CS 61:
Database Systems

Data analytics/warehousing

Practice: Normalization

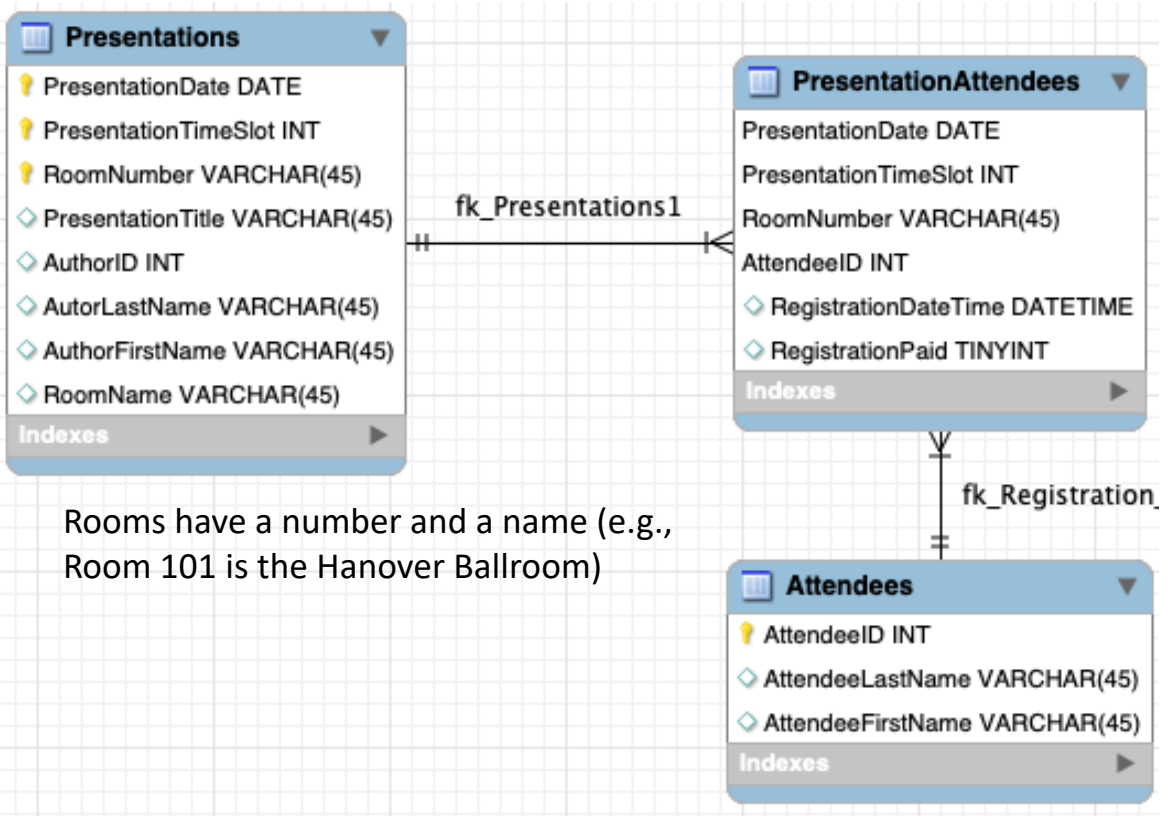
You run a computer science conference where authors present their work to groups of conference attendees

- Assume only one author gives a given presentation, but an author may give multiple presentations
- Presentations can be uniquely identified by the date, time slot, and room number
- Attendees can sign up for multiple presentations (but must pay for each separately)

A junior database administrator created this ERD for you. He says you don't need to worry about any dependencies in the Presentations table.

- Do you agree?
- What dependencies are present in that table?
- What would you change?

Download presentations.mwb from the course web page and make changes to bring the tables into 3NF



