Data analytics; data governance

ISYS2120 Data and Information Management

Prof Alan Fekete

University of Sydney

Acknowledge: slides from Uwe Roehm and Alan Fekete, and from the materials associated with reference books (c) McGraw-Hill, Pearson

Agenda

- **Motivation, define OLAP**
- **Data Warehousing**
  - **Issues and the ETL Process**
- **OLAP in Relational DBMS**
  - **Star Schema**
  - **Data cube representation (for display)**
  - **CUBE Operator**
  - **WINDOW clause**
- **Schema-late approaches (“Data Lakes” and “Lakehouses”)**
- **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 product quality is 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,…

Analytics

- Managers may need standard reports
  - Eg (university manager) average grade earned in each unit of study
  - Eg (sales manager) average sales of each product category, each month
- Managers also explore the data, looking for patterns or trends that can provide insights
  - These queries are not predictable, and often, the output from asking one, will lead to asking another related query

Data Analysis in the Enterprise

- Traditional Analytics:
  - Data Warehousing:
    - Consolidate data from many databases 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.
- Recent trends:
  - data mining and machine learning to find patterns and make predictions (techniques are not covered in isys2120 – instead study in comp3008, comp4318)
  - schema-late or schema-free data lakes

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
    - Transactions are well-known (parameters may vary)
    - Relatively frequent updates to data
    - Each transaction accesses only a small fraction of the database
    - Each transaction typically needs data about the current state of the domain (not the history over a long time)
- On Line Analytic Processing – ***OLAP***
  - Uses information in database to guide strategic decisions. Characteristics:
    - Complex queries
    - Infrequent updates (typically only on periodic refresh of data)
    - A single query may access a large fraction of the database
    - Data need not be perfectly up-to-date for query to be useful
    - Transactions often need a lot of historic data

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)

Agenda

- **Motivation, define OLAP**
- **Data Warehousing**
  - **Issues and the ETL Process**
- **OLAP in Relational DBMS**
  - **Star Schema**
  - **Data cube representation (for display)**
  - **CUBE Operator**
  - **WINDOW clause**
- **Schema-late approaches (“Data Lakes” and “Lakehouses”)**
- **Data Governance**

