OLAP

Alvin Cheung
Aditya Parameswaran

Berkeley
cs186

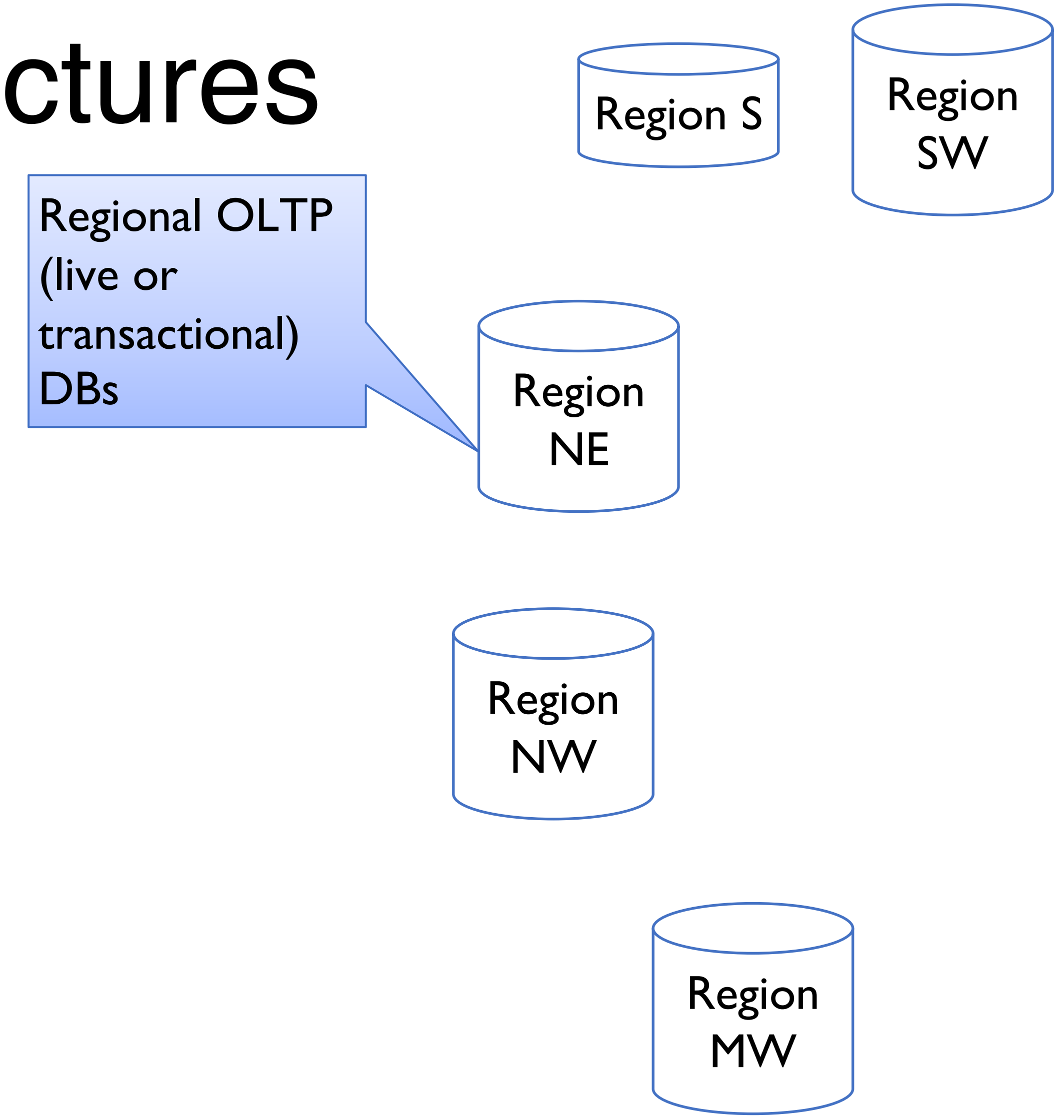
What is OLAP?

- OLAP = OnLine Analytical Processing
- Aka *decision support* or *business intelligence* (BI)
  - But now BI has diversified (e.g., ML)

- What is OLAP?
  - A specialization of DBMSs that prioritizes reading and summarizing large volumes (PBs) of data to understand trends and patterns
    - E.g., total sales of each type of Honda car over time for each county
  - “Read-only” queries
- Contrast to OLTP: OnLine Transaction Processing
  - “Read-write” queries
  - Usually touch a small amount of data
    - e.g., append a new car sale into the sales table

Typical Architectures

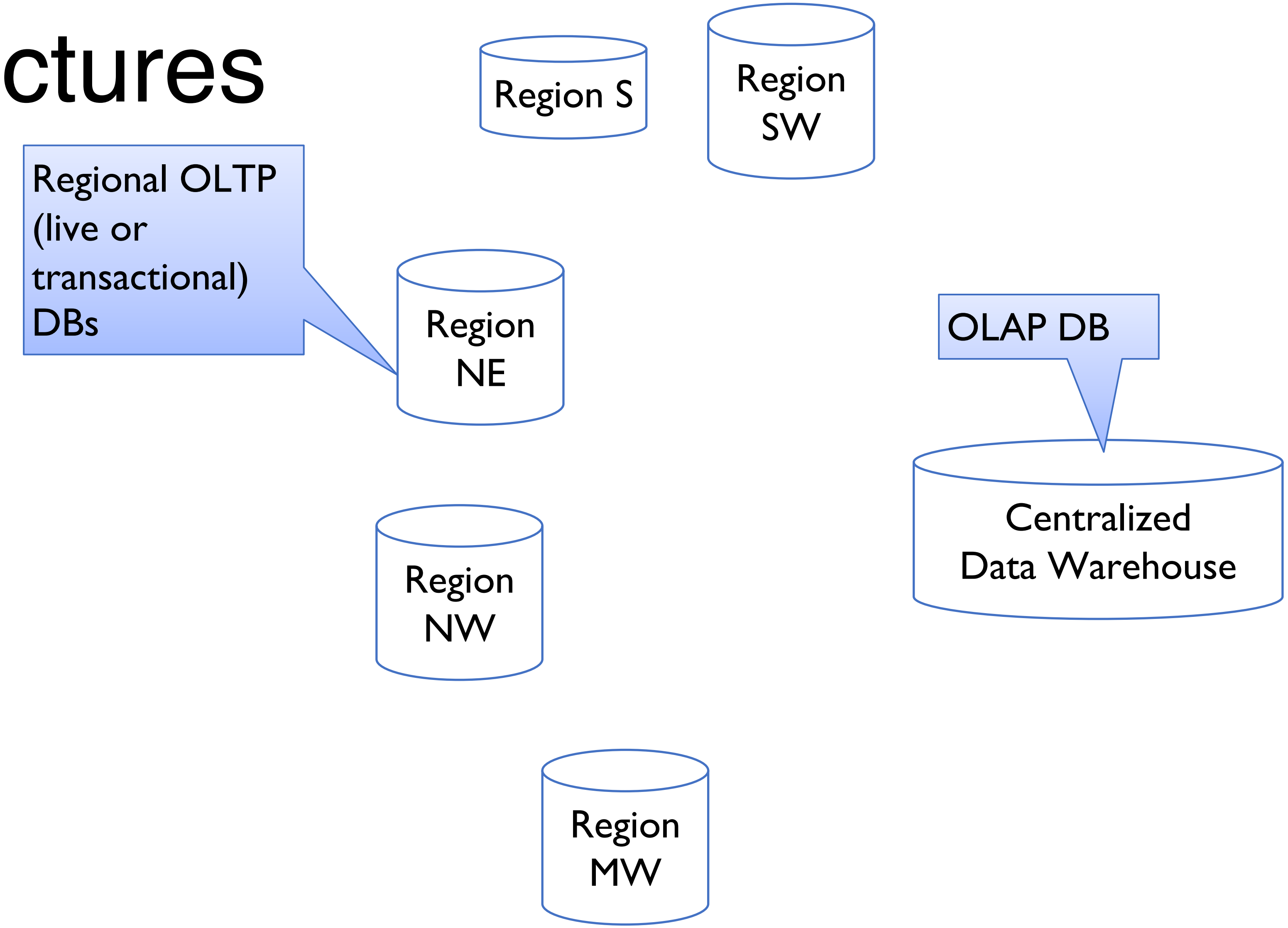
- Imagine Honda USA
- Many sales regions



