ISTM 6212 - Week 7
Conformed Dimension Design

Daniel Chudnov, dchud@gwu.edu

2016-11-01

Agenda

- Schedule check
- Exercise 04 follow up
- A little more UNIX background
- Conformed Dimensions
- Issues in Dimension Design
- Project 02

Schedule check

Exercise 04 follow up

Good results

- You're getting the hang of SQL
- Many creative approaches
- Use of referential constraints, PL/pgSQL - great!
- Let's not eat at Dion's Pizza

Tips & tricks

- "LIKE '%28%'" or "LIKE '%COMPLIANCE%'"
- shuf -n 17556 inspections.csv | csvstat
- use DISTINCT and GROUP BY rather than head/tail to determine ranges, missing values, etc.
- use ORDER BY for consistency

Tips & tricks (2)

- CHAR(6) vs. VARCHAR(6)
- VARCHAR(6) vs. VARCHAR(255)
- plot time left-to-right
- show "_desc" when counting codes
- don't cut-n-paste metadata or queries :)

- w/txt msgs b casual s'ok
- When you write prose sentences, you should always, always, always use proper capitalization and punctuation, period.

Examples of good techniques

- Wei's seaborn charts:
  - https://github.com/zhengweifz/istm-6212/blob/master/exercise4_Wei.ipynb
- Kamran's use of referential constraints
  - https://github.com/kamran1310/istm-6212/blob/master/exercise-04.ipynb

A little more UNIX background

Servers, processes, ports, users

- database engines like PostgreSQL and MySQL are servers
- connect to servers with clients
- clients use software libraries to talk w/servers
- e.g.: Psycopg is a client software library for PostgreSQL

Clients and databases

- SQLite is just a library; no separate server, just a file
- to connect to a SQLite db:
  - %sql sqlite:///boating.db (in Jupyter)
  - % sqlite3 boating.db (in terminal shell)
  - import sqlite3; conn = sqlite3.connect('boating.db')

Servers have hosts, ports, users

- To connect to a PostgreSQL db:
  - %sql postgresql://dbuser:dbpass@localhost:5432/boating
  - % psql -U dbuser -p 5432 -h localhost -W boating
  - import psycopg2
  - conn = psycopg2.connect("dbname='boating' user='dbuser' host='localhost'")

Common ports

- HTTP 80
- HTTPS 443
- SSH 22
- Email (SMTP) 25/465, (POP) 110/995, (IMAP) 143/993
- MySQL 3306
- PgSQL 5432

datanotebook.org

- an AWS auto-scaling group of AMIs hosting temporary docker containers serving Jupyter notebooks proxied to port 80 (HTTP) that scales up with active users

- a VM that runs on one or more servers hosting Jupyter notebooks depending on many of you are using it

- Restarting PostgreSQL is *not* normal. You have to do it at the top of your notebooks only because of the weird way I set up Jupyter within Docker containers.
- I'm sorry about that.
- Don't expect to be restarting servers all the time! (Or at least I hope you don't have to.)

Conformed Dimensions

first, a quick review

Facts and dimensions

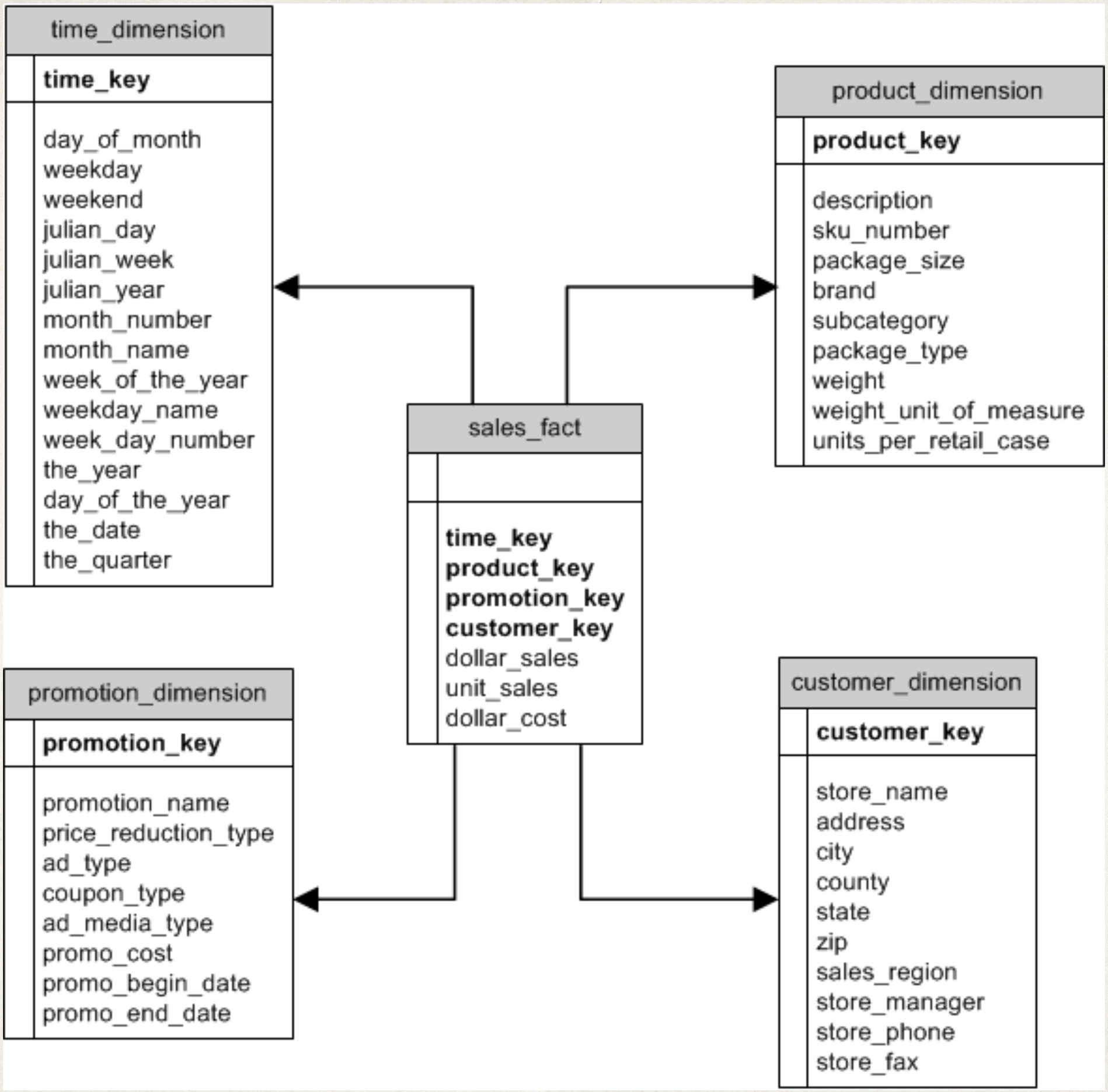
- **Facts** are instances of business processes worthy of measurement

