical size of the types:

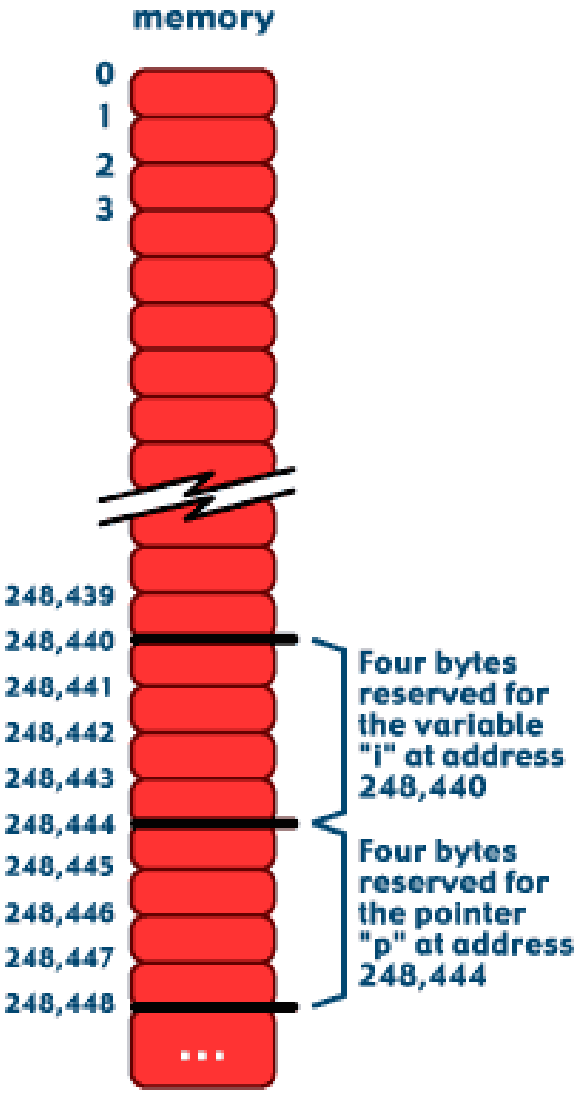
- bool: 1 byte
- char: 1 byte
- int and float: 4 bytes
- double: 8 bytes

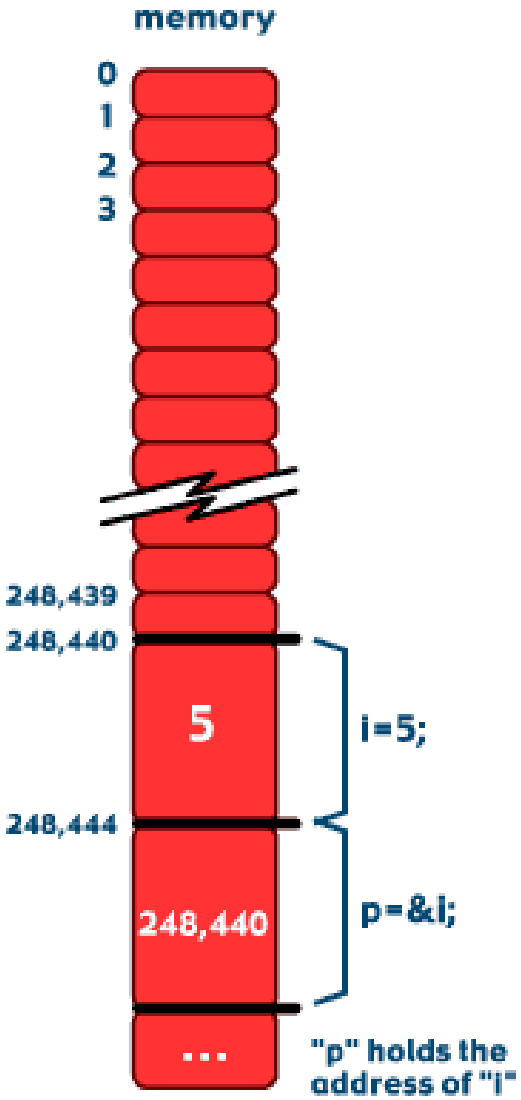




typename	description, and an example of variable initialization
bool	Boolean (truth values), which may only hold the values of either **true** or **false** *e.g.* **bool** finished = **false**;
char	character values, to each hold a single values such as an alphabetic character, a digit character, a space, a tab... *e.g.* **char** initial = 'C';
int	integer values, negative, positive, and zero *e.g.* **int** year = 2006;
float	floating point values, with a typical precision of 10 decimal digits *e.g.* **float** inflation = 4.1;
double	"bigger" floating point values, with a typical precision of 17 decimal digits *e.g.* **double** pi = 3.1415926535897932;

Variables—Main Memory (aside)





- Variables are locations in memory.
  - ✓ it has many addressable memory locations.
- 4 bytes to hold a single integer value
- Any variable can only hold a single value at any time.