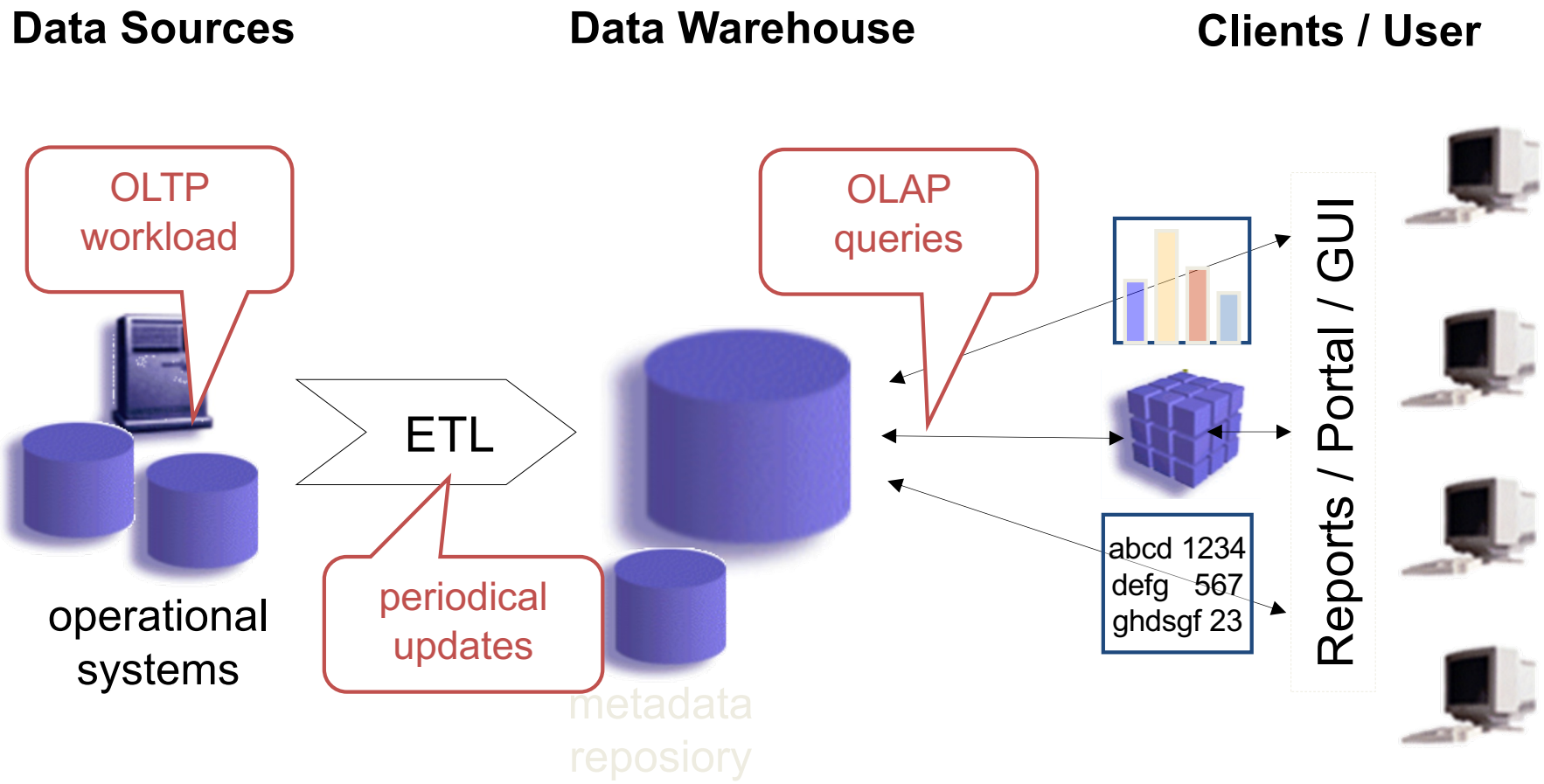
Data Warehouse

- OLAP (and data mining) is often done on a database which is stored on a special server called ***data warehouse***:
  - Used in support of management decision-making processes
  - *Integrated* data spanning long time periods, often augmented with summary information.
  - Can accommodate the huge amount of data generated by multiple OLTP systems
  - Interactive response times are expected for complex queries
  - Read-only, except for brief periods when contents are refreshed
  - Allow OLAP queries 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
- Recent research tries to find a system design that can support both efficiently (called HTAP "Hybrid transaction and analytics processing")