- **Dimensions** are the contexts in which those processes occurred and through which their measurement may be framed

Facts are sparse; dimensions wide

- Facts represent individual events; no records for "all possible events", only what actually happened

- Dimensions represent possible contexts; records for many possible combinations of filter / aggregation attributions


time_dimension
time_key
day_of_month
weekday
weekend
julian_day
julian_week
julian_year
month_number
month_name
week_of_the_year
weekday_name
week_day_number
the_year
day_of_the_year
the_date
the_quarter
product_dimension
product_key
description
sku_number
package_size
brand
subcategory
package_type
weight
weight_unit_of_measure
units_per_retail_case
sales_fact
time_key
product_key
promotion_key
customer_key
dollar_sales
unit_sales
dollar_cost
promotion_dimension
promotion_key
promotion_name
price_reduction_type
ad_type
coupon_type
ad_media_type
promo_cost
promo_begin_date
promo_end_date
customer_dimension
customer_key
store_name
address
city
county
state
zip
sales_region
store_manager
store_phone
store_fax

Functions of dimensions

- filter queries or reports
- control scope of fact aggregation
- order and sort information
- provide context to facts on reports
- define hierarchy, group, subtotal, and summary

Functions of facts

- hold measurable data about processes/events
- enable aggregation ("additivity")
- define the **grain**, its level of detail
- hold as low a level of grain as possible
- allow query by context (dimensions)

Separating facts and processes

- Key questions:
  - Do two facts/processes occur simultaneously?
  - Are both available at the same grain?
- If "no" to either, you have more than one fact

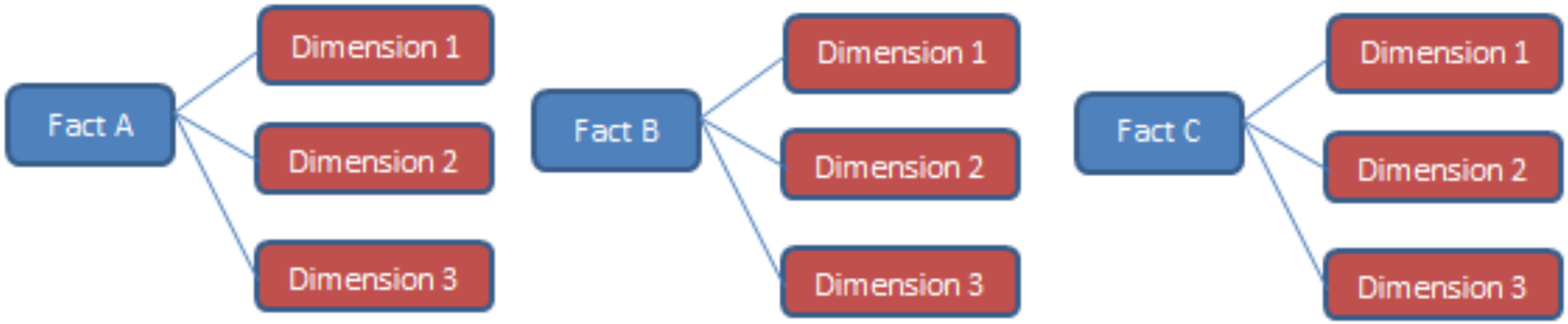
Types of keys

- **Natural keys** - attributes that were likely primary / foreign keys in source data but do not necessarily work as such in dimensional designs

- **Surrogate keys** - primary keys generated for analytical dimension tables, foreign keys on analytical fact tables; no meaning w/r/t source systems