Performance Implications

THE UNIVERSITY OF WESTERN AUSTRALIA

FactTable
Foreign key(s) Measure 1 Measure 2 Measure 3

- Let's consider the extremes
- Assumptions:
  - 100 million fact rows
  - 3 four-byte measurement columns in the fact table
  - 100 dimensional attributes, average size = 20 bytes

- Three modelling options
  - One "Everything" dimension
  - Each attribute gets its own dimension table
  - 5 dimensions (Date, Product, Store, Promotion, Transaction ID)

Option A

THE UNIVERSITY OF WESTERN AUSTRALIA

FactTable
Dimension key
Measure 1
Measure 2
Measure 3

- **Option A: One "Everything" dimension**
  - Fact table is very thin (16 bytes per row)
    - 3 four-byte fact columns
    - 1 four-byte foreign key to the Everything dimension
  - Dimension table is very wide (2000 bytes per row)
    - 100 attributes * 20 bytes each
  - Dimension table may has as many rows as fact table!
    - Each fact row has a different combination of attributes
  - Total space = 1.6 GB fact + 200 GB dimension
    - 16 bytes * 100 million rows = 1.6 GB
    - 2000 bytes * 100 million rows = 200 GB

Option B

- **Option B: Each attribute gets its own dimension table**
  - Manager First Name dimension, Manager Last Name dimension, etc.
  - Fact table is wide (212 bytes per row)
    - Assume 2-byte key each dimension table
    - 2 bytes x 100 dimensions = 200 bytes
    - 2 bytes per key is a generous assumption
  - Dimension tables are very thin, have few rows
  - Space for fact table = 21.2 GB
    - 212 bytes per row x 100 million rows = 21.2 GB
  - Space for dimension tables = negligible
    - < 132 MB total for all dimensions
    - No dimension table has more than 60,000 rows
      - 2-byte key has 2^16 unique values ≈ 60,000.
    - Each dimension row is 22 bytes
      - 2 bytes for key + 20 bytes for dimensional attributes
    - 100 dimension tables x 60,000 rows per table x 22 bytes per row ≈ 132 MB

FactTable
Measure 1
Measure 2
Measure 3
Dimension key1
Dimension key2
Dimension key3
…
Dimension key100

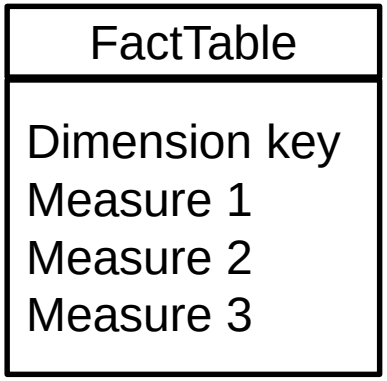
Option C

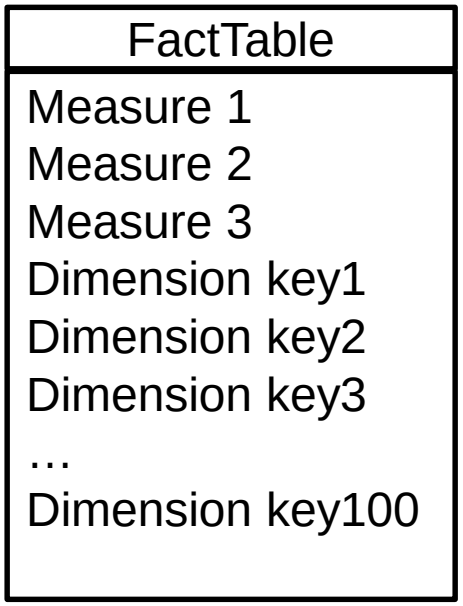
- **Option C: 4 dimensions (Date, Product, Store, Promotion)**
  - Fact table is quite thin (28 bytes per row)
    - 2-byte keys for Date and Store = 4 bytes
    - 4-byte keys for Product, Promotion, Transaction ID = 12 bytes
    - 4-byte fact columns for 3 measures = 12 bytes
  - Dimension tables are wide, have few rows
    - No dimension table has more than 60,000 rows
  - Space for fact table = 2.8 GB
    - 28 bytes * 100 million rows
  - Space for dimension tables = negligible
    - < 130 MB for all dimensions
    - 4 dimension tables x 60,000 rows per table x 25 attributes per row x 20 bytes per attribute = 120 MB
    - 4 bytes per key x 60,000 rows x 4 tables ≈ 1 MB

