On-Line Analytical Processing (OLAP)

Warehouses
Data Cubes
Outer Join

Instructor: Shel Finkelstein

Reference:

A First Course in Database Systems, 3rd edition, Sections 5.2, 6.3.8, 10.6 and 10.7

Important Notices

- Lecture slides and other class information will be posted on Piazza (not Canvas).
  - Slides are posted under Resources→Lectures
  - Lecture Capture recordings are available to all students under Yuja.
    - That includes classes given over Zoom.
  - There's no Lecture Capture for Lab Sections.
- All Lab Sections (and Office Hours) normally meet In-Person, with rare exceptions announced on Piazza.
  - Some slides for Lab Sections have been posted on Piazza under Resources→Lab Section Notes
- Office Hours for TAs and me are posted in Syllabus and Lecture1, as well as in Piazza notice @7; that post also now includes hours for Group Tutors.
- Some suggestions about use (and non-use) of Generative AI systems such as ChatGPT were posted on Piazza in @41, with specific discussion about the Movies tables and the four Avatar queries in Lecture 4.

Important Notices

- The Fifth Gradiance Assignment, CSE 180 Winter 2024 #5, on Design Theory, was assigned on Friday, March 8.
  - It is due by 11:59pm on **Friday, March 15**, the last day of class.
- Explanation of two difficult Gradiance questions on transactions was posted on Piazza on Sunday, March 2 in @122.
- Answers for two of the "Practice" Relational Algebra queries (in magenta) were posted on Piazza on Sunday, March 2 in @126.
- Winter 2024 Student Experience of Teaching Surveys - SETs opened on Monday, March 4.
  - SETs close on Sunday March 17 at 11:59pm.
  - Instructors are not able to identify individual responses.
  - Constructive responses help improve future courses.

Important Notices

- Lab4 was due on **Tuesday, March 12 by 11:59pm**.
  - Our Lab4 solution was posted on Piazza on Wednesday, March 13.
- Lab Sections, Office Hours and Tutoring continue throughout the last week of classes, even though Lab Assignments are complete.
  - Topics may include Lab4 solution, Gradiance #5, the W23 CSE 180 Final, and recent Lecture material.
- Plan for lecture on last day of classes, Friday, March 15
  - Will complete Lecture 14 (OLAP).
  - Will go over the two "Practice" Relational Algebra queries (Lecture 9, in magenta) that were posted on Piazza in @126.
  - Will review discussion of BCNF and 3NF from Lecture 13.
  - May begin Lecture 15 (Non-First Normal Form), but new material in Lecture 15 will not be included on the CSE 180 Final.

Important Notices: Final

- CSE 180 Final is on **Wednesday, March 20, 8:00-11:00am, and it will be given in-person in our classroom, except for students who receive Remote Exam permission.**
  - **No** early/late Exams. **No** make-up Exams. **No** devices (except for Remote students).
  - 3 hours, extended for DRC students, covering the entire quarter.
    - All DRC students should have recently received email about the final.
    - Final will be harder than the Midterm.
  - **You may bring one double-sided 8.5 x 11 sheet to the Final, with anything that you want written or printed on it that you can read unassisted) …**
    - (Okay, you may bring two sheets with writing only on one side of each sheet.)
    - **But you must not use any other material or receive any help during the Final, whether you're in the classroom, remote, or in a DRC room**
    - As the Syllabus emphasizes, Academic Integrity violations have serious consequences!
  - We will assign seats as you enter the room. You may not choose your own seat.
- CSE 180 Final from Winter 2023 was posted on Piazza under Resources→Exams on Sunday, March 3.
  - Solution to that Final was posted on Piazza on Sunday, March 10 … but take it yourself first, rather than just reading the solution.

Important Notices: In-Person Final

Final includes a Multiple Choice Section and a Longer Answers Section.

- In-Person students should bring a Red Scantron sheet (ParSCORE form number f-1712) sold at Bookstore for about 25 cents, and #2 pencils for Multiple Choice Section.
- You'll answer Multiple Choice Section on your Scantron sheet, entering your name, your student ID, and which version ("Test Form Letter") of the Multiple Choice Section you're answering.
  - Write your Multiple Choice answers on the Scantron sheet using a #2 pencil.
  - Ink and #3 pencils don't work.
- You'll answer Longer Answers Section on the Long Answers Section, using either ink or #2 pencil (as on the Midterm).
- Be sure to answer all questions readably!!
- Do not hand in your 8.5 x 11 sheet or your Multiple Choice Section. Just hand in your Scantron Sheet and your Longer Answers Section.
  - But show us your Multiple Choice Section, so that we can check that you filled in the Test Form Letter (A, B, C or D) for that Section on your Scantron Sheet.