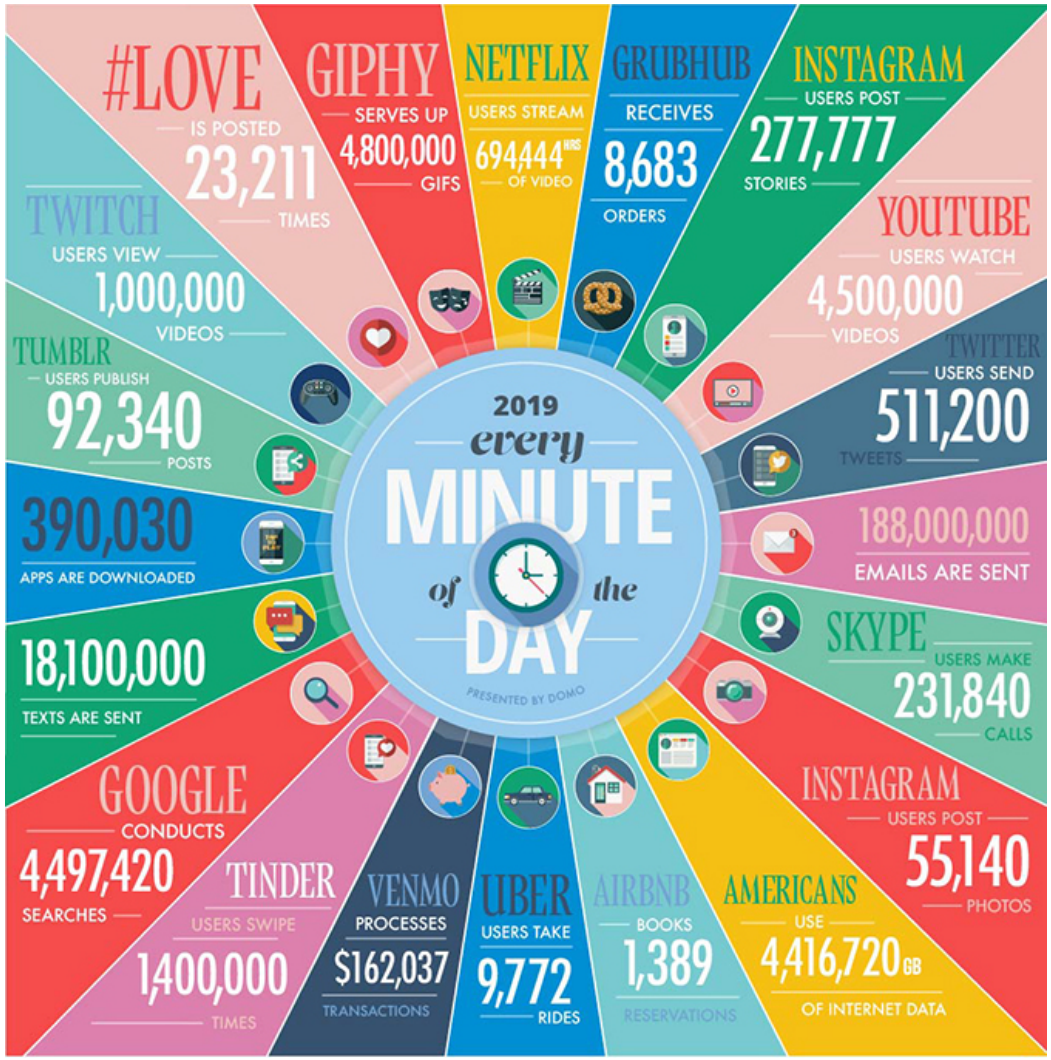
Adapted from Coronel and Morris

Agenda

1. Data warehousing/analytics
2. Excel vlookups, pivot tables
3. Rollup/Rank/top k queries

Today we collect lots of data...

Five V's of big data

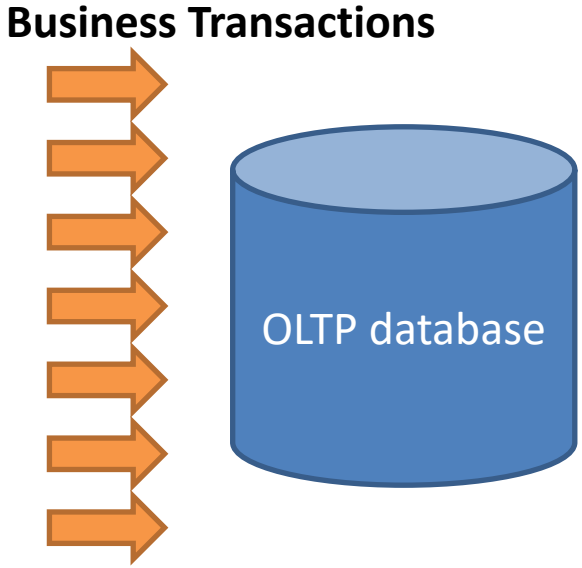


https://www.domo.com/learn/data-never-sleeps-7

Data characterized by five V's:

1. **Volume**: quantity of data to be stored, systems can be scaled
   - Vertically : "get a bigger box"
   - Horizontally: "get more boxes"
2. **Velocity**: speed at which data must be processed
   - Stream processing: analyze data as it comes
   - Feedback loop: data generates recommendations, recommendations lead to more data
3. **Variety**: store data in many forms
   - Structured data: fits into predefined data model
   - Unstructured data: does not fit data model
4. **Veracity**: can the data be trusted?
5. **Value**: can we exact value from the data, perhaps by correlating with other data?

We need tools to analyze this data for insight



Business intelligence queries can hamper transaction performance

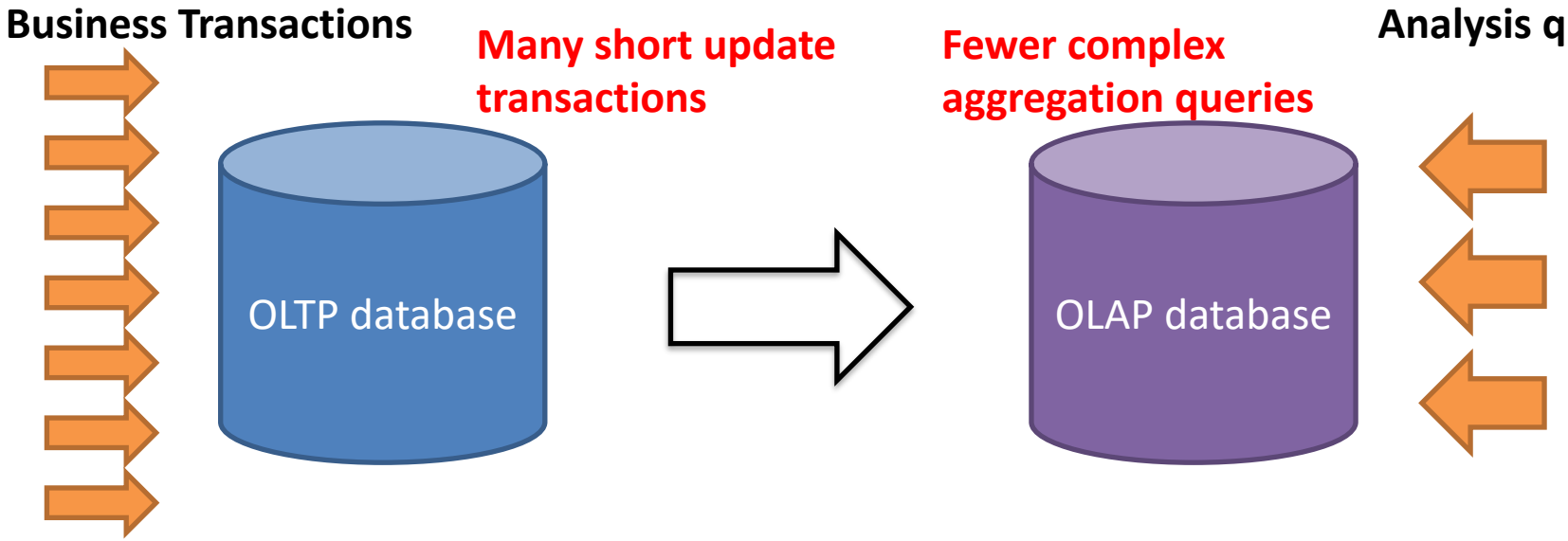
Operational data often not well suited for business analysis

Solution: create a separate database optimized for data analysis

Online Transaction Processing databases (OLTP)

- Our focus thus far
- Handles daily business operations
- Data often highly normalized
- Transactions mainly updates
- Speed is crucial!

