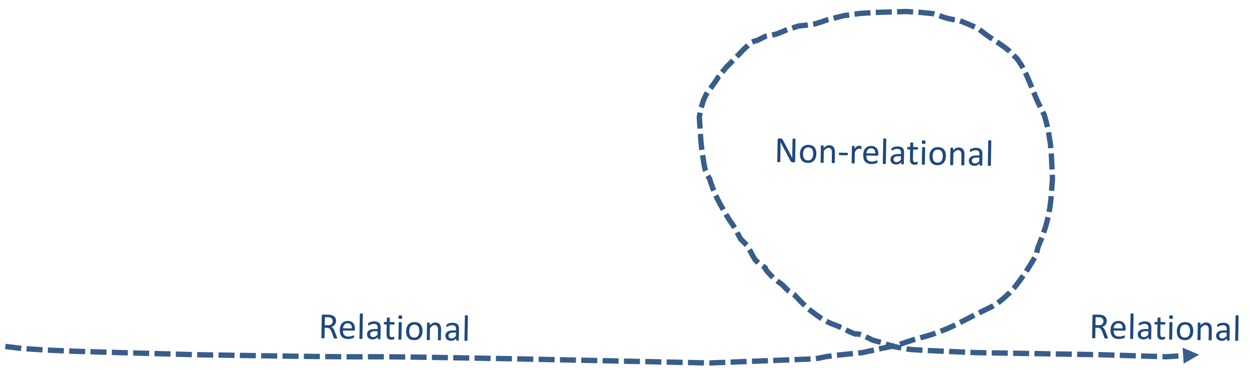
COMP207
Database Development

Lecture 27

Data Warehousing, OLAP, and Data Mining:
Introduction

- End of module questioner can be completed during first lecture tomorrow

The Final Chapter



- Return to **relational databases**
  - So, again relational databases, SQL, …
- Different use cases: **data analysis**
- Different technology: **data warehouses**

GROUP BY SQL

- SQL has a GROUP BY keyword
- It has not been important so far
  - the kind of queries we have focused on have been simpler

```
SELECT Lecturers.name, Modules.name
FROM Modules, Lecturers
WHERE code=module;
```

- How to count the number of modules each lecturer has been teaching?

```
SELECT Lecturers.name, COUNT(*)
FROM Modules, Lecturers
WHERE code=module
GROUP BY Lecturers.name;
```

GROUP BY SQL - intuitively

```
SELECT *
FROM Modules, Lecturers
WHERE code=module;
```

Lecturers.name	module	code	Modules.name
J. Fearnley	COMP105	COMP105	PL. paradigms
J. Fearnley	COMP396	COMP396	Automated trading project
M. Gairing	COMP211	COMP211	Computer networks

GROUP BY SQL - intuitively

- GROUP BY intuitively does as follows:

```
SELECT Lecturers.name, COUNT(*)
FROM Modules, Lecturers
WHERE code=module
GROUP BY Lecturers.name;
```

<table>
<tr><th>Lecturers.name</th><th></th></tr>
<tr><td>J. Fearnley</td><td>
<table>
<tr><th>module</th><th>code</th><th>Modules.name</th></tr>
<tr><td>COMP105</td><td>COMP105</td><td>PL. paradigms</td></tr>
<tr><td>COMP396</td><td>COMP396</td><td>Automated trading project</td></tr>
</table>
</td></tr>
<tr><td>M. Gairing</td><td>
<table>
<tr><th>module</th><th>code</th><th>Modules.name</th></tr>
<tr><td>COMP211</td><td>COMP211</td><td>Computer networks</td></tr>
</table>
</td></tr>
</table>

HAVING SQL

- In SQL WHERE comes and is interpreted before GROUP BY
- If you want to do a WHERE after GROUP BY, use HAVING

```
SELECT Lecturers.name, COUNT(*)
FROM Modules, Lecturers
WHERE code=module
GROUP BY Lecturers.name
HAVING COUNT(*)>1
ORDER BY Lecturers.name;
```

- Other general aggregate functions: MIN, MAX
- Only for number attributes: AVG, SUM
- ORDER BY is still last…

