Information Integration

MEDIATORS

WAREHOUSING

ANSWERING QUERIES USING VIEWS

Information Integration

Information integration is the process of taking several databases and making the data in these sources work together as if they were a single database.

The integrated database may be

- Physical (”data warehouse”)
- Virtual (”mediator”) that may be queried even though it does not exist physically

Information-integration systems require special kinds of query-optimization techniques for their efficient operation.

Why Information Integration?

If we could put data always in a single database, there would be no need for information integration.

However, in the real world, matters are rather different..

- Databases are created intependently, even if they later need to work together.
- The use of databases evolves, so we cannot design a database to support every possible future use.

Example Applications

1. Enterprise Information Integration: making separate DB's, all owned by one company, work together.
2. Scientific DB's, e.g., genome DB's.
3. Catalog integration: combining product information from all your suppliers.

Challenges

1. *Legacy databases* : DB's get used for many applications.
   - You can't change its structure for the sake of one application, because it will cause others to break.
2. *Incompatibilities* (heterogenity problem): Two, supposedly similar databases, will mismatch in many ways.

Examples: Incompatibilities

Lexical : `addr` in one DB is `address` in another.

Value mismatches : is a "BL" car the same color in each DB (blue versus black)? Is 20 degrees Fahrenheit or Centigrade?

Semantic : are "employees" in each database the same? What about consultants? Retirees? Contractors?

Query-Language heterogenity : Relational database (SQL) verus XML (Xquery)

Data Type differences : Serial numbers might be represented as *string* in one source and *integer* in another source.

Examples: Schema Heterogeneity

One dealer might store cars in a single relation that look like:

- `Cars(serialNo, model, color, autoTrans, navi, ...)`

Another dealer might use a schema in which options are seperated out into a second relation, such as:

- `Autos (serial, model, color)`
- `Options (serial, option)`

What Do You Do About It?

Grubby, handwritten translation at each interface.

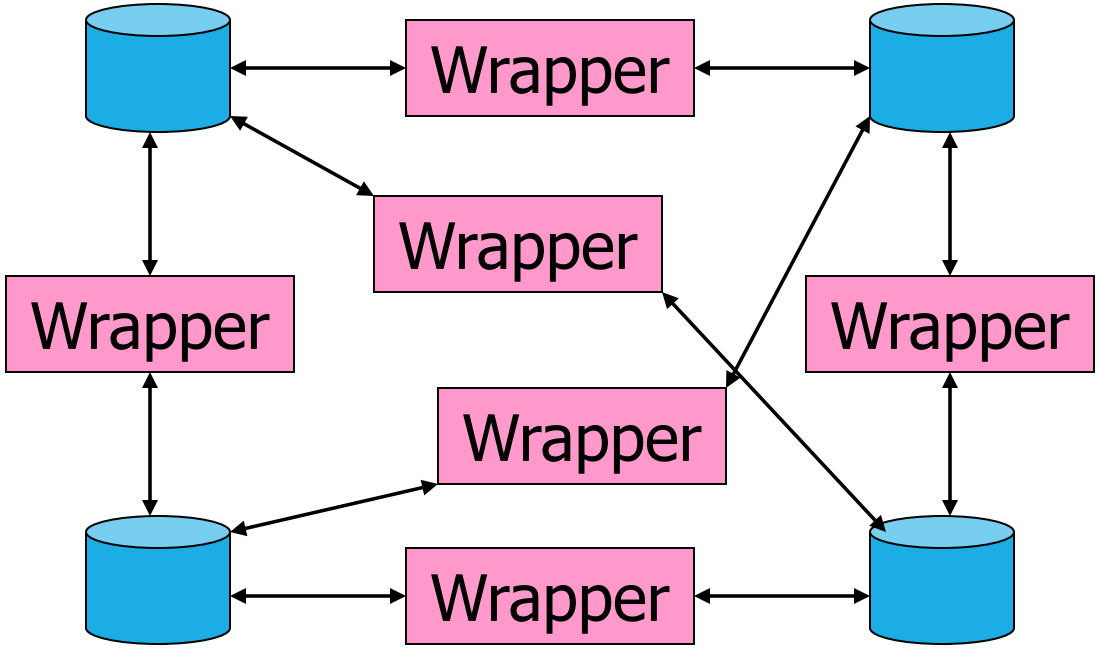
- Some research on automatic inference of relationships.