We need tools to analyze this data for insight



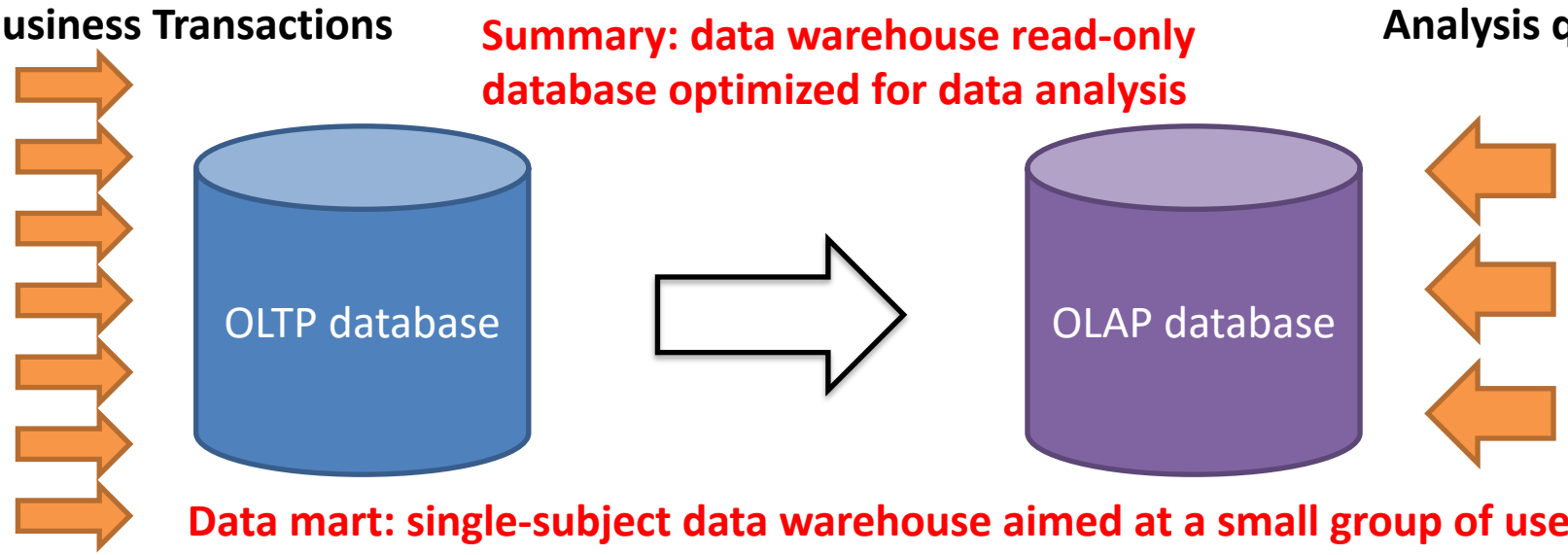
Online Transaction Processing databases (OLTP)

- Our focus thus far
- Handles daily business operations
- Data often highly normalized
- Transactions mainly updates
- Speed is crucial!

Online Analytical Processing databases (OLAP)

- Designed for analysis of "so what" of the data (get insight into data) to make decisions
- Contains summaries of data (e.g., product sales by year by region)
- Transactions mainly reads
- Speed less critical

We need tools to analyze this data for insight



Extract, Transform, Load (ETL) data from OLTP to OLAP database

- Extract data periodically from OLTP (and other sources) in a batch (how often?)
- Filter, integrate, and aggregate data (what level of aggregation?)
- Store data for easy business analysis (denormalize data! Yep, you read that right!)
- Data warehouse is an "integrated, subject-oriented, time-variant, nonvolatile" collection of data
  - Integrated – consolidate data from many sources
  - Subject-oriented – data optimized by topic such as sales, marketing, finance
  - Time-variant – represent the flow of data through time (even projected data)
  - Nonvolatile – data in warehouse not removed (or updated unless error)

Bill Inmon and Chuck Kelly, "The twelve rules of data warehouse for a client/server world", *Data Management Review*, May 1994

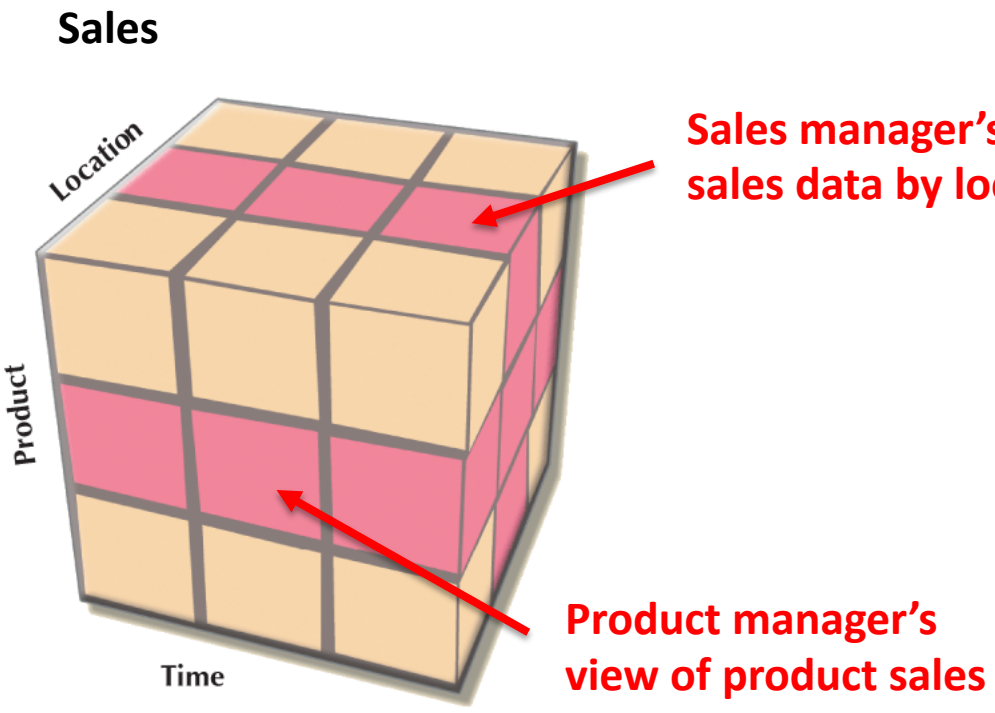
A data warehouse conforms to 12 rules

12 rules for a data warehouse

Rule	Description
1	The data warehouse and operational environments are separated
2	The data warehouse is integrated (data from multiple sources)
3	The data warehouse contains historical data over a long time
4	The data warehouse data is a snapshot captured at a given point in time
5	The data warehouse is subject oriented
6	The data warehouse data is mainly read-only with periodic batch updates
7	The data warehouse is data driven, operational database is process driven
8	The data warehouse contains data with several levels of detail (current/old, summarized at various levels)
9	The data warehouse is characterized by read-only queries of very large data sets
10	The data warehouse has a system that traces data sources, transforms, and storage
11	The data warehouse's metadata is critically important
12	The data warehouse enforces optimal use of the data by end users

