ISYS2120: Data & Information Management

Week 12: Data Analysis – OLAP and Data Warehousing

Alan Fekete

Based on slides from Kifer/Bernstein/Lewis (2006) "Database Systems" and from Ramakrishnan/Gehrke (2003) "Database Management Systems", and also including material from Fekete and Roehm.

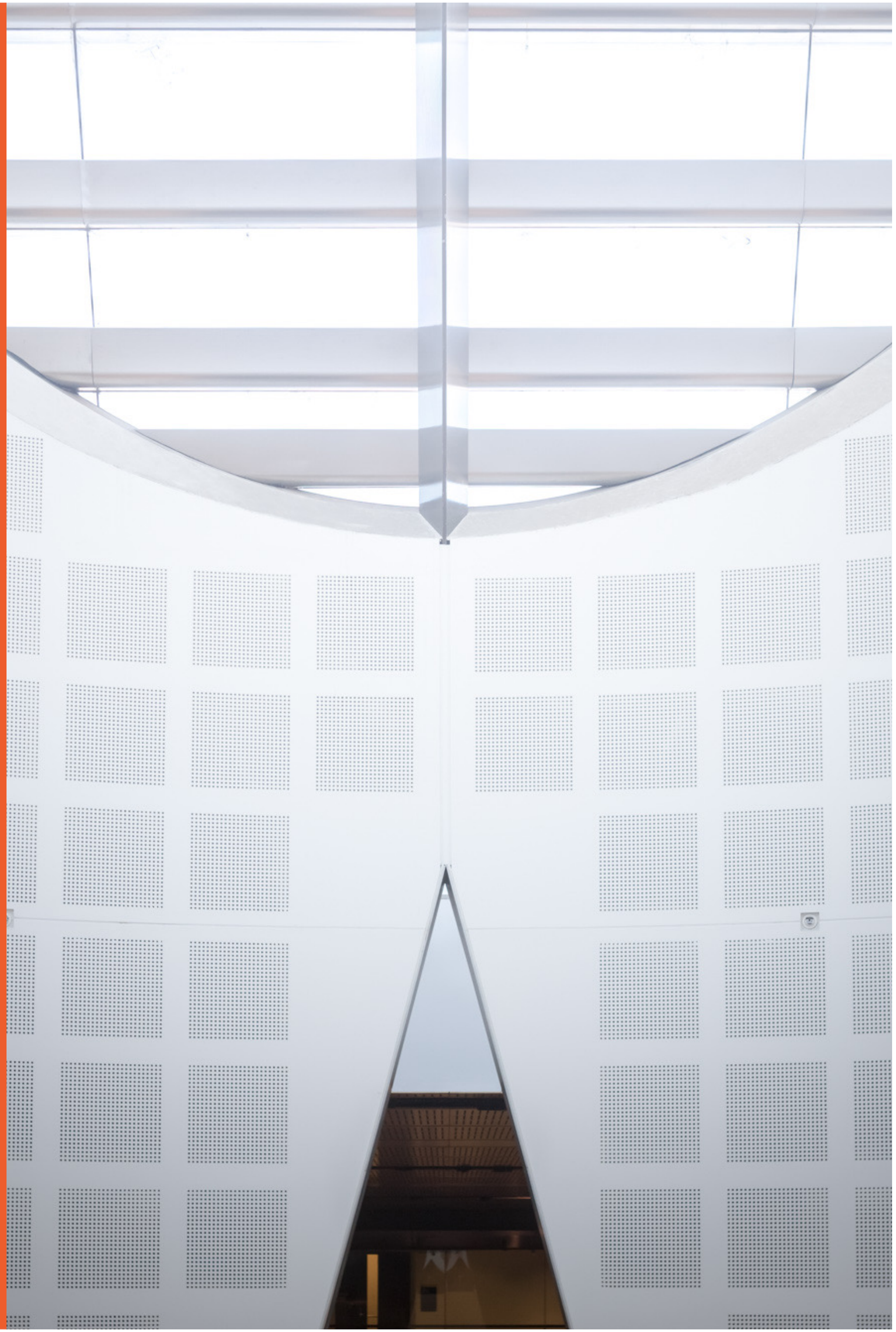
Cf. Kifer/Bernstein/Lewis – Chapter 17

Ramakrishnan/Gehrke – Chapter 15

Silberschatz/Korth/Sudarshan – Chapter 11

Ullman/Widom – Chapter 10.6 & 10.7

THE UNIVERSITY OF SYDNEY



Agenda

- The Problem / Motivation
- Data Warehousing
  - Issues and the ETL Process
- OLAP in Relational DBMS
  - Star Schema
  - CUBE Operator
- Data Lakes, Data Governance

Data Management in an Enterprise

- So far we have focused on one DBMS and the data it holds
- Typical enterprise has many databases stored among multiple DBMS instances (and indeed, from varying platforms)
  - Often, the data management follows organisational structure” each department has its own data, in a dedicated DBMS instance
  - Eg One database for HR department (information about employees etc), another for Finance department, (accounts etc) yet another for the logistics department (inventory table, etc)
- This may have arisen from history (each department got automated at different time) or from buying a software system that was written for a specific DBMS and schema.