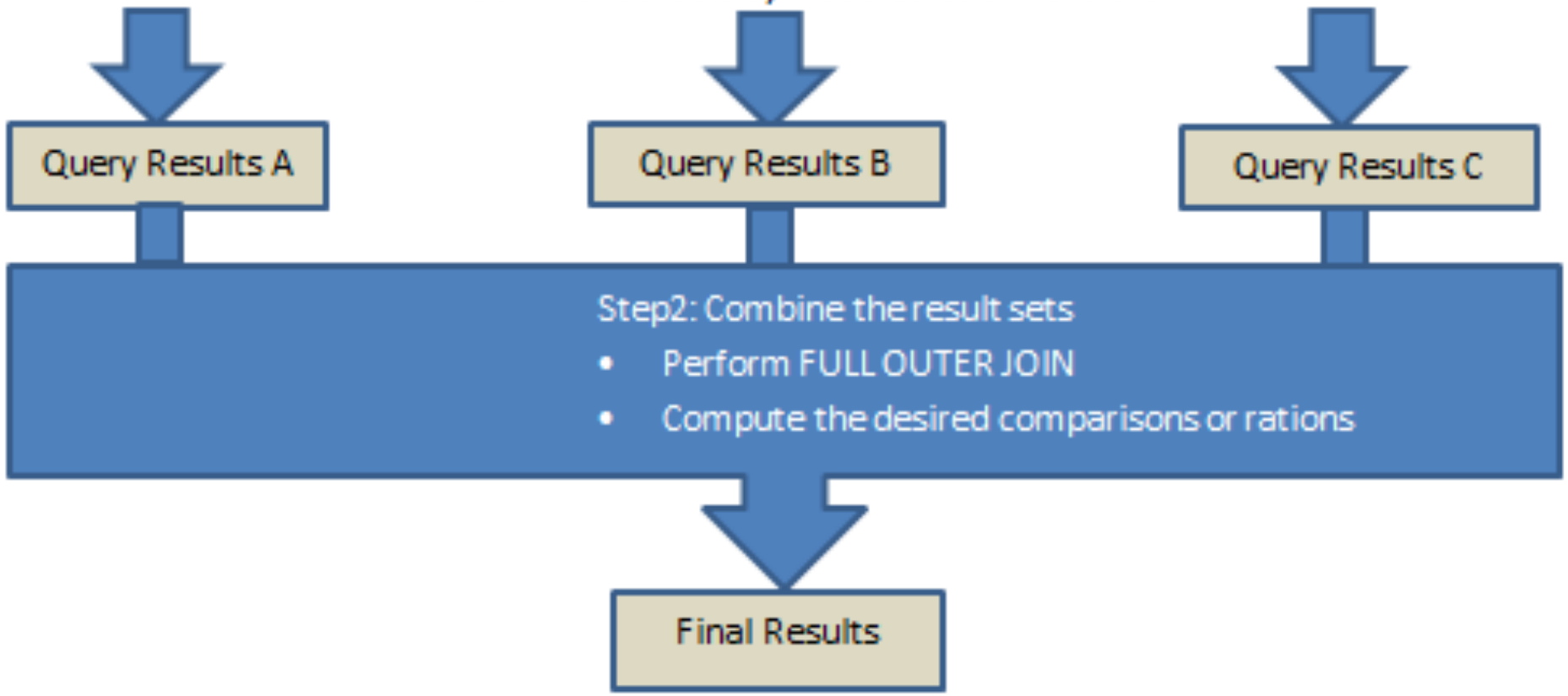
Querying multiple facts

- **don't** join fact tables: remember Cartesian product!
- **do** "drill across":
  - summarize each fact into common dimensions
  - join based on common dimensions
  - add computations/comparisons as needed

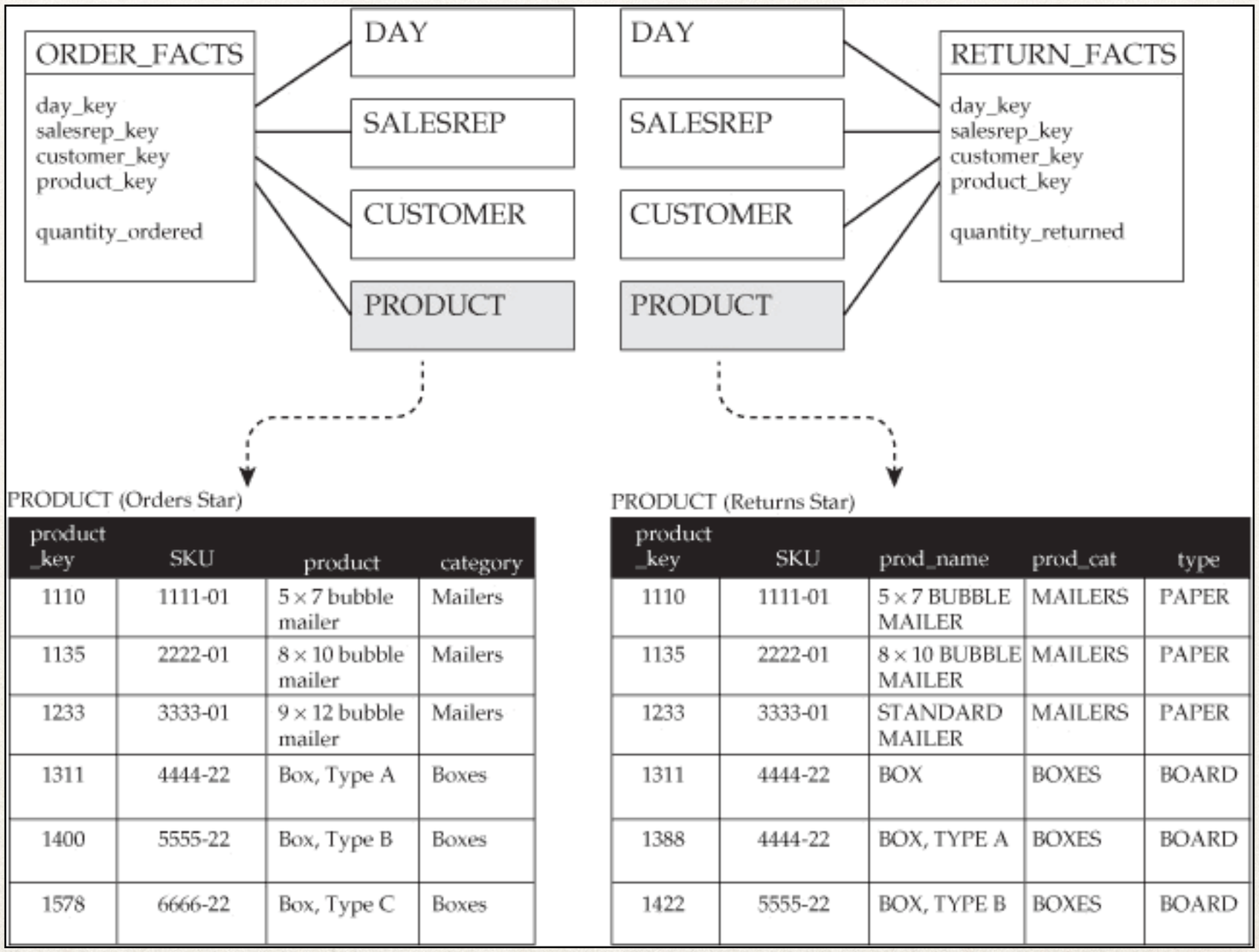


Step 1: Issue a separate query for each fact table

- Create each query as needed.
- Analyze the facts on the same dimensions
- Summarize facts by the selected dimensions



What causes failure?