Data Analysis

A Data Analysis Scenario

- A big company wants to *analyse its product sales*…
- Has database with schema:

Sales(productNo, date, store_name, price)

Stores(name, city, country, phone)

- Example query:

Requires most of the data

```
SELECT country, AVG(price)
FROM Sales, Stores
WHERE Sales.store_name = Stores.name AND
      date >= '2019-06-01'
GROUP BY country;
```

Scenario 2: Healthcare Analytics

- A hospital wants to *analyse risks, best mode of treatment, …*
- Possible schema:

> **EpisodesOfCare**(date, patientID, diagnosisID, treatmentID, …)
> **Patients**(patientID, name, age, gender, …)
> **Diagnoses**(diagnosisID, …)
> **Treatments**(treatmentID, …)

- Analyse for
  - Best mode of treatment given a certain diagnosis
  - Average length of stay
  - …

Again: requires most of the data

What Is The Problem?

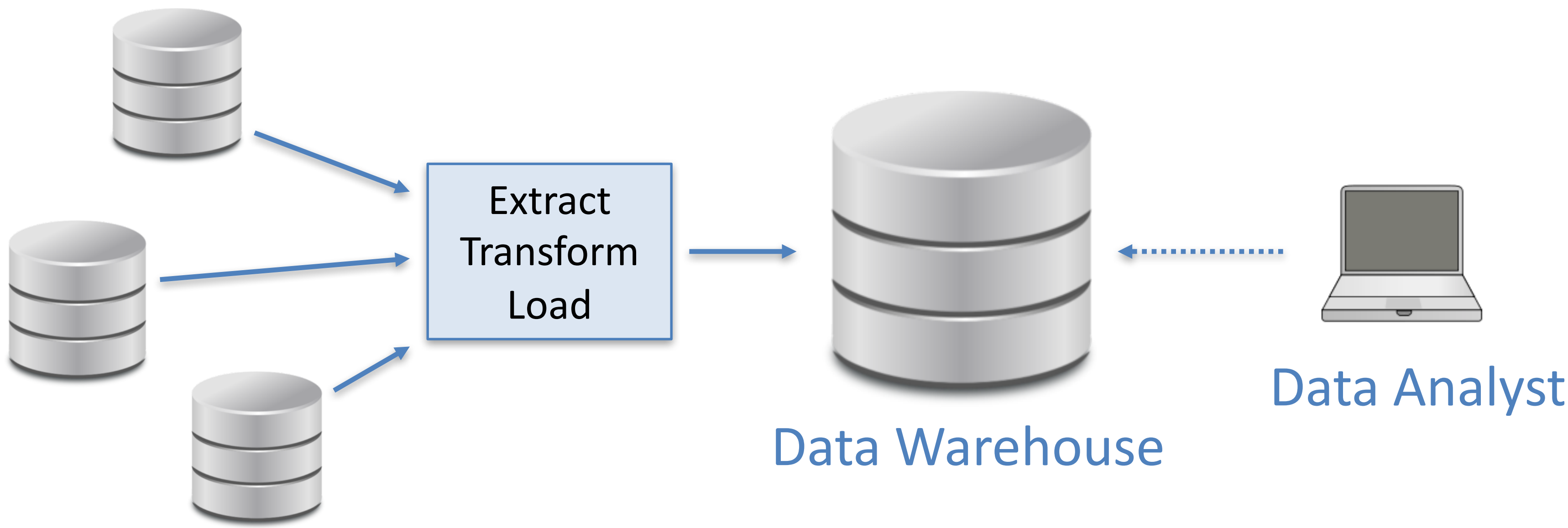
- Queries in these applications tend to…
  - be **complex**: use aggregates & other advanced features
  - **Examine large parts of the data**

```
SELECT country, AVG(price)
FROM Sales, Stores
WHERE Sales.store_name = Stores.name AND
      date >= ‘2019-06-01’
GROUP BY country;
```

- Block large parts of database
- Should avoid execution on DBMS serving many users