Analytics

- Operational activities often can be done involving a single database
- However, managers are interested in insights that may depend on connecting information from different departments
  - Eg explore how profits are impacted by staffing levels
  - Eg look for seasonal trends in sales, that could be used to improve inventory management
- So there is a need to have a way to look at information that comes in different databases, managed by different DBMS instances
- Terms include data analytics, business analytics, data science, business intelligence, decision support,…

Data Analysis in the Enterprise

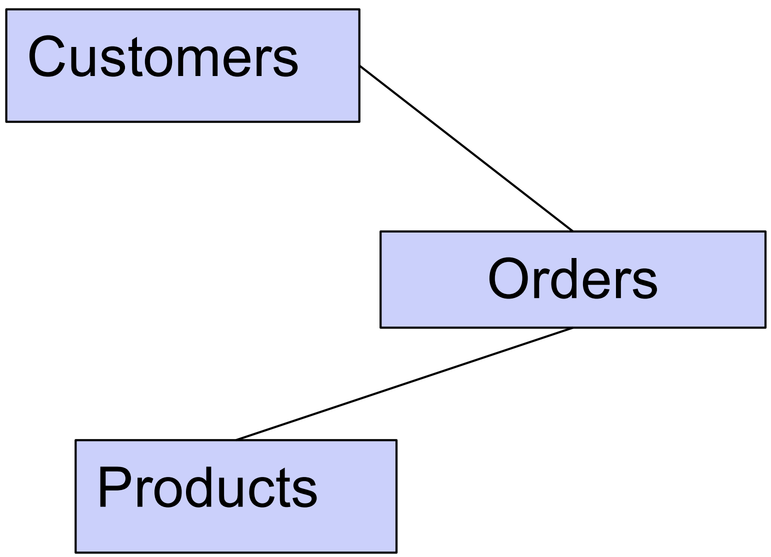
- Three Complementary Trends:
  - Data Warehousing: Consolidate data from many sources in one large repository.
    - Loading, periodic synchronization of replicas.
    - Semantic integration.
  - OLAP (Online Analytical Processing):
    - Complex SQL queries and views.
    - Interactive and “online” queries based on spreadsheet-style operations and “multidimensional” view of data.
  - Data Mining and Machine Learning: Exploratory search for interesting trends and anomalies. (Another unit!)

Comparison of OLTP and OLAP

- On Line Transaction Processing – ***OLTP***
  - Maintains a database that is an accurate model of some real-world enterprise. Supports day-to-day operations. Characteristics:
    - Short simple transactions
    - Relatively frequent updates
    - Transactions access only a small fraction of the database
- On Line Analytic Processing – ***OLAP***
  - Uses information in database to guide strategic decisions. Characteristics:
    - Complex queries
    - Infrequent updates
    - Transactions may access a large fraction of the database
    - Data need not be up-to-date
    - More historic data

OLTP vs OLAP (Example)

OLTP



OLAP

cust_id	orders	month	products

This is called a “fact table”

Example: The Internet Grocer

- OLTP-style transaction:
  - John Smith, from Schenectady, N.Y., just bought a box of tomatoes; charge his account; deliver the tomatoes from our Schenectady warehouse; decrease our inventory of tomatoes from that warehouse

- OLAP-style transaction:
  - Traditional
    - How many cases of tomatoes were sold in all northeast warehouses in the years 2019 and 2020?
  - Newer
    - Prepare a profile of the grocery purchases of John Smith for the years 2019 and 2020 (so that we can customize our marketing to him and get more of his business)

Data Mining

- *Data Mining* is an attempt at knowledge discovery – to extract knowledge from a database
- Comparison with OLAP
  - *OLAP*:
    - What percentage of people who make over $50,000 defaulted on their mortgage in the year 2000?
  - *Data Mining*:
    - How can information about salary, net worth, and other historical data be used to *predict* who will default on their mortgage?

Data Warehouses

- OLAP and data mining databases are frequently stored on special servers called ***data warehouses***:
  - A subject-oriented, integrated, time-variant, non-updatable collection of data used in support of management decision-making processes
  - Can accommodate the huge amount of data generated by OLTP systems
  - Allow OLAP queries and data mining to be run off-line so as not to impact the performance of OLTP

Why separate servers?