Typical Architectures

- Imagine Honda USA
- Many sales regions

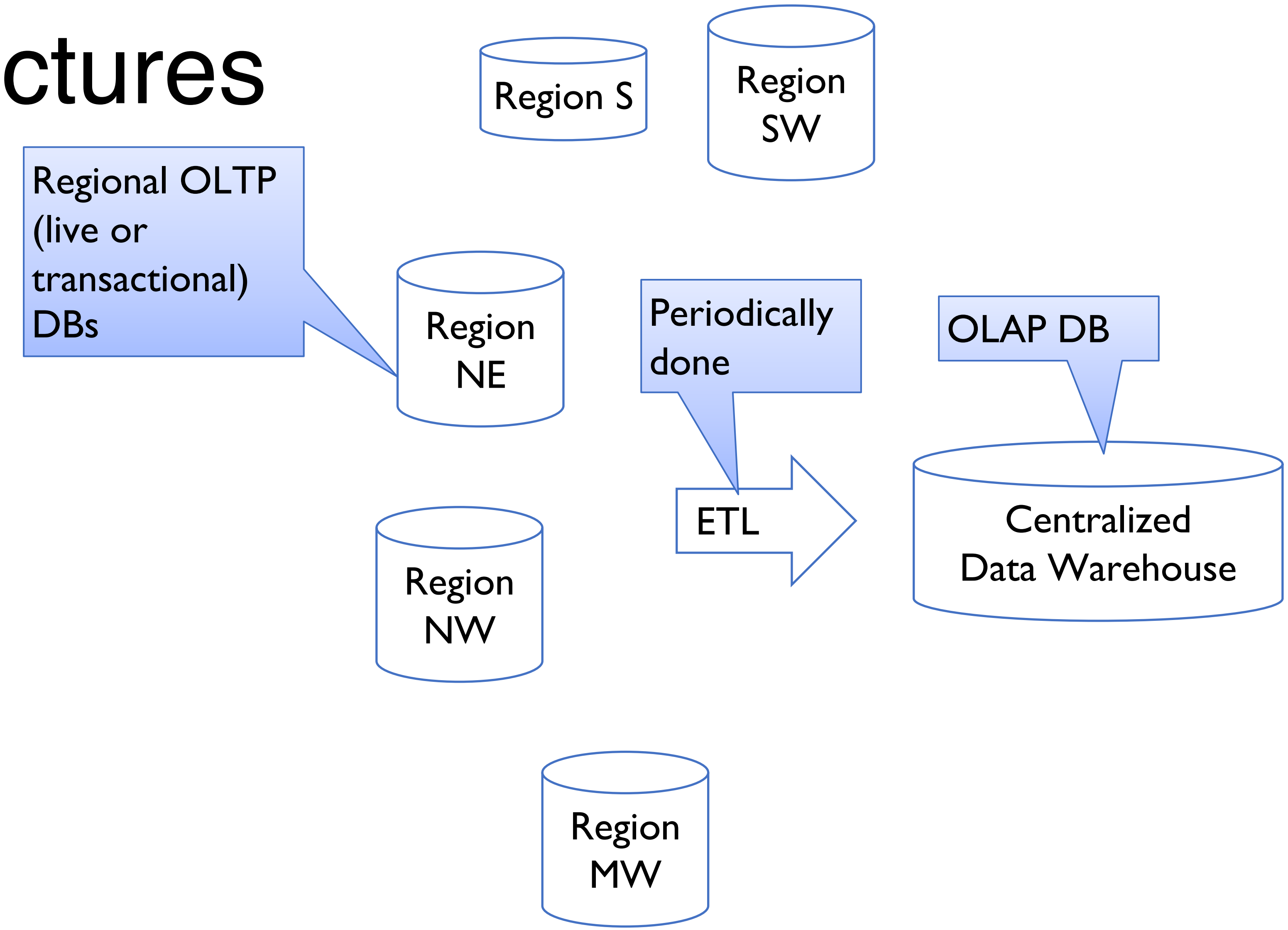
- OLAP is performed on a separate *data warehouse* away from the critical path of OLTP.
- Post-hoc large-scale analysis happens separate from txns



Typical Architectures

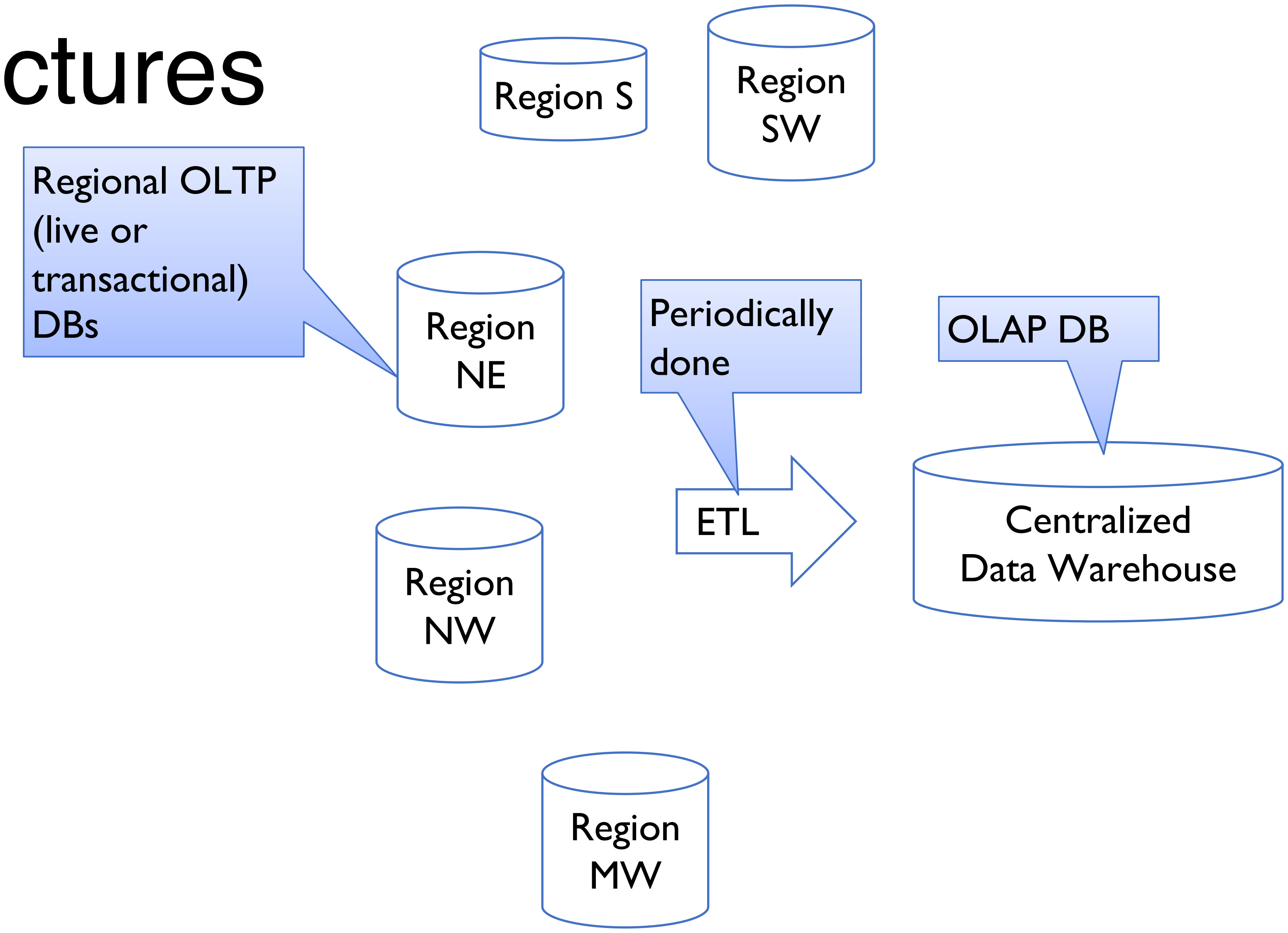
- Imagine Honda USA
- Many sales regions