Wrapper (aka “adapter”) translates incoming queries and outgoing answers.

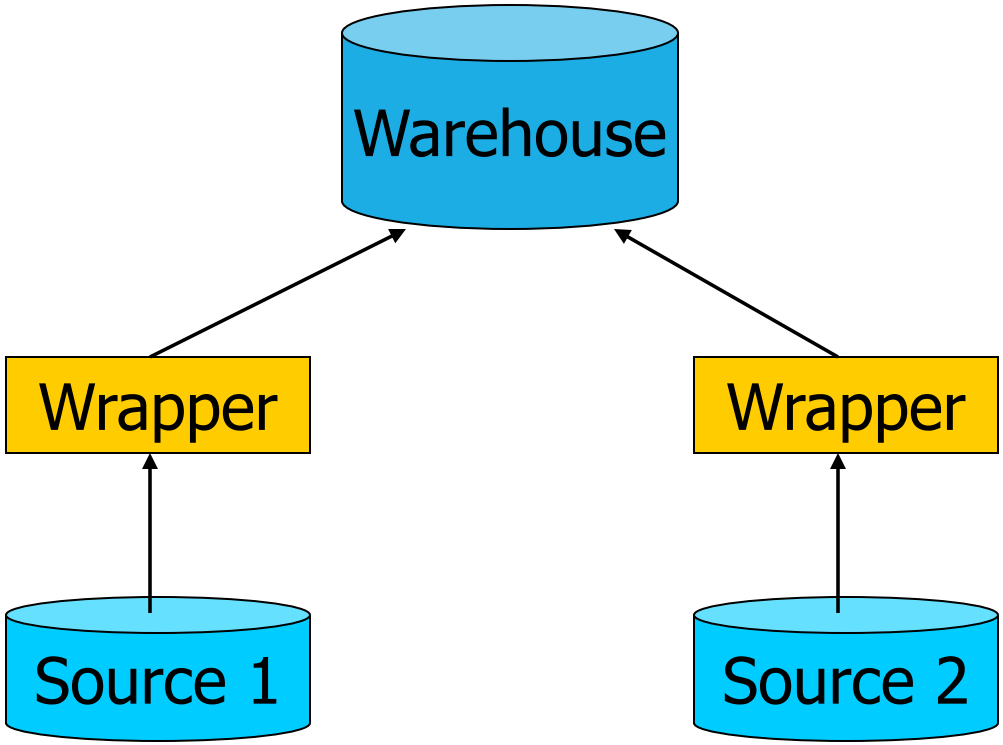
Integration Architectures

1. *Federation* : everybody talks directly to everyone else.
2. *Warehouse* : Sources are translated from their local schema to a global schema and copied to a central DB.
3. *Mediator* : *Virtual warehouse* --- turns a user query into a sequence of source queries.