Data Warehousing



Issues in Data Warehousing

Populating the warehouse is not simple

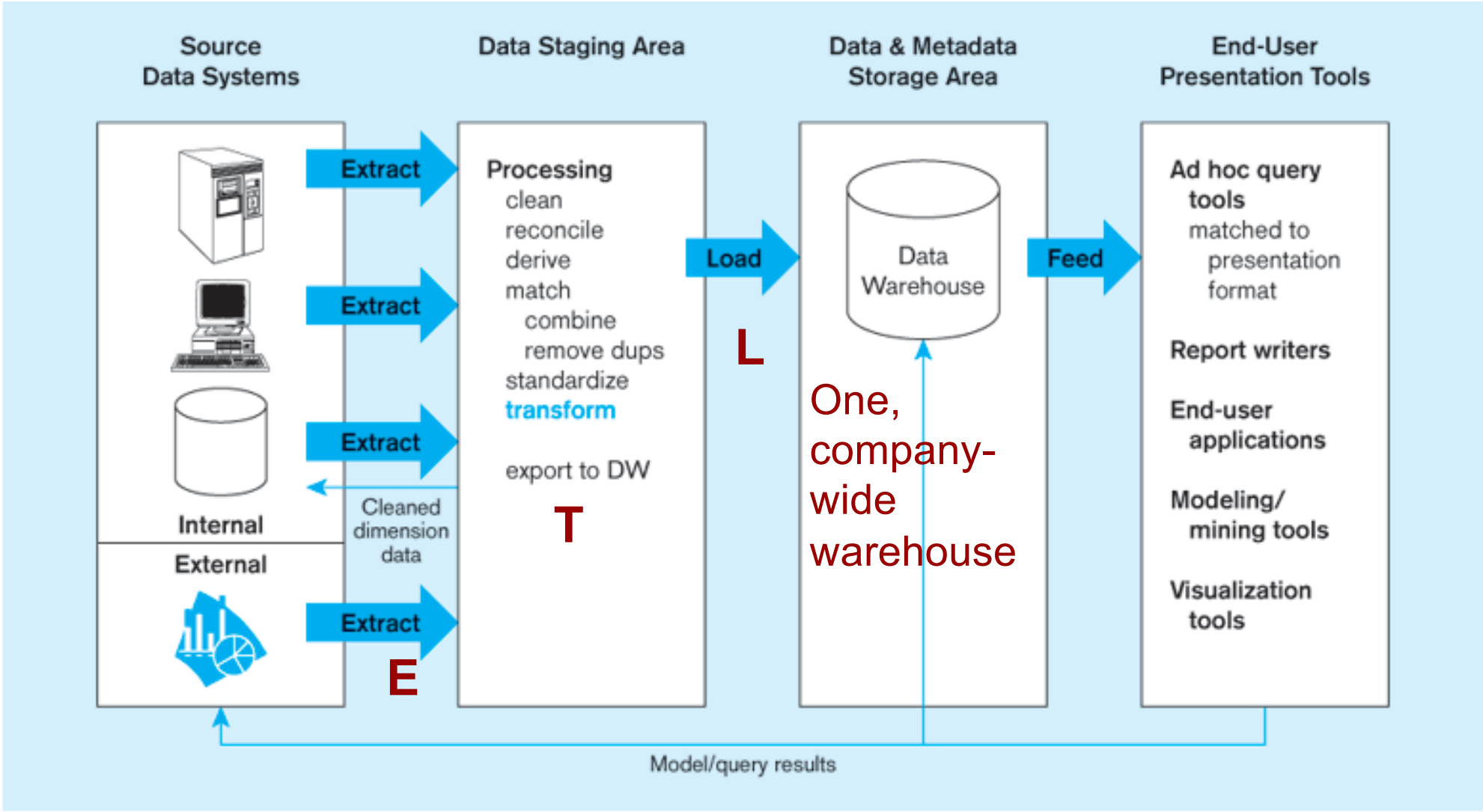
- **Semantic Integration:** When getting data from multiple sources, must eliminate mismatches, e.g., different currencies, schemas.
  - *E.g. schema* used in different DMBSs for the same data might differ
    - Attribute names: SSN vs. Ssnum
    - Attribute domains: Integer vs. String
  - *Semantic*: 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

Agenda

- **Motivation, define OLAP**
- **Data Warehousing**
  - **Issues and the ETL Process**
- **OLAP in Relational DBMS**
  - **Star Schema**
  - **Data cube representation (for display)**
  - **CUBE Operator**
  - **WINDOW clause**
- **Schema-late approaches (“Data Lakes” and “Lakehouses”)**
- **Data Governance**

Relational OLAP

- Managers often access the data warehouse through visual interfaces, that provide charts, direct manipulation, etc
  - Eg PowerBI, Tableau, etc
- However, underneath, the warehouse is often running a relational dbms
  - User actions are converted by the tool to SQL queries, and the results are converted to a more visual format for display

Fact Tables

- Relational OLAP applications are based on a ***fact table***
  - For example, a supermarket application might be based on a table