- Data warehouse periodically loaded w/ new data
  - E.g, regional sales data gets consolidated at end of each day



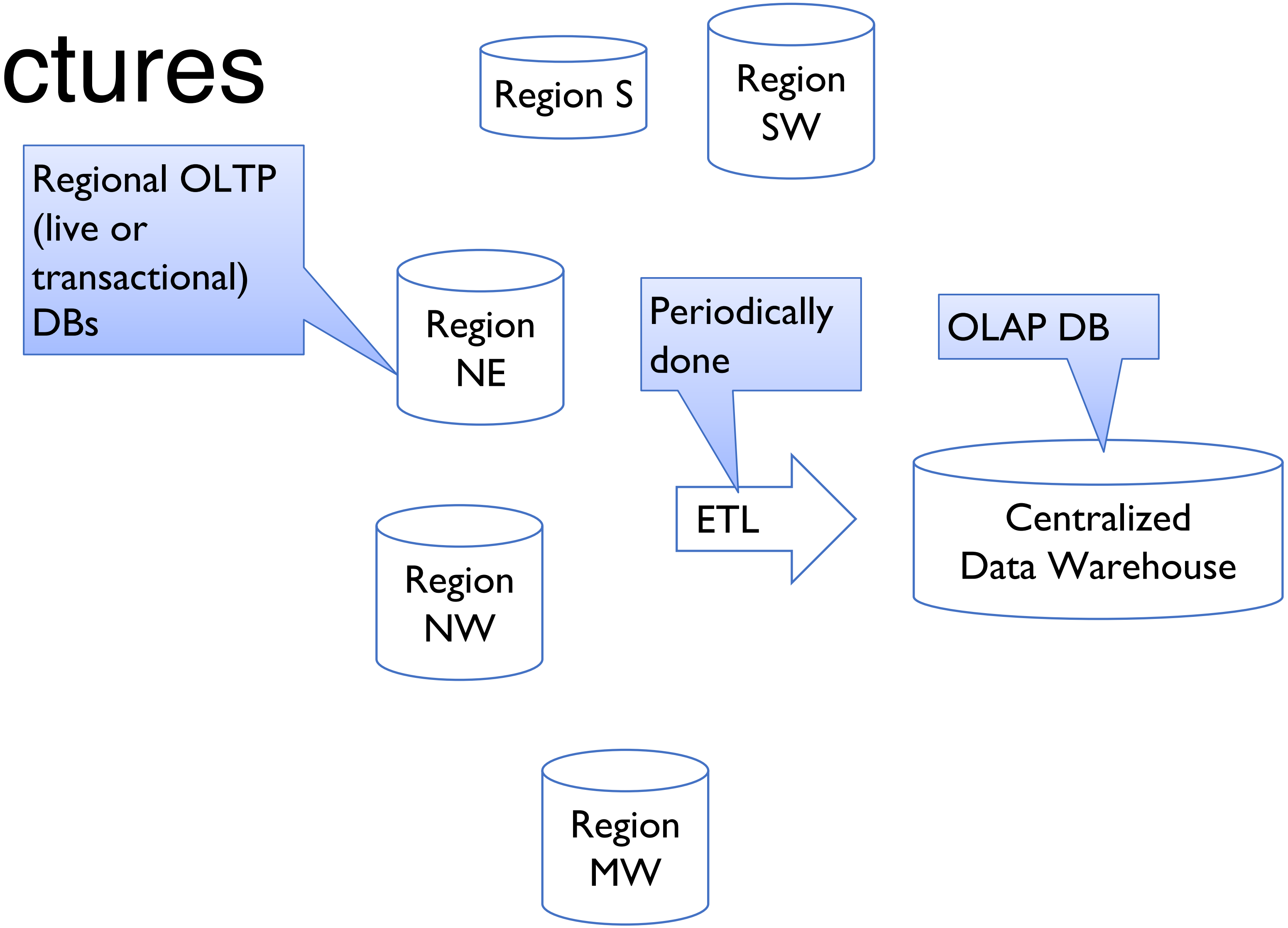
Typical Architectures

- Consolidation happens through ETL
  - Extract, Transform, Load
  - Extract useful business info to be summarized, transform it (e.g., canonicalize, clean up), load it in the warehouse
  - Ready for analysis!



Typical Architectures

- Challenge: staleness
- Still, usually a reasonable tradeoff:
  - Large scale OLAP (read) queries may delay txns
    - Crucial to ensure that sales are not prevented than a report for a manager is generated promptly
    - Latter will anyway take a long time, so might as well have them wait a bit longer
  - OK if the analysis results are a bit stale