Important Notices: Remote Final

CSE 180 Final **will be given in-person in our classroom, except for students who receive Remote Exam permission.**

- If you need to take the Final Remotely, send me an email whose subject is "Taking the CSE 180 Final Remotely" by Tuesday, March 19 at 6:00pm that includes your strong justification for that request.
  - Only students whose requests are approved may take the Final Remotely.
    - If you don't receive a response from me by 9:00pm, please send email again!
- I'll send instructions to you for taking the Final before 8:00am on Wednesday, March 20 which include a Zoom link
  - You must be on Zoom while you are taking the Final.
    - Please make sure that your face is visible on Zoom video, but that your microphone is muted.
    - Please do not have headphones on during the Final.
    - If there are any bugs on the Final, they will be conveyed to all Remote students via Zoom Chat.
    - If you have a question during the Final, please send it to me (just to me) directly using Zoom Chat, not by speaking.
  - You'll have to complete the exam by 11:00am, just like all other students …
  - … except for DRC students, who will receive extra time, whether they take the exam In-Person (in a different room) or Remotely.
  - You won't need a red Scantron Sheet if you're taking the Final Exam Remotely.

A Word to the Unwise

- This is a tough class for some students since it involves a combination of theory and practice.
  - The second half of CSE 180 is much harder than the first half of the course.
- Students who regularly attend Lectures and Lab Sections (and Office Hours and Tutoring) often do well; students who don't regularly attend often do poorly.
  - We don't take attendance; you're responsible for your own choices.
- After the course ends, your course grade will be determined by your scores on Exams, Labs and Gradiance, as described on the “Course Evaluation” and “Grading” Slides in the Syllabus.
  - You won't be able to do any additional work to improve your grade.

Overview

- Originally, database systems were tuned to many, small, simple queries (OLTP).
- Many applications use fewer, more time-consuming, *analytical* queries (OLAP).
- New architectures were developed to handle analytical queries efficiently.

Data Warehouses

- One common approach to data integration of multiple data sources
  - Copy sources into a single DB (*warehouse*), and try to keep data reasonably up-to-date.
  - Methods:
    - Periodic reconstruction of the warehouse, perhaps overnight
    - Periodic incremental update of warehouses
    - "Continuous" incremental update of warehouse
- Warehouses are frequently used for analytical queries.
  - Alternative approach: Leave data in separate data sources, and execute "mediated" query across those sources
  - Advantages and disadvantages of Warehouse vs. Mediation?

OLTP

- Database operations we've discussed before involve *On-Line Transaction Processing* (OLTP).
  - Short, simple, frequent queries and/or modifications, each involving a relatively small number of tuples.
  - Examples: Answering queries from a Web interface, sales at cash registers, selling airline tickets.

OLAP

- *On-Line Analytical Processing* (OLAP, or "analytic") queries are different from OLTP.
  - Fewer but more complex queries, which may take minutes to execute.
  - Databases may be quite large—terabytes is very common, but warehouses may have petabytes, exabytes or more.
  - Sometimes, it's okay to run queries on data this is not fully up-to-date.
    - Why/when is it okay to use data that is not absolutely current, or data that is incomplete?

From Bytes to Yottabytes

Multiples of bytes V · T · E				
SI decimal prefixes		Binary usage	IEC binary prefixes	
Name (Symbol)	Value		Name (Symbol)	Value
kilobyte (kB)	10^3	2^{10}	kibibyte (KiB)	2^{10}
megabyte (MB)	10^6	2^{20}	mebibyte (MiB)	2^{20}
gigabyte (GB)	10^9	2^{30}	gibibyte (GiB)	2^{30}
terabyte (TB)	10^{12}	2^{40}	tebibyte (TiB)	2^{40}
petabyte (PB)	10^{15}	2^{50}	pebibyte (PiB)	2^{50}
exabyte (EB)	10^{18}	2^{60}	exbibyte (EiB)	2^{60}
zettabyte (ZB)	10^{21}	2^{70}	zebibyte (ZiB)	2^{70}
yottabyte (YB)	10^{24}	2^{80}	yobibyte (YiB)	2^{80}
See also: Multiples of bits · Orders of magnitude of data				

OLAP Examples

1. Amazon analyzes purchases by its customers to come up with a personalized screen listing products that are likely of interest to the customer.
2. Analysts at Wal-Mart look for items whose sales in some region are increasing.
   - Send trucks to move merchandise between stores.
3. Google identified advertising “segments” (categories) of population, and displays advertisements based on individual’s segment, as well as personal history.
4. Summary reports of product sales by Consumer Goods companies may be created monthly/weekly/daily/hourly …
   - Automatically report trends and anomalies and react to them

Common Architecture-1

1-Before Cloud Computing:

- Database systems at store branches handle OLTP.
- Local store databases are copied periodically to a Data Warehouse.
  - … or more likely the Warehouse is incrementally updated, with only the changed data copied.
  - Analysts use the Warehouse for OLAP.
- Older data may be archived.
  - But even “Cold” data is kept for a long time, possibly forever. (Why?)