Bill Inmon and Chuck Kelly, "The twelve rules of data warehouse for a client/server world", *Data Management Review*, May 1994

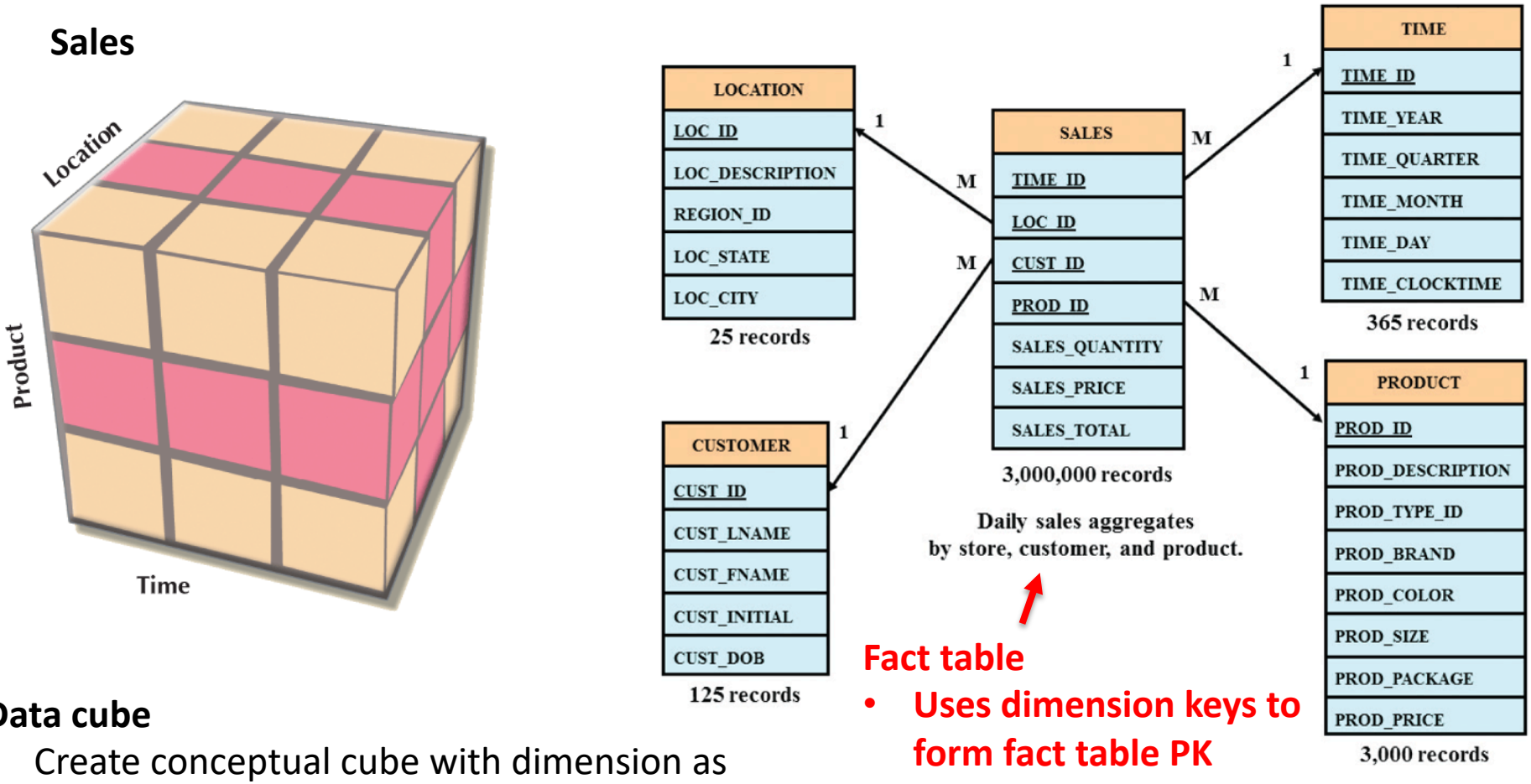
Data warehouses are often implemented using a Star Schema



Data cube

- Create conceptual cube with dimension as sides of cube
- Each cube element contains a fact (sales $)
- Allows rapid slicing and dicing
- Uses fact and dimension tables to store data