Schemas in Data Warehouses

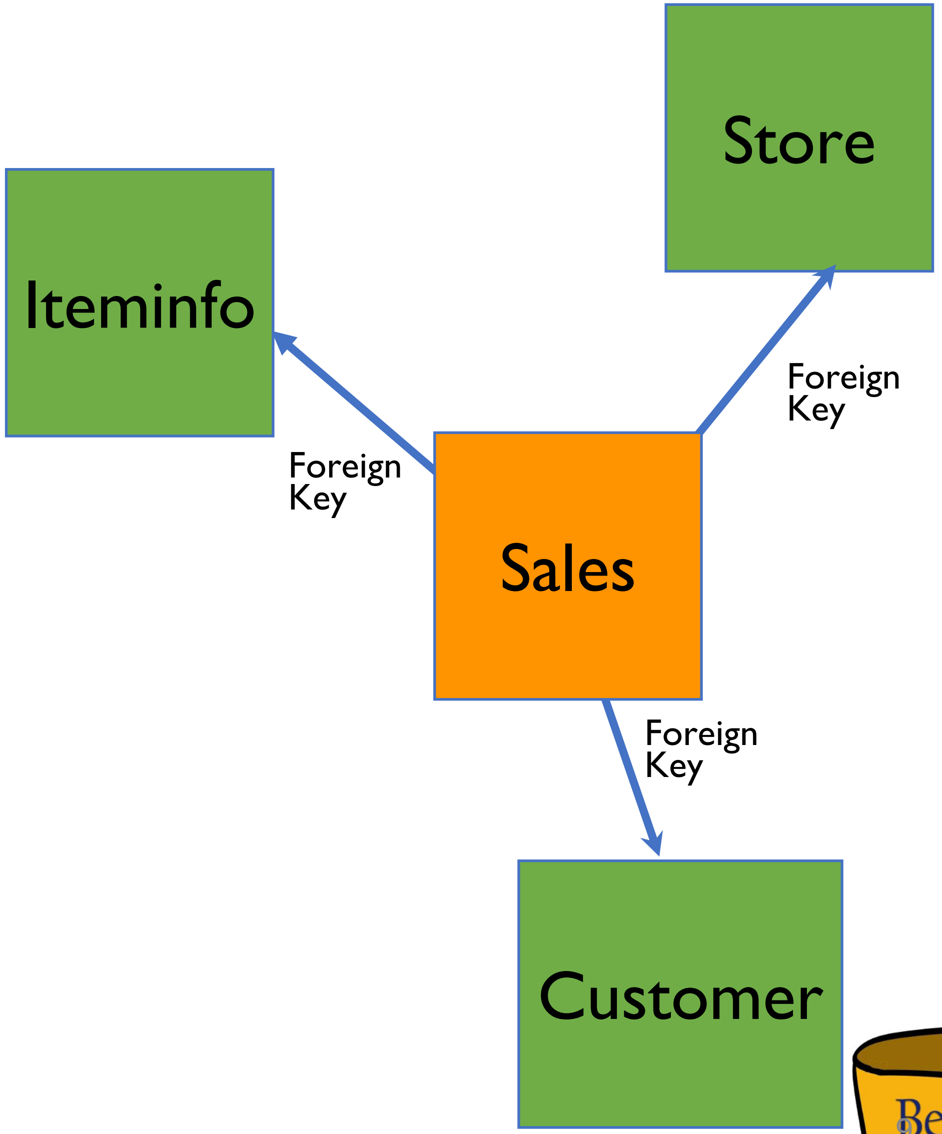
Centralized Data Warehouse

- Usually employs a *star* (or sometimes *snowflake*) schema
- One *fact table* and many *dim. tables*
- Fact tables contain *dimension attr.* and *measure attr.*

- Canonical data warehouse example:
  - Fact table: **Sales** (itemid, storeid, customerid, date, number, price)
  - Dim table: **Iteminfo** (itemid, itemname, color, size, category)
  - Dim table: **Store** (storeid, city, state, country)
  - Dim table: **Customer** (customerid, name, street, city, state)

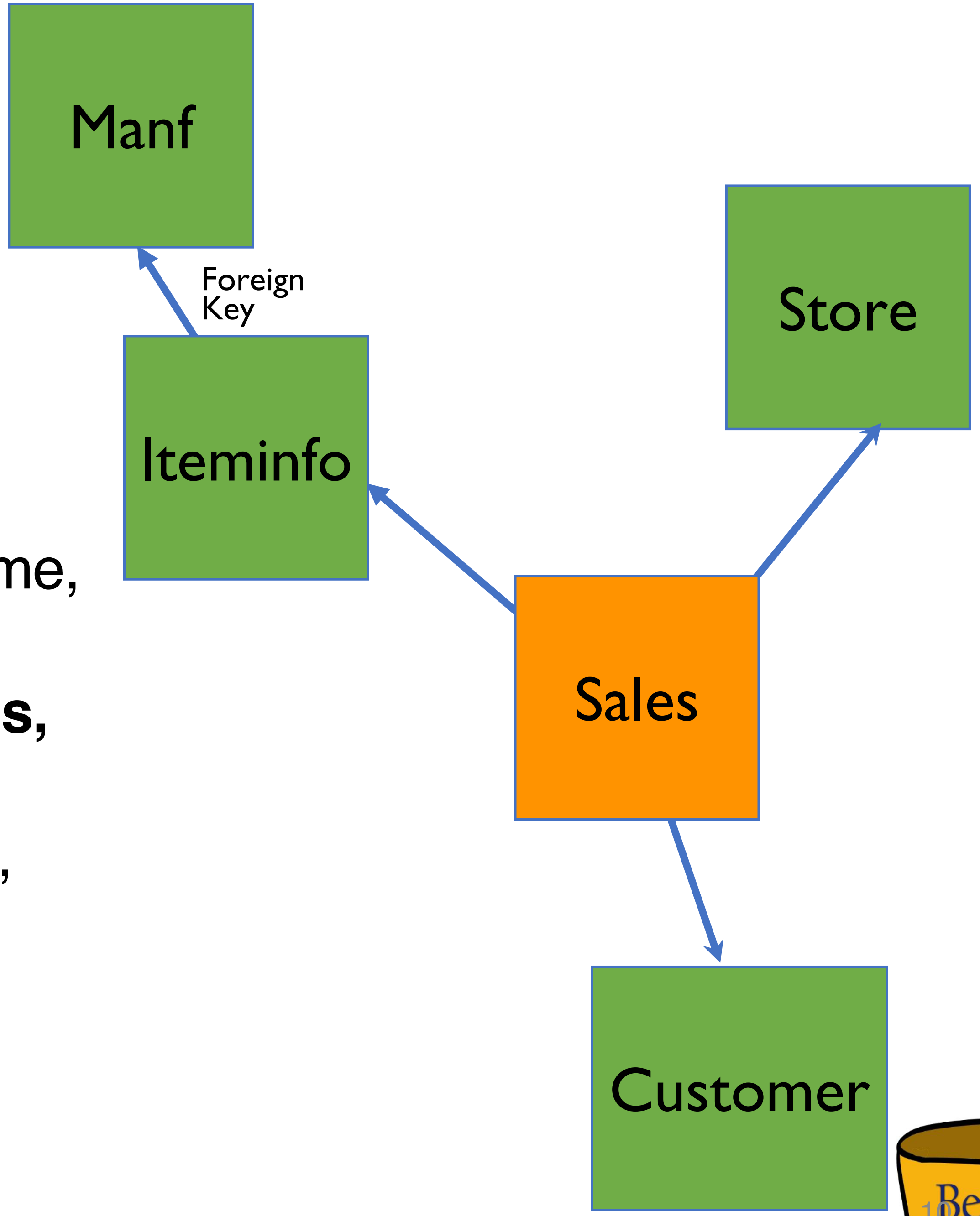
Star Schema

- Example Schema
  - Fact table: **Sales** (itemid, storeid, customerid, date, number, price)
  - Dim table: **Iteminfo** (itemid, itemname, color, size, category)
  - Dim table: **Store** (storeid, city, state, country)
  - Dim table: **Customer** (customerid, name, street, city, state)

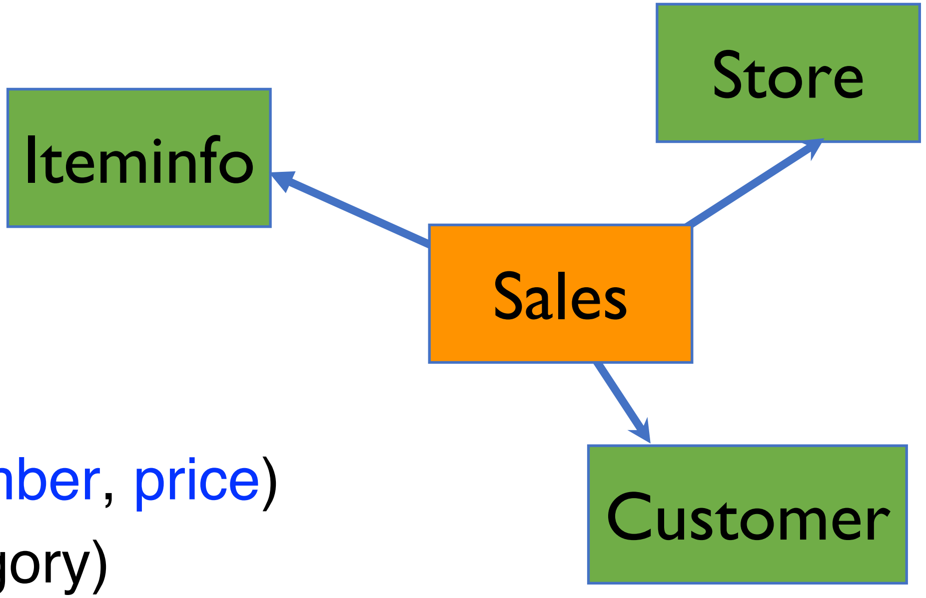


Star to Snowflake

- Extending the example
  - Fact table: **Sales** (itemid, storeid, customerid, date, number, price)
  - Dim table: **Iteminfo** (itemid, itemname, color, size, category, manfname)
    - **Dim table: Manf (name, address, owner)**
  - Dim table: **Store** (storeid, city, state, country)
  - Dim table: **Customer** (customerid, name, street, city, state)