- The system internals (hardware and software) that work well for OLTP often don't perform well for OLAP, and vice versa
- The updates in OLTP often cause delays to the long-running report calculations of OLAP, and vice versa
- So, it makes sense to operate
  - one system optimized for OLAP, where OLAP runs
  - one system optimized for OLTP, where OLTP runs

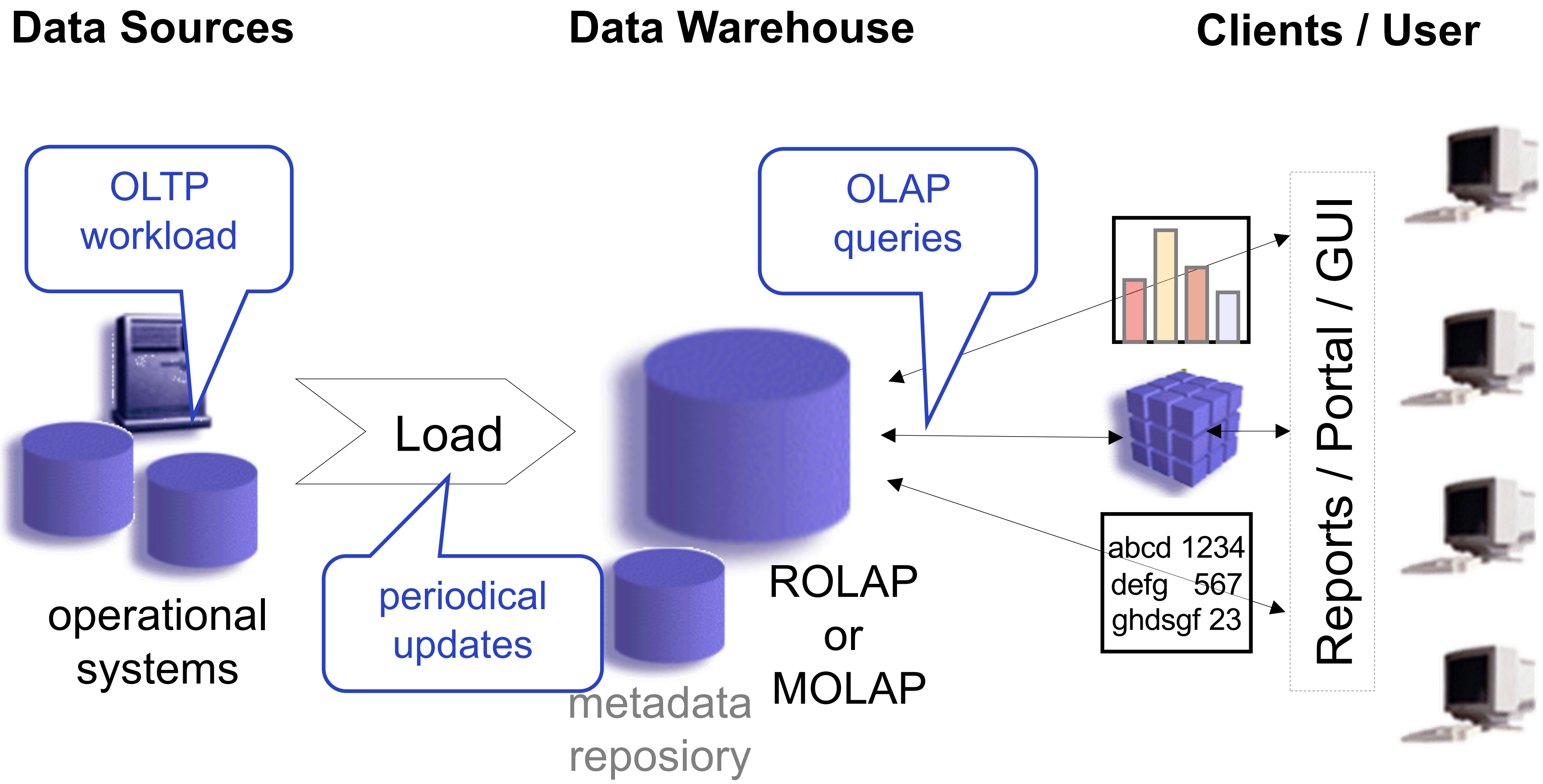
OLAP, Data Mining and Analysis

- The “A” in OLAP stands for “Analytical”
- Many OLAP and Data Mining applications involve sophisticated analysis methods from the fields of mathematics, statistical analysis, and artificial intelligence
- Our main interest is in the database aspects of these fields, not the sophisticated analysis techniques

Data Warehouse

- Data (often derived from OLTP) for both OLAP and data mining applications is usually stored in a special database called a ***data warehouse***
- Data warehouses are generally large and contain data that has been gathered at different times from DBMSs provided by different vendors and with different schemas
  - *Integrated* data spanning long time periods, often augmented with summary information.
  - Several gigabytes to terabytes common.
  - Interactive response times expected for complex queries; ad-hoc updates uncommon.
  - Read-only, periodically refreshed
- Populating such a data warehouse is not trivial

