ISTM 6212 - Week 7
Warehouses & Dimensional Design

Daniel Chudnov, dchud@gwu.edu

2016-10-18

Agenda

- Schedule check
- Exercise 03 & Project Reviews 01 follow up
- Analytic / Dimensional database designs
- Dimension and Fact tables
- More basic ETL in SQL
- Exercise 04

Schedule check

- work in pairs on upcoming exercises?
- work in pairs on upcoming project?

Exercise 03 wrap-up

Good results

- My apologies for the lousy tests!
- Most of you worked it out correctly
- Nice use of JOIN/UNION for #10
- Creative SQL formatting - okay!

Tips & tricks

- echo "   Chart Title" # chart plot title workaround
- no need to connect to db again w/python
- usually no need to import pandas or matplotlib (%matplotlib inline & get data frame from SQL result)
- import matplotlib for full plot control (title, color, etc.)

Final thought

- *all* the data was about Connecticut registrations!

Project reviews 01 wrap-up

Nice work!

- Most of you nailed it
- Very positive tone - a good default
- Nice use of GitHub issues - response, closing ticket
- Markdown formatting within issue text
- Repeat on upcoming project

Analytic / Dimensional database designs

Why analytical processing?

- to **measure business processes**
  - operational/transactional systems support **execution**
  - analytical systems support **evaluation**
- improved **analyst UX** over transactional system

OLTP vs OLAP (Fig 1-1)

process execution	process measurement
CRUD operations	query operations
individual transactions	aggregated transactions
current focus	current+historical focus
ER / 3NF design	dimensional design

Facts and dimensions

- "How were last year's sales by quarter in each territory?

- "Which products were most popular for each demographic group during the last three holiday sales seasons?"

- "How does delivery performance vary by warehouse, driver, and by day of the week?"

Facts and *dimensions*

- "How were *last year's* **sales** *by quarter* in each *territory*?
- "Which *products* were **returned** the most by each *demographic group* during the *last three holiday sales seasons*?"
- "How does **delivery performance** vary *by warehouse, driver,* and *by day of the week*?"

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 and dimensions (Fig 1-3)

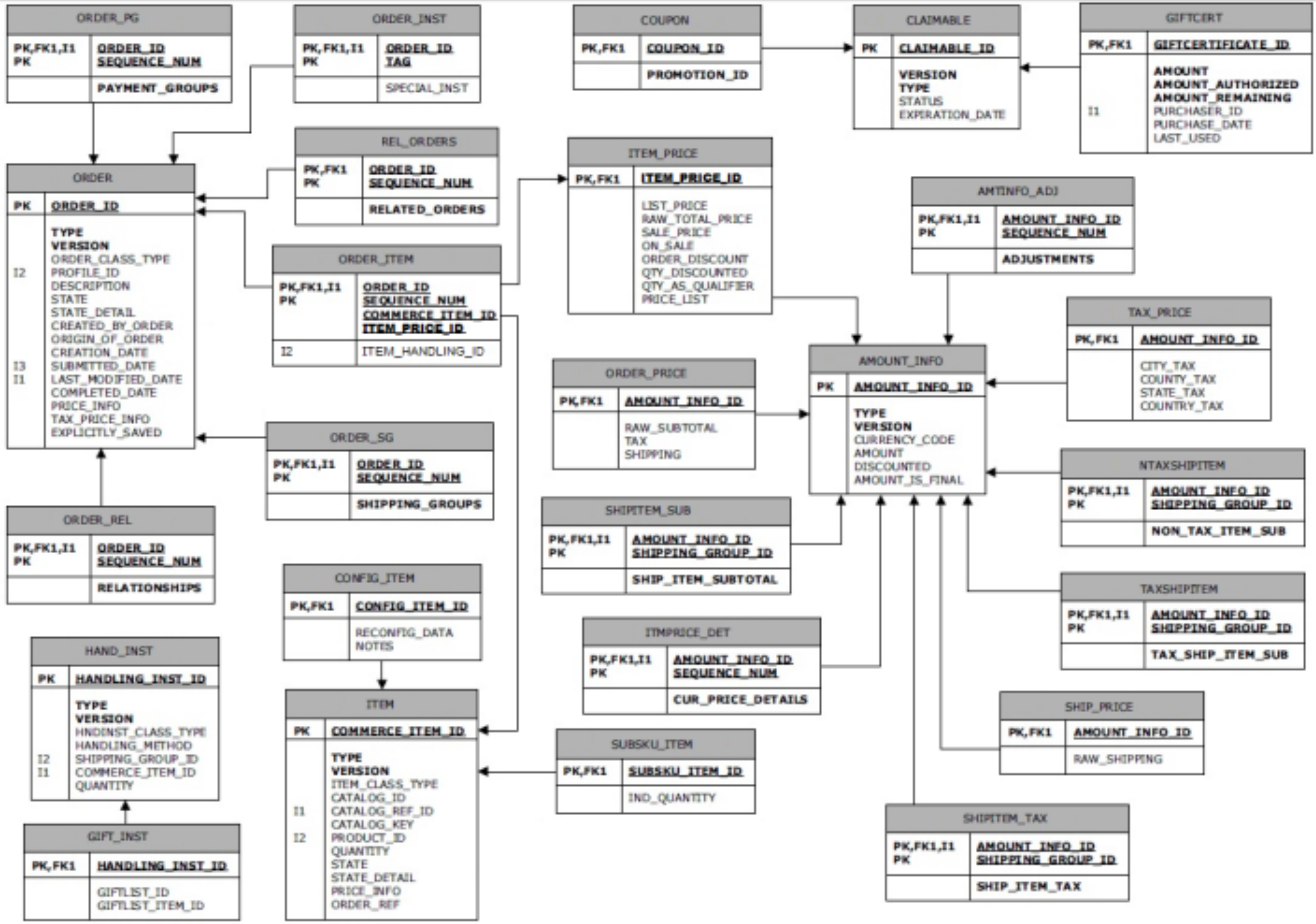
- order dollars, cost dollars, quantity ordered

- product, product description, SKU, brand code, brand, category code, category, order date, order month, order quarter, salesperson id, salesperson, territory, territory code, region, region code, customer, etc.