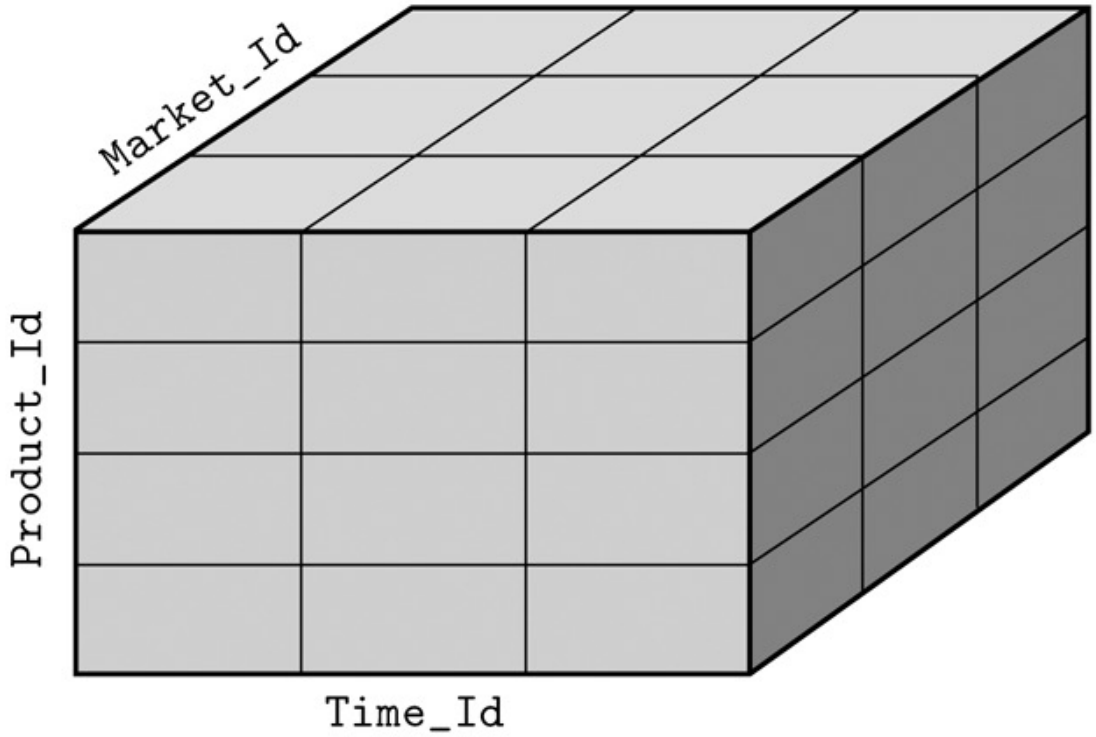
Sales (*Market_Id*, *Product_Id*, *Time_Id*, *Sales_Amt*)

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

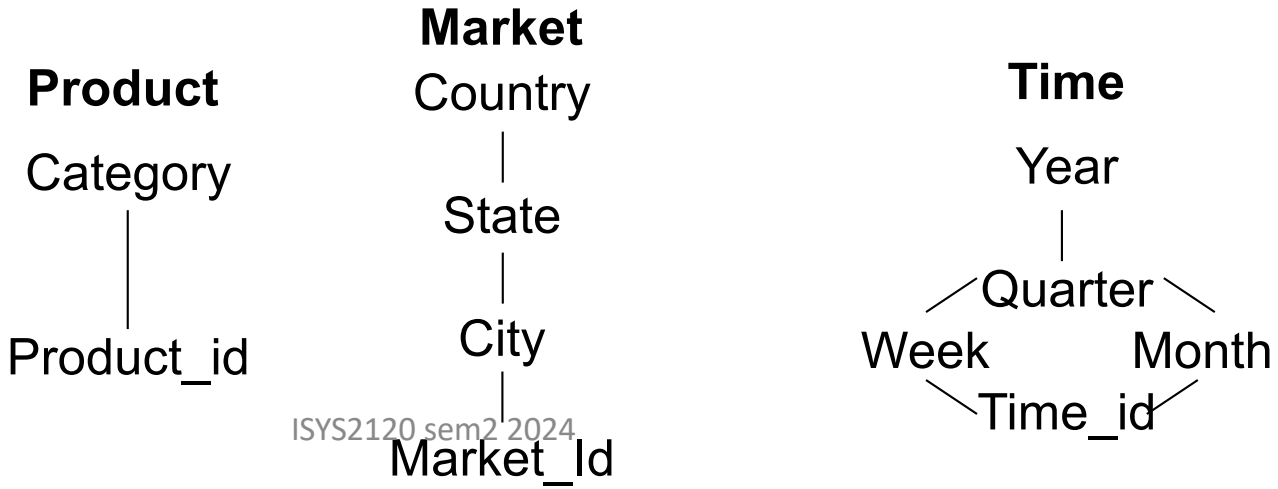
- Fact tables can be displayed as an N-dimensional *data cube* (3-dimensional in our example)
  - The entries in the cube are the values for *Sales_Amts*