Data Warehousing



Issues in Data Warehousing

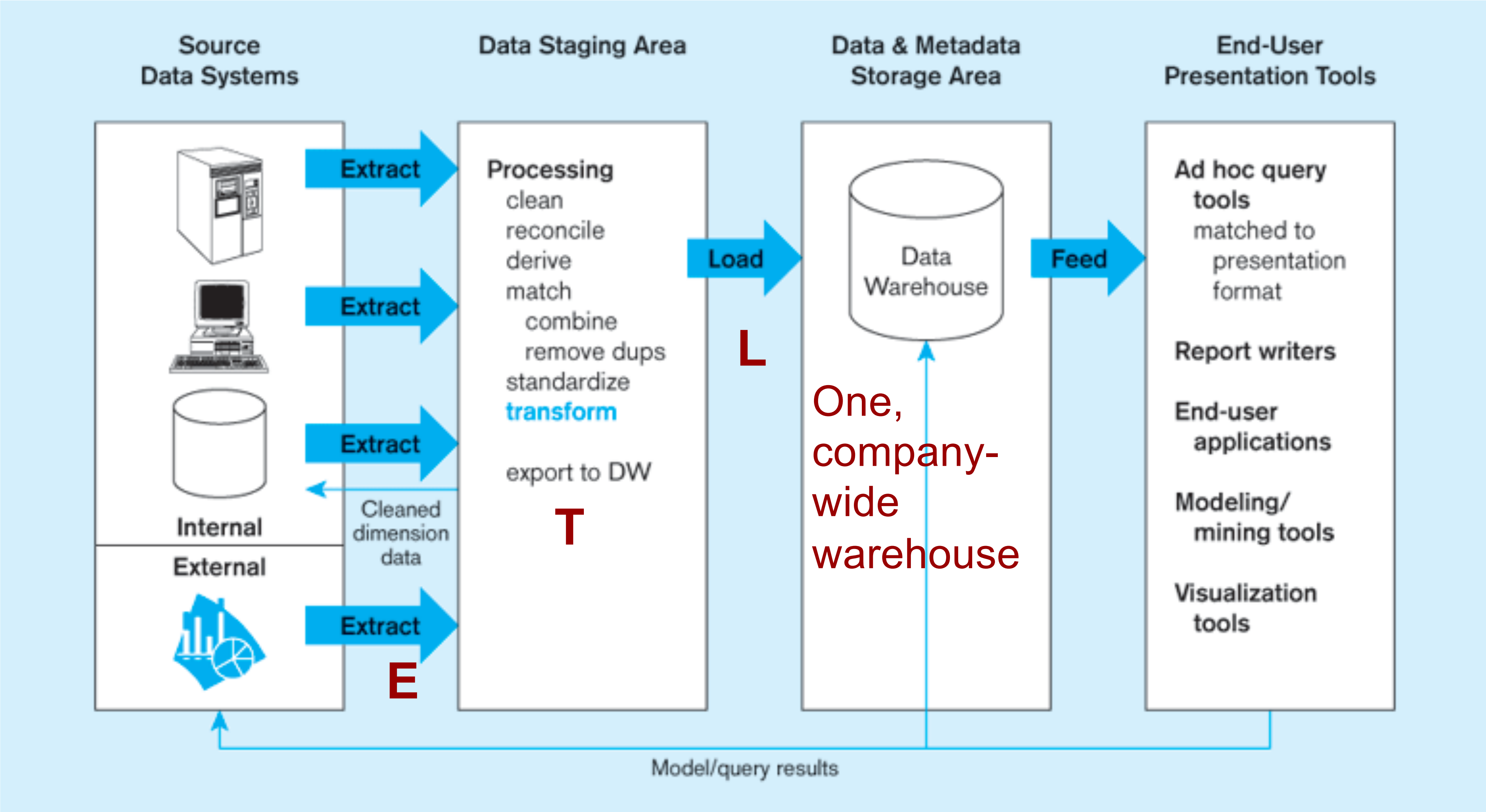
- **Semantic Integration:** When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemas.
  - *E.g. schema* used in different DMBSs for the same data might differ
    - Attribute names: SSN vs. Ssnum
    - Attribute domains: Integer vs. String
  - *Semantic*: semantics might be different
    - Summarizing sales on a daily basis vs. summarizing sales monthly basis
- **Heterogeneous Sources:** Must access data from a variety of source formats and repositories.
  - Replication capabilities can be exploited here.
- **Load, Refresh, Purge:** Must load data, periodically refresh it, and purge too-old data.
  - E.g. *Data Cleaning:* Removing errors and inconsistencies in data
- **Metadata Management:** Must keep track of source, loading time, and other information for all data in the warehouse.