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



ORDER_PG

PK,FK1,I1 PK	ORDER_ID SEQUENCE_NUM
	PAYMENT_GROUPS

ORDER_INST

PK,FK1,I1 PK	ORDER_ID TAG
	SPECIAL_INST

COUPON

PK,FK1	COUPON_ID
	PROMOTION_ID

CLAIMABLE

PK	CLAIMABLE_ID
	VERSION TYPE STATUS EXPIRATION_DATE

GIFTCERT

PK,FK1	GIFTCERTIFICATE_ID
   I1	AMOUNT AMOUNT_AUTHORIZED AMOUNT_REMAINING PURCHASER_ID PURCHASE_DATE LAST_USED

REL_ORDERS

PK,FK1 PK	ORDER_ID SEQUENCE_NUM
	RELATED_ORDERS

ITEM_PRICE

PK,FK1	ITEM_PRICE_ID
	LIST_PRICE RAW_TOTAL_PRICE SALE_PRICE ON_SALE ORDER_DISCOUNT QTY_DISCOUNTED QTY_AS_QUALIFIER PRICE_LIST

ORDER

PK	ORDER_ID
   I2 I3 I1	TYPE VERSION ORDER_CLASS_TYPE PROFILE_ID DESCRIPTION STATE STATE_DETAIL CREATED_BY_ORDER ORIGIN_OF_ORDER CREATION_DATE SUBMITTED_DATE LAST_MODIFIED_DATE COMPLETED_DATE PRICE_INFO TAX_PRICE_INFO EXPLICITLY_SAVED

AMTINFO_ADJ

PK,FK1,I1 PK	AMOUNT_INFO_ID SEQUENCE_NUM
	ADJUSTMENTS

ORDER_ITEM

PK,FK1,I1 PK	ORDER_ID SEQUENCE_NUM COMMERCE_ITEM_ID ITEM_PRICE_ID
I2	ITEM_HANDLING_ID

TAX_PRICE

PK,FK1	AMOUNT_INFO_ID
	CITY_TAX COUNTY_TAX STATE_TAX COUNTRY_TAX

ORDER_PRICE

PK,FK1	AMOUNT_INFO_ID
	RAW_SUBTOTAL TAX SHIPPING

AMOUNT_INFO

PK	AMOUNT_INFO_ID
	TYPE VERSION CURRENCY_CODE AMOUNT DISCOUNTED AMOUNT_IS_FINAL

ORDER_SG

PK,FK1,I1 PK	ORDER_ID SEQUENCE_NUM
	SHIPPING_GROUPS

NTAXSHIPITEM

PK,FK1,I1 PK	AMOUNT_INFO_ID SHIPPING_GROUP_ID
	NON_TAX_ITEM_SUB

SHIPITEM_SUB

PK,FK1,I1 PK	AMOUNT_INFO_ID SHIPPING_GROUP_ID
	SHIP_ITEM_SUBTOTAL

ORDER_REL

PK,FK1,I1 PK	ORDER_ID SEQUENCE_NUM
	RELATIONSHIPS

CONFIG_ITEM

PK,FK1	CONFIG_ITEM_ID
	RECONFIG_DATA NOTES

TAXSHIPITEM

PK,FK1,I1 PK	AMOUNT_INFO_ID SHIPPING_GROUP_ID
	TAX_SHIP_ITEM_SUB

ITMPRICE_DET

PK,FK1,I1 PK	AMOUNT_INFO_ID SEQUENCE_NUM
	CUR_PRICE_DETAILS

HAND_INST

PK	HANDLING_INST_ID
    I2 I1	TYPE VERSION HNDINST_CLASS_TYPE HANDLING_METHOD SHIPPING_GROUP_ID COMMERCE_ITEM_ID QUANTITY

ITEM

PK	COMMERCE_ITEM_ID
    I1 I2	TYPE VERSION ITEM_CLASS_TYPE CATALOG_ID CATALOG_REF_ID CATALOG_KEY PRODUCT_ID QUANTITY STATE STATE_DETAIL PRICE_INFO ORDER_REF