Common Architecture-2

2-With Cloud Computing:

- Data systems for store branches may be stored in a cloud, using database schema suitable for OLTP.
  - May be in one or more databases.
  - Current OLTP data may be used for decision support.
- Local store databases are copied periodically to a Data Warehouse that is also stored in the Cloud.
  - … or more likely the Warehouse is incrementally updated, with only the changed data copied.
  - Analysts use the Warehouse for OLAP.
  - Older data may be archived.
- There also are systems for HTAP (Hybrid Transactional/Analytical Processing), a combination of OLTP and OLAP.
  - Can do this both with/without Cloud.

Some Modern Architectures

There are modern system approaches that combine capabilities for:

- OLTP transactions, such as Banking transactions and Order Entry
- OLAP analytics, which we'll discuss in this lecture
  - Data Science analytics, such as customer categorization
- Streaming data (new events, such as sensor data and stock quotes)

Some of these approaches involved multiple copies of data for availability and for performance.

- Some systems use row-based representation for OLTP and column-based representation for OLAP.
- Physical-independence means that applications are correct no matter what the physical data representation is!
  - But physical representation does affect performance.

Increasingly, data is stored in the Cloud, often in files rather than in databases.

- "Data Lakes" running in the Cloud can store data in its raw, unstructured, uncleaned form, perhaps transforming and cleaning it later.
- Cloud is dominating, although some data is still stored in "on-premises" systems.
  - "On-premises" means managed directly/indirectly by company, outside cloud.

Star Schemas

- A *Star Schema* is a common organization for data in a Warehouse. A Star Schema consists of Dimension Tables and a Fact Table.
  1. *Dimension Tables*: Smaller, largely static (unchanging) information describing the data that's in the Facts.
     - Examples: Product, Customer, Store
  2. *Fact Table*: A very large accumulation of Facts, such as Sales.
     - A Fact gives the Sales of a specific product to a specific customer in a specific store on a specific date.
     - The key for a Fact Table consists of values from its Dimension Tables (foreign keys).
     - Facts may be "insert-mostly", with some updates.

Example: Star Schema

- Suppose that we want to record the information about every beer sale:
  - the bar,
  - the brand of beer,
  - the drinker who bought the beer,
  - the day of the purchase, and
  - the price charged

- The Fact Table is a relation:

  Sales(bar, beer, drinker, day, price)

Example -- Continued

- The Dimension Tables provide information about the bar, beer, and drinker "dimensions":

  Bars(bar, addr, license)

  Beers(beer, manf)

  Drinkers(drinker, addr, phone)

  Days(month, daynumber, year)

Example: Star Schema (modified)

- Suppose that we want to record the information about every beer sale:
  - the bar,
  - the brand of beer,
  - the drinker who bought the beer,
  - the day of the purchase, and
  - the price charged

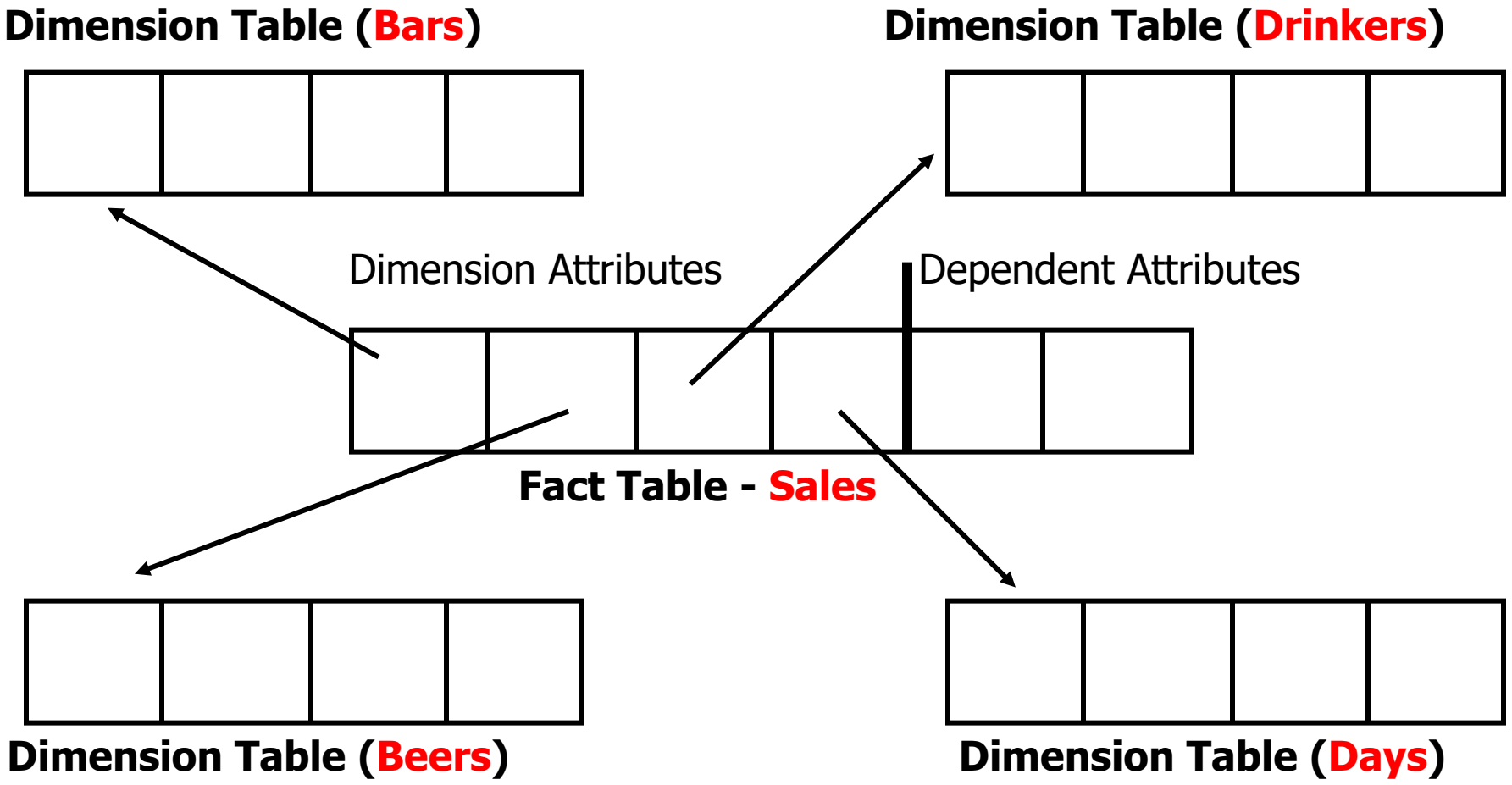
- The Fact Table is a relation:

  Sales(bar, beer, drinker, day, price)

  - Since primary key of Days is month, daynumber, year, should be:

    Sales(bar, beer, drinker, month, daynumber, year, price)

Visualization – Star Schema



Dimension and Dependent Attributes

- Two classes of Fact Table attributes:
  1. *Dimension Attribute*: the key of a dimension table.
  2. *Dependent Attribute*: a fact value determined by the dimension attributes of the tuple.
     - The sales info (e.g., price, quantity, salesperson) for a specific product to a specific customer at a specific store
     - The price of a specific beer purchased by a specific drinker at a specific bar on a specific day

Dimension and Dependent Attributes and Fact Table

The key of a Fact Table is the combination of the keys of its dimensions.

- Values of dimension attributes in any Fact must match dimension attribute values in the Dimension Tables.
  - Fact Table has Foreign Key constraint for each Dimension Table!
- But there don't have to be Facts for every combination of Dimension Table values.

Example:

- If there's a Fact saying that:

  George bought Bud at Joe's Bar on Jan 23, 2015 (at some price),

  then those green values must be in corresponding Dimension Tables.
- But even though those values are in the Dimension Tables, there doesn't have to be a Fact for them.

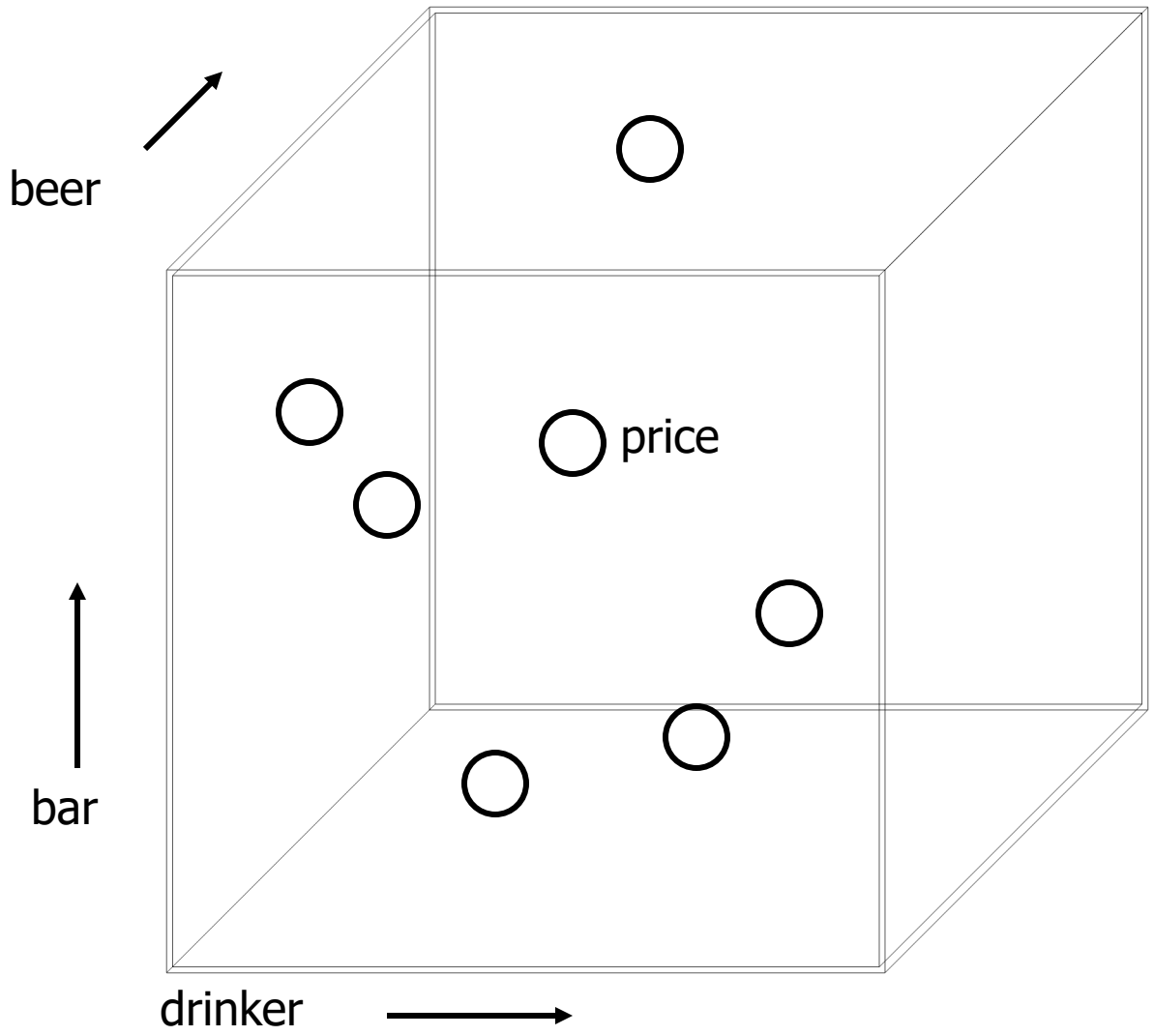
Example: Dependent Attribute

- price is a dependent attribute in our example Sales relation.
- That attribute is determined by the combination of dimension attributes: bar, beer, drinker and day.
- Time is sometimes treated as a dimension (specific month, day or hour) and sometimes treated as a dependent attribute.
  - For example, if you're recording specific second/msec, you're probably treating time as a dependent attribute, not as a dimension.

Data Cubes

- Keys of dimension tables are the dimensions of a hypercube.
  - Example: For the beer Sales data, the four dimensions are bar, beer, drinker and day.
- Dependent attributes (e.g., price and quantity) appear as labels for points in the cube.