ETL Process

- Typical operational data is:
  - Transient – not historical
  - Restricted in scope – not comprehensive
  - Sometimes poor quality – inconsistencies and errors
- **ETL (Extract-Transform-Load) Process**
  - Capture/Extract - Data Cleansing & Transform - Load
- After ETL, data should be:
  - Detailed – not summarized yet
  - Historical – periodic
  - Comprehensive – enterprise-wide perspective
  - In the right, uniform format of the data warehouse
  - Quality controlled – accurate with full integrity

Populating a Data Warehouse: ETL Process

ETL : Capture/**E**xtract, **T**ransform, and **L**oad



Periodic extraction ➔ data is not completely current in warehouse

Transform

- The Transform step must be automated, and there are many complex aspects
- Adjust structure for the warehouse target schema
  - May require joining source tables, splitting source fields etc
- Adjust values for the warehouse target schema
  - May involve calculations (eg deg C to deg F), lookup in translation tables (eg use currency exchange rates); also data cleaning to fix mistakes in source data
- ETL Tools often offer proprietary language for programming the Transform step

Metadata

- As with other databases, a warehouse must include a ***metadata repository***
  - Information about physical and logical organization of data
    - dimensions and facts
    - available reports and predefined queries
    - …
  - Also, keep information about the source of each data item and the dates on which it was loaded and refreshed
    - how data is derived from operational data store, including derivation rules
    - responsible people, etc.

Incremental Updates

- The large volume of data in a data warehouse makes loading and updating a significant task
- For efficiency, updating is usually incremental
  - Different parts are updated at different times
- Incremental updates might result in the database being in an inconsistent state
  - Usually not important because queries involve only statistical summaries of data, which are not greatly affected by such inconsistencies

ROLAP and MOLAP

- Relational OLAP: **ROLAP**
  - OLAP data is stored in a relational database
  - Accessed through SQL queries
  - Data cube is a conceptual view – way to *think about* a fact table

- Multidimensional OLAP: **MOLAP**
  - Vendor provides an OLAP server that *implements* a fact table as a data cube using a special multi-dimensional (non-relational) structure.
  - Multidimensional data is stored physically in a (disk-resident, persistent) array
  - No standard query language for MOLAP databases
  - Many MOLAP vendors (and many ROLAP vendors) provide proprietary visual languages that allow casual users to make queries that involve pivots, drilling down, or rolling up

- In the following slides, we will look at ROLAP in more detail

Fact Tables

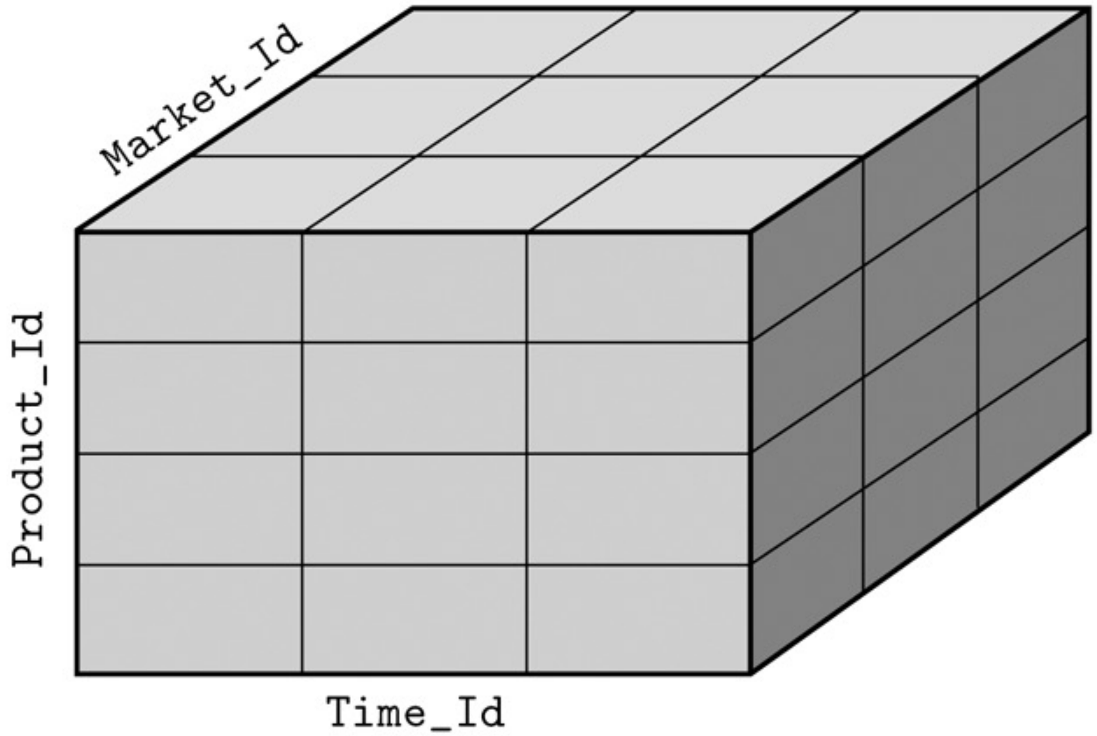
- Relational OLAP applications are based on a ***fact table***
  - For example, a supermarket application might be based on a table
    Sales (*Market_Id*, *Product_Id*, *Time_Id*, *Sales_Amt*)