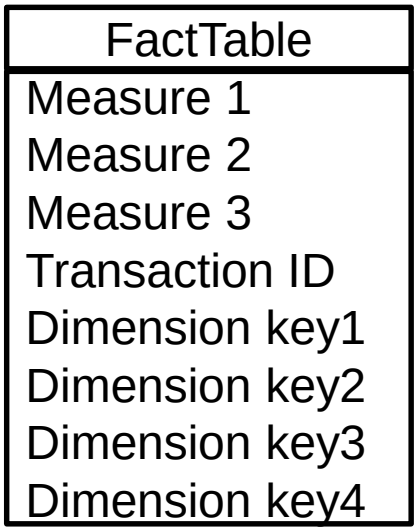
FactTable
Measure 1
Measure 2
Measure 3
Transaction ID
Dimension key1
Dimension key2
Dimension key3
Dimension key4

Summary of Option A, B and C

- **Option A: One "Everything" dimension**
  - Space consumption: 200 GB
- **Option B: Each attribute gets its own dimension table**
  - Space consumption: 21.2 GB
- **Option C: 4 dimensions (Date, Product, Store, Promotion)**
  - Space consumption: 2.8 GB







Why Option C is the best?

- **Attributes that pertain to the same logical object have a high degree of correlation**
  - Correlated attributes (Option C makes use of)
    - (product name, brand)
    - Number of distinct combinations = number of distinct products
    - Product name and brand are completely correlated
  - Uncorrelated attributes (Option A)
    - (product name, date)
    - Number of distinct combinations = number of products * number of dates
    - No correlation between date and product
    - Most possible combinations of values will appear in the fact table
  - Combining non-correlated attributes in the same dimension leads to blow-up in size of dimension table (Option A)
- **When attributes are semi-correlated, designer has a choice**
  - Frequently, multiple types of promotion occur together
  - E.g. product being promoted has ad, coupon, and in-store display
  - Number of (ad, coupon, discount, display) combinations is small
  - Combining them in a single Promotion dimension is reasonable

Lecture Outline

- **Dimension Topics**
  - How many dimensions?
  - Date/Time Dimensions
  - Surrogate Keys
- **Fact Topics**
  - Semi-additive facts
  - “Factless” fact tables
- **Slowly Changing Dimensions**
  - Overwrite history
  - Preserve history
  - Hybrid schemes
- **More dimension topics**
  - Dimension roles
  - Junk dimension
- **More fact topics**
  - Multiple currencies
  - Master/Detail facts and fact allocation
  - Accumulating Snapshot fact tables

Fact tables – what to store (Lecture 4)

- **Additive facts (measures) are easy to work with**
  - Summing the fact value gives meaningful results
  - Additive facts:
    - Quantity sold
    - Total dollar sales
  - Non-additive facts:
    - Averages (average sales price, unit price)
    - Percentages (% discount)
    - Ratios (gross margin)

Month	Quantity Sold
June	12
July	10
August	14
OVERALL	36

Month	Avg. Sales Price
June	$35
July	$28
August	$30
OVERALL	$93 ← Wrong!

Store Additive Quantities (Lecture 4)

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Store additive quantities in the fact table**
- **Example:**
  - Don't store "unit price"
  - Store "quantity sold" and "total price" instead
- **Store additive summaries used for distributive aggregates**
  - Numerator and denominator for averages, percentages, ratios
- **Big disadvantage of non-additive quantities:**
  - Cannot pre-compute aggregates!

Transactional vs. Snapshot facts

- **Transactional**
  - Each fact row represents a discrete event
  - Provides the most granular, detailed information
- **Snapshot**
  - Each fact row represents a point-in-time snapshot
  - Snapshots are taken at predefined time intervals
    - Examples: Hourly, daily, or weekly snapshots
  - Provides a cumulative view
  - Used for continuous processes / measures of intensity
  - Examples:
    - Account balance
    - Inventory level
    - Room temperature

Transactional vs. Snapshot example

THE UNIVERSITY OF WESTERN AUSTRALIA

Transactional

Brian	Oct. 1	CREDIT	+40
Rajeev	Oct. 1	CREDIT	+10
Brian	Oct. 3	DEBIT	-10
Rajeev	Oct. 3	CREDIT	+20
Brian	Oct. 4	DEBIT	-5
Brian	Oct. 4	CREDIT	+15
Rajeev	Oct. 4	CREDIT	+50
Brian	Oct. 5	DEBIT	-20
Rajeev	Oct. 5	DEBIT	-10
Rajeev	Oct. 5	DEBIT	-15

Snapshot

Brian	Oct. 1	40
Rajeev	Oct. 1	10
Brian	Oct. 2	40
Rajeev	Oct. 2	10
Brian	Oct. 3	30
Rajeev	Oct. 3	30
Brian	Oct. 4	40
Rajeev	Oct. 4	80
Brian	Oct. 5	40
Rajeev	Oct. 5	55

Why snapshot facts?