OLAP Queries



- Example Schema:
  - Fact table: Sales (itemid, storeid, customerid, date, number, price)
  - Dim table: Iteminfo (itemid, itemname, color, size, category)
  - Dim table: Store (storeid, city, state, country)
  - Dim table: Customer (customerid, name, street, city, state)

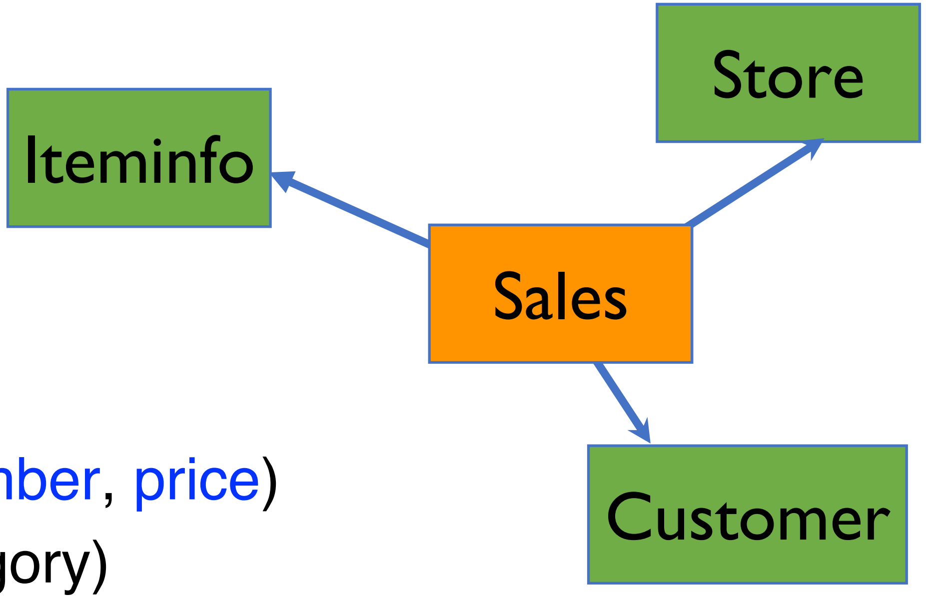
- Typical "report" queries GROUP BY some dim. attrs., aggregate some measure attrs.

  SELECT category, country, COUNT(number)

  FROM Sales NATURAL JOIN Iteminfo NATURAL JOIN Store

  GROUP BY category, country
- Q: What does this query return?

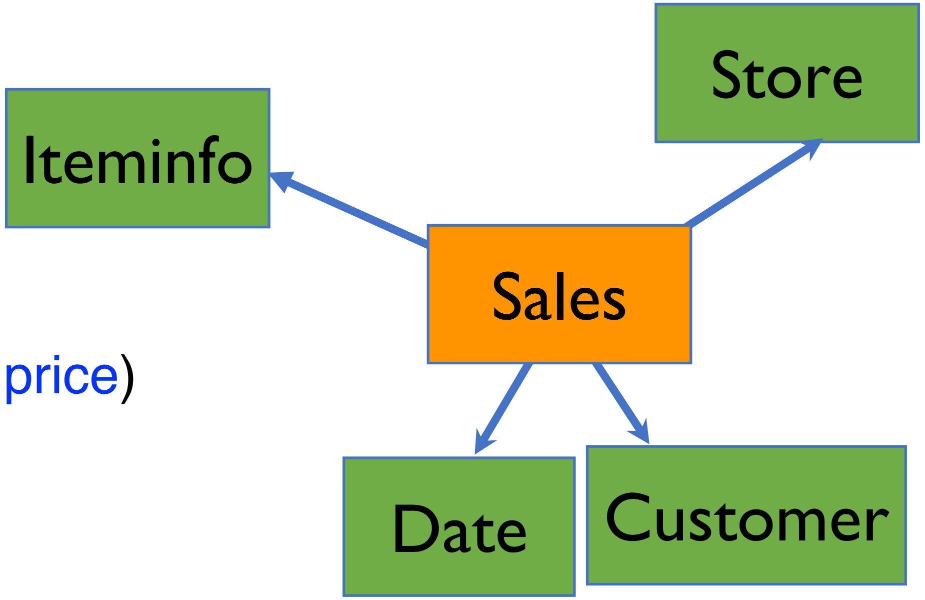
Dates in OLAP



- Example Schema:
  - Fact table: Sales (itemid, storeid, customerid, date, number, price)
  - Dim table: Iteminfo (itemid, itemname, color, size, category)
  - Dim table: Store (storeid, city, state, country)
  - Dim table: Customer (customerid, name, street, city, state)

- Not very useful to "aggregate" date; better to treat date as an implicit dimension!
  - Allows us to group by and see trends across dates (e.g., sales by year)
  - Fact table: Sales (itemid, storeid, customerid, date, number, price)
  - **Implicit Dim table: Dateinfo (date, month, quarter, year)**

Star Schema



- Example Schema:
  - Fact table: Sales (itemid, storeid, customerid, **date**, number, price)
  - Dim table: Iteminfo (itemid, itemname, color, size, category)
  - Dim table: Store (storeid, city, state, country)
  - Dim table: Customer (customerid, name, street, city, state)
  - Implicit Dim table: **Dateinfo (date, month, quarter, year)**

- Example query:

```
SELECT category, country, month, COUNT(number)
FROM Sales NATURAL JOIN Iteminfo NATURAL JOIN Store NATURAL JOIN Dateinfo
GROUP BY category, country, month
```

- Actual query may be (Postgres, SQL Server, Snowflake, …)

```
SELECT category, country, datepart('month', date) as month, COUNT(number)
FROM Sales NATURAL JOIN Iteminfo NATURAL JOIN Store
GROUP BY category, country, month
```

Introducing Data Cubes

- For now, will operate on a "denormalized view", where fact and dim. tables are joined
  - same considerations for star/snowflake schema
- Consider a simpler inventory relation: (item, color, size, number)

Item	Color	Size	Number
Jacket	Blue	Small	1
Jacket	Red	Medium	1
Jeans	Black	Large	2
…	…	…	…

Vanilla GROUP BY across all groups

- Q: If there are n item names, m colors, and k sizes, what are the number of rows?

Item	Color	Size	Number
Jacket	Blue	Small	23
Jacket	Blue	Medium	17
Jacket	Blue	Large	34
Jacket	Red	Small	18
…	…	…	…
Jeans	Blue	Small	14
…	…	…	13

Say I was interested in only the color and item…

- Might want to see a *cross-tabulation* of aggs corr. to Item and Color

- Q: How could you get this via a GROUP BY?