market_id	product_id	time_id	sales_amt
M1	P1	T1	3000
M1	P2	T1	1000
M1	P3	T1	500
M2	P1	T1	100
M2	P2	T1	1100
M2	P3	…	…
…	…		

- The table can be viewed as *multidimensional*
  - Collection of numeric measures, which depend on a set of dimensions
    - E.g. *Market_Id*, *Product_Id*, *Time_Id* are the dimensions that represent specific supermarkets, products, and time intervals
    - *Sales_Amt* is a function of the other three

Data Cube

- Fact tables can be viewed as an N-dimensional *data cube* (3-dimensional in our example)
  - The entries in the cube are the values for *Sales_Amts*



Dimension Tables

- The dimensions of the fact table are further described with **dimension tables**
  - Supermarket Example: Fact table
    - Sales (*Market_id*, *Product_Id*, *Time_Id*, *Sales_Amt*)
  - Dimension Tables:
    - Market (*Market_Id*, *City, State, Region*)
    - Product (*Product_Id*, *Name, Category, Price*)
    - Time (*Time_Id*, *Week, Month, Quarter*)

- For each dimension, the set of values can be organized in a hierarchy:

Product

category
pname

Location

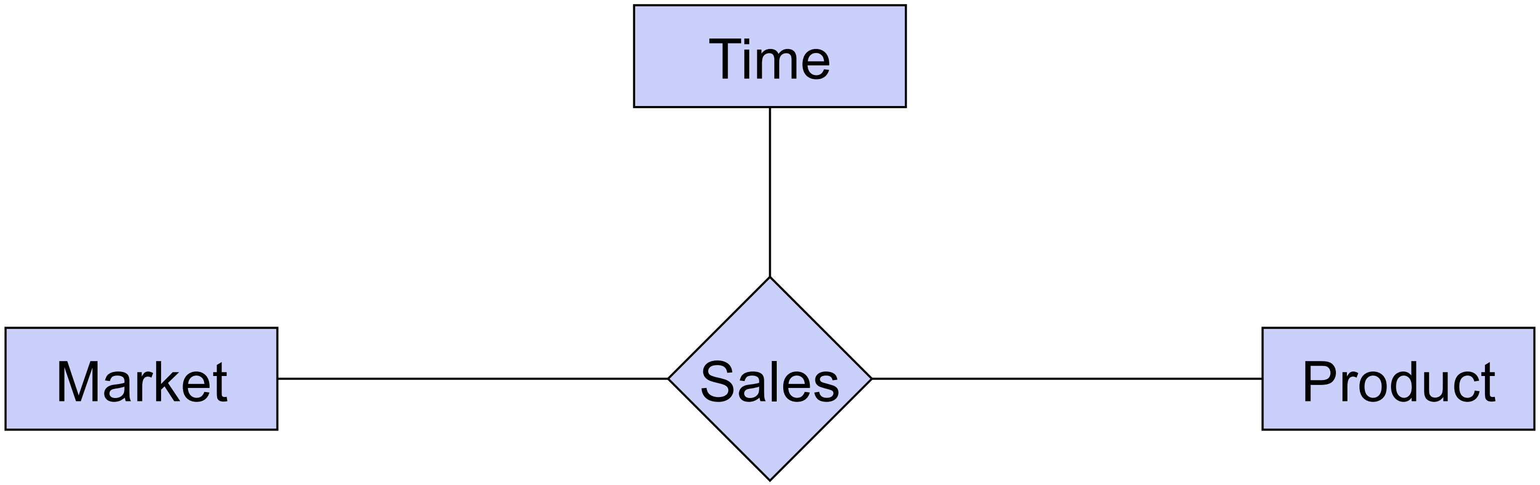
country
state
city

Time

year
quarter
week month
date

Star Schema

- The fact and dimension relations can be displayed in an E-R diagram, which looks like a star and is called a ***star schema***



- If we map this to relations
  - 1 central fact table
  - *n* dimension tables with foreign key relationships from the fact table

    (the fact table holds the FKs referencing the dimension tables)

OLAP Queries: Aggregation

- Many OLAP queries involve ***aggregation*** of the data in the fact table
- For example, to find the total sales (over time) of each product in each market, we might use

```
SELECT     S.Market_Id, S.Product_Id, SUM (S.Sales_Amt)
FROM       Sales S
GROUP BY   S.Market_Id, S.Product_Id
```

- The aggregation is over the entire time dimension and thus produces a two-dimensional view of the data. (Note: aggregation here is over time, not supermarkets or products.)