SHIP_PRICE

PK,FK1	AMOUNT_INFO_ID
	RAW_SHIPPING

SUBSKU_ITEM

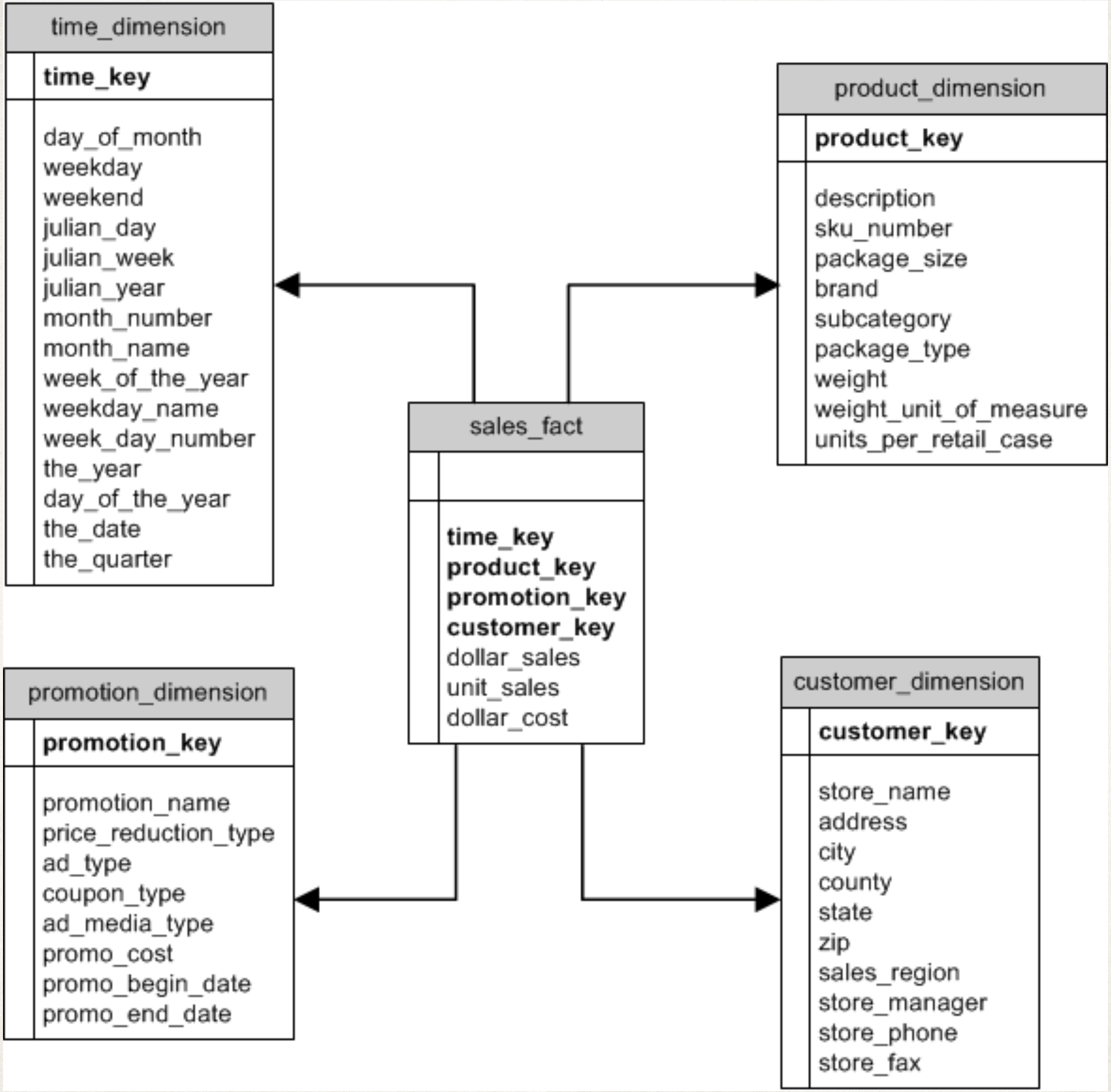
PK,FK1	SUBSKU_ITEM_ID
	IND_QUANTITY

GIFT_INST

PK,FK1	HANDLING_INST_ID
	GIFTLIST_ID GIFTLIST_ITEM_ID

SHIPITEM_TAX

PK,FK1,I1 PK	AMOUNT_INFO_ID SHIPPING_GROUP_ID
	SHIP_ITEM_TAX



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

Key schema differences

- Denormalized codes and categories
- Redundant data
- Many simple one-level joins
- Optimized for query, not operations