Visualization -- Data Cubes

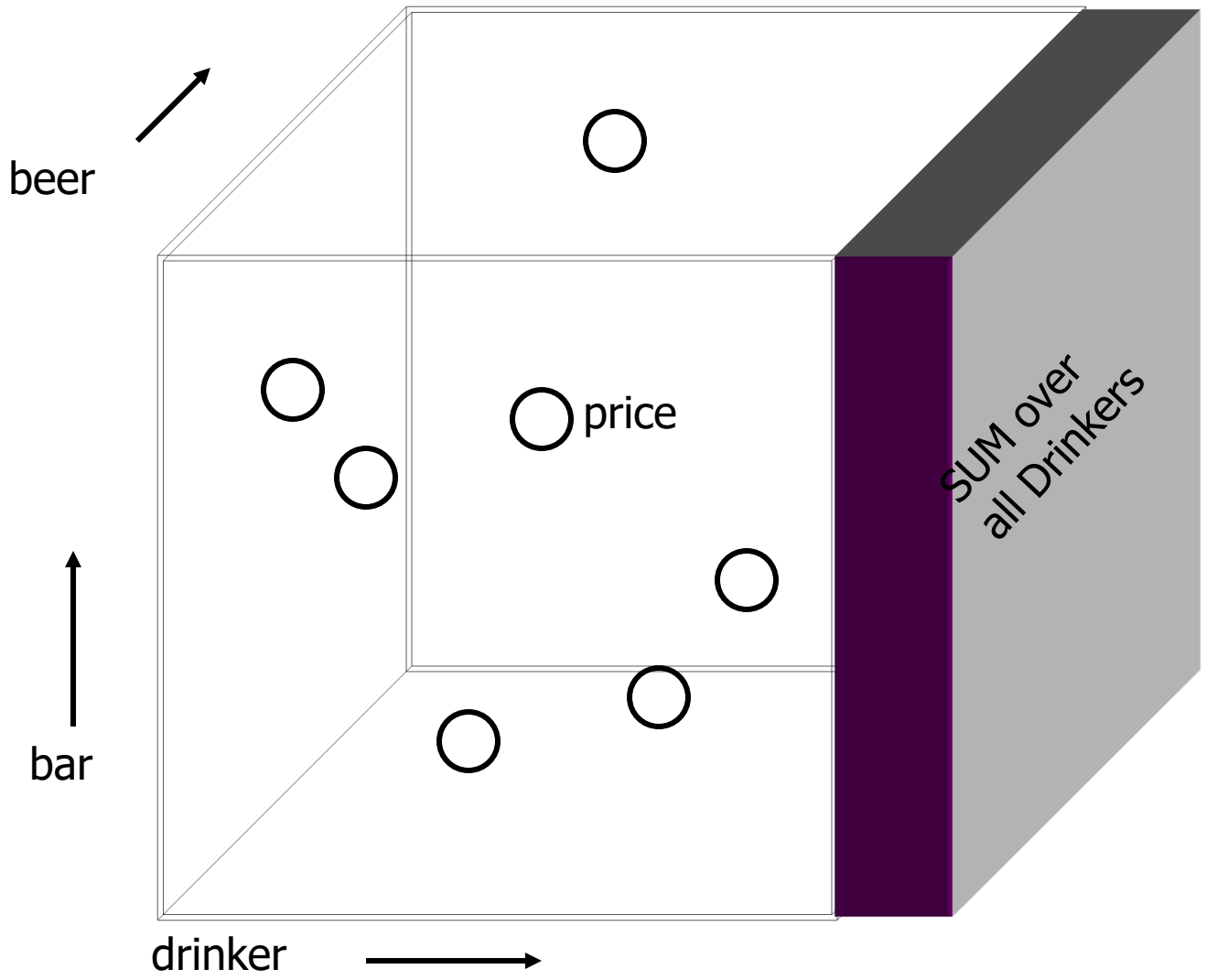


Measures

- The Data Cube also (logically) includes aggregations (e.g., SUM) along the margins of the cube.
- These *measures* (which textbook calls *marginals*) include aggregations over one dimension, two dimensions, …

Visualization --- Data Cube with Aggregation

What's the total sales of each beer sold by each bar?



Sums on the Cube

How can we find the total price for each bar and beer, summing over all drinkers, as a GROUP BY?

```
SELECT bar, beer, SUM(price)
FROM Sales
GROUP BY bar, beer;
```

What does this represent?

```
SELECT drinker, SUM(price)
FROM Sales
GROUP BY drinker;
```

The total spent by each drinker, summing over all bars and beers.

Example: Measures

- Our 3-dimensional Sales cube includes the sum of price over each bar, each beer, and each drinker.
  - Summing over each drinker was first example on previous slide.
  - Could go 4-dimensional and have day as fourth dimension.
- It could also have the sum of price for each drinker over all bar-beer pairs, (see second example on previous slide), all beer-day pairs, …, all bar-drinker-day triples, …
- Question: Do the aggregates have to be stored, and maintained every time a relevant fact is inserted, updated or deleted?

Structure of the Cube

- Think of each dimension as having an additional value *, meaning “everything”.
- A point with one or more *’s in its coordinates aggregates over the dimensions with the *’s.
- Examples:
  - Sales(”Joe’s Bar”, ”Bud”, *, *) holds the Sum (over all drinkers and all days) of the Bud beers consumed at Joe’s Bar.