Data warehouses are often implemented using a Star Schema



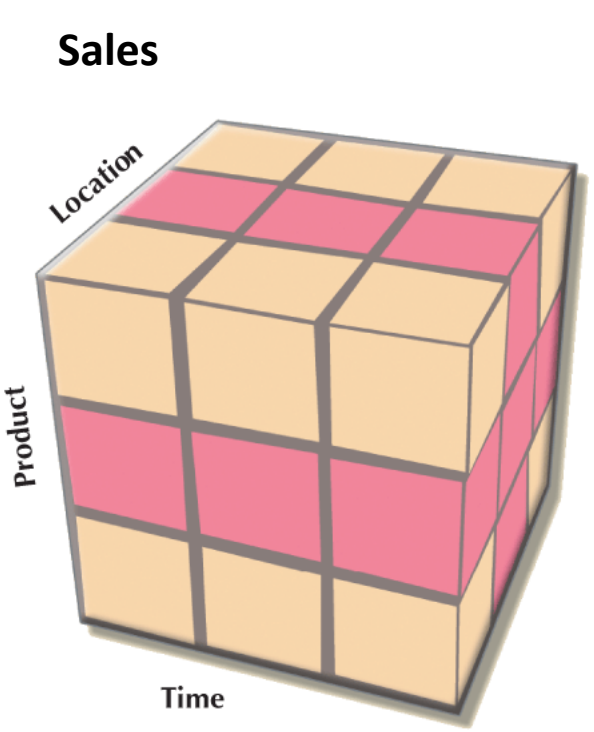
Data cube

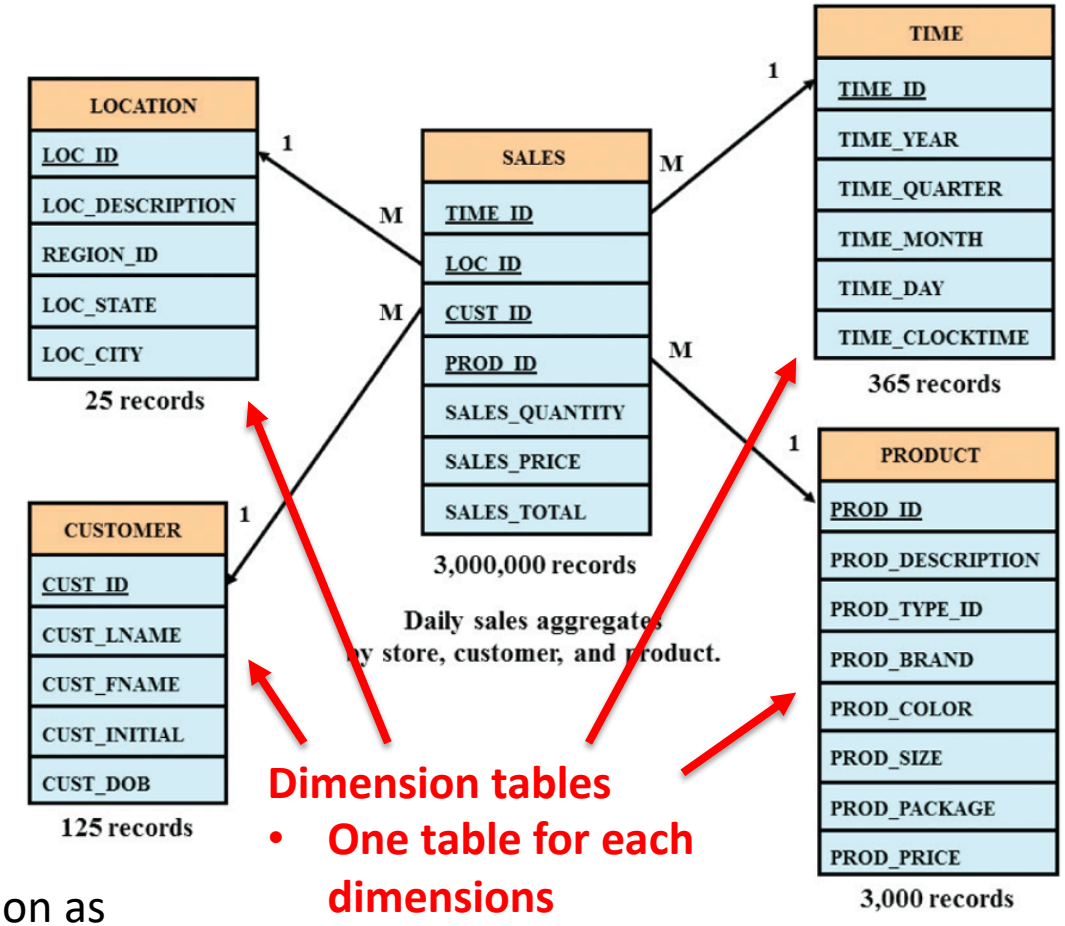
- Create conceptual cube with dimension as sides of cube
- Each cube element contains a fact (sales $)
- Allows rapid slicing and dicing
- Uses fact and dimension tables to store data

Fact table

- **Uses dimension keys to form fact table PK**
- **Denormalized data (same data stored many times)**
- **May have multiple attributes**

Data warehouses are often implemented using a Star Schema





Data cube

- Create conceptual cube with dimension as sides of cube
- Each cube element contains a fact (sales $)
- Allows rapid slicing and dicing
- Uses fact and dimension tables to store data

Dimension tables

- **One table for each dimensions**
- **Keys form PK on fact table**
- **Each table normalized with attributes for dimension**

Alternative is one *large* table

Agenda

1. Data warehousing/analytics
2. Excel vlookups, pivot tables
3. Rollup/Rank/top k queries

Excel pivot table tutorial

- Relative vs. absolute references
- Download csv file of Restaurants and Cuisine tables
- Create VLOOKUP for CuisineID
- Create pivot table over data
  - Filter by Boro
  - Rows: Cuisine
  - Sum InspectionCount
  - Sort by InspectionCount

Practice

Given data from day17.xlsx

Using Excel, create a pivot table to answer:

- What were the value of pens sold in the Southern region in 2016
- What was the value of pens sold by Victor in all years
- How did Victor's sales break down by region?

day17.xlsx

Year	Region	Agent	Product	Value
2016	East	Carlos	Erasers	50
2016	East	Tere	Erasers	12
2016	North	Carlos	Paper	120
2016	North	Tere	Paper	100
2016	North	Carlos	Paper	30
2016	South	Victor	Pens	145
2016	South	Victor	Pens	34
2016	South	Victor	Pens	80
2016	West	Mary	Pencils	89
2016	West	Mary	Pencils	56
2017	East	Carlos	Pencils	45
2017	East	Victor	Pens	55
2017	North	Mary	Pencils	60
2017	North	Victor	Erasers	20
2017	South	Carlos	Paper	30
2017	South	Mary	Paper	75
2017	South	Mary	Paper	50
2017	South	Tere	Pens	70
2017	South	Tere	Erasers	90
2017	West	Carlos	Paper	25
2017	West	Tere	Pens	100

Practice

Given data from day17.xlsx
Using Excel, create a pivot table to answer:

- What were the value of pens sold in the Southern region in 2016
- What was the value of pens sold by Victor in all years
- How did Victor's sales break down by region?