Types of keys

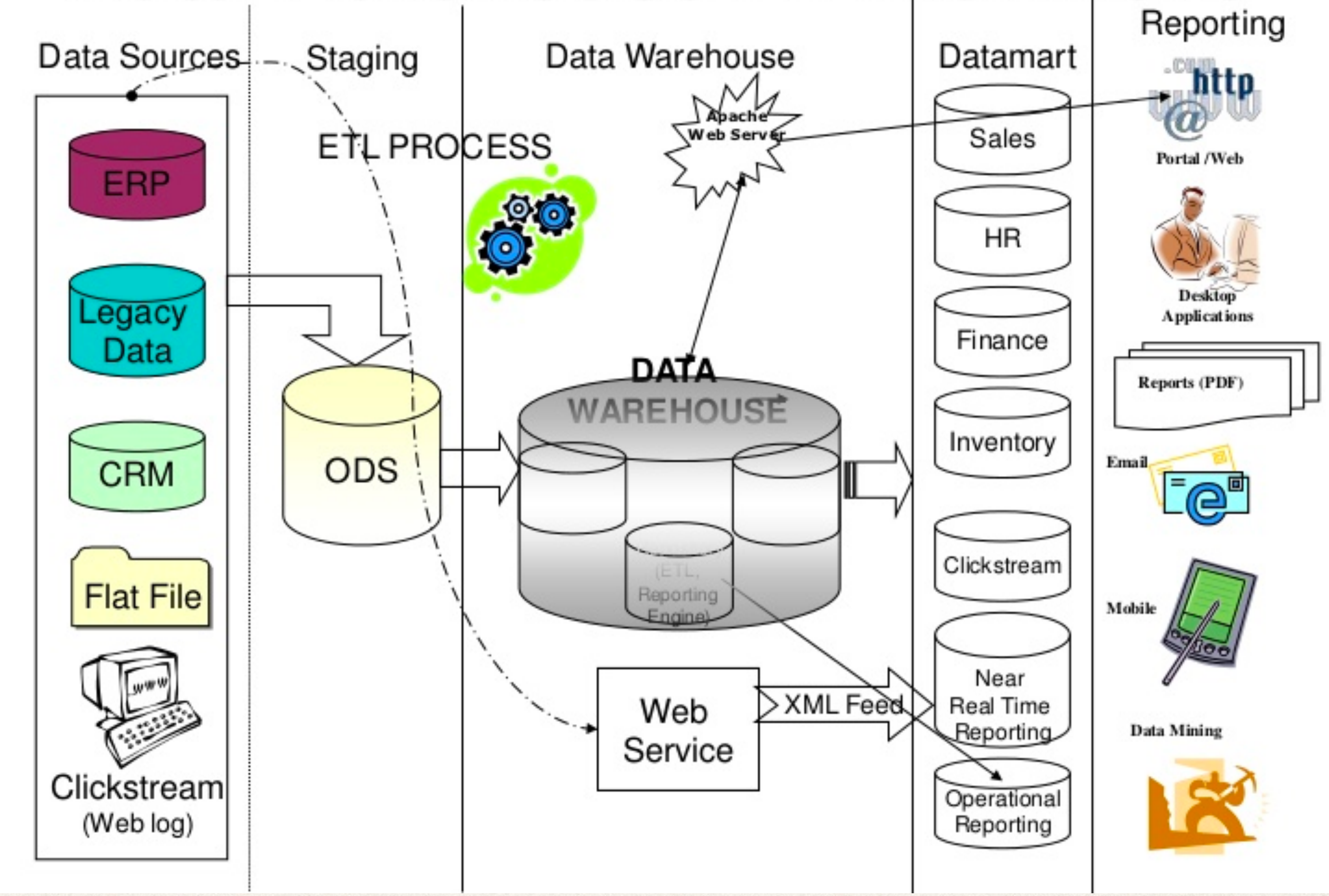
- **Natural keys** - attributes that were likely primary/ foreign keys in source data but do not necessarily work as such in dimensional designs

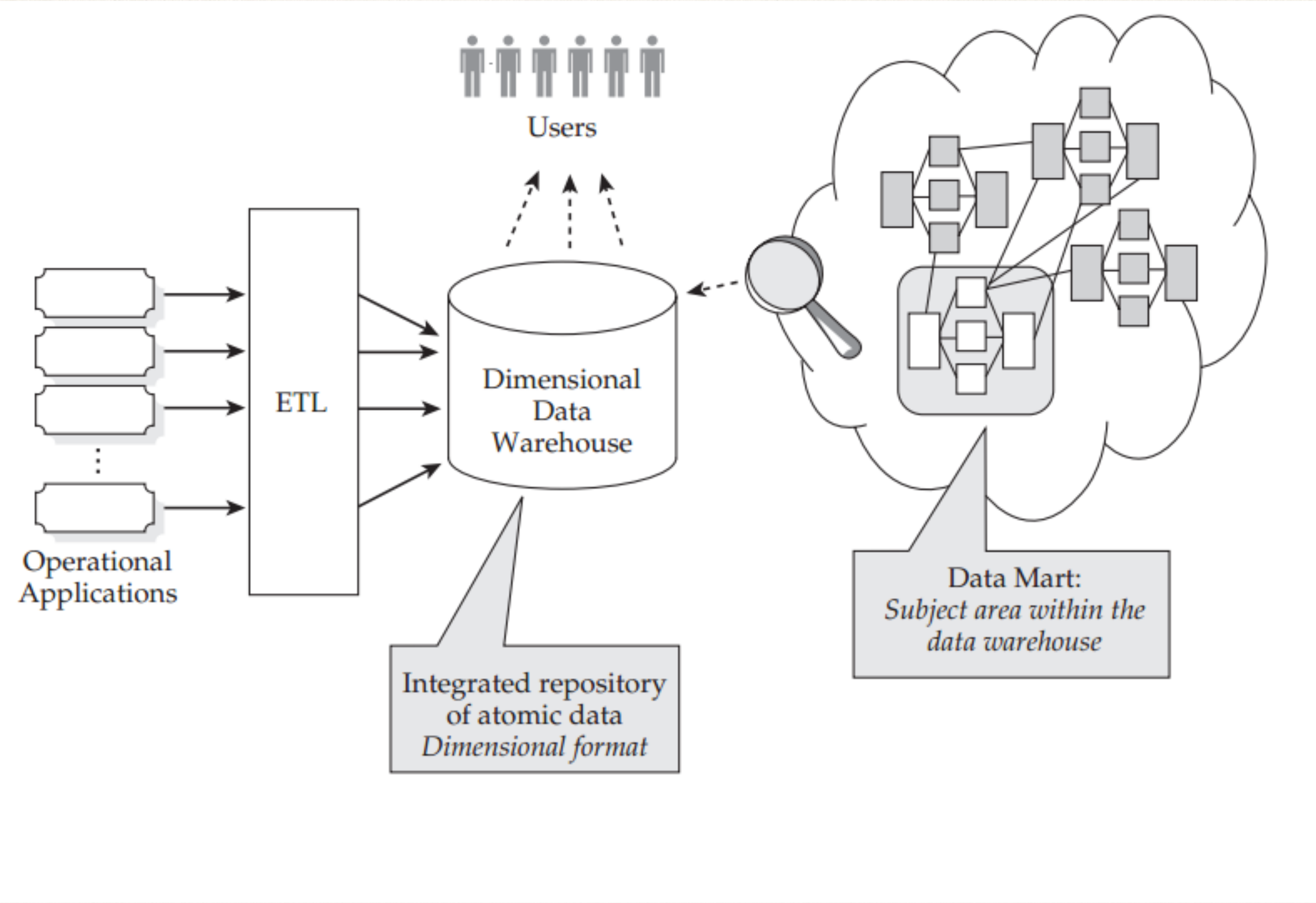
- **Surrogate keys** - primary keys generated for analytical dimension tables, foreign keys on analytical fact tables; no meaning w/r/t source systems