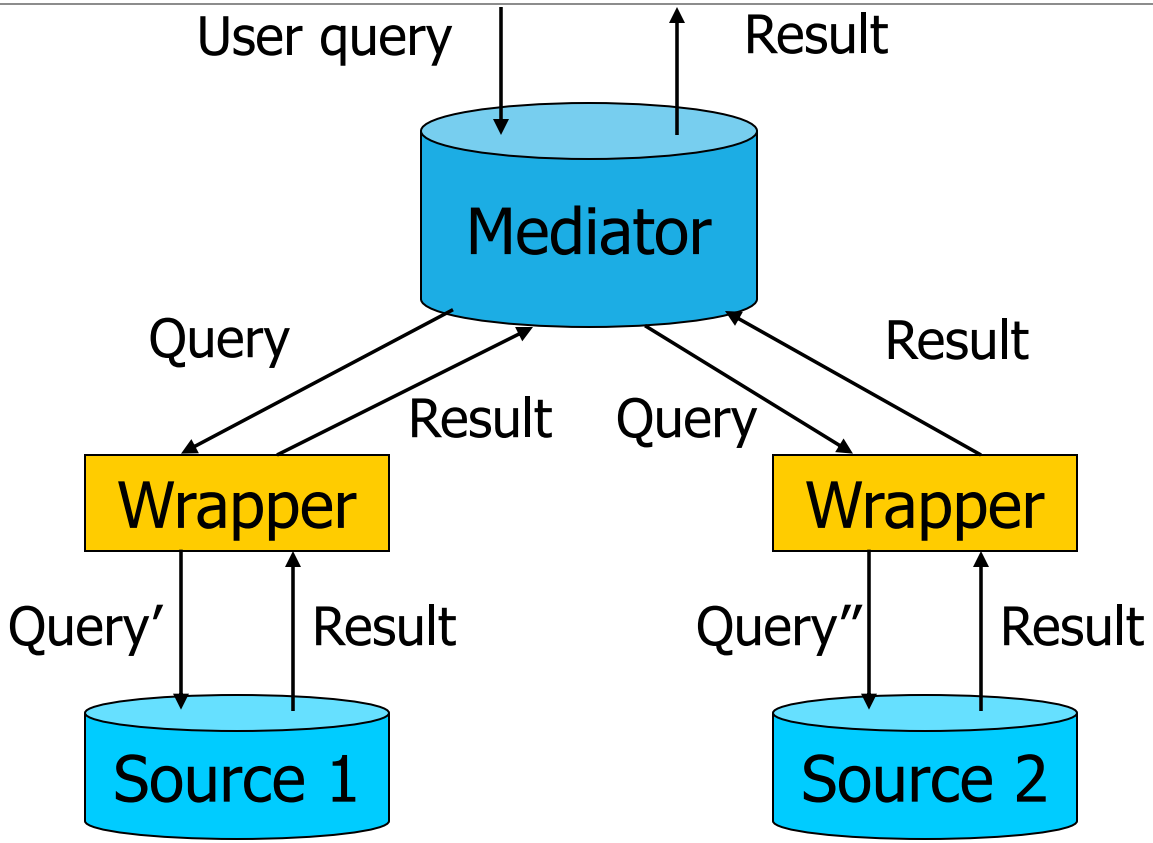
Federations



Warehouse Diagram



A Mediator


User query
Result
Mediator
Query
Result
Result
Query
Wrapper
Wrapper
Query'
Result
Query''
Result
Source 1
Source 2

Schema Heterogeneity

One dealer might store cars in a single relation that look like:

- `Cars(serialNo, model, color, autoTrans, navi, ...)`

Another dealer might use a schema in which options are seperated out into a second relation, such as:

- `Autos (serial, model, color)`
- `Options (serial, option)`

Example: Mediatior

Suppose mediator integrates the same two automobile sources into a view that is a single relation with schema:

- `AutosMed (serialNo, model, color, autoTrans, dealer)`

Assume the user asks the mediator about red cars, with the query:

```
SELECT serialNo, model
FROM AutosMed
WHERE color = 'red';
```

Example: Mediatior

The wrapper for Dealer 1 translates the query into the terms of the dealer's schema:

```
SELECT SerialNo, model
FROM Cars
WHERE color = 'red'
```

At the same time, the wrapper for Dealer 2 translates the same query into the schema of that dealer:

```
SELECT serial, model

FROM Autos

WHERE color = 'red';
```

The mediator takes union of these sets and returns the result to the user.

Mediation Approach

Mediator processes queries into steps executed at sources.

Actions

Read Chapter *Information Integration* (21.1-2)

On-Line Application Processing

WAREHOUSING

DATA CUBES

DATA MINING

Overview

Traditional database systems are tuned to many, small, simple queries.