Dimension Tables

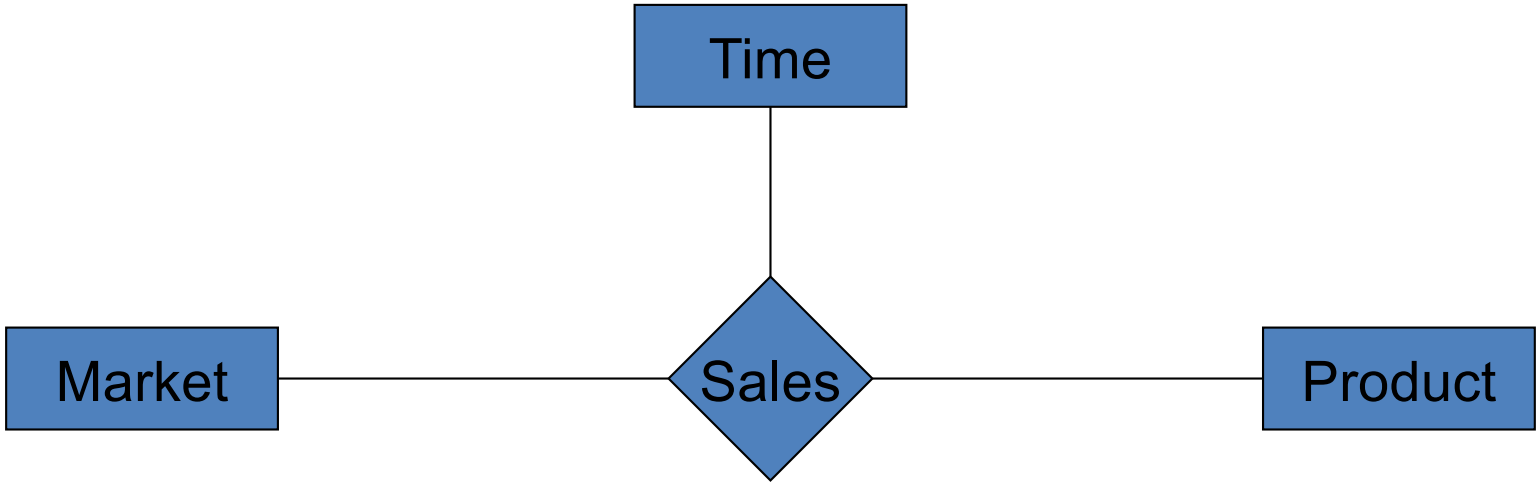
- The dimensions of the fact table are further described with **dimension tables**
  - Supermarket Example: Fact table
    - Sales (*Market_id*, *Product_Id*, *Time_Id*, *Sales_Amt*)
  - Dimension Tables:
    - Market (*Market_Id*, *City, State, Region*)
    - Product (*Product_Id*, *Name, Category,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

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



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, displayed as a "cube"

Market_Id

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

Product_Id

Drilling Down and Rolling Up

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

  Market_Id -> City -> State -> Country

- Executing a series of queries that moves down a hierarchy (*e.g.,* from aggregation over countries 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 countries) 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 *Country* to *State*

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

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

  This is often called "dicing" – it divides each cell into several smaller ones, which can reveal differences between