  - Sales(bar, beer, *, *) holds the Sum (over all drinkers and all days) for any bar and beer.
- Sum isn’t the only “Measure/Marginal”.
  - Average, Count and more complex statistical formulas are sometimes used.

A ClothingSales Relation Instance

item_name	*color*	*clothes_size*	*quantity*
dress	dark	small	2
dress	dark	medium	6
dress	dark	large	12
dress	pastel	small	4
dress	pastel	medium	3
dress	pastel	large	3
dress	white	small	2
dress	white	medium	3
dress	white	large	0
pants	dark	small	14
pants	dark	medium	6
pants	dark	large	0
pants	pastel	small	1
pants	pastel	medium	0
pants	pastel	large	1
pants	white	small	3
pants	white	medium	0
pants	white	large	2
shirt	dark	small	2
shirt	dark	medium	6
shirt	dark	large	6
shirt	pastel	small	4
shirt	pastel	medium	1
shirt	pastel	large	2
shirt	white	small	17
shirt	white	medium	1
shirt	white	large	10
skirt	dark	small	2
skirt	dark	medium	5
…	…	…	…
…	…	…	…

Cross Tabulation of *ClothingSales* by *item_name* and *color*

clothes_size **all**

item_name	*color*: dark	pastel	white	total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pants	20	2	5	27
total	62	54	48	164

- The table above is an example of a **cross-tabulation** (**cross-tab**), also referred to as a **pivot-table**.
  - Values for one of the dimension attributes (item_name) form the row headers.
  - Values for a second dimension attribute (color) form the column headers.
  - Other dimension attributes (clothes_size) are listed in a menu at the top.
  - Values in individual cells are (aggregates of) the values of the dimension attributes that specify the cell.