Data warehouse architectures

- Three main models:
  - Corporate Information Factory (Inmon model)
  - Dimensional Data Warehouse (Kimball model)
  - Stand-alone Data Marts
- In practice, you will see mixes of all three

Data Warehouse Environment





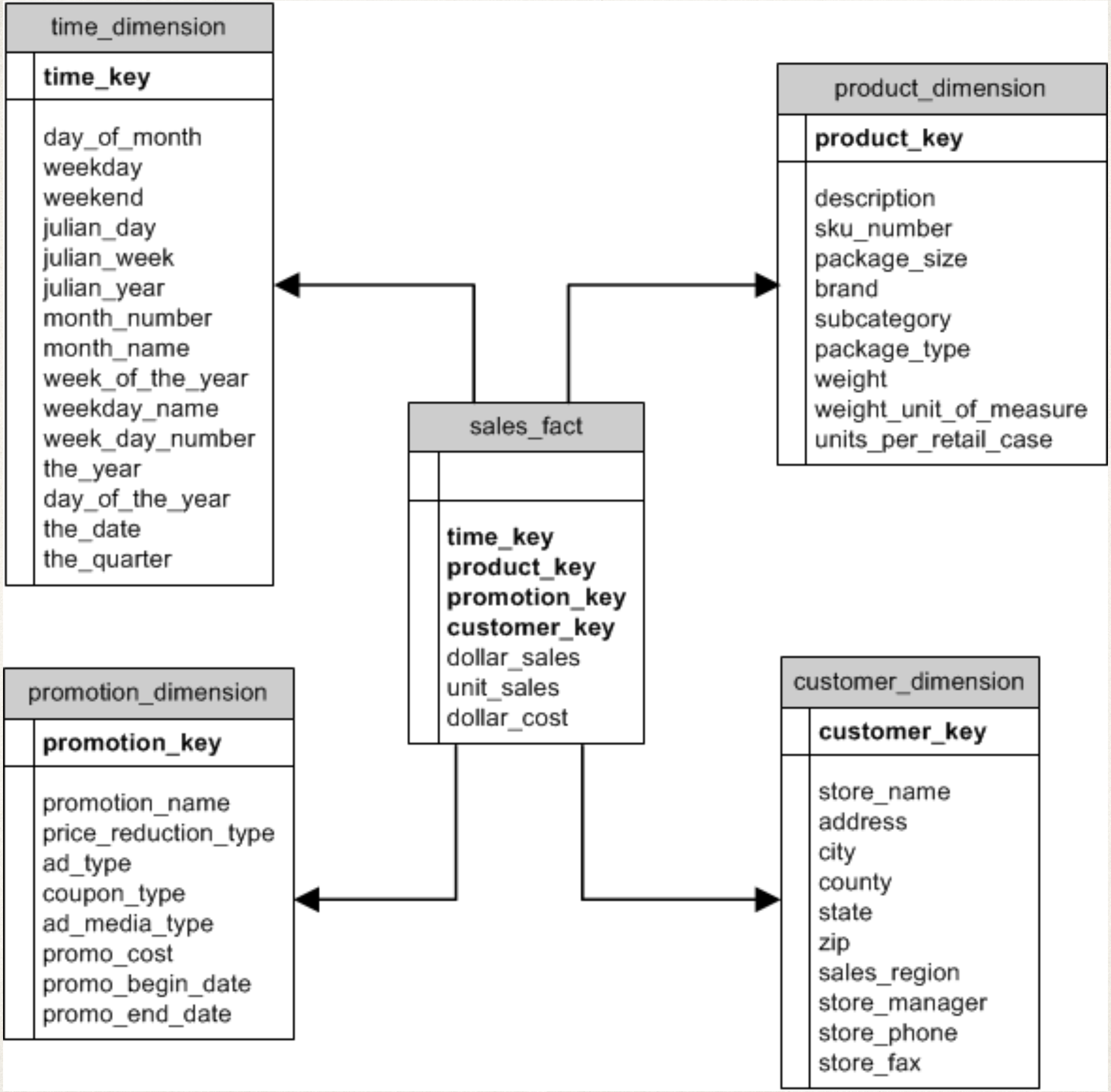
Integrated repository of atomic data *Dimensional format*

Data Mart: *Subject area within the data warehouse*

Key points on architecture

- ETL extracts data from disparate source systems
- ETL can be casual and ad hoc or rigorous and formalized
- Some settings connect users to DW via data marts; others directly through views of the DW
- ETL tools help on back end; BI tools help on front end

Dimension and fact tables



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

Dimensional denormalization

- common combinations (e.g. names)
- codes and descriptions
- flags and values
- multi-part values split up

Dimension affinity

- products, date/time, geography, customers, vendors have related attributes so they fit together naturally

- **junk dimensions** offer a "catch-all" for meaningful dimensional attributes that don't group well

- **snowflakes** allow normalization of some dimensional attributes where valuable