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

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 *Country*
  - 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.Country, SUM (T.Sales_Amt)
FROM      State_Sales T, Market M
WHERE     M.State = T.State
GROUP BY T.Product_Id, M.Country
```

This calculation in the system might reuse the results of query 1, rather than computing from scratch

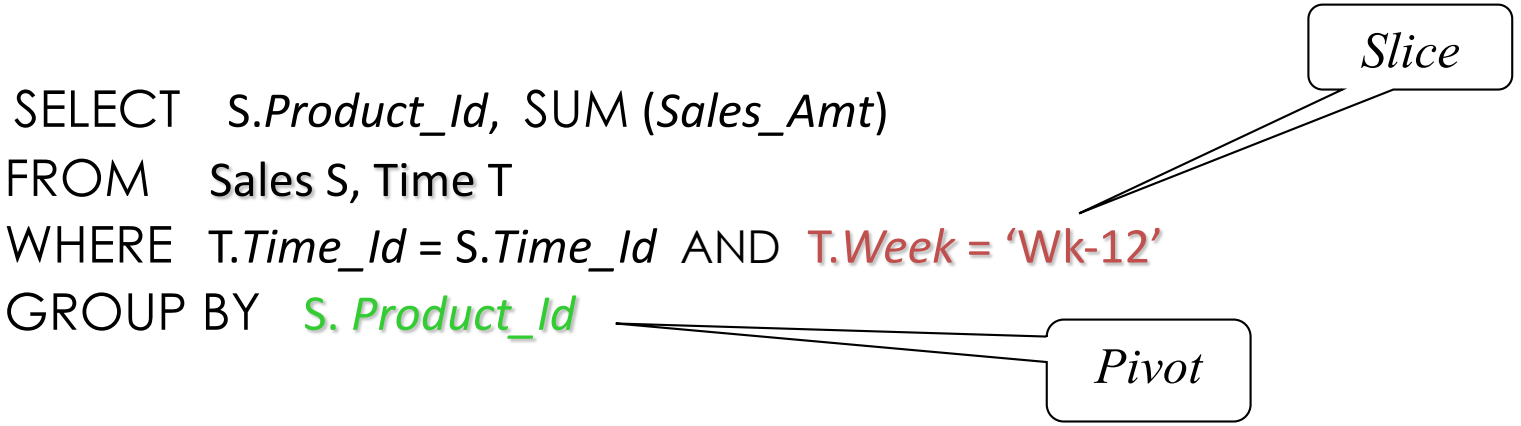
Pivoting

- 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

- When we to restrict our query to 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*)

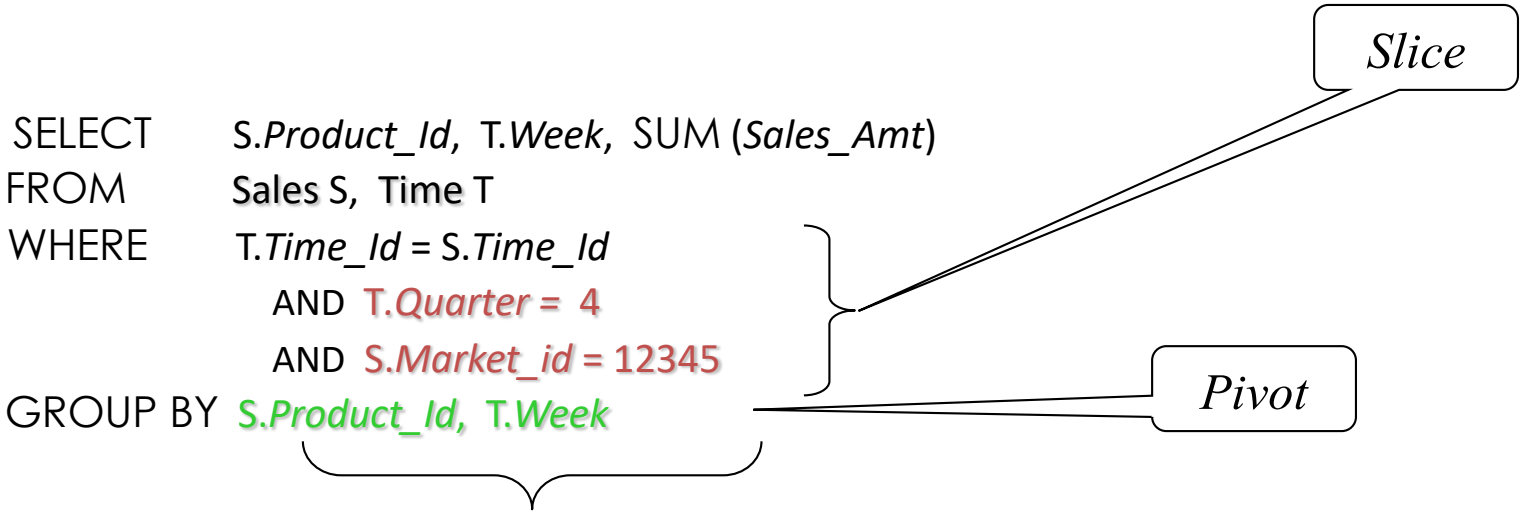


Slicing and Dicing

- Typically, exploring an issue involves a sequence of several queries doing slicing and dicing, to find some display which shows an interesting and meaningful pattern of difference between cells

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)



Cross-tab with marginals

- Understanding data is often helped if as well as the values for the cells, we see the totals for the cells of each row/column of the cube (these are called "marginal values")

Market_Id

SUM(*Sales_Amt*)	M1	M2	M3	*Total*
P1	3003	1503	…	…
P2	6003	2402	…	…
P3	4503	3	…	…
P4	7503	7000	…	…
Total	…	10908	…	…

Product_Id

Total of Sales in Market M2, over the products P1, P2, P3 and P4

Relational calculation of marginals

- To construct the values for a two-dimensional crosstab with marginals, would take 4 GROUP-BY 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	…	10908	…	…

Total of Sales in Market M2, over the products P1, P2, P3 and P4

The Four GROUP-BY 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)
```