Aggregation over Time

- The output of the previous query

Market_Id

Product_Id

SUM(*Sales_Amt*)	M1	M2	M3	M4
P1	3003	1503	…	
P2	6003	2402	…	
P3	4503	3	…	
P4	7503	7000	…	
P5	…	…	…	

Drilling Down and Rolling Up

- Some dimension tables form an ***aggregation hierarchy***

  Market_Id -> City -> State -> Region

- Executing a series of queries that moves down a hierarchy (*e.g.,* from aggregation over regions to that over states) is called ***drilling down***
  - Requires the use of the fact table or information more specific than the requested aggregation (*e.g*., cities)
- Executing a series of queries that moves up the hierarchy (e.g., from states to regions) is called ***rolling up***
  - Note: In a rollup, coarser aggregations can be computed using prior queries for finer aggregations

Drilling Down

- Drilling down on market: from *Region* to *State*

  Sales (*Market_Id, Product_Id, Time_Id, Sales_Amt*)

  Market (*Market_Id, City, State, Region*)

1. ```
   SELECT    S.Product_Id, M.Region, SUM (S.Sales_Amt)
   FROM      Sales S, Market M
   WHERE     M.Market_Id = S.Market_Id
   GROUP BY  S.Product_Id, M.Region
   ```

2. ```
   SELECT    S.Product_Id, M.State, SUM (S.Sales_Amt)
   FROM      Sales S, Market M
   WHERE     M.Market_Id = S.Market_Id
   GROUP BY  S.Product_Id, M.State
   ```

Rolling Up

- Rolling up on market, from *State* to *Region*
  - If we have already created a table, State_Sales, using

1. SELECT S.*Product_Id*, M.*State*, SUM (S.*Sales_Amt*)
   FROM Sales S, Market M
   WHERE M.*Market_Id* = S.*Market_Id*
   GROUP BY S.*Product_Id*, M.*State*

then we can roll up from there to:

2. SELECT T.*Product_Id*, M.*Region*, SUM (T.*Sales_Amt*)
   FROM State_Sales T, Market M
   WHERE M.*State* = T.*State*
   GROUP BY T.*Product_Id*, M.*Region*

Can reuse the results of query 1.

Pivoting

- When we view the data as a multi-dimensional cube and group on a subset of the axes, we are said to be performing a *pivot* on those axes
  - Pivoting on dimensions $D_1,\ldots,D_k$ in a data cube $D_1,\ldots,D_k,D_{k+1},\ldots,D_n$ means that we use GROUP BY $A_1,\ldots,A_k$ and aggregate over $A_{k+1},\ldots A_n$, where A_i is an attribute of the dimension D_i
  - *Example*: Pivoting on Product and Time corresponds to grouping on *Product_id* and *Quarter* and aggregating *Sales_Amt* over *Market_id:*

```
SELECT     S.Product_Id, T.Quarter, SUM (S.Sales_Amt)
FROM       Sales S, Time T
WHERE      T.Time_Id = S.Time_Id
GROUP BY   S.Product_Id, T.Quarter
```

Slicing and Dicing

- When we use WHERE to specify a particular value for an axis (or several axes), we are performing a ***slice***
  - Slicing the data cube in the Time dimension (choosing sales only in week 12) then pivoting to *Product_id* (aggregating over *Market_id*)

```
SELECT    S.Product_Id,  SUM (Sales_Amt)
FROM      Sales S, Time T
WHERE     T.Time_Id = S.Time_Id  AND  T.Week = 'Wk-12'
GROUP BY  S. Product_Id
```

Slice

Pivot

Slicing and Dicing

- Typically slicing and dicing involves several queries to find the “right slice.”

  For instance, change the slice & the axes (from the prev. example):

  - Slicing on Time and Market dimensions then pivoting to *Product_id* and *Week* (in the time dimension)

```
SELECT      S.Product_Id,  T.Week,  SUM (Sales_Amt)
FROM         Sales S,  Time T
WHERE        T.Time_Id = S.Time_Id
                   AND  T.Quarter =  4
                   AND  S.Market_id = 12345
GROUP BY  S.Product_Id,  T.Week
```

Slice

Pivot

The CUBE Operator

- To construct the following table, would take 4 queries (next slide)

Market_Id

Product_Id

SUM(*Sales_Amt*)	M1	M2	M3	*Total*
P1	3003	1503	…	…
P2	6003	2402	…	…
P3	4503	3	…	…
P4	7503	7000	…	…
Total	…	…	…	…