PRODUCT (Orders Star)

product_key	SKU	product	category
1110	1111-01	5 × 7 bubble mailer	Mailers
1135	2222-01	8 × 10 bubble mailer	Mailers
1233	3333-01	9 × 12 bubble mailer	Mailers
1311	4444-22	Box, Type A	Boxes
1400	5555-22	Box, Type B	Boxes
1578	6666-22	Box, Type C	Boxes

PRODUCT (Returns Star)

product_key	SKU	prod_name	prod_cat	type
1110	1111-01	5 × 7 BUBBLE MAILER	MAILERS	PAPER
1135	2222-01	8 × 10 BUBBLE MAILER	MAILERS	PAPER
1233	3333-01	STANDARD MAILER	MAILERS	PAPER
1311	4444-22	BOX	BOXES	BOARD
1388	4444-22	BOX, TYPE A	BOXES	BOARD
1422	5555-22	BOX, TYPE B	BOXES	BOARD

Fig. 5-2

Consistency in dimensions

- Same structure: same attributes, names, types
- Same content: same values, casing, abbreviations
- Queries can account for differences, but early planning and proper ETL can make drilling across easier
- Attributes must match even if tables don't

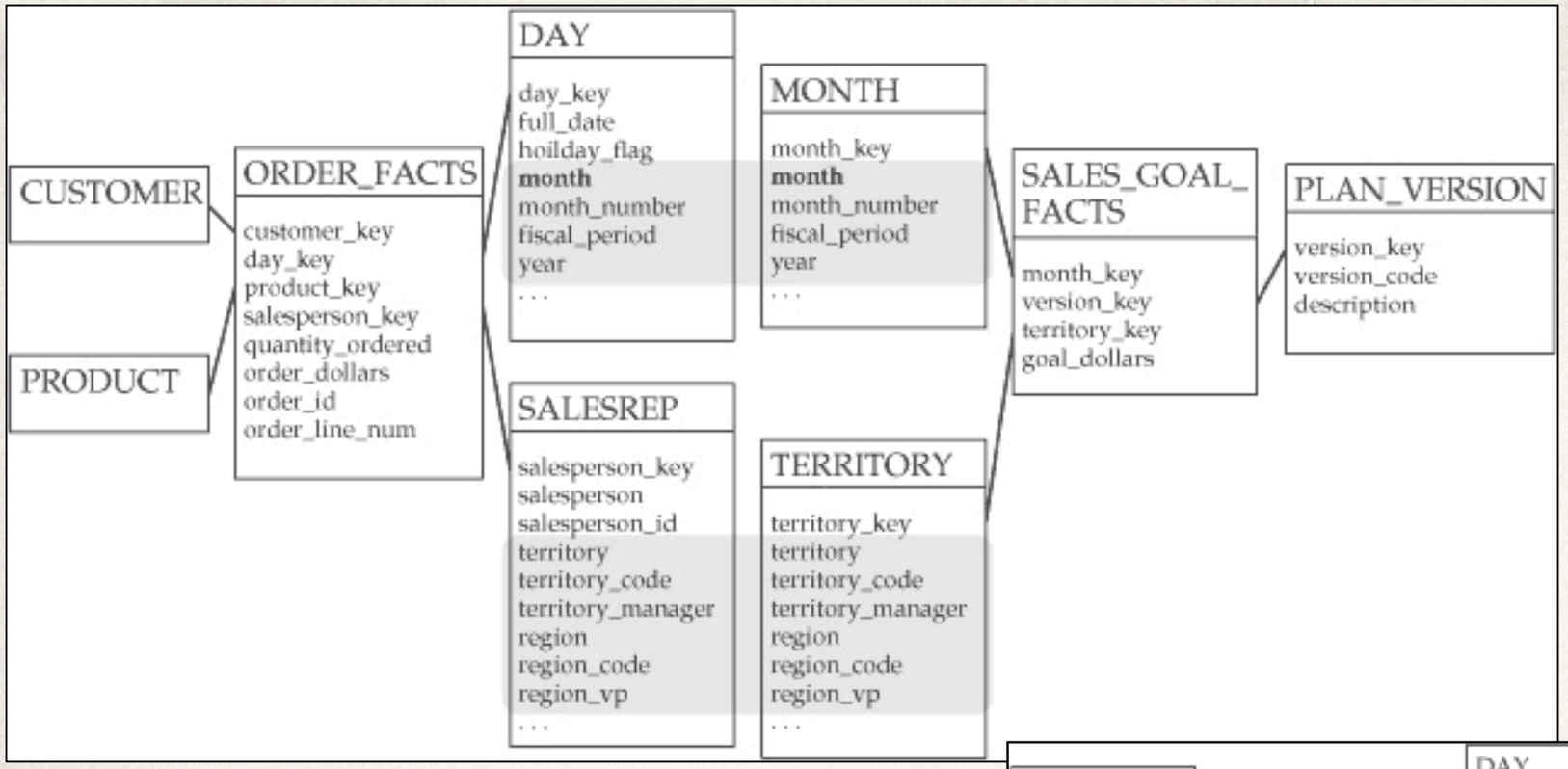

CUSTOMER
PRODUCT
ORDER_FACTS
customer_key
day_key
product_key
salesperson_key
quantity_ordered
order_dollars
order_id
order_line_num
DAY
day_key
full_date
hoilday_flag
month
month_number
fiscal_period
year
. . .
MONTH
month_key
month
month_number
fiscal_period
year
. . .
SALESREP
salesperson_key
salesperson
salesperson_id
territory
territory_code
territory_manager
region
region_code
region_vp
. . .
TERRITORY
territory_key
territory
territory_code
territory_manager
region
region_code
region_vp
. . .
SALES_GOAL_FACTS
month_key
version_key
territory_key
goal_dollars
PLAN_VERSION
version_key
version_code
description