Online Analytical Processing Operations

- **Pivoting:** Changing the dimensions used in a cross-tab.
  - E.g., Moving clothes_size to column names.
- **Slicing:** Creating a cross-tab for fixed values only.
  - E.g., Fixing color to white and clothes_size to small.
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.
- **Rollup:** Moving from finer-granularity data to a coarser granularity.
  - E.g., Aggregating along one or more dimensions.
  - E.g., Moving from aggregates by day to aggregates by month or year.
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data.
  - Involves estimates if you don't have the original data.

Roll-Up

- *Roll-up* means aggregate along one or more dimensions.
- Example: Given a Fact Table showing how much Bud each drinker consumes at each bar, roll it up into a table giving the total amount of Bud consumed by each drinker.

Drill-Down

- *Drill-Down* means "dis-aggregate", that is, break an aggregate into its constituent parts.
- Example: Having determined that Joe's Bar sells very few Anheuser-Busch beers, break down his sales by each particular A-B beer.
- Drill-Down can be done accurately only when we also have the underlying Fact Table, which has full data for every A-B beer.
  - Doing Drill-Down approximately still has value; how can we do that?

Example: Roll-Up and Drill-Down

$ of Anheuser-Busch by drinker/bar

	Jim	Bob	Mary
Joe's Bar	45	33	30
Nut-House	50	36	42
Blue Chalk	38	31	40

Roll-up by Bar

$ of A-B / drinker

Jim	Bob	Mary
133	100	112

Drill-down by Beer

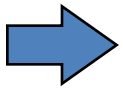
$ of A-B Beers / drinker

	Jim	Bob	Mary
Bud	40	29	40
M'lob	45	31	37
Bud Light	48	40	35

Aggregates and Roll-Up (1)

- Add up the amounts for day 1
- In SQL: SELECT SUM(amt) FROM SALES
  WHERE date = 1

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

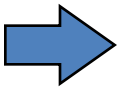


81

Aggregates and Roll-Up (2)

- Add up amounts by day
- In SQL: SELECT date, SUM(amt) FROM SALES
  GROUP BY date

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

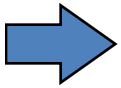


ans	date	sum
	1	81
	2	48

Roll-Up and Drill-Down

- Add up amounts by product, day
- In SQL: SELECT prodid, date, SUM(amt) FROM SALES
  GROUP BY date, prodId

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4



sale	prodId	date	amt
	p1	1	62
	p2	1	19
	p1	2	48

Roll-Up

Drill-Down

Outer Join Motivation

- Suppose we join relations R and S on some join condition.
- A tuple of *R* that has no tuple of *S* with which it joins is said to be *dangling*.
  - Similarly for a tuple of *S*.
- **Outer Join** preserves dangling tuples by padding them with NULL.
  - Even if attributes don't allow NULL.

Reminder: Join (Inner Join)

```
SELECT * FROM R, S WHERE R.B = S.B;
SELECT * FROM R JOIN S ON R.B = S.B;
SELECT * FROM R INNER JOIN S ON R.B=S.B;
```

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

(1,2) joins with (2,3), but the other two tuples, (4,5) and (6,7), are dangling.

Example: Full Outer Join (Full Join)

SELECT * FROM R FULL OUTER JOIN S ON R.B = S.B;

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

SELECT *
FROM R FULL OUTER JOIN S
ON R.B = S.B;

A	R.B	S.B	C
1	2	2	3
4	5	NULL	NULL
NULL	NULL	6	7

Tricky: these are **NOT** equivalent!

SELECT * FROM R FULL OUTER JOIN S ON R.B = S.B WHERE R.A = 4;
SELECT * FROM R FULL OUTER JOIN S ON R.B = S.B AND R.A = 4;

Variations of Outer Join

Must have LEFT, RIGHT, or FULL before OUTER JOIN.

- LEFT means pad dangling tuples of R only.
- RIGHT means pad dangling tuples of S only.
- FULL means pad both; this choice is the default.

Must have ON clause.

Left and Right Outer Join (Left Join and Right Join)

What is the result of the following?

SELECT * FROM R **LEFT** OUTER JOIN S ON R.B = S.B;

What is the result of the following?

SELECT * FROM R **RIGHT** OUTER JOIN S ON R.B = S.B;

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

Examples: Left /Right Outer Join

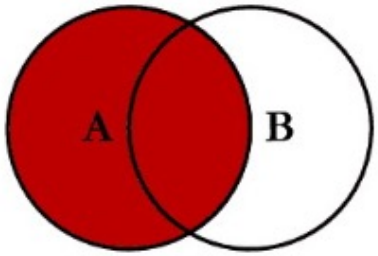
```
SELECT *
FROM Movies LEFT OUTER JOIN StarsIn
ON Movies.title = StarsIn.movieTitle AND Movies.year = StarsIn.movieYear
```

This gives Movie tuples with any star who StarsIn that movie,
and NULL-padded Movie tuples for which there's no star who StarsIn that movie,
but won't include StarsIn tuples where stars doesn't star in any movie in Movies.

