Warehousing

What we will cover...

1. Analytics database goals
2. Star Schema

Normalized Shmormalized

It's worth reviewing *why* we wanted normalization. We saw that normalization:

1. Saved space
2. Made updates faster
3. (This made ACID easier)

Normalized Shmormalized

Space is cheap.

What if we aren't updating our data in the same way we were in an operational database?

Normalized Shmormalized

In analytics, we want to *read* the data and read *lots* of data.

It should be clear that this is an entirely different paradigm than that of the operational database we saw last week.

Operational vs Analytics (OLTP vs OLAP)

Operational: Many small reads and many updates from thousands of users who are interested only in "their" rows. Data integrity, consistency, is important because the rest of the software relies on it.

Analytics: Few users, rarely writing (batch updates?), making *giant* read queries that might span *all* the rows, usually making aggregations. Importance is speed of reads and flexibility to write any needed query.

An Analytic Query

Let's consider our ecommerce database of classic models. Here's a simple question we might want to ask:

What were our profits per month?
How much profit did each product bring in?
What were our profits per product each quarter?

Let's design a simple table to answer all these questions.

Profits per month

```
SELECT month, SUM(profit) FROM table GROUP BY month;
```

order	line_number	profit	month
1	1	$1.50	1
1	2	$3.00	1
2	1	$2.78	1
3	1	$1.98	2
3	2	$5.67	2
3	3	$1.98	2

Profits by product

```
SELECT product, SUM(profit) FROM table GROUP BY product;
```

order	line_number	profit	month	product
1	1	$1.50	1	53
1	2	$3.00	1	989
2	1	$2.78	1	78
3	1	$1.98	2	211
3	2	$5.67	2	987
3	3	$1.98	2	78

Profits by product/quarter

```
SELECT quarter, product, SUM(profit) FROM table GROUP BY quarter, product;
```

order	line_number	profit	month	product	quarter
1	1	$1.50	1	53	1
1	2	$3.00	1	989	1
2	1	$2.78	1	78	1
3	1	$1.98	2	211	1
3	2	$5.67	2	987	1
3	3	$1.98	2	78	1

Profits by manufacturer

```
SELECT manufacturer, SUM(profit) FROM table GROUP BY manufacturer;
```

order	line_number	profit	month	product	quarter	manufacturer
1	1	$1.50	1	53	1	TruxRUs
1	2	$3.00	1	989	1	TruxRUs
2	1	$2.78	1	78	1	Holidayz
3	1	$1.98	2	211	1	Holidayz
3	2	$5.67	2	987	1	TruxRUs
3	3	$1.98	2	78	1	Disney

Profits by manufacturer's country

```
SELECT man_country, SUM(profit) FROM table GROUP BY man_country;
```

order	ln	profit	month	product	quarter	manufacturer	man_country
1	1	$1.50	1	53	1	TruxRUs	USA
1	2	$3.00	1	989	1	TruxRUs	USA
2	1	$2.78	1	78	1	Holidayz	Russia
3	1	$1.98	2	211	1	Holidayz	Russia
3	2	$5.67	2	987	1	TruxRUs	USA
3	3	$1.98	2	78	1	Disney	Canada

Facts and Dimensions

We can see that the tables we designed to answer our queries consisted of two parts:

1. The core **facts** of our business (an item sold), along with derived information (profit) about each fact.
2. A set of **dimensions** by which we'd like to slice/dice our data for aggregating (time, product, etc.)

Star Schema

The star schema will consist of two main parts:

1. Measure (center of the star)
2. Dimensions (the points of the star)

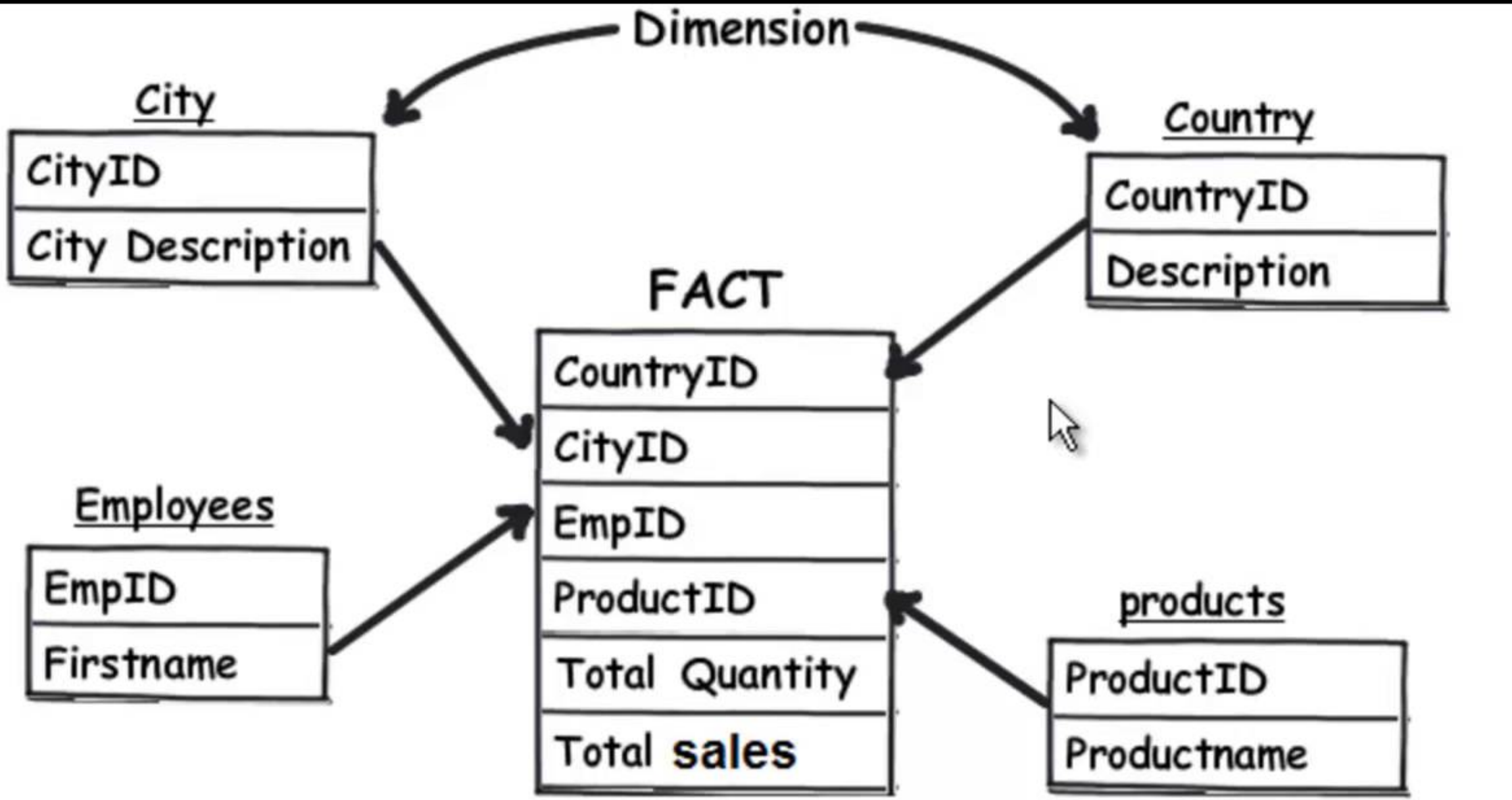

Figure: - Star design

Measures / Fact Tables

These are the things we want to "measure," and generally correspond one-to-one to a business process, often a "transaction" of some sort.

We think of these being big N! Many rows of the measure and we want to aggregate them.

Because we aggregate them, we usually think of these being continuous variables, preferably that are additive.

Grain!

Dimension Tables

How do we aggregate the facts? We aggregate them by *dimensions*.

Dimension tables usually consist of the discrete variables by which we want to aggregate our measures to ask different questions.

Example - Fact Table

id	profit	quantity	product_id	time_id	order_id
1	$1.50	1	53	987	1
2	$3.00	15	989	876	1
3	$2.78	13	78	788	2
4	$1.98	2	211	333	3
5	$5.67	1	53	400	3
6	$1.98	43	78	93	3

Example - Dimension Table

```
SELECT dow, SUM(profit) FROM facts left join time GROUP BY dow;
```

time_id	date	month	dow
93	23	may	tue
987	15	june	wed
788	10	may	wed
400	30	may	sat

Example - Dimension Table

```
SELECT holiday, SUM(profit) FROM facts left join time GROUP BY holiday;
```

time_id	date	month	dow	holiday
93	23	may	tue	true
987	15	june	wed	false
788	10	may	wed	false
400	30	may	sat	false

Review

1. Analytics database goals
2. Star Schema