After you've answered those questions, create the pivot table shown below

Sum of Value	Column Labels					
Row Labels	East	North	South	West	(blank)	Grand Total
Carlos	**95**	**150**	**30**	**25**		**300**
Erasers	50					50
Paper		150	30	25		205
Pencils	45					45
Mary		**60**	**125**	**145**		**330**
Paper			125			125
Pencils		60		145		205
Tere	**12**	**100**	**160**	**100**		**372**
Erasers	12		90			102
Paper		100				100
Pens			70	100		170
Victor	**55**	**20**	**259**			**334**
Erasers		20				20
Pens	55		259			314
(blank)						
(blank)						
Grand Total	**162**	**330**	**574**	**270**		**1336**

day17.xlsx

Year	Region	Agent	Product	Value
2016	East	Carlos	Erasers	50
2016	East	Tere	Erasers	12
2016	North	Carlos	Paper	120
2016	North	Tere	Paper	100
2016	North	Carlos	Paper	30
2016	South	Victor	Pens	145
2016	South	Victor	Pens	34
2016	South	Victor	Pens	80
2016	West	Mary	Pencils	89
2016	West	Mary	Pencils	56
2017	East	Carlos	Pencils	45
2017	East	Victor	Pens	55
2017	North	Mary	Pencils	60
2017	North	Victor	Erasers	20
2017	South	Carlos	Paper	30
2017	South	Mary	Paper	75
2017	South	Mary	Paper	50
2017	South	Tere	Pens	70
2017	South	Tere	Erasers	90
2017	West	Carlos	Paper	25
2017	West	Tere	Pens	100

Agenda

1. Data warehousing/analytics
2. Excel vlookups, pivot tables
3. Rollup/Rank/top k queries

We have previously seen how to use GROUP BY to aggregate data

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

Can use group by to get sales per product line

```
SELECT
  productline,
  SUM(orderValue) AS totalOrderValue
FROM  sales
GROUP BY productline;
```

productline	totalOrderValue
Vintage Cars	12245.78
Classic Cars	19668.13
Trucks and Buses	14194.95
Trains	9021.03
Ships	13147.86
Planes	11700.79
Motorcycles	9044.15

No total line of all sales, just sales by product line

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

You can add a summary row by using UNION

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

UNION adds new *rows* to result (JOINs adds new columns)

```
SELECT
  productline,
  SUM(orderValue) totalOrderValue
FROM sales
GROUP BY productline
UNION ALL
SELECT
  NULL,
  SUM(orderValue) totalOrderValue
FROM sales;
```

UNION ALL returns duplicate rows (UNION DISTINCT) does not

Must have same number of columns with compatible data types

SQL has an easier way to add the summary row using ROLLUP

productline	totalOrderValue
Vintage Cars	12245.78
Classic Cars	19668.13
Trucks and Buses	14194.95
Trains	9021.03
Ships	13147.86
Planes	11700.79
Motorcycles	9044.15
NULL	89022.69

- **UNION returns new row with total**
- **Note: NULL for productline in second SELECT**

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

ROLLUP can be used similarly to create subtotals based on grouping

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

ROLLUP creates total

```
SELECT
  productLine,
  SUM(orderValue) totalOrderValue
FROM  sales
GROUP BY productline WITH ROLLUP;
```

WITH ROLLUP adds extra row with totals for grouped by attributes like UNION did

productLine	totalOrderValue
Classic Cars	19668.13
Motorcycles	9044.15
Planes	11700.79
Ships	13147.86
Trains	9021.03
Trucks and Buses	14194.95
Vintage Cars	12245.78
NULL	89022.69

Now have total for all sales in a row called a super-aggregate

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

ROLLUP can operate over multiple columns

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

Can roll up multiple attributes

```
SELECT
  productLine,
  orderYear,
  SUM(orderValue) totalOrderValue
FROM sales
GROUP BY productline, orderYear WITH ROLLUP;
```

productLine	orderYear	totalOrderValue
Classic Cars	2003	5571.80
Classic Cars	2004	8124.98
Classic Cars	2005	5971.35
Classic Cars	NULL	19668.13
Motorcycles	2003	2440.50
Motorcycles	2004	2598.77
Motorcycles	2005	4004.88
Motorcycles	NULL	9044.15
Planes	2003	4825.44
Planes	2004	2857.35
Planes	2005	4018.00
Planes	NULL	11700.79
Ships	2003	5072.71
Ships	2004	4301.15
Ships	2005	3774.00
Ships	NULL	13147.86
Trains	2003	2770.95
Trains	2004	4646.88
Trains	2005	1603.20
Trains	NULL	9021.03
Trucks and Buses	2003	3284.28
Trucks and Buses	2004	4615.64
Trucks and Buses	2005	6295.03
Trucks and Buses	NULL	14194.95
Vintage Cars	2003	4080.00
Vintage Cars	2004	2819.28
Vintage Cars	2005	5346.50
Vintage Cars	NULL	12245.78
NULL	NULL	89022.69

Now have super-aggregate row by product line

Hierarchy determined by GROUP BY order (product line first)

Also have grand total over all super-aggregate rows

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

ROLLUP can operate over multiple columns

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

Can roll up multiple attributes

```
SELECT
  orderYear,
  productLine,
  SUM(orderValue) totalOrderValue
FROM sales
GROUP BY orderYear,productline WITH ROLLUP;
```

GROUP BY reversed

orderYear	productLine	totalOrderValue
2003	Classic Cars	5571.80
2003	Motorcycles	2440.50
2003	Planes	4825.44
2003	Ships	5072.71
2003	Trains	2770.95
2003	Trucks and Buses	3284.28
2003	Vintage Cars	4080.00
2003	NULL	28045.68
2004	Classic Cars	8124.98
2004	Motorcycles	2598.77
2004	Planes	2857.35
2004	Ships	4301.15
2004	Trains	4646.88
2004	Trucks and Buses	4615.64
2004	Vintage Cars	2819.28
2004	NULL	29964.05
2005	Classic Cars	5971.35
2005	Motorcycles	4004.88
2005	Planes	4018.00
2005	Ships	3774.00
2005	Trains	1603.20
2005	Trucks and Buses	6295.03
2005	Vintage Cars	5346.50
2005	NULL	31012.96
NULL	NULL	89022.69

Grouping order reversed (orderYear first)