- Two complementary organisations
- Information content is similar
  - Snapshot view can be always derived from transactional fact
  - But not the other way around.

- Why use snapshot facts?
  - Sampling is the only option for continuous processes
    - E.g. sensor readings
  - Data compression
    - Recording all transactional activity may be too much data!
    - Stock price at each trade vs. opening/closing price
  - Query expressiveness
    - Some queries are much easier to ask/answer with snapshot fact
    - Example: Average daily balance

A difficult SQL Exercise

How to generate snapshot fact from transactional fact?

Brian	Oct. 1	CREDIT	+40		Oct. 1	40
Rajeev	Oct. 1	CREDIT	+		Oct. 1	10
Brian	Oct. 3	DEBIT		B	Oct. 2	40
Rajeev	Oct. 3	CREDIT	+20		Oct. 2	10
Brian	Oct. 4	DEBIT	-5		Oct. 3	30
Brian	Oct. 4	CREDIT	+15	ajeev	Oct. 3	30
Rajeev	Oct. 4	CREDIT	+50	Brian	Oct. 4	40
Brian	Oct. 5	DEBIT	-20	ajeev	Oct. 4	80
Rajeev	Oct. 5	DEBIT	-10	rian	Oct. 5	40
Rajeev	Oct. 5	DEBIT	-15	Rajeev	Oct. 5	55