- When seen as relational output (not displayed in cube format), the marginals are shown as rows which have NULL for any grouping column that is aggregated over

GROUPING SETs

- Latest SQL standard includes a generic GROUP BY extension that generalises from *cube*: **GROUPING SETS**
- Allows to explicitly specify a set of grouping operations which should be performed by a single query
  - Pro: Less overhead than individual OLAP queries; single scan of fact table only
- Example:

```
SELECT prod_id, cust_id, channel_id, SUM(quantity_sold)
  FROM Sales
 WHERE cust_id < 3
 GROUP BY GROUPING SETS (
    (prod_id), (cust_id, channel_id)
 );
```

Query Sequences in SQL:1999

- Trend analysis was difficult to do in the original SQL-92:
  - Find the % change in monthly sales
  - Find the top 5 product by total sales
  - Find the trailing *n*-day moving average of sales
  - The first two queries can be expressed with difficulty, but the third cannot even be expressed in SQL-92 if *n* is a parameter of the query.
- The WINDOW clause in SQL:1999 allows us to write such queries over a table viewed as a **sequence** (implicitly, based on user-specified sort keys)

The WINDOW Clause in SQL:1999

- A *window* is an *ordered* group of tuples around each (reference) tuple of a table.
- The *order* within a window is determined based on an attribute specified by the SQL statement.
- The *width* of the window is also specified by the SQL statement.
- The tuples of the window can be aggregated using the standard (set-oriented) SQL aggregate functions (SUM, AVG, COUNT, . . .).
- SQL:1999 also introduces some new (sequence-oriented) aggregate functions, in particular
  RANK, DENSE_RANK, PERCENT_RANK, NTILE, ROW_NUMBER.

Only concept is examinable, not syntax details!

Window Clause Example

```
SELECT M.state, T.month, AVG(S.sales) OVER W AS movavg
  FROM Sales S, Time T, Market M
 WHERE S.timeId=T.timeId AND S.marketId=M.marketId
WINDOW W AS (PARTITION BY M.state
             ORDER BY T.month
              ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)
```

- Let the result of the FROM and WHERE clauses be “Temp”.
- (Conceptually) Temp is partitioned according to the PARTITION BY clause.
  - Similar to GROUP BY, but the answer has one row for each row in a partition, not one row per partition!
- Each partition is sorted according to the ORDER BY clause.
- For each row in a partition, the WINDOW clause creates a “window” of nearby (preceding or succeeding) tuples.
  - Can be value-based, as in example, using RANGE
  - Can be based on number of rows to include in the window, using ROWS clause
- The aggregate function is evaluated for each row in the partition using the corresponding window.
  - New aggregate functions that are useful with windowing include RANK (position of a row within its partition) and its variants DENSE_RANK, PERCENT_RANK, CUME_DIST, NTILE(*n*).