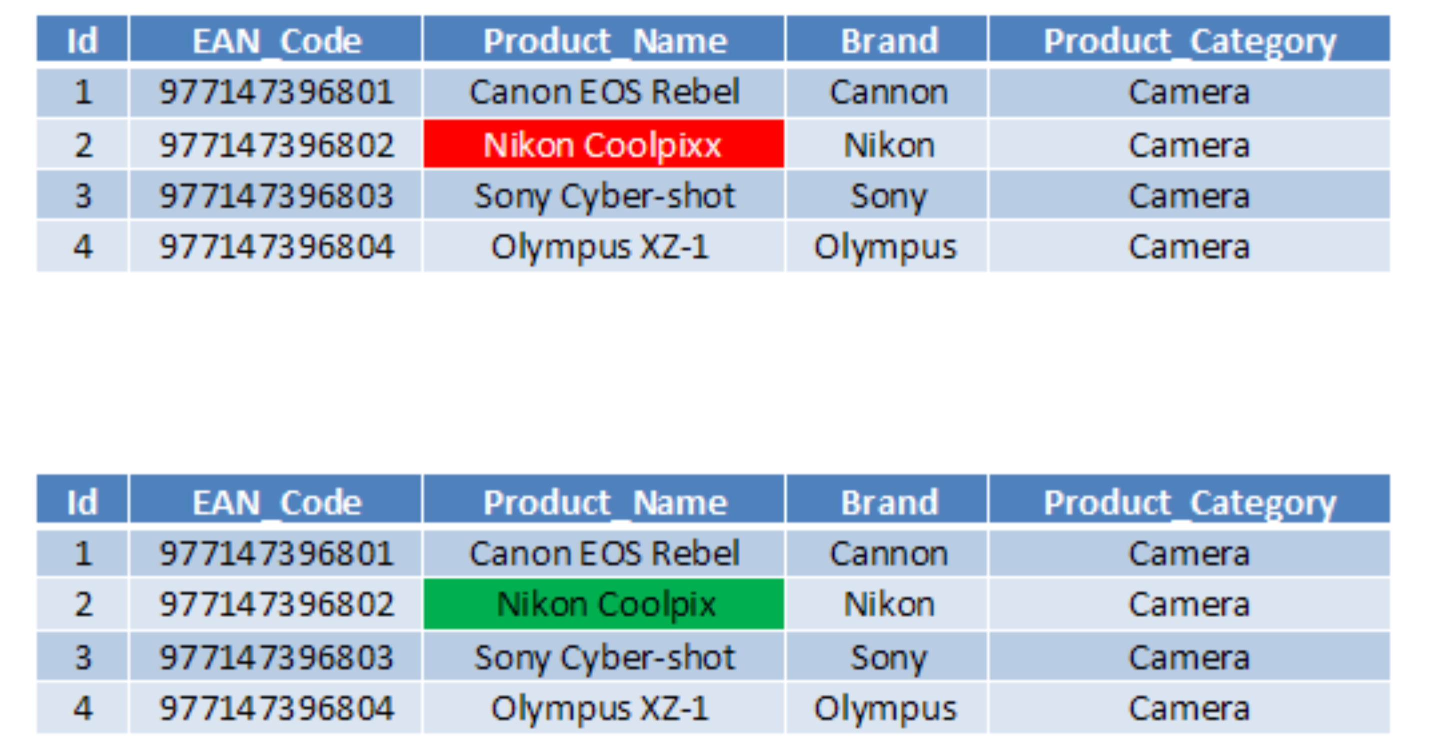
Degenerate dimensions

- data unique to processes but don't fit in dimensions
- added to fact tables, (sort of) treated as dimensions
- "transaction id" or "order id" are canonical examples
- may be natural keys from source system

Slowly changing dimensions

- address how to handle changes in source data
  - **Type 1** - corrections, update data, no history
  - **Type 2** - updates, insert data, keep history
- these decisions part of DW schema design