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

GROUPING can give super-aggregate rows a meaningful label

Given sales table

productLine	orderYear	orderValue
Vintage Cars	2003	4080.00
Classic Cars	2003	5571.80
Trucks and Buses	2003	3284.28
Trains	2003	2770.95
Ships	2003	5072.71
Planes	2003	4825.44
Motorcycles	2003	2440.50
Classic Cars	2004	8124.98
Vintage Cars	2004	2819.28
Trains	2004	4646.88
Ships	2004	4301.15
Planes	2004	2857.35
Motorcycles	2004	2598.77
Trucks and Buses	2004	4615.64
Motorcycles	2005	4004.88
Classic Cars	2005	5971.35
Vintage Cars	2005	5346.50
Trucks and Buses	2005	6295.03
Trains	2005	1603.20
Ships	2005	3774.00
Planes	2005	4018.00

GROUPING returns 1 if super-aggregate row, 0 otherwise

```
SELECT IF(GROUPING(orderYear),'All Years',orderYear)AS orderYear,
IF(GROUPING(productLine),'All Products',productLine)AS productLine,
SUM(orderValue)AS totalOrderValue
FROM  sales
GROUP BY orderYear , productline WITH ROLLUP;
```

orderYear	productLine	totalOrderValue
2003	Classic Cars	5571.80
2003	Motorcycles	2440.50
2003	Planes	4825.44
2003	Ships	5072.71
2003	Trains	2770.95
2003	Trucks and Buses	3284.28
2003	Vintage Cars	4080.00
2003	All Product Lines	28045.68
2004	Classic Cars	8124.98
2004	Motorcycles	2598.77
2004	Planes	2857.35
2004	Ships	4301.15
2004	Trains	4646.88
2004	Trucks and Buses	4615.64
2004	Vintage Cars	2819.28
2004	All Product Lines	29964.05
2005	Classic Cars	5971.35
2005	Motorcycles	4004.88
2005	Planes	4018.00
2005	Ships	3774.00
2005	Trains	1603.20
2005	Trucks and Buses	6295.03
2005	Vintage Cars	5346.50
2005	All Product Lines	31012.96
All Years	All Product Lines	89022.69

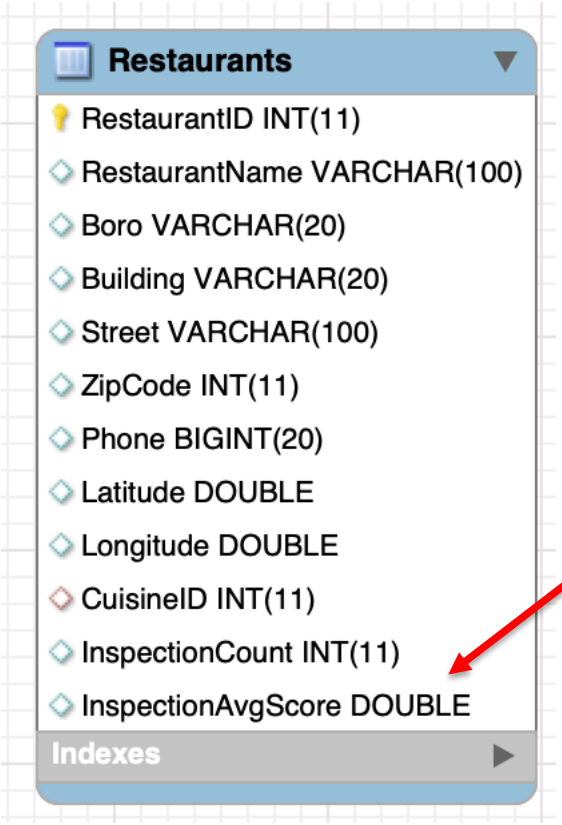
Remember how IF works: first value if true, otherwise second value

Super-aggregate rows now have reasonable names (not just NULL)

Adapted from: https://www.mysqltutorial.org/mysql-rollup/

Practice

use nyc_inspections;



Reminder:
Restaurants table has columns for how many times each restaurant has been inspected and its average score

Practice

use nyc_inspections;

Create a rollup with a count of the number of inspections by Boro and by Cuisine type (e.g., 2,103 inspections of American cuisine restaurants in the Bronx)

- Fill in ROLLUP Nulls with 'All boros' and ''All cuisines' using IF and GROUPING
- Format your count to have commas at thousands (e.g., 1,234)
- Make sure your super-aggregate rows come at the bottom of your groups (e.g., the total count of inspections in the Bronx come at the end of the Bronx rows)
- Output should look like:

Note: a few restaurants have a Boro of 0

Boro	Cuisine	Inspections
0	American	21
0	Hawaiian	5
0	Scandinavian	6
0	All Cuisines	32
Bronx	African	175
Bronx	American	2,103
Bronx	Armenian	3

RANK assigns an increasing number to each row returned

RANK

Sometimes you want to assign a numerical value to rows to indicate their rank (e.g., first row has rank 1, second row has rank 2, …)

- RANK() assigns a rank to each row within the partition of a result set
- The rank of a row is specified by one plus the number of ranks that come before it

Format:

SELECT RANK() **OVER** (
  PARTITION BY A_1 [,A_2,…A_n]
  ORDER BY <expression> [ASC|DESC], [{,<expression>...}]) **AS** RankName

PARTITION BY works like GROUP BY – splits results into groups based on attribute listed

Can have more than one partition (partition by cuisine type, then boro for example)

Sort each partition by ORDER BY

- **Rank numbering starts at 1 for each partition**
- **If tie on partition, all tying rows get same rank (e.g., if three row tie for first, all three get rank of 1, next row gets rank of 4)**
- **Use ROW_NUMBER() instead of RANK() to ensure no gaps between rank values assigned (e.g., first three ties get rank 1 though three, next row still gets rank of 4)**

Adapted from: https://www.mysqltutorial.org/mysql-window-functions/mysql-rank-function/

RANK assigns an increasing number to each row returned

RANK

PARTITION BY (works like GROUP BY) by Boro and sorted by average inspection score ascending (default)

Example:

```
SELECT
    RANK() OVER (PARTITION BY Boro ORDER BY InspectionAvgScore) AS `Rank`,
    RestaurantID, RestaurantName, Boro, InspectionAvgScore
FROM Restaurants
WHERE CuisineID = 83; -- only fruits/veg
```

Note: Tim's and Ono Bowls tie for second in Manhattan boro, so both get rank of 2
Juke Box Juice gets rank 4 (not 3) due to tie