Data Warehouses

- Database systems designed to support data analysis
  - …to answer queries like those in the previous scenarios
- Typically integrate different data sources



OLAP vs OLTP

- **OLAP (Online Analytic Processing)**: refers to the process of analysing complex data stored in a data warehouse
- **OLAP query**: a query used in OLAP
  - Typical OLAP query:

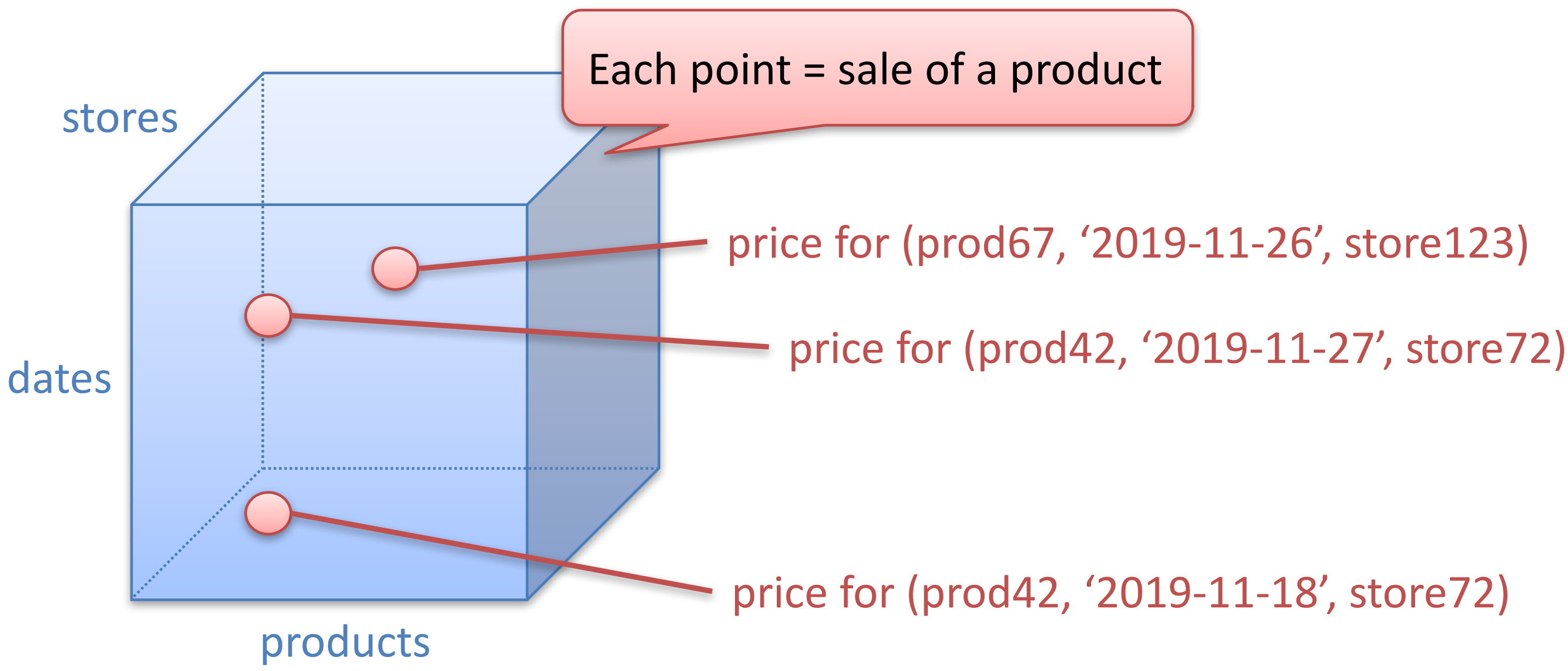
```
SELECT country, AVG(price)
FROM Sales, Stores
WHERE Sales.store_name = Stores.name AND
      date >= '2019-06-01'
GROUP BY country;
```

- **OLTP (Online Transaction Processing)**: traditional DBMS tasks
  - queries and updates that can be executed fast
  - affect a small portion of a database