The Four Queries

- For the table entries, without the totals (aggregation on time)

```
SELECT      S.Market_Id,  S.Product_Id,  SUM (S.Sales_Amt)
FROM        Sales S
GROUP BY    S.Market_Id, S.Product_Id
```

- For the row totals (aggregation on time and markets)

```
SELECT      S.Product_Id,  SUM (S.Sales_Amt)
FROM        Sales S
GROUP BY    S.Product_Id
```

- For the column totals (aggregation on time and products)

```
SELECT      S.Market_Id,  SUM (S.Sales)
FROM        Sales S
GROUP BY    S.Market_Id
```

- For the grand total (aggregation on time, markets, and products)

```
SELECT      SUM (S.Sales)
FROM        Sales S
```

Definition of the CUBE Operator

- Doing these three queries is wasteful
  - Generalizing the previous example, if there are *k* dimensions, we have 2^k possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.
  - The first does much of the work of the other two: if we could save that result and aggregate over *Market_Id* and *Product_Id*, we could compute the other queries more efficiently
- The CUBE clause is part of SQL:1999
  - GROUP BY CUBE (v1, v2, …, vn)
  - Equivalent to a collection of GROUP BYs, one for each of the 2^n subsets of v1, v2, …, vn

Example of CUBE Operator

- The following query returns all the information needed to make the previous products/markets table:

```
SELECT  S.Market_Id, S.Product_Id,        SUM (S.Sales_Amt)
FROM  Sales S
GROUP BY CUBE (S.Market_Id, S.Product_Id)
```

Data Warehouse Limits

- Data warehouse often has only a subset of enterprise data
- Warehouse has bounded volume
  - The software and hardware that runs an enterprise-scale data warehouse is very expensive
  - The free DBMS don't scale up well to huge datasets, and the commercial ones charge a lot, proportional to data size
  - So most enterprises limit the volume of data they keep in the DW
  - typically, not much more than one year of detailed data
- Warehouse setup is very expensive in effort
  - Choose schema, code the ETL, etc
  - This effort is done only for the most valuable data
- Warehouse technology often is relational, so limited in data types it handles well
  - A lot of enterprise data is text, image (scan of document, etc)

Data Lake

- The enterprise has lots of old data, of many types, and wants to get value from it
  - solution: a data lake
  - store all the old historic data, just as it comes, across lots of cheap CPUs and storage
  - run programs in Hadoop or similar analysis frameworks to explore this data
    - highly parallel, fault-tolerant (because with many cheap machines, crashes are common)
- The programs are often written as needed, for some investigation
  - And saved, in case they could be useful (directly, or modified a bit) for other investigations
  - Use tags, community mechanisms to find releavant information and analysis coce

Data Governance

- The enterprise needs to put considerable effort to tracking where its data is kept, and what is done with it
- Compliance
  - Rules vary from country to country (even state to state)
  - Rules about valid uses, conditions where data must be kept, conditions where data must be removed, conditions on reporting about data
  - Rules may be different for different kinds of data (personal, financial, etc)
- Security concerns
  - Sometimes required by government etc, but even when not, security is an important business need (risk mitigation)

Governance support

- Tools that track data location, uses, etc
- Security tools
- Processes such as audits, penetration testing etc

Summary

- Decision support is an emerging, rapidly growing subarea of databases.
- Involves the creation of large, consolidated data repositories called *data warehouses*.
  - Populating such warehouses is non-trivial (data integration etc.)
- Warehouses exploited using new analysis techniques: complex SQL queries and OLAP “multidimensional” queries (influenced by both SQL and spreadsheets).
- New techniques for database design, indexing, and analytical querying need to be supported.
  - Star Schema
  - Drill-down / Rollup queries; “slicing-and-dicing”
  - SQL:1999 support for CUBE and ROLLUP operators, as well as new WINDOW clause

References

- Kifer/Bernstein/Lewis (2nd edition)
  - Chapter 17.1-17.6
    a whole chapter dedicated to OLAP and data mining; the latter we do not cover in this lecture (but in depth in COMP5318). The slides here follow basically this chapter up-to 17.6 (except the window-query part, see below)
- Ramakrishnan/Gehrke (3rd edition - the 'Cow' book)
  - Chapter 25
    also a whole chapter dedicated to OLAP (star schemas, CUBE/ROLLUP). Only book that also covers several implementation details to improve the performance of OLAP queries such as bitmap&join indexes, materialized views, view maintenance, and a good part on WINDOW queries (it's the only book covering this at the moment; slides of this part are from here).
- Ullman/Widom (3rd edition – 'A First Course in Database Systems')
  - Chapter 10.6 and 10.7
    a short overview of OLAP, star schemas and the CUBE/ROLLUP operators in SQL. Easy to read with good examples, but meant as just an introduction.