Fig. 5-3

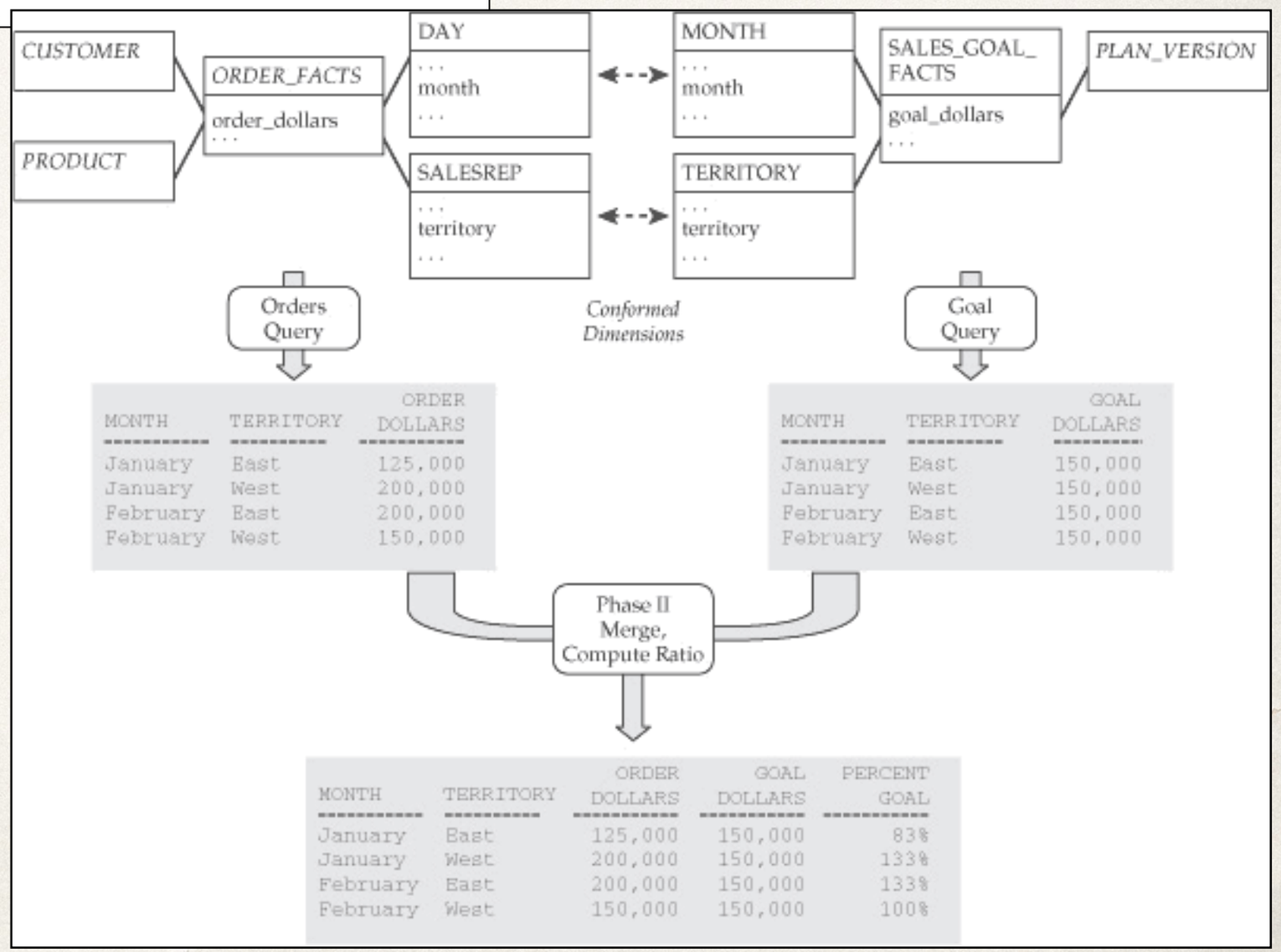

CUSTOMER
PRODUCT
ORDER_FACTS
order_dollars
. . .
DAY
. . .
month
. . .
MONTH
. . .
month
. . .
SALESREP
. . .
territory
. . .
TERRITORY
. . .
territory
. . .
SALES_GOAL_FACTS
goal_dollars
. . .
PLAN_VERSION
Conformed Dimensions
Orders Query
Goal Query
MONTH TERRITORY ORDER DOLLARS
January East 125,000
January West 200,000
February East 200,000
February West 150,000
MONTH TERRITORY GOAL DOLLARS
January East 150,000
January West 150,000
February East 150,000
February West 150,000
Phase II Merge, Compute Ratio
MONTH TERRITORY ORDER DOLLARS GOAL DOLLARS PERCENT GOAL
January East 125,000 150,000 83%
January West 200,000 150,000 133%
February East 200,000 150,000 133%
February West 150,000 150,000 100%

Fig. 5-4

Types of conforming dimensions

- Shared dimensions (using same dimension tables)
- Conformed rollups (one is aggregate of other)
- Overlapping attributes (like previous slide)
- Degenerate dimensions (e.g. order lines)

Planning conformance

- Substantial up-front planning required
- Extensive enterprise-wide focus now, or cleanup cost later?

	day			product		salesrep					
	day	*month*	*quarter*	*product*	*category*	*salesrep*	*territory*	*region*	*customer*	*warehouse*	*order_line*
order_facts	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
shipment_facts	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
return_facts	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
inventory_facts	✓	✓	✓	✓	✓					✓	
receivables_facts	✓	✓	✓						✓		✓
sales_goal_facts		✓	✓				✓	✓			
demand_forecast_facts			✓		✓			✓			

Fig. 5-7

Beyond databases

- this problem applies to all data everywhere, not just data warehouses

- everywhere with more than one system, with more than one source, with more than one table

- warehouses evolved to focus on solving this integration problem to enable analysis

CSV, Data Lakes, APIs, Everything

- one Bikeshare data file, one weather history file: non-conforming dimensions
- one voting district, one census tract/block group: non-conforming dimensions
- one owner, multiple facilities: non-conforming dimensions
- data from Twitter, Facebook, Instagram accounts: non-conforming dimensions
- ...this will keep happening...