Data Model for Data Warehouses

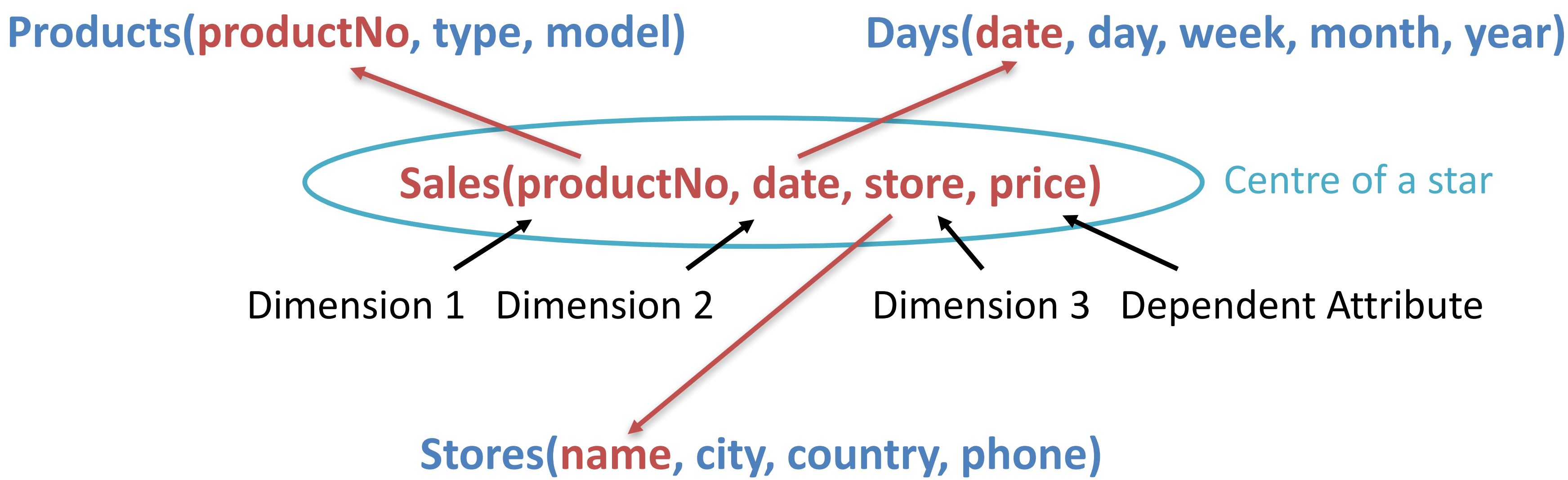
Fact Tables & Data Cubes

- In OLAP applications, there is typically a **unique fact table**
  - Represents events & objects of interest for the analysis
- Example fact table: **Sales(productNo, date, store, price)**
- May be thought of as representing a **data cube**



Star Schemas

- One of the most common data warehouse architectures
  - Unique fact table: contains points in the data cube
  - Dimension tables: describe values along each axis



Star Schemas: More Precisely

- A **star schema** describes a database consisting of:
  - A **fact table** $R(A_1,\ldots A_n,B_1,\ldots,B_m)$
    - $A_1, \ldots, A_n$ are called **dimensions**
    - $B_1, \ldots, B_m$ are called **dependent attributes**
  - A **dimension table** $D_i(A_i,\ldots)$ for each dimension A_i

Key of D_i

$R(A_1,\ldots A_n,B_1,\ldots,B_m)$

$D_1(A_1,\ldots)$ $D_2(A_2,\ldots)$ $D_n(A_n,\ldots)$

- More general: snowflake schema (not here)

Characteristics of Star Schemas

- Key feature: **denormalised schema**

  Unlike typical databases

  - Main data in one table (fact table)
  - Rest of the data can be joined with fact table very quickly

- Gains:
  - Queries **don't require many joins**
  - **Performance gains**, especially when processing queries that would require many joins in a normalised database
  - **Faster aggregation** of data