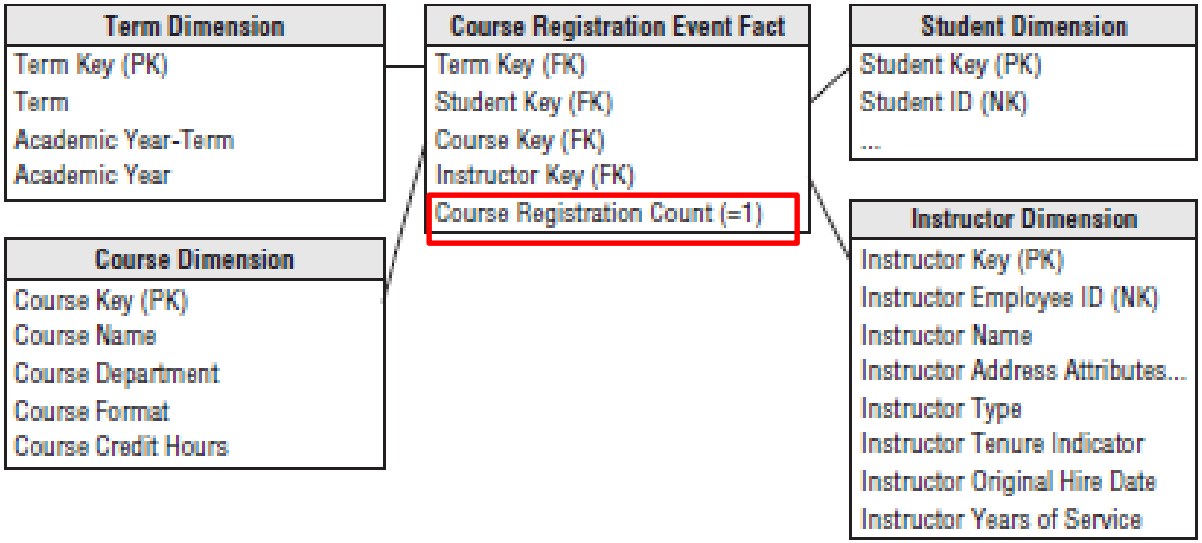
Semi-additive facts

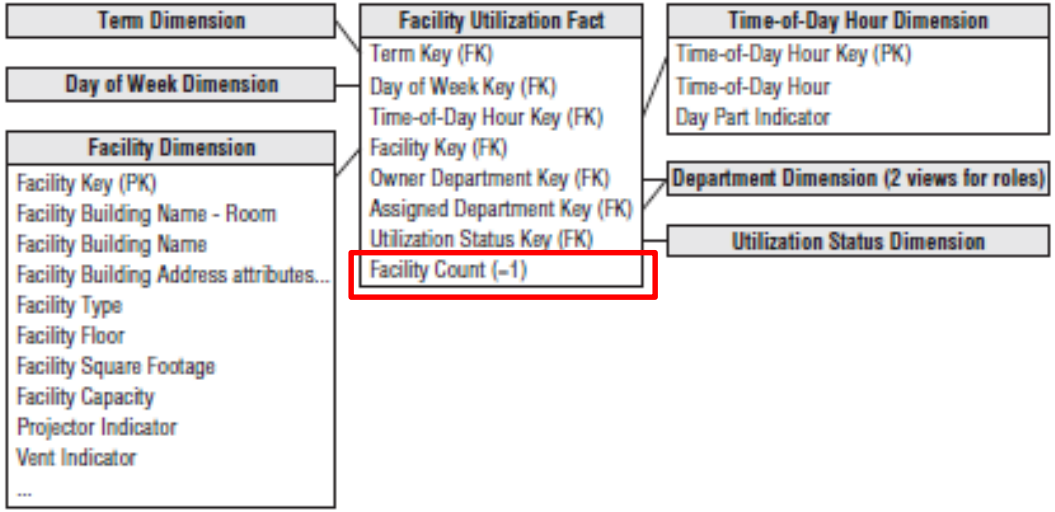
- Snapshot facts are semi-additive
- Additive across non-date dimensions
- Not additive across date dimension
- Example:
  - Total account balance on Oct 1 = OK
  - Total account balance for Brian = NOT OK
- Time averages
  - Example: Average daily balance
  - Can be computed from snapshot fact
    - First compute sum across all time periods
    - Then divide by the number of time periods
    - Can't just use the SQL AVG() operator

Factless Fact Tables

- **Transactional fact tables don't have rows for non-events**
  - Example: No rows for products that didn't sell
- **This has good and bad points.**
  - Good: Take advantage of sparsity
    - Much less data to store if events are "rare"
  - Bad: No record of non-events
    - Example: What products on promotion didn't sell?
- **"Factless" fact table**
  - A fact table without numeric fact columns
  - Used to capture relationships between dimensions
  - Include a dummy fact column that always has value 1
- **Examples:**
  - Promotion coverage fact table
    - Which products were on promotion in which stores for which days?
    - Sort of like a periodic snapshot fact
  - Student/department mapping fact table
    - What is the major field of study for each student?
    - Even for students who didn't enroll in any courses…

Education Factless Fact Tables





Lecture Outline

- **Dimension Topics**
  - How many dimensions?
  - Date/Time Dimensions
  - Surrogate Keys
- **Fact Topics**
  - Semi-additive facts
  - “Factless” fact tables
- **Slowly Changing Dimensions**
  - Overwrite history
  - Preserve history
  - Hybrid schemes
- **More dimension topics**
  - Dimension roles
  - Junk dimension
- **More fact topics**
  - Multiple currencies
  - Master/Detail facts and fact allocation
  - Accumulating Snapshot fact tables

Slowly Changing Dimensions

THE UNIVERSITY OF WESTERN AUSTRALIA

- Compared to fact tables, contents of dimension tables are relatively stable.
  - New sales transactions occur constantly.
  - New products are introduced rarely.
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally change over time
  - Customer moves to a new address
  - Grouping of stores into districts, regions changes due to corporate re-organisation
- How to handle gradual changes to dimensions?
  - Option 1: Overwrite history
  - Option 2: Preserve history

Pros and Cons

- Type 1: Overwrite existing value
  - \+ Simple to implement
- Type 2: Add a new dimension row
  - \+ Accurate historical reporting
  - \+ Pre-computed aggregates unaffected
  - \- Dimension table grows over time
- Type 2 SCD requires surrogate keys
  - Store mapping from operational key to **most current** surrogate key in data staging area
- To report on Brian's activity over time, constrain on natural key
  - WHERE name = 'Brian'

To overwrite or not?