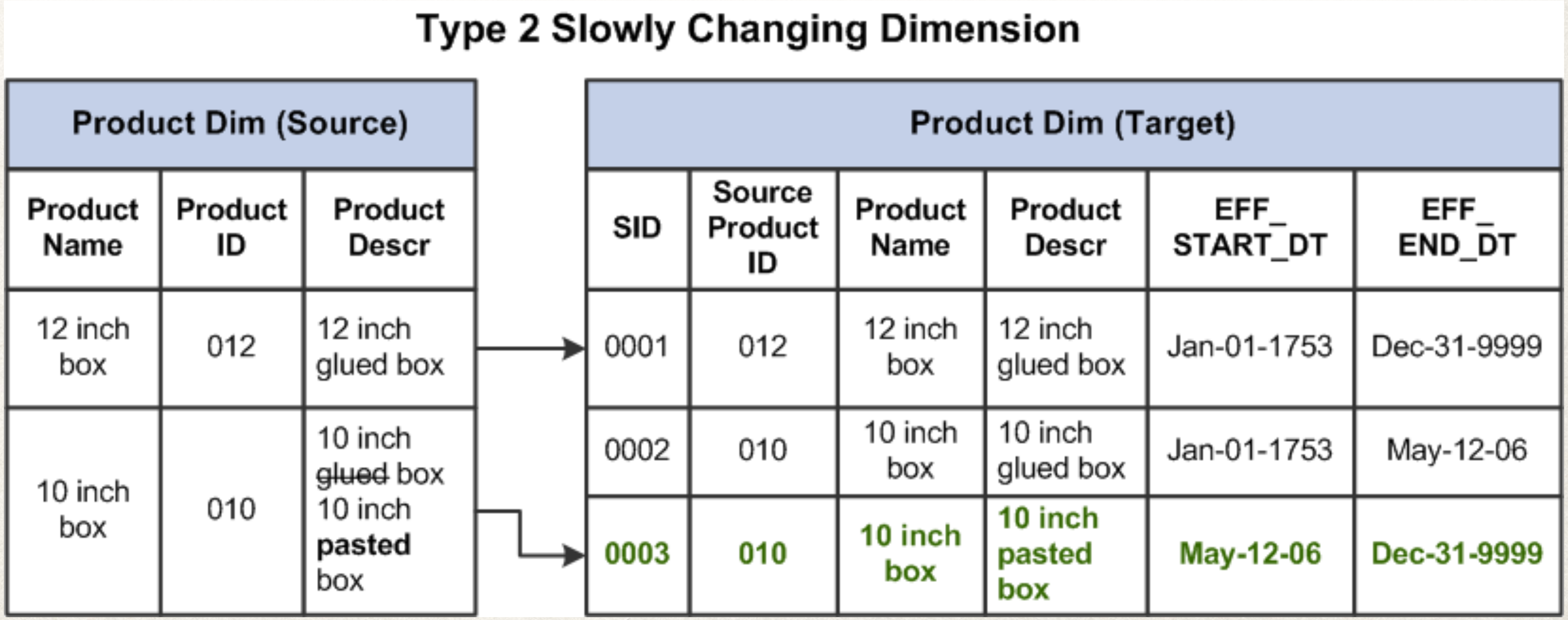
Type 1 - correction



Id	EAN_Code	Product_Name	Brand	Product_Category
1	977147396801	Canon EOS Rebel	Cannon	Camera
2	977147396802	Nikon Coolpixx	Nikon	Camera
3	977147396803	Sony Cyber-shot	Sony	Camera
4	977147396804	Olympus XZ-1	Olympus	Camera

Id	EAN_Code	Product_Name	Brand	Product_Category
1	977147396801	Canon EOS Rebel	Cannon	Camera
2	977147396802	Nikon Coolpix	Nikon	Camera
3	977147396803	Sony Cyber-shot	Sony	Camera
4	977147396804	Olympus XZ-1	Olympus	Camera

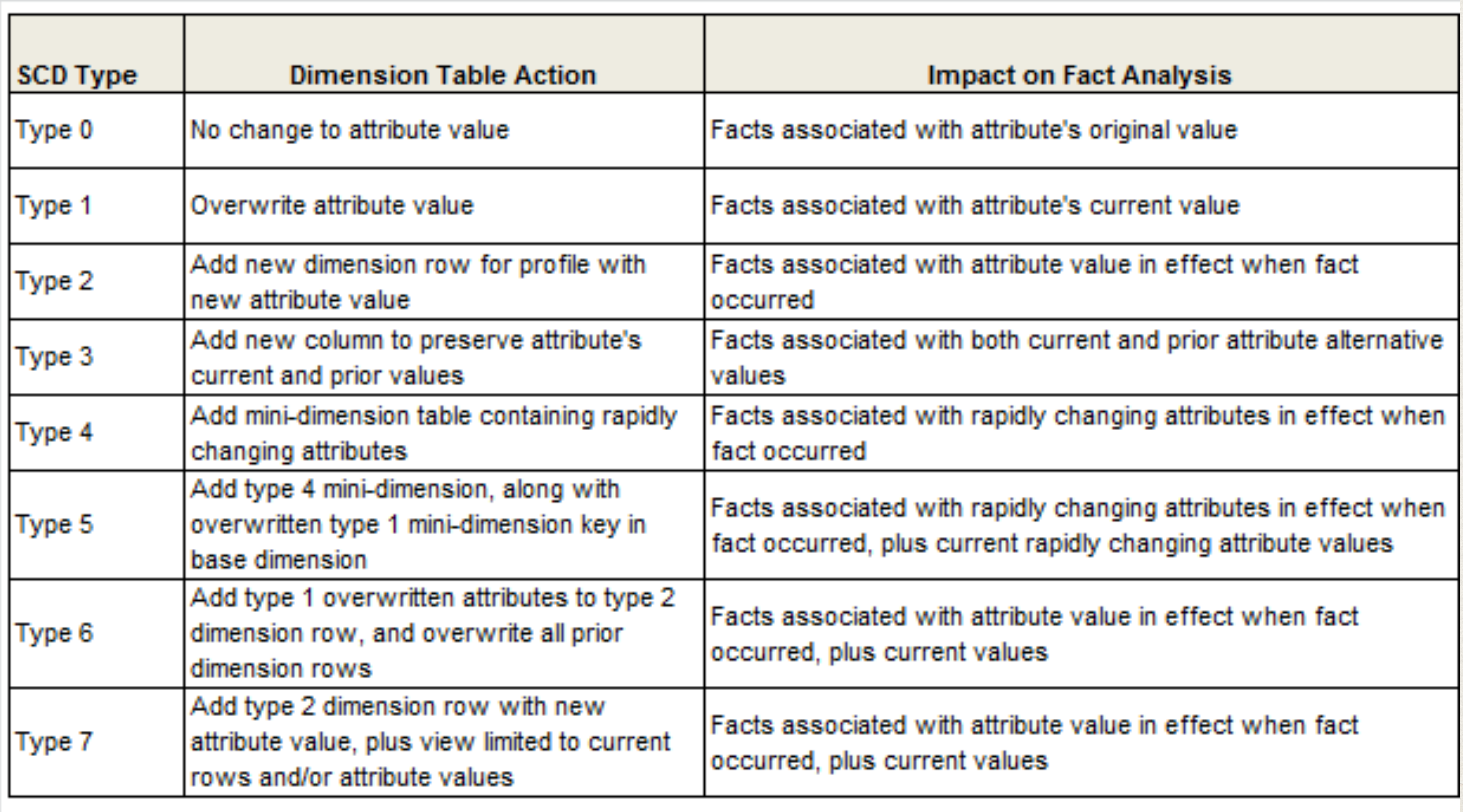
Type 2 - insertion



Type 2 Slowly Changing Dimension

Product Dim (Source)		
Product Name	**Product ID**	**Product Descr**
12 inch box	012	12 inch glued box
10 inch box	010	10 inch ~~glued~~ box 10 inch **pasted** box