	Blue	Red	…	Total
Jacket	23	45	…	234
Jeans	24	28	…	462
…	…	…	…	…
Total	89	132	…	2384

Say I was interested in only the color and item…

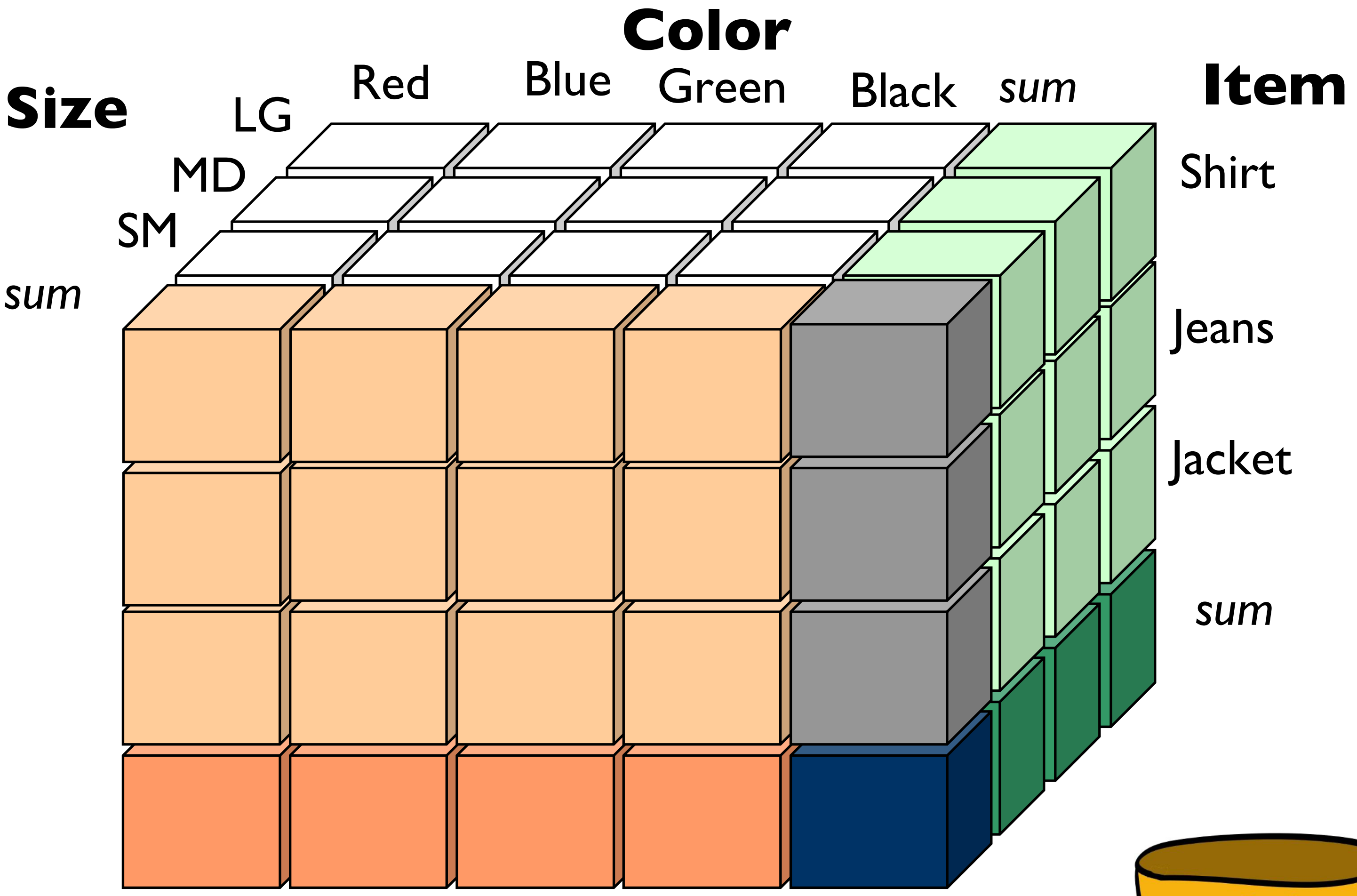
- Might want to see a *cross-tabulation* of aggs corr. to Item and Color
- Q: How could you get this via a GROUP BY?
- A: group by combinations, and individual values, and overall count

	Blue	Red	…	Total
Jacket	23	45	…	234
Jeans	24	28	…	462
…	…	…	…	…
Total	89	132	…	2384

Another way to view this…

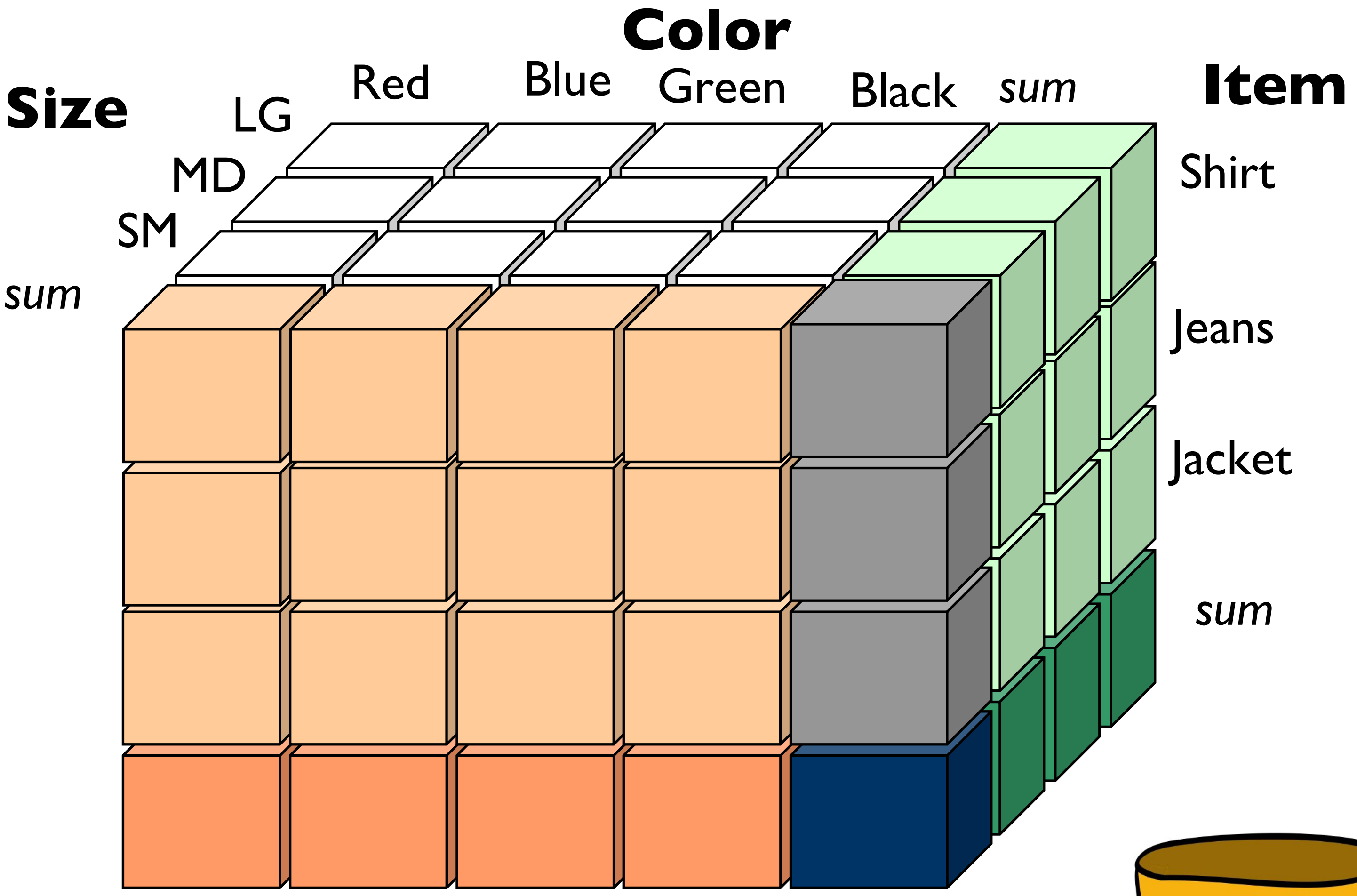
- Crosstab of item and color = vertical plane closest to us

	Blue	Red	…	Total
Jacket	23	45	…	234
Jeans	24	28	…	462
…	…	…	…	…
Total	89	132	…	2384