Rank	RestaurantID	RestaurantName	Boro	InspectionAvgScore
1	50011980	NEWKIRK FRUIT	Brooklyn	13
1	50048030	JAMBA JUICE	Manhattan	5.666666666
2	1111	Tim's Tasty Treats	Manhattan	8
2	50098776	ONO BOWLS	Manhattan	8
4	41705768	JUKE BOX JUICE & SALAD	Manhattan	12.4
1	50087753	DJ CLUB INC KTV & LOU…	Queens	15
1	41516689	THE JUICE BAR	Staten Island	17.9

Use WITH and LIMIT to get top *k* results

RANK

Will limit to top 2 restaurants

Example:

```
SET @k = 2; -- return top k=2
WITH RestaurantRanks AS (
    SELECT RANK() OVER (ORDER BY InspectionAvgScore ) AS `Rank`,
        RestaurantID, RestaurantName, Boro, InspectionAvgScore
    FROM Restaurants WHERE CuisineID = 83) – only fruits/vegetable restaurants
SELECT * FROM RestaurantRanks WHERE `Rank`<= @k; -- top 2
```

Note: PARTITION BY is optional, if omitted, use all rows (here all boros)

WITH created temporary table, SELECT from that on RANK to get top k results

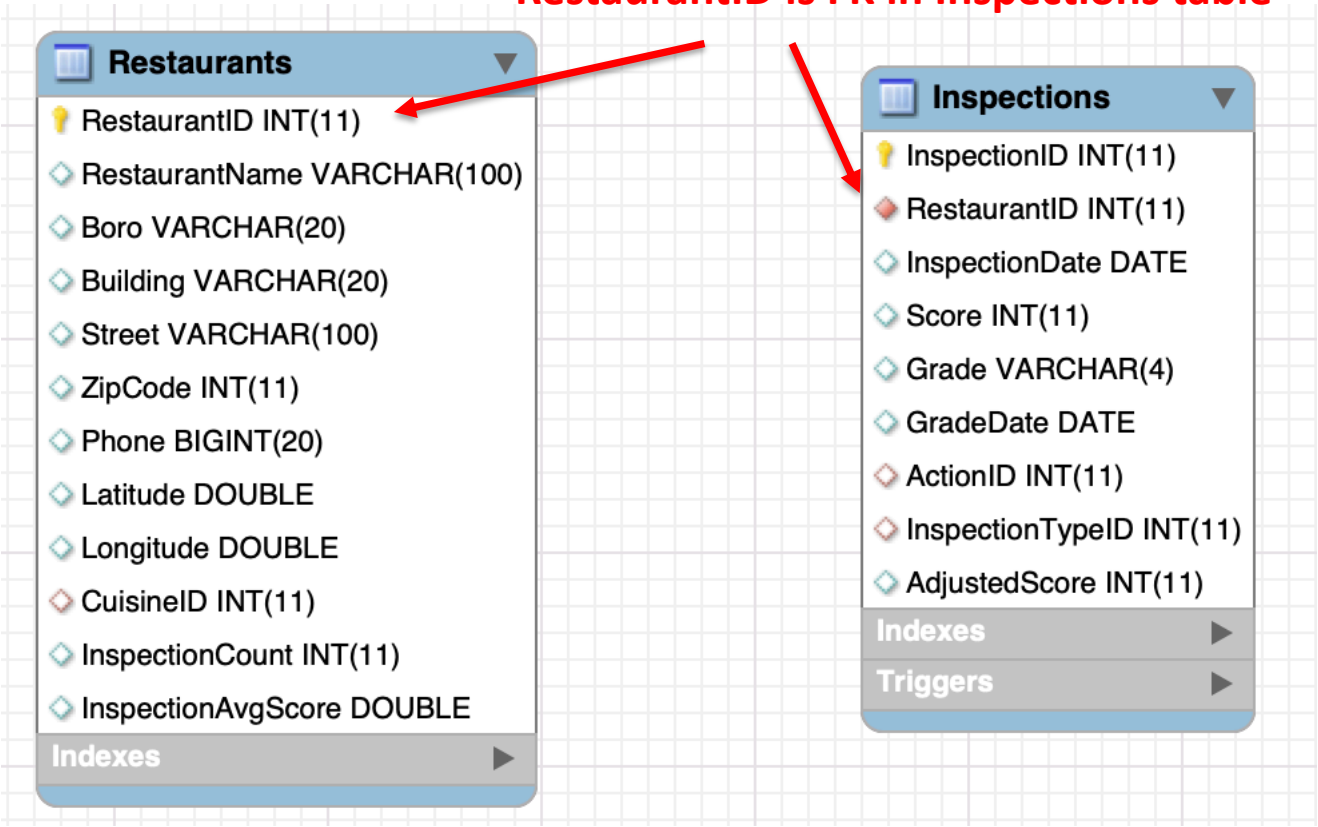
Rank	RestaurantID	RestaurantName	Boro	InspectionAvgScore
1	50048030	JAMBA JUICE	Manhattan	5.666666666
2	1111	Tim's Tasty Treats	Manhattan	8
2	50098776	ONO BOWLS	Manhattan	8

Here the top results happen to be in Manhattan (could have been other boros)
Also, note that this returned 3 restaurants due to tie, how could you force only 2?

Practice

use nyc_inspections;

Reminder:
There is an entry in the Inspections table each time a restaurant was inspected
RestaurantID is FK in Inspections table



Practice

use nyc_inspections;

1. Update InspectionCount and InspectionAvgScore in Restaurants table using data from Inspections table. (Hint: Use UPDATE on both columns)
2. Insert a new Inspection for Tim's Tasty Treats with a score of 6 (other values can be Null) and confirm triggers updated count to 2 and avg score to 8
3. Select all Fruits/Vegetables restaurants (there should be 7 of them including Tim's)
4. Rank all Fruits/Vegetables restaurants by best average inspection score (lowest inspection score is best), return rank without ties
5. List the restaurants with Rank <=2 for all cuisine types in the Manhattan Boro (e.g., top two ranked Italian/Pizza shops, top two ranked American). Only return two per Boro and do not consider restaurants that have not been inspected

Practice

use nyc_data;

1. Create a stored procedure that takes the boro and number of restaurants k as parameters and returns the top k restaurants of each cuisine type in the given boro based on average inspection score

2. Create the same query, but return your data as JSON