- Both choices are commonly used
- Easy to “mix and match”
  - Preserve history for some attributes
  - Overwrite history for other attributes

- Questions to ask:
  - Will queries want to use the original attribute value or the new attribute value?
    - In most cases preserving history is desirable
  - Does the performance impact of additional dimension rows outweigh the benefit of preserving history?
    - Some fields like “customer phone number” are not very useful for reporting
    - Adding extra rows to preserve phone number history may not be worth it

Hybrid Solutions

- Suppose we want to be able to report using either old or new values
  - Mostly useful for corporate reorganisations!
  - Example: Sales districts are re-drawn annually
- Solution: Create two dimension columns

- Approach #1: “Previous District” and “Current District”
  - Allows reporting using either the old or the new scheme
  - Whenever district assignments change, all “Current District” values are moved to “Previous District”
  - “Type 3” Slowly Changing Dimension

- Approach #2: “Historical District” and “Current District”
  - Allows reports with the original scheme or the current scheme
  - When district assignment changes, do two things:
    - Create a new dimension row with “Historical District” = old value
    - Overwrite relevant dim. rows to set “Current District” = new value

Example comparison

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	9999-12-31	Current

Rows in Product dimension following department reassignment:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	2013-01-31	Expired
25984	ABC922-Z	IntelliKidz	Strategy	...	2013-02-01	9999-12-31	Current

Figure 5-6: SCD type 2 sample rows.

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education

Updated row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Strategy

Figure 5-5: SCD type 1 sample rows.

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education

Updated row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	Prior Department Name
12345	ABC922-Z	IntelliKidz	Strategy	Education

Figure 5-8: SCD type 3 sample rows.

Lecture Outline

- **Dimension Topics**
  - How many dimensions?
  - Date/Time Dimensions
  - Surrogate Keys
- **Fact Topics**
  - Semi-additive facts
  - “Factless” fact tables
- **Slowly Changing Dimensions**
  - Overwrite history
  - Preserve history
  - Hybrid schemes
- **More dimension topics**
  - Dimension roles
  - Junk dimension
- **More fact topics**
  - Multiple currencies
  - Master/Detail facts and fact allocation
  - Accumulating Snapshot fact tables

Dimension Roles

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Let's consider an online auction data mart**
- **We'll model auction results**
  - Grain: one fact row per auction.
  - Bidding history stored in a different fact
- **Dimensions:**
  - Auction Start Date
  - Auction Close Date
  - Selling User
  - Buying User
  - Product
- **"Date" and "User" occur twice**
  - Date and User dimensions play multiple roles
  - Don't create separate "Auction Start Date" and "Auction End Date" dimension tables
  - Do create a single Date dimension table
  - Do create two separate foreign keys to Date in the fact table