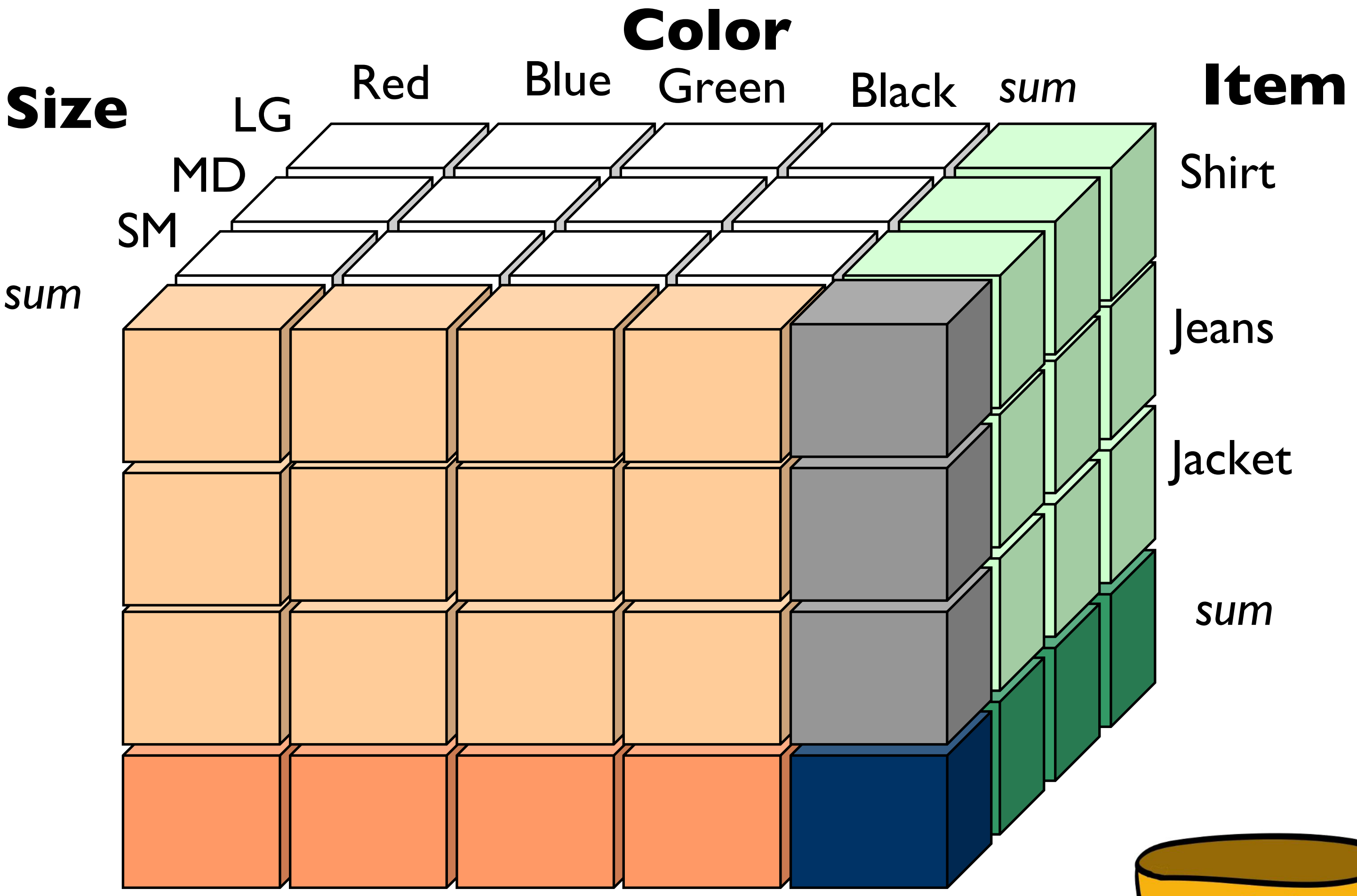
Another way to view this…

- This is a *data cube*
- Has 3 dimensions (hence the name dimensions for those attributes!)
- Can be *sliced* and *diced* in various ways



Another way to view this…

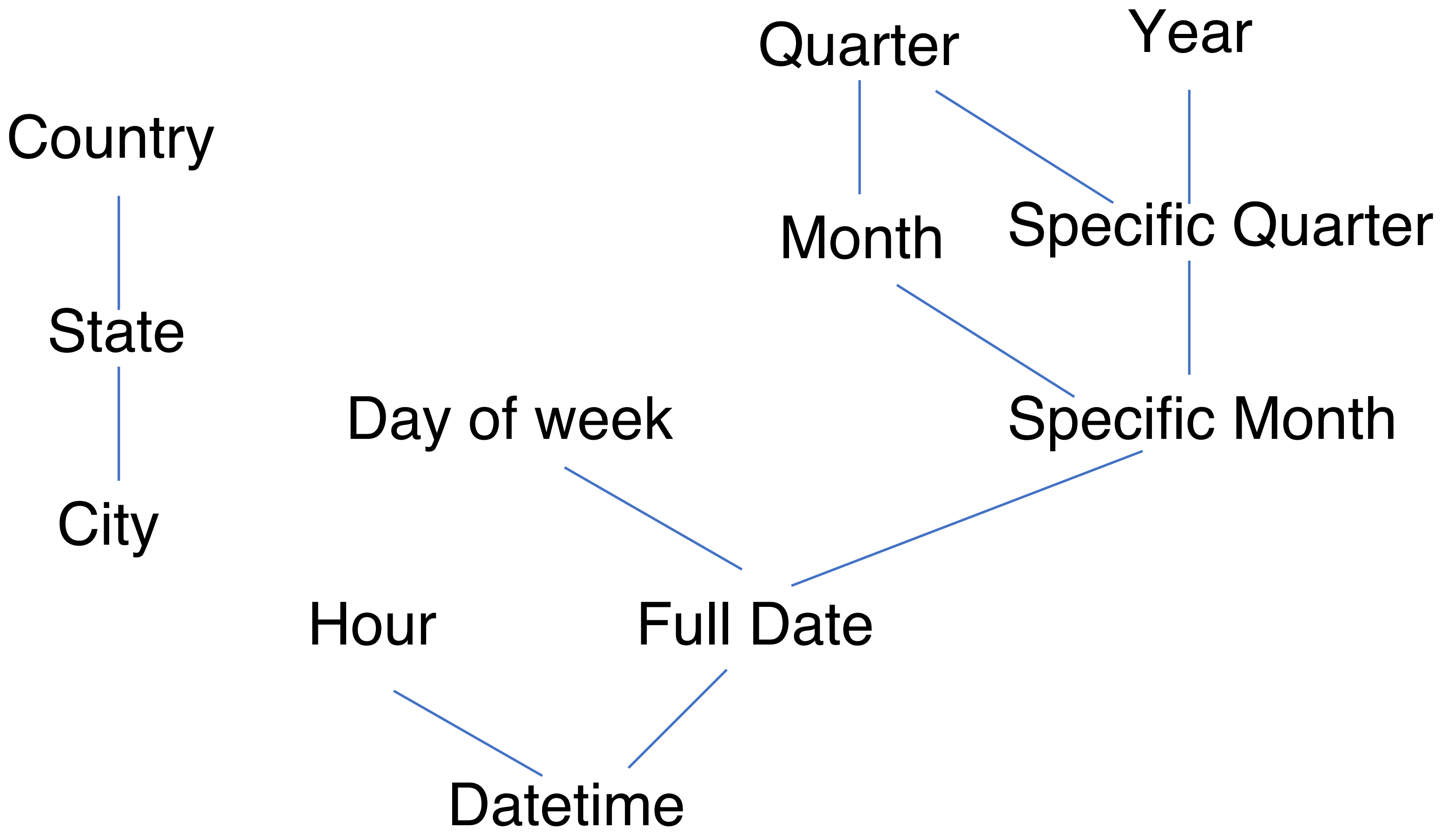
- *Slicing* = adding a condition to one/more of the dimensions
  - e.g., green color
- *Dicing* = partitioning of the dimension
  - Here, we partitioned based on distinct values, but can partition in more coarse-grained ways
  - e.g., lights and darks for color
- Especially useful for the date dimension:
  - Can group by months, days, years, weeks, etc.