Some new applications use fewer, more time-consuming, *analytic* queries.

New architectures have been developed to handle analytic queries efficiently.

The Data Warehouse

The most common form of data integration.

- Copy sources into a single DB (*warehouse*) and try to keep it up-to-date.
- Usual method: periodic reconstruction of the warehouse, perhaps overnight.
- Frequently essential for analytic queries.

OLTP

Most database operations involve *On-Line Transaction Processing* (OLTP).

- Short, simple, frequent queries and/or modifications, each involving a small number of tuples.
- Examples: Answering queries from a Web interface, sales at cash registers, selling airline tickets.

OLAP

On-Line Application Processing (OLAP, or "analytic") queries are, typically:

- Few, but complex queries --- may run for hours.
- Queries do not depend on having an absolutely up-to-date database.

OLAP Examples

1. Amazon analyzes purchases by its customers to come up with an individual screen with products of likely interest to the customer.
2. Analysts at Wal-Mart look for items with increasing sales in some region.
   - Use empty trucks to move merchandise between stores.

Common Architecture

Databases at store branches handle OLTP.

Local store databases copied to a central warehouse overnight.

Analysts use the data warehouse for OLAP.

Star Schemas

A *star schema* is a common organization for data at a warehouse. It consists of:

1. *Fact table* : a very large accumulation of facts such as sales.
   - Often "insert-only."
2. *Dimension tables* : smaller, generally static information about the entities involved in the facts.
   - e.g., information about products, customers and dates.

Example: Star Schema

Suppose we want to record in a warehouse information about every beer sale: the bar, the brand of beer, the drinker who bought the beer, the day, the time, and the price charged.

The fact table is a relation:

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

Example -- Continued

The dimension tables include information about the bar, beer, and drinker "dimensions":

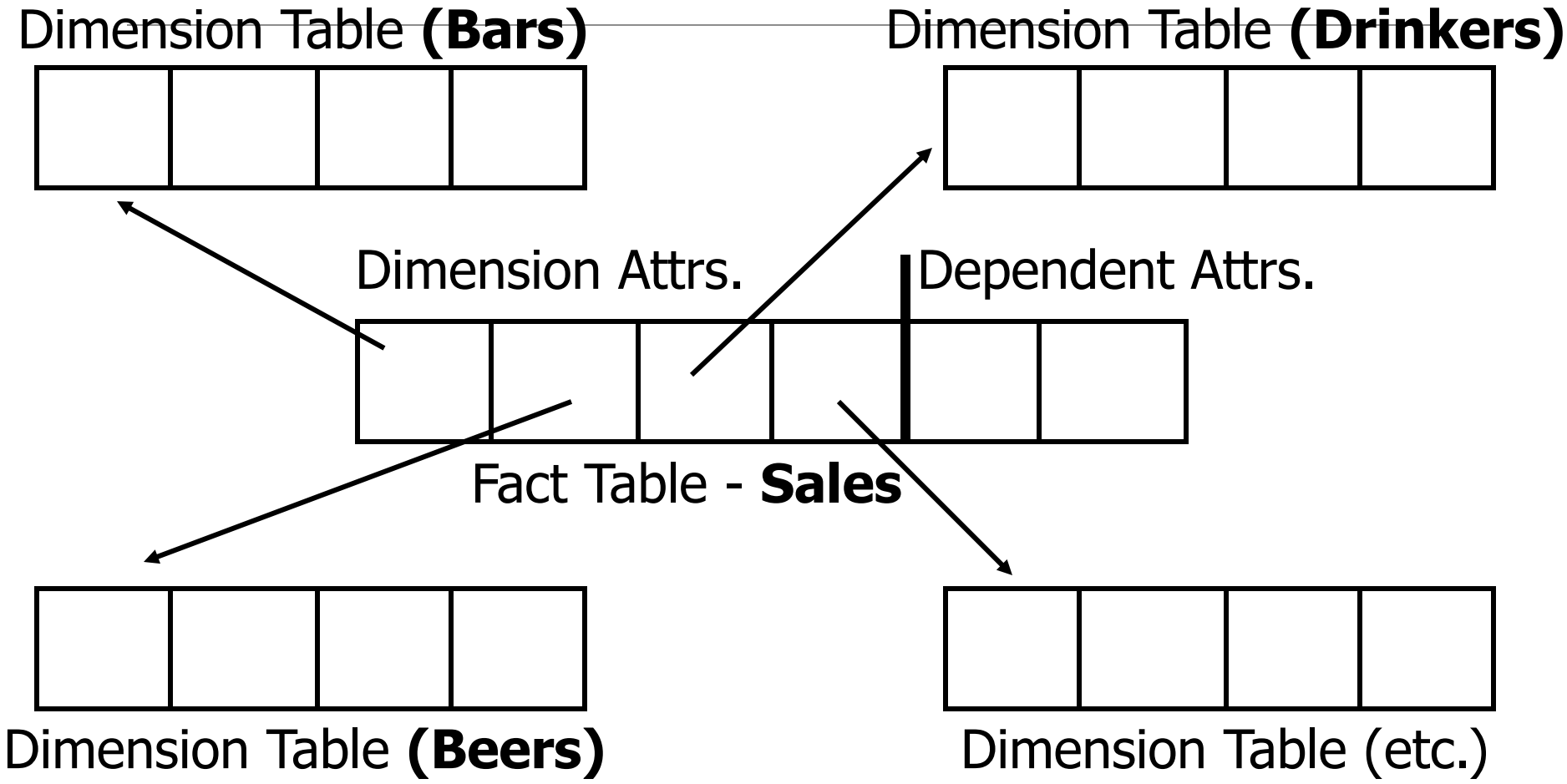
Bars(bar, addr, license)