Role playing date dimensions

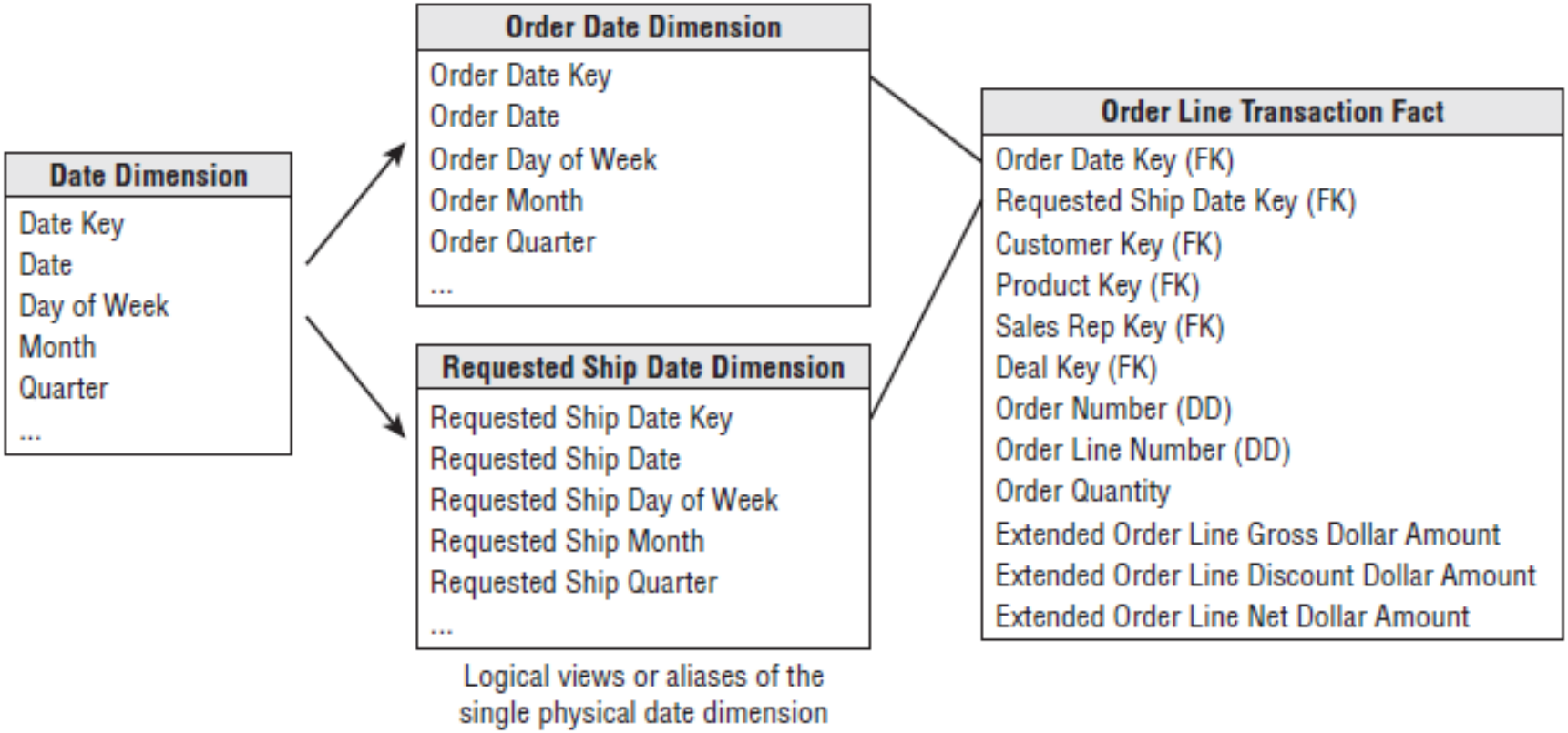


Figure 6-3: Role-playing date dimensions.

Junk Dimension

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Sometimes certain attributes don't fit nicely into any dimension**
  - Payment method (Cash vs. Credit Card vs. Check)
  - Bagging type (Paper vs. Plastic vs. None)
- **Create one or more "miscellaneous" dimensions**
  - Group together several leftover attributes as a dimension even if they aren't logically related
  - Reduces number of dimension tables, width of fact table
  - Works best if leftover attributes are
    - Few in number
    - Low in cardinality
    - Correlated
- **Some alternatives**
  - Each leftover attribute becomes its own dimension
  - Eliminate leftover attributes that are not useful

Example – Order Indicators

THE UNIVERSITY OF WESTERN AUSTRALIA

A junk dimension is a grouping of low-cardinality flags and indicators. By creating a junk dimension, you remove the flags from the fact table and place them into a useful dimensional framework.

Order Indicator Key	Payment Type Description	Payment Type Group	Order Type	Commission Credit Indicator
1	Cash	Cash	Inbound	Commissionable
2	Cash	Cash	Inbound	Non-Commissionable
3	Cash	Cash	Outbound	Commissionable
4	Cash	Cash	Outbound	Non-Commissionable
5	Visa	Credit	Inbound	Commissionable
6	Visa	Credit	Inbound	Non-Commissionable
7	Visa	Credit	Outbound	Commissionable
8	Visa	Credit	Outbound	Non-Commissionable
9	MasterCard	Credit	Inbound	Commissionable
10	MasterCard	Credit	Inbound	Non-Commissionable
11	MasterCard	Credit	Outbound	Non-Commissionable
12	MasterCard	Credit	Outbound	Commissionable

Figure 6-8: Sample rows of order indicator junk dimension.

Lecture Outline

- **Dimension Topics**
  - How many dimensions?
  - Date/Time Dimensions
  - Surrogate Keys
- **Fact Topics**
  - Semi-additive facts
  - “Factless” fact tables
- **Slowly Changing Dimensions**
  - Overwrite history
  - Preserve history
  - Hybrid schemes
- **More dimension topics**
  - Dimension roles
  - Junk dimension
- **More fact topics**
  - Multiple currencies
  - Master/Detail facts and fact allocation
  - Accumulating Snapshot fact tables

International Issues

- **International organisations often have facts denominated in different currencies**
  - Some transactions are in dollars, others in Euros, still others in yen, etc.
- **Reporting requirements may be diverse**
  - Standard currency vs. local currency
  - Historical exchange rate vs. current exchange rate
- **Time zones cause a similar problem**
  - Sometimes local time is most meaningful
    - E.g. buying patterns are different in morning vs. afternoon
  - Sometimes standardised time (e.g. GMT) is better
    - Correctly express relative order of events

Handling Multiple Currencies

- **Add a Currency dimension to the fact table**
  - Values are US Dollars, Yen, Euros, etc.
- **Each currency-denominated fact gets 2 fact columns**
  - One column uses the local currency of the transaction
  - The other column stores the equivalent value in standard currency
  - Currency dimension is used to indicate the units being used in the local currency column
  - Historical exchange rate in effect the day of the transaction is used for the conversion
- **Create a special currency conversion table**
  - Store current conversion factor between each pair of currencies
  - Used to generate reports in any currency of interest