Issues in Dimensional Design

Affinity in dimensions

- salesperson + territory vs. salesperson + customer
- product + brand + model
- team + player
- is affinity natural, or process/event-based?
- is affinity within one context or several?

Check for "browsability"

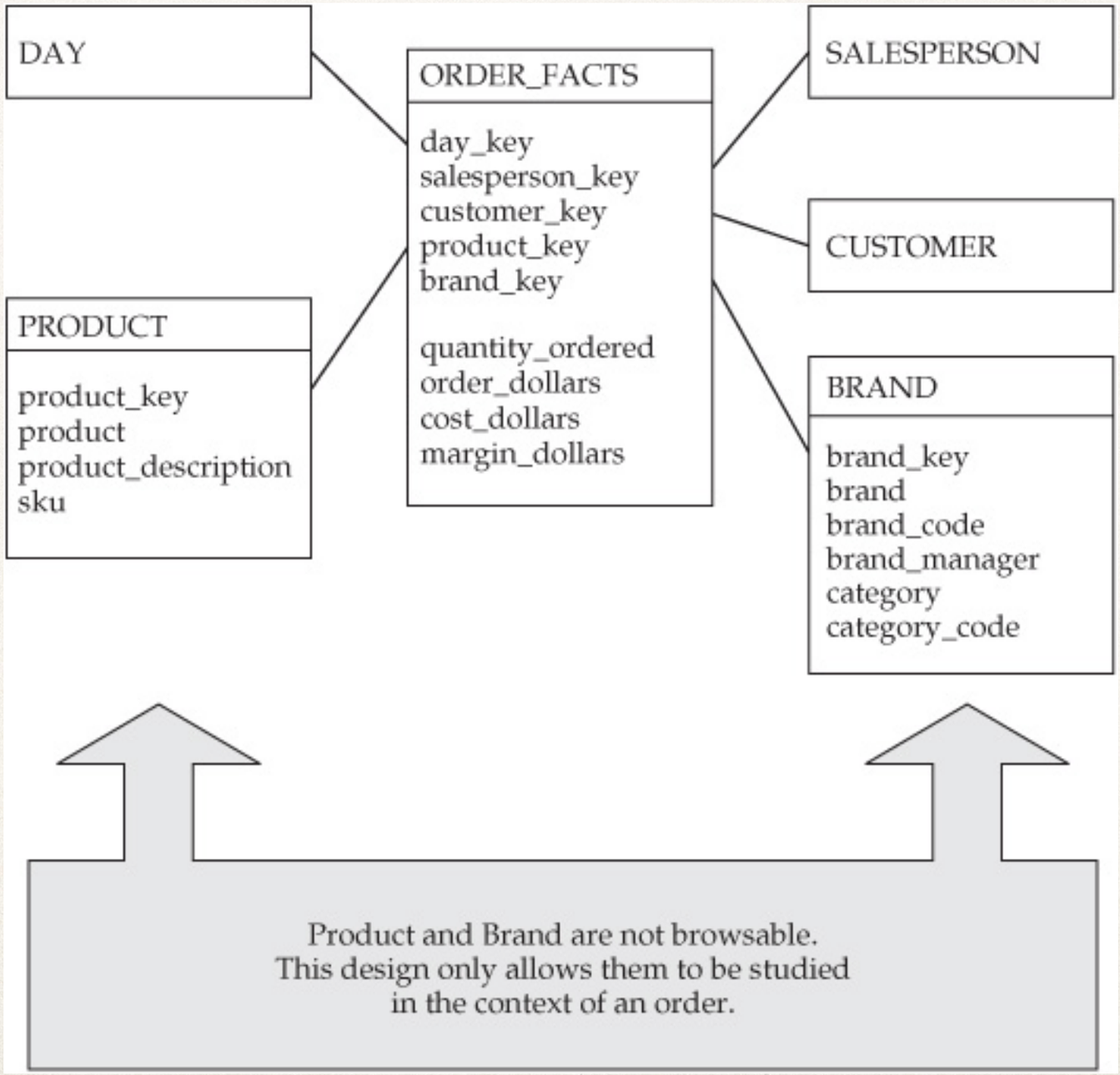


Fig. 6-2

Large (wide) dimensions

- arbitrary splits work but complicate queries
- "L" part of ETL process staging issues
- BI tool integration suffers