Beers(beer, manf)

Drinkers(drinker, addr, phone)

...

Visualization – Star Schema



Dimensions and Dependent Attributes

Two classes of fact-table attributes:

1. *Dimension attributes* : the foreign key of a dimension table.
2. *Dependent attributes* : a value determined by the dimension attributes of the tuple.

Example: Dependent Attribute

price is the dependent attribute of our example Sales relation.

It is determined by the combination of dimension attributes: bar, beer, drinker, and the time (combination of day and time-of-day attributes).

Approaches to Building Warehouses

1. *ROLAP* = “relational OLAP”: Tune a relational DBMS to support star schemas.
2. *MOLAP* = “multidimensional OLAP”: Use a specialized DBMS with a model such as the “data cube.”

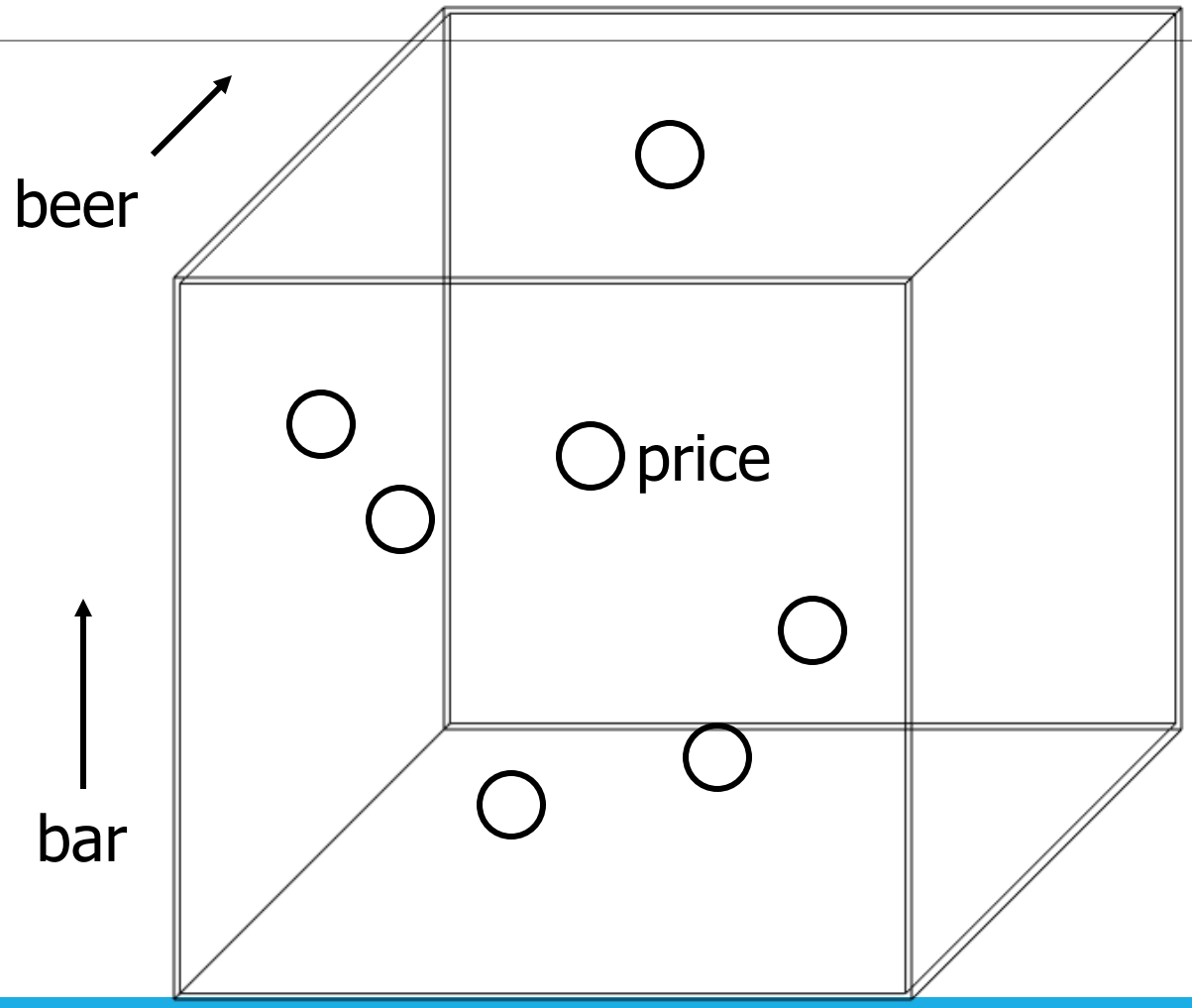
MOLAP and Data Cubes

Keys of dimension tables are the dimensions of a hypercube.

- Example: for the Sales data, the four dimensions are bar, beer, drinker, and time.

Dependent attributes (e.g., price) appear at the points of the cube.