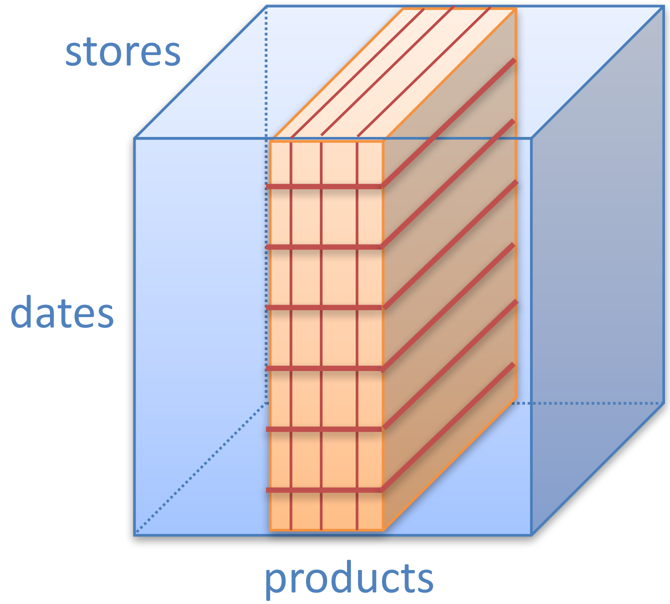
- Also: easy aggregation and grouping of data in many different ways

Example

Sales(productNo, date, store_name, price)
Products(productNo, type, model)
Stores(name, city, country, phone)

- Suppose product type 'X' doesn't sell too well…
- Let's try to find out which model doesn't sell well…

```
SELECT model, SUM(price)
FROM Sales NATURAL JOIN Products
WHERE type='X'
GROUP BY model;
```



- Refine by month:

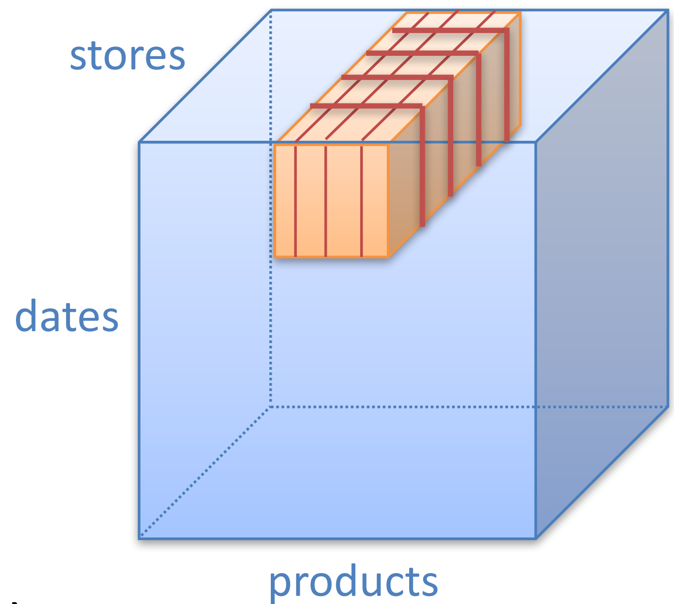
```
SELECT model, month, SUM(price)
FROM (Sales NATURAL JOIN Products)
     NATURAL JOIN Days
WHERE type='X'
GROUP BY model, month;
```

Example

Sales(productNo, date, store_name, price)
Products(productNo, type, model)
Stores(name, city, country, phone)

- Say, product 'X' didn't sell well recently
- Which models and in which stores?

```
SELECT model, store, SUM(price)
FROM Sales NATURAL JOIN Products
WHERE type='X' AND
      (month='Oct' OR month='Nov')
GROUP BY model, store;
```



- Technique is called
  - **Slicing** (done by the where clause) &
  - **Dicing** (done by the group by clause)

Summary

- **Data warehouses** are database systems that support the analysis of data
- Analysing data in a data warehouse is called **OLAP**
  - Contrast to OLTP (traditional transaction processing)
  - Queries in an OLAP setting are called **OLAP queries**
- Data in a data warehouse is typically structured as a **data cube** (e.g., using a **star schema**)
  - Performance gains
  - Easy and flexible aggregation
- Next lecture: data analysis using **data mining**