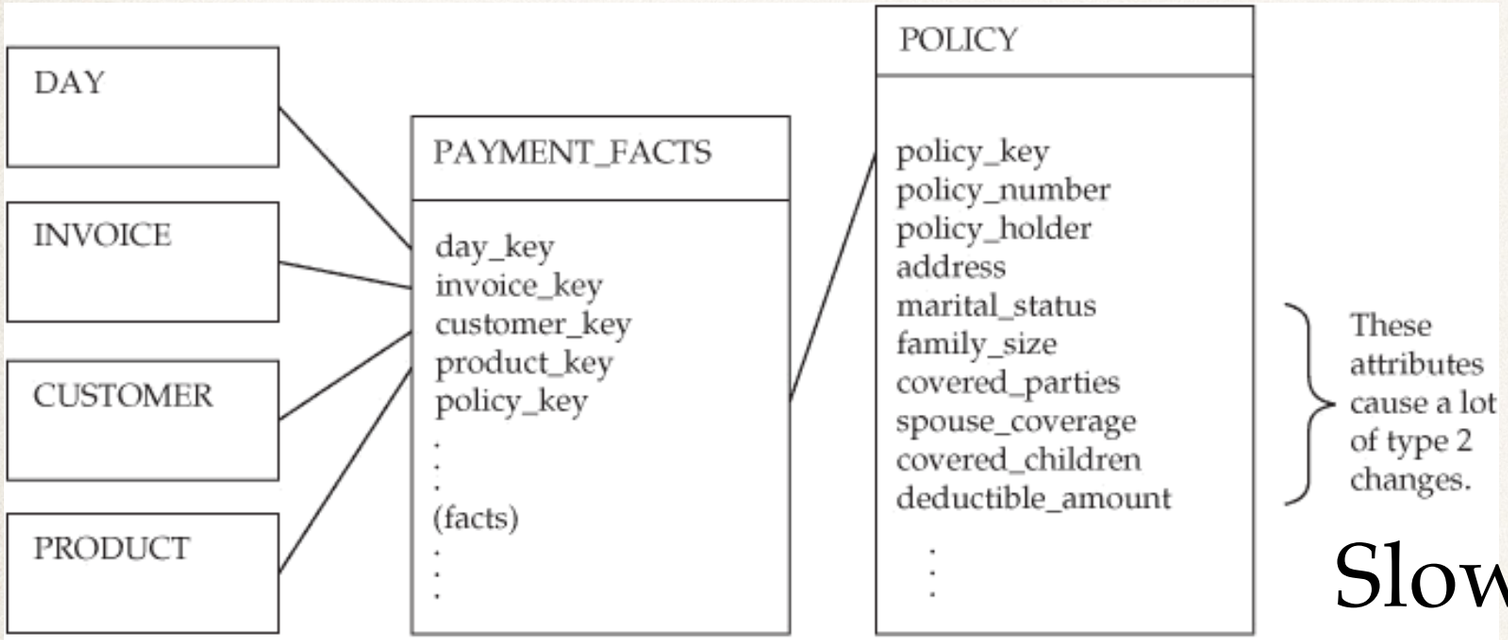
Addressing large dimensions

- is it really more than one dimension? split them up.

- extract subtype specifics to new dimensions (e.g. related product types)

- mini-dimensions: split out possibilities

Mini-dimensions - simple, elegant

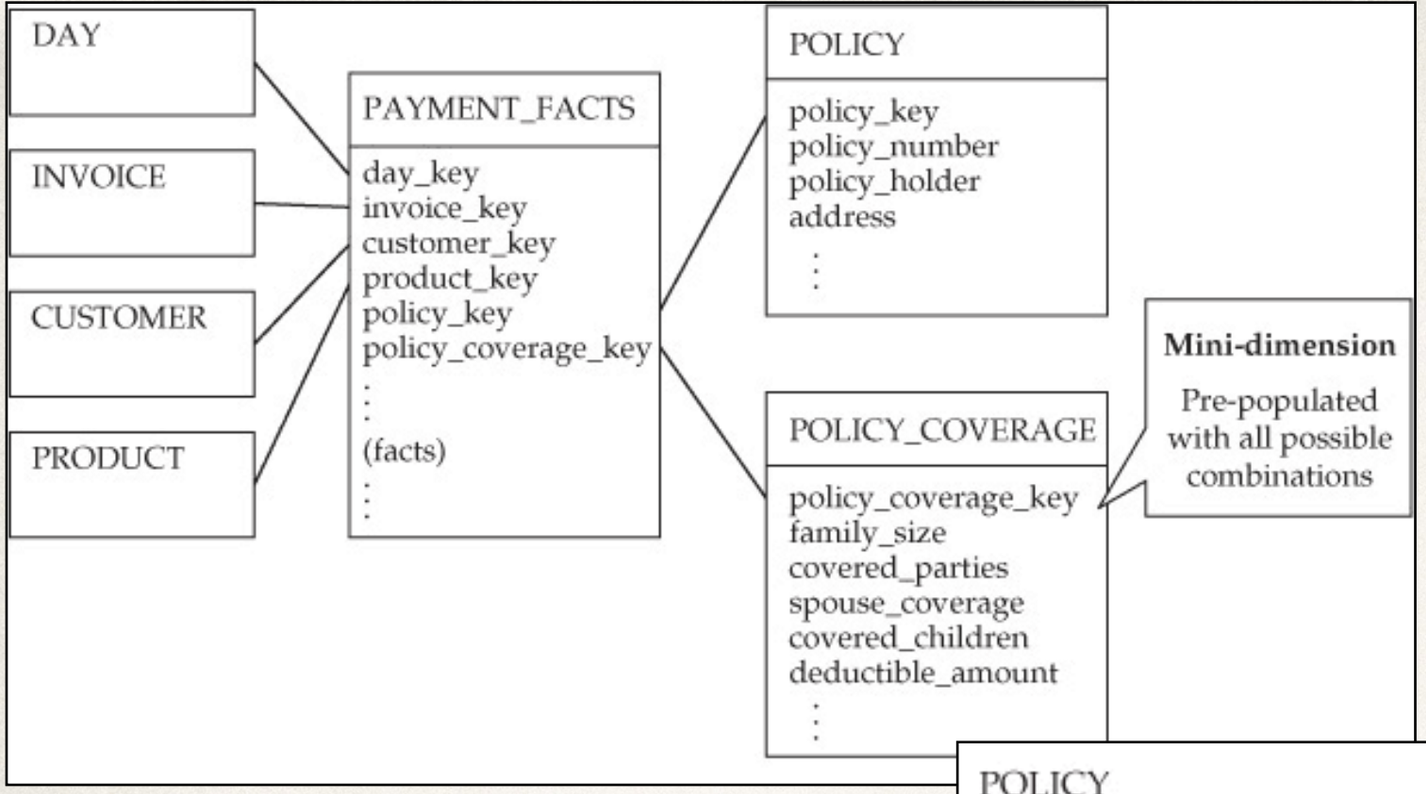


Slow but frequent-changing dimensions: lots of rows

POLICY

policy_ key	policy_ number	policy_ holder	address	marital_ status	family_ size	covered_ parties	covered_ children	deductible _amount
12882	40111	Smith, Hal	113 Random Rd.	Single	1	1	0	250
12911	40111	Smith, Hal	113 Random Rd.	Married	2	1	0	250
13400	40111	Smith, Hal	113 Random Rd.	Married	2	2	0	250
14779	40111	Smith, Hal	113 Random Rd.	Married	3	3	1	250
14922	40111	Smith, Hal	113 Random Rd.	Married	3	3	1	500
18911	40111	Smith, Hal	113 Random Rd.	Married	2	2	0	500

Fig. 6-4



New dimension table captures potential variety

Original dimension now Type 1

POLICY

policy_key	policy_number	policy_holder	address
12882	40111	Smith, Hal	113 Random Rd.