Berkeley cs186

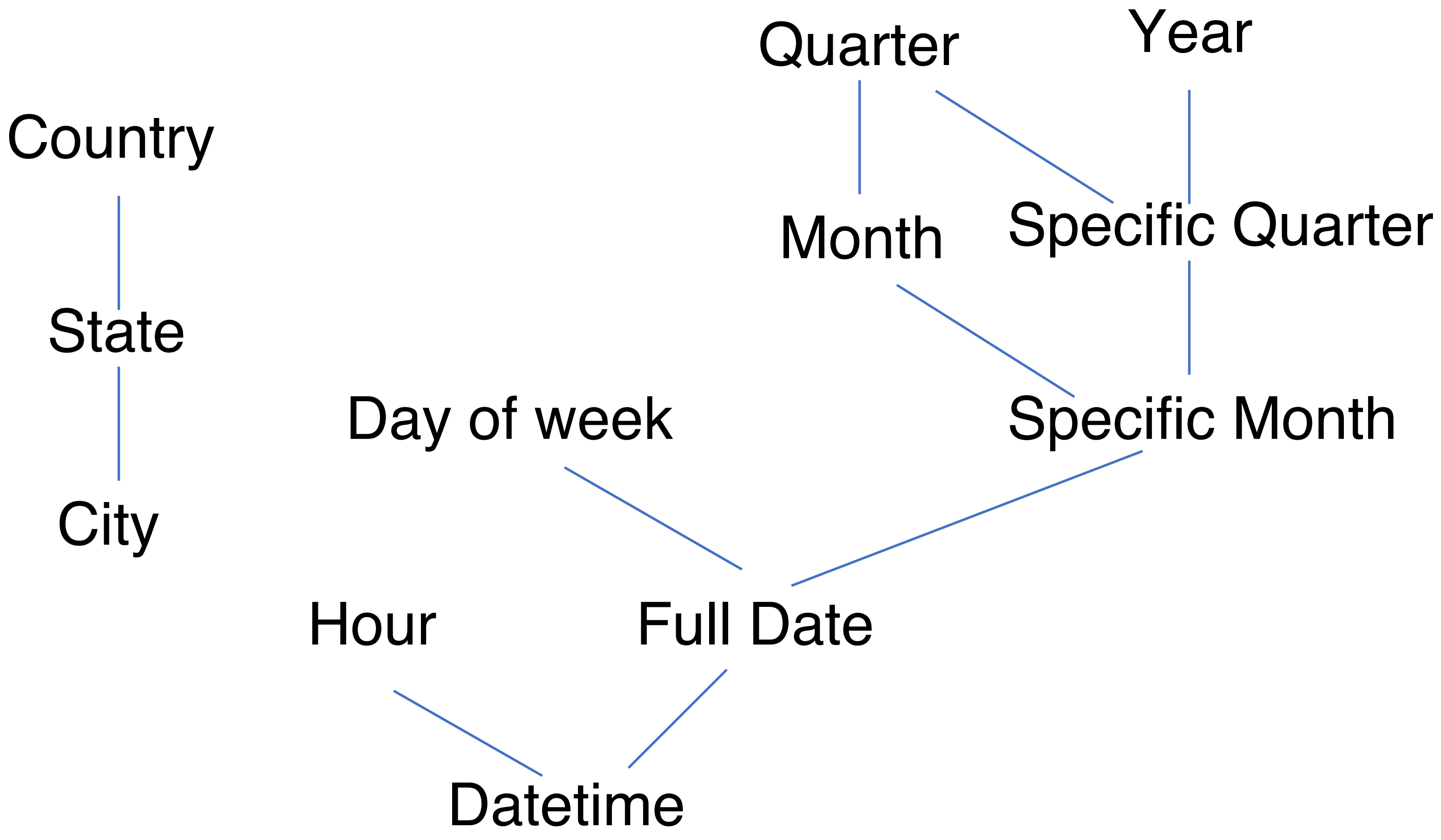
Hierarchies for Setting the Partitioning Granularity

- The partitioning granularity can be set based on user needs…
- Different partitioning may be useful for different applications



Hierarchies for Setting the Partitioning Granularity

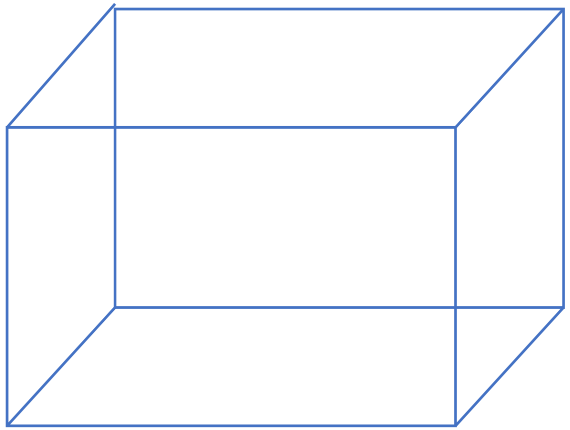
- Q: I want the aggregates per month. What if we computed a data cube based on Full Date? Can I avoid recomputing the cube (or cross-tab)?
- Q: What about if we had computed a data cube based on Year?



Moves in the Hierarchy and Corresponding Cube

- Moving from a finer to a coarser granularity is called a *rollup*
- Moving from a coarser to a finer granularity is called a *drill-down*

- Unit steps from coarse to fine (drill-down):
  - Move down the hierarchy for one of the dimensions, OR
  - Move from
    - the origin to an edge, or
    - an edge to a plane, or
    - a plane to the cube



OLAP in SQL: CUBE

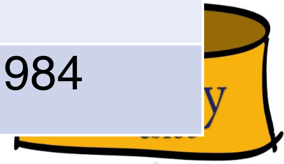
- "NULL" is used to indicate "ALL"

  SELECT item, color, SUM(number)
  FROM Sales
  GROUP BY CUBE (item, color)

- Any attr. may be replaced with NULL(ALL)

- May result in too many combinations

Item	Color	Number
Jacket	Blue	23
…	…	…
Jacket	Green	34
Jeans	Blue	28
…	…	…
Jeans	Green	17
Jacket	NULL	185
Jeans	NULL	200
…	…	…
NULL	Blue	94
NULL	Red	74
…	…	…
NULL	NULL	984



ROLLUP is an alternative to CUBE

ROLLUP targets a smaller # of combinations

SELECT item, color, size, SUM(number)
FROM Sales
GROUP BY ROLLUP (item), ROLLUP (color, size)

- Every combination of:
  - specific item or ALL for the first rollup
  - for the second rollup
    - specific color and specific size
    - specific color and ALL sizes
    - ALL colors and ALL sizes
- Thus, combinations include
  - {(item, color, size), (item, color), (item), (color, size), (color), ()}

Picking the right CUBE/ROLLUP query

- First, why does this matter?
  - If this query is being run once on a PB- sized warehouse, it is important to get it right!
  - Results usually materialized and used in dashboards, presentations, spreadsheets, ….

- Approach:
  - Think about all the ways you want to slice and dice your data
  - Pick granularity to recreate all aggregates you want, without blowing up the query result
    - Result size grows exponentially in the attrs; can be quite bad in large snowflake schemas
    - Known as the *curse of dimensionality*

So, why did we learn OLAP?

- OLAP is a specialization of DBMSs to support analytical proc. and report generation
  - Typically done in large “batch” ops on the entire DB
  - Rule of thumb: pick as “coarse-grained” query results as will allow you to construct all necessary cross-tabs
- Concepts of data cubes, hierarchies, slicing/dicing, and rollup/drill-down are valuable to describe what you’re doing when exploring your data
- Conveniences that come in SQL: ROLLUP and CUBE operators
- Next, systems that specialize for OLAP: column stores!