Product Dim (Target)					
SID	**Source Product ID**	**Product Name**	**Product Descr**	**EFF_ START_DT**	**EFF_ END_DT**
0001	012	12 inch box	12 inch glued box	Jan-01-1753	Dec-31-9999
0002	010	10 inch box	10 inch glued box	Jan-01-1753	May-12-06
0003	**010**	**10 inch box**	**10 inch pasted box**	**May-12-06**	**Dec-31-9999**



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

www.kimballgroup.com/2013/02/design-tip-152-slowly-changing-dimension-types-0-4-5-6-7/

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

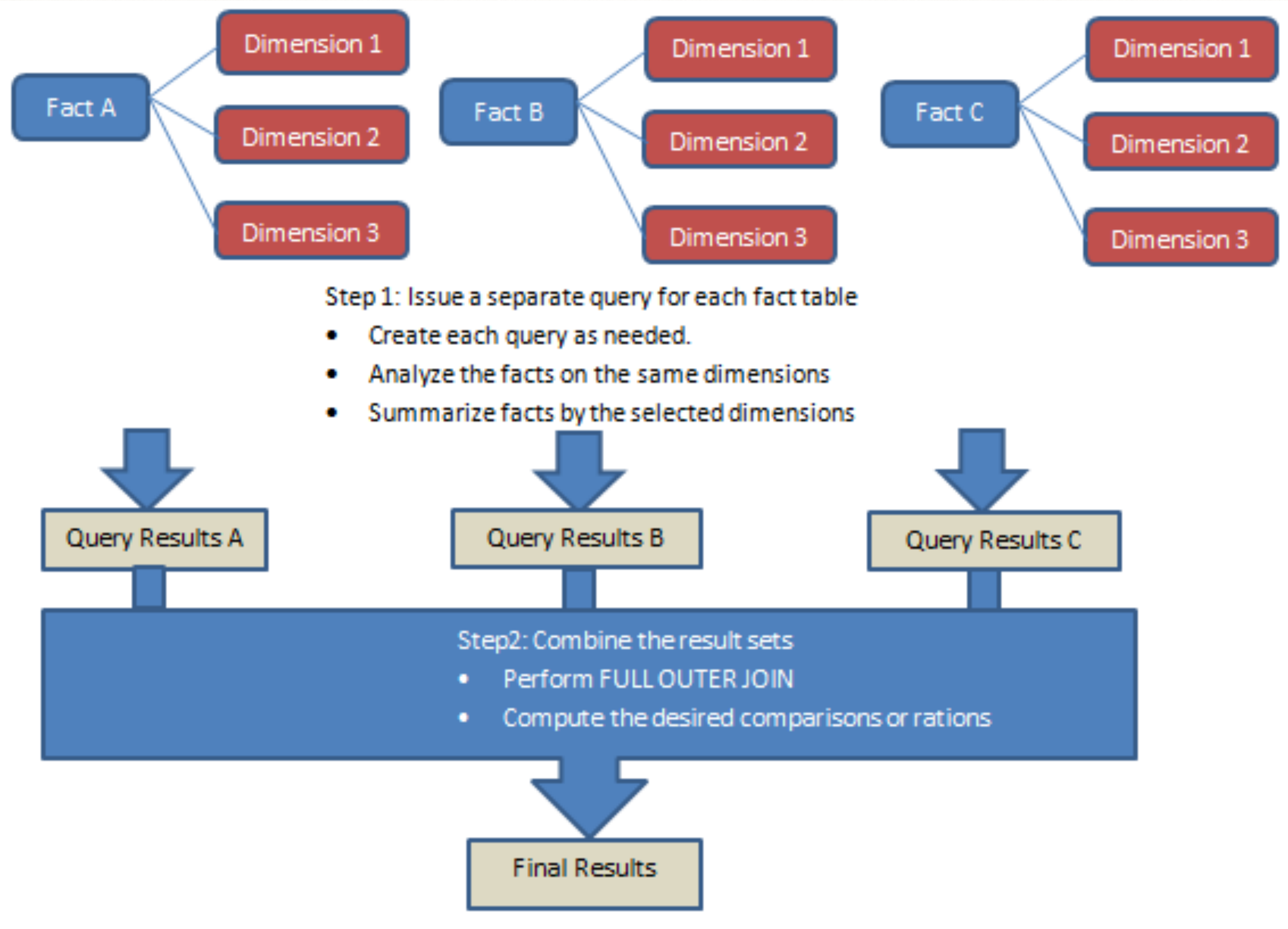
- Key questions:
  - Do two facts/processes occur simultaneously?
  - Are both available at the same grain?
- If "no" to either, you have more than one fact

Distinguishing different facts

- Different timing: e.g. sales and shipping
  - sale ends with financial transaction; shipping starts with end of sale and ends with delivery
- Different grain: e.g. sales and shipping
  - measurement of sales reflects customer preferences and pricing; measurement of shipping reflects inventory mgmt, shipper performance, reliability

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

Step2: Combine the result sets

- Perform FULL OUTER JOIN
- Compute the desired comparisons or rations

More basic ETL in SQL

Exercise 04