Coverage changes don't generate new dimension rows...

POLICY_COVERAGE

policy coverage_key	marital_status	family_size	covered_parties	covered_children
1	Single	1	1	0
2	Married	2	1	0
3	Married	2	2	0
4	Married	3	1	0
5	Married	3	2	0
6	Married	3	2	1
7	Married	3	3	1
8	Married	4	1	0

... and they don't generate new mini-dimension rows

Fig. 6-5

SQL ALIAS for roles

SQL Query

```
SELECT
   officer.employee_name AS officer_name,
   processor.employee_name AS processor_name,
   underwriter.employee_name AS underwriter_name
FROM
   -- Alias the employee table 3 times:
   --
   employee ALIAS officer,
   employee ALIAS processor,
   employee ALIAS underwriter,
   --
   --
   mortgage_closing_facts
WHERE
   --
   -- join to each alias using correct key:
   --
   mortgage_closing_facts.employee_key_officer = officer.employee_key AND
   mortgage_closing_facts.employee_key_processor = processor.employee_key AND
   mortgage_closing_facts.employee_key_underwriter = underwriter.employee_key AND
   --
   --
   mortgage_closing_facts.application_key = 77777
```

Query Results

```
officer_name      processor_name    underwriter_name
===============   ===============   ================
Eve Adelson       Dan Roberts       Chiu Mieng
```

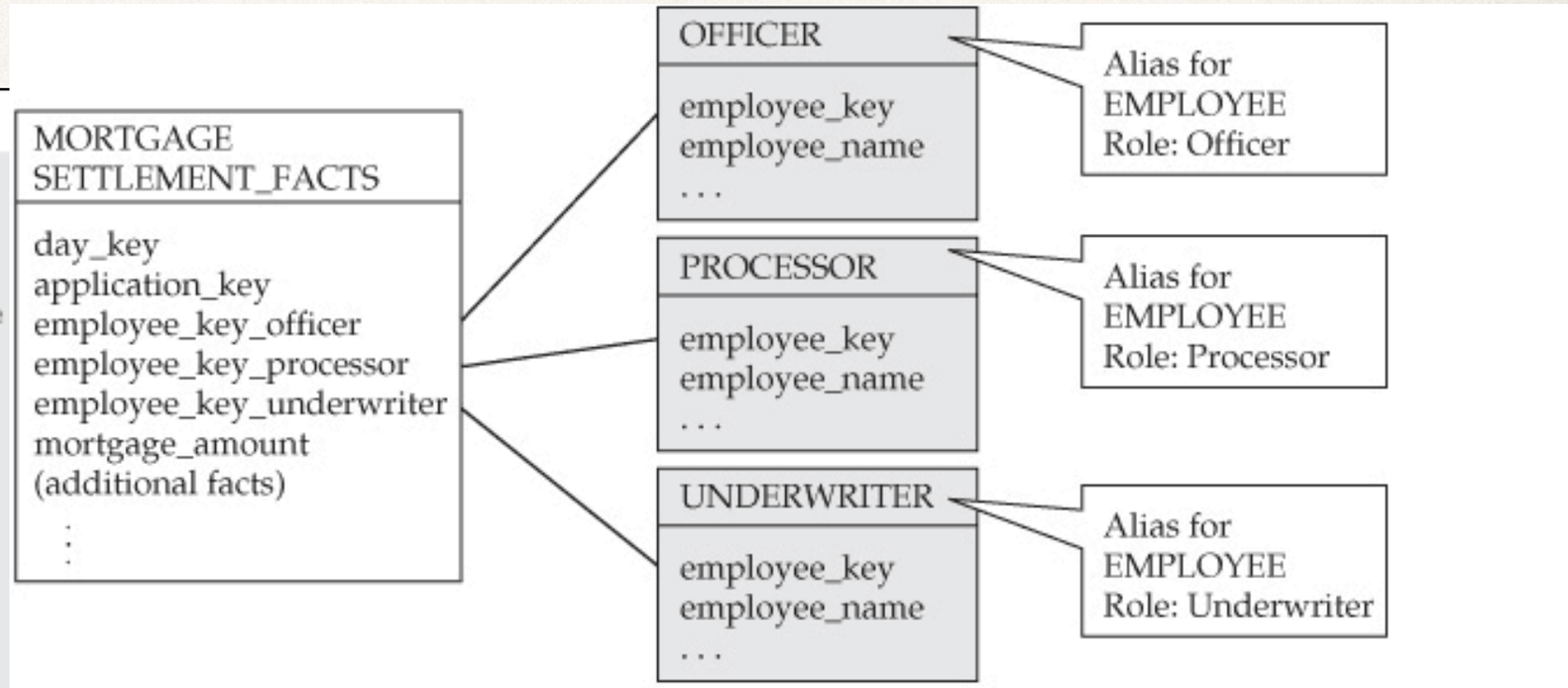


Fig. 6-8

Avoid NULL in dimensions

- Special-case roles for missing or bad data

Used when a fact is supplied with an invalid product_code

Used when a fact arrives prior to dimensional context

PRODUCT

product_key	row_type	product_code	product_name	
0	Invalid	n/a	n/a	
1	Unknown	n/a	n/a	
101	Product	B57330-1	Cardboard Box	
102	Product	B47770-2	Bubble Envelope	

Fig. 6-11

Behavioral dimensions

- Read this closely!
- Use facts to create new dimensions:
  - categorize customers by sales level / frequency
  - categorize products by popularity / seasonality
- A prelude to **feature engineering** in data mining

Facts -> dimensions

- using past behavior as context for studying current (or future) behavior
- add attributes to dimension w/natural affinity
- don't forget to consider how to handle change/update

Project 02 - work in pairs!