Multi-currency example

THE UNIVERSITY OF WESTERN AUSTRALIA

Product	Date	Currency	AmtLocal	AmtUSD
443	87	1	400	400
1287	87	4	1250	1447
34	88	2	3500	380

Key	Name	Abrv	Country
1	US Dollar	USD	USA
2	Japanese Yen	JPY	Japan
3	Pound Sterling	GBP	UK
4	Euro	EUR	Europe

From	To	Factor
1	2	111.3
1	3	.562
1	4	.814
2	1	.0089
…	…	…

Another multi-currency example

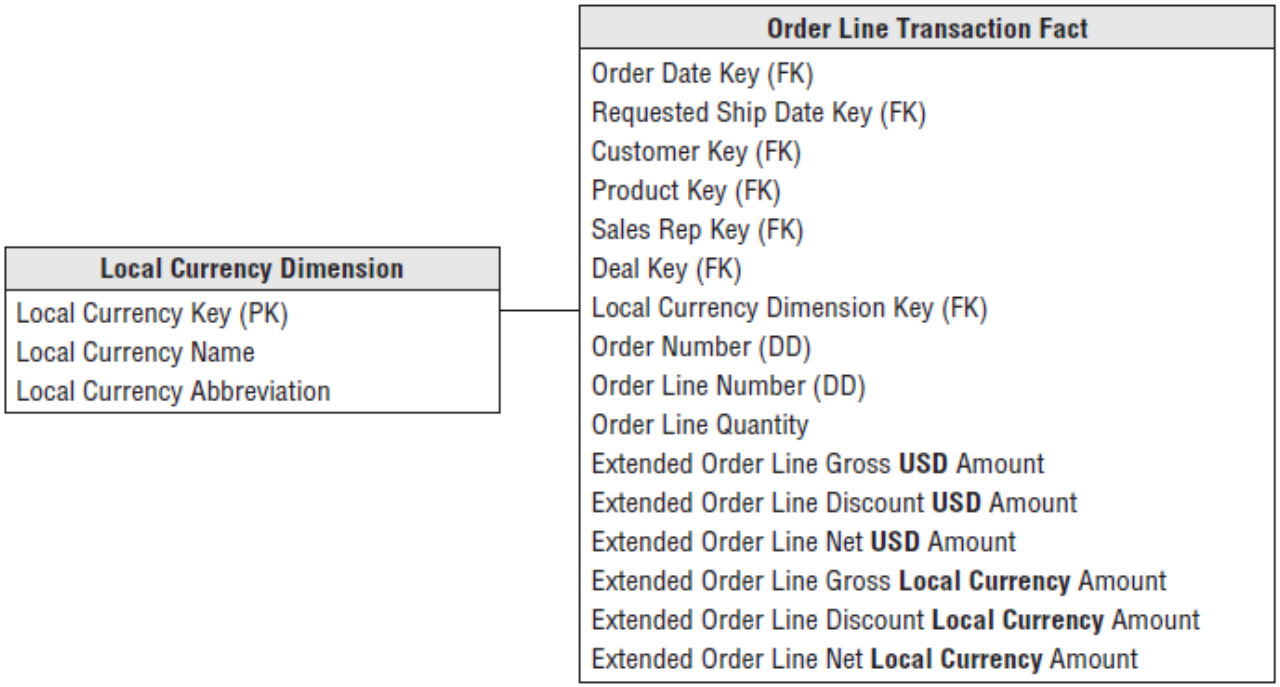


Figure 6-10: Metrics in multiple currencies within the fact table.

Currency Conversion Fact
Conversion Date Key (FK)
Source Currency Key (FK)
Destination Currency Key (FK)
Source-Destination Exchange Rate
Destination-Source Exchange Rate

Reference

- **Stanford Lecture Notes for CS345**
  - http://web.stanford.edu/class/cs345/
- **Ralph Kimball, Margy Ross. The Data Warehouse Toolkit (Third Edition)**
  - Chapter 3, 5, and 6.

Copyright Notice

Copyright Notice

Material used in this recording may have been reproduced and communicated to you by or on behalf of **The University of Western Australia** in accordance with section 113P of the *Copyright Act 1968*.

Unless stated otherwise, all teaching and learning materials provided to you by the University are protected under the Copyright Act and is for your personal use only. This material must not be shared or distributed without the permission of the University and the copyright owner/s.