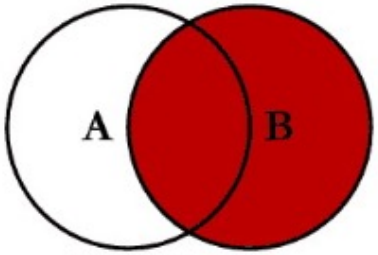
```
SELECT *
FROM Movies RIGHT OUTER JOIN StarsIn
ON Movies.title = StarsIn.movieTitle AND Movies.year = StartsIn.movieYear
```

This gives StarsIn tuples where the listed movie is in Movies,
and NULL-padded StarsIn tuples for which movie listed isn't in Movies,
but won't include movies for which there's no star that's in StarsIn.

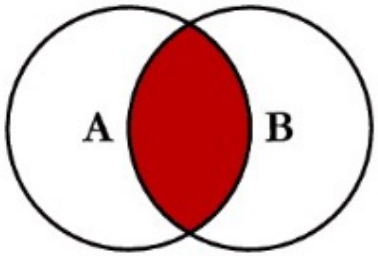
SQL JOINS



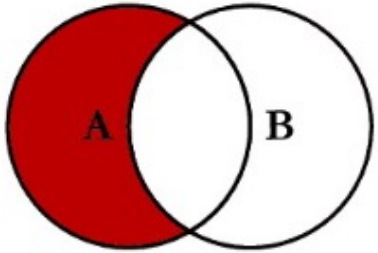

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



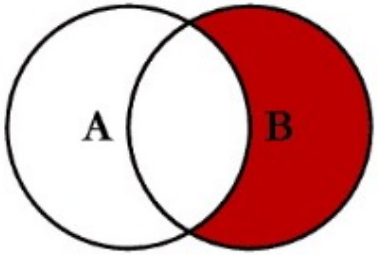

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



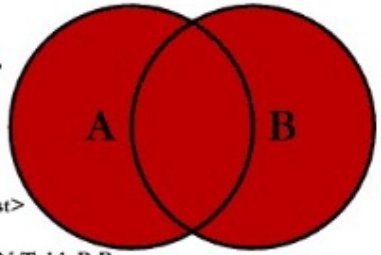

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



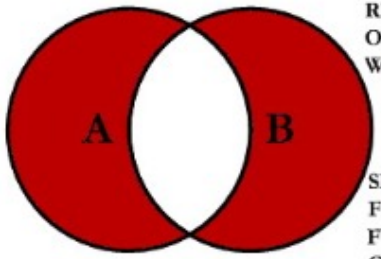

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

OLAP and Outer Join

Taking the Cartesian Product of the Dimension Table Keys …

… and then taking LEFT OUTER JOIN of that with the Fact Table …
… will give you entries for **every** combination of Dimensions, …
… **not just the ones** that have entries in the Fact Table.

There may not be any Facts for a given Day, but you'd like that Day to show up in your report.

- Example: Summing up Sales amount across all Stores and Products and Days, that "missing" Day's total Sales amount should be 0.
- You can get that by using Outer Join. (Well, actually you get NULL.)

How do you change NULL value to 0?

- One common way to do this is with the **Coalesce** function.
- COALESCE(x, 0) has value x if x isn't NULL, and value 0 if x is NULL.

Calculating # of Roles for Each Actor without Outer Join

Actors(actorID, isActorsEquity, equityJoinDate, roleCount)

Roles(playTitle, productionNum, actorID, characterName)

For each actor, find the actorId and roleCount, and calculate the number of roles in Roles for that actor, **including for actor who have no roles.** The number of roles should appear as calulatedRoles in your result. (That might not equal roleCount!)

```
SELECT a.actorID, a.roleCount,
                    COUNT(*) AS calculatedRoles
FROM Actors a, Roles r
WHERE a.actorID = r.actorID
GROUP BY a.actorID

UNION

SELECT a.actorID, a.roleCount, 0 As calulatedRoles
FROM Actors a
WHERE NOT EXISTS
      ( SELECT * FROM Roles r
        WHERE a.actorID = r.actorID );
```

Why don't we need a.roleCount in the GROUP BY clause?

Calculating # of Roles for Each Actor with Outer Join

Actors(actorID, isActorsEquity, equityJoinDate, roleCount)

Roles(playTitle, productionNum, actorID, characterName)

FoFor each actor, find the actorId and roleCount, and calculate the number of roles in Roles for that actor, **including for actor who have no roles.** The number of roles should appear as calulatedRoles in your result. (That might not equal roleCount!)

```
SELECT a.actorID, a.roleCount,
                    COUNT(r.actorID) AS calculatedRoles
FROM Actors a LEFT OUTER JOIN Roles r
          ON a.actorID = r.actorID
GROUP BY a.actorID;
```

Could the COUNT that's in the SELECT be applied instead to any Roles table attribute?