Visualization -- Data Cubes

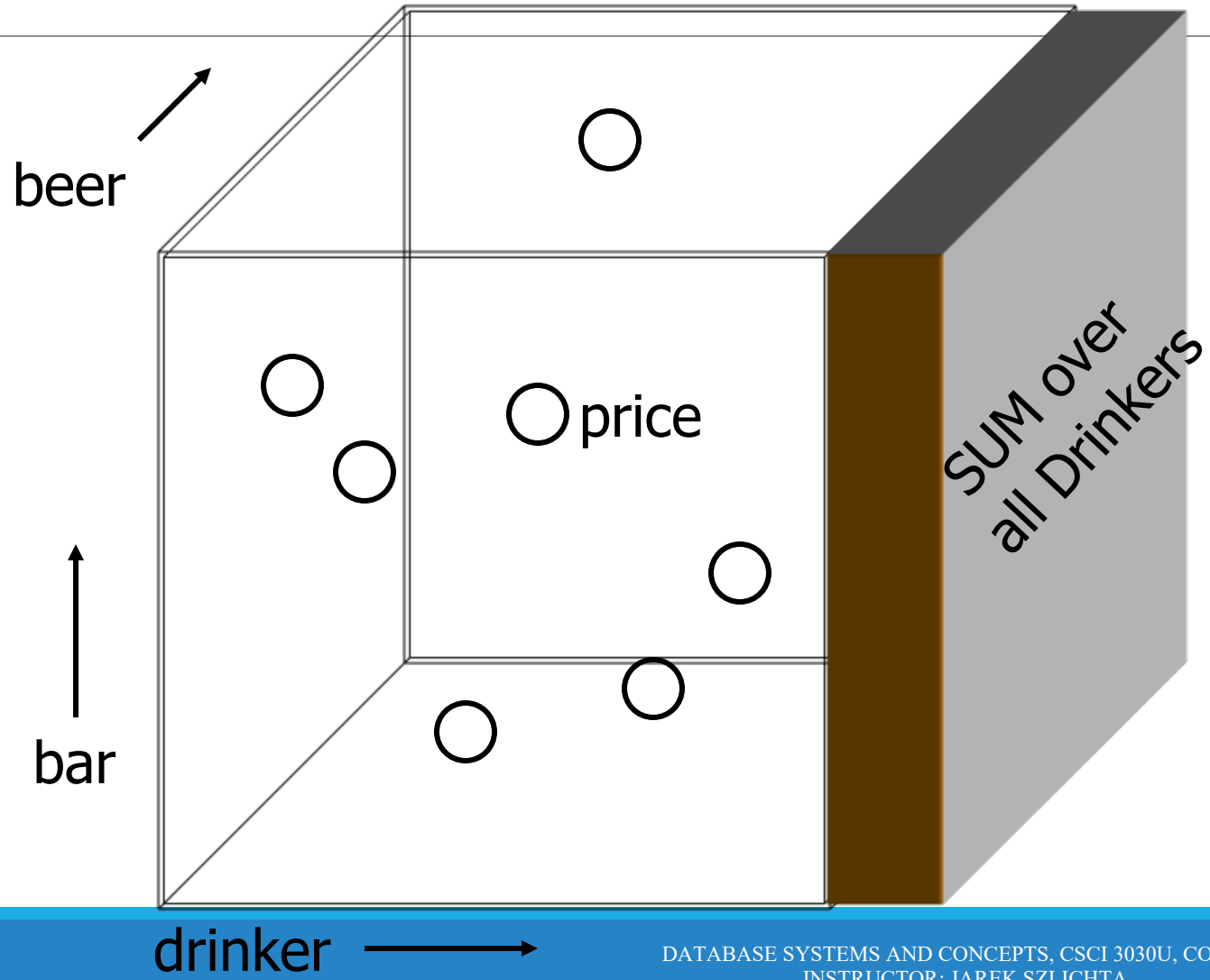


Marginals

The data cube also includes aggregation (typically SUM) along the margins of the cube.

The *marginals* include aggregations over one dimension, two dimensions,…

Visualization --- Data Cube w/Aggregation



Example: Marginals

Our 4-dimensional Sales cube includes the sum of price over each bar, each beer, each drinker, and each time unit (perhaps days).

It would also have the sum of price over all bar-beer pairs, all bar-drinker-day triples,…

Structure of the Cube

Think of each dimension as having an additional value *.

A point with one or more *'s in its coordinates aggregates over the dimensions with the *'s.

Example: Sales("Joe's Bar", "Bud", *, *) holds the sum, over all drinkers and all time, of the Bud consumed at Joe's.

Drill-Down

Drill-down = “de-aggregate” = break an aggregate into its constituents.

Example: having determined that Joe’s Bar sells very few Anheuser-Busch beers, break down his sales by particular A.-B. beer.

Roll-Up

Roll-up = aggregate along one or more dimensions.

Example: given a table of how much Bud each drinker consumes at each **bar**, roll it up into a table giving total amount of Bud consumed by each drinker (at all bars).

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

Course Plug

Big Data Analytics – Winter 2019

Actions

Read chapters: Information Integration (21.1, 21.2) and Data Mining (22.1, 22.2).