Windowing vs. Grouping

- GROUP BY
  - partitions data into a **SET** of rows
  - which all share the same partition key(s)
  - Aggregation for whole partition results into **one** aggregate per group
    - 1 group, 1 aggregation
  - set-based aggregates

- OVER … WINDOW
  - partitions data to a **SEQUENCE** of rows
  - from a range 'around' the current reference row
  - Aggregation over window **per each** reference row
    - 1 window, *N* aggregations
  - set-based and new list-based aggregates

Not examinable

Further Examples

- SELECT SUM(sales_amt)
  FROM Sales

Sales
80863

- SELECT market_id, SUM(sales_amt)
  FROM Sales
  GROUP BY market_id

Market	Sales
M1	21012
M2	10909
M3	48942

- SELECT DISTINCT market_id, SUM(sales_amt) OVER W
  FROM Sales
  WINDOW W AS (ORDER BY market_id)

Market	Sales
M1	21012
M2	31921
M3	80863

  - Note the need for DISTINCT above;
    window gets computed *per each row* of Sales!

Not examinable

Semantics of Window Clause

```
SELECT AGG(…) OVER name FROM …
WINDOW name AS (
   [ PARTITION BY attributelist ]
   [ ORDER BY attributelist  ]
   [(RANGE|ROWS) BETWEEN v1 PRECEEDING AND v2 FOLLOWING] )
```

- Four Steps:
  - 1. Partitioning (PARTITION BY clause)
    - By which attribute shall the table be partitioned; Optional!
  - 2. Ordering (ORDER BY clause)
    - How should the rows in a window be ordered? optional.
  - 3. Framing (RANGE or ROWS clause) optional.
    - How large shall the window be '**around' the current reference row**?
    - RANGE defines window based on values of the order by attribute; requires a data type which allows addition and substraction such as INT or DATE)
    - ROWS defines window based on number of rows before/after
    - Possibly half-open window: UNBOUNDED PRECEEDING | FOLLOWING
  - 4. Aggregation (OVER clause) Note: anonymous window possible by using OVER(*window-def.*)

Not examinable

RANK Query Example

Rank our top sales

```
SELECT S.*, RANK() OVER (ORDER BY sales_amt
  DESC)
  FROM Sales S
```

By using aggregation inside the window definition, we can combine grouping with windowing:

Rank our top markets by total sales

```
SELECT market_id, SUM(sales_amt), RANK() OVER
  W
    FROM Sales
   GROUP BY market_id
  WINDOW W AS (ORDER BY SUM(sales_amt) DESC)
```

Not examinable

Tertiles Query Example

Categorise sales into 3 buckets

```
SELECT S.*,NTILE(3) OVER (ORDER BY sales_amt
  DESC)
  FROM Sales S;
```

Market	Prodcut	Sales	NTILE(3)
M3	P2	8002	1
M3	P1	5002	1
M2	P4	3334	2
M3	P4	3301	2
M1	P4	2502	3
M1	P2	2002	3

Note: You can configure the number of 'buckets' by the integer parameter on NTILE(), so NTILE(5) would give you quintile values

Not examinable

Rank() vs. Dense_Rank() vs. NTile() vs. Row_Number()

```
SELECT firstName, lastName, zipcode,
     ,ROW_NUMBER() OVER (ORDER BY zipcode) AS "Row Num"
     ,RANK()       OVER (ORDER BY zipcode) AS Rank
     ,DENSE_RANK() OVER (ORDER BY zipcode) AS "Dense Rank"
     ,NTILE(4)     OVER (ORDER BY zipcode) AS Quartile
FROM Person
```

firstName	Lastname	zipcode	Row Num	Rank	Dense Rank	Quartile
Jilian	Carson	2006	1	1	1	1
Peter	Pan	2006	2	1	1	1
Shu	Ito	2006	3	1	1	2
David	McDonald	2015	4	4	2	2
Lynn	Tsofliakes	2015	5	4	2	3
Mark	Reiter	2020	6	6	3	3
Rachel	Valdez	2020	7	6	3	4
Rajit	Pak	2037	8	8	4	4

ROLAP and MOLAP

- Our focus was 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, and display it to managers for decision making exploration

- Alternative platform type is 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 exists 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

Agenda

- **Motivation, define OLAP**
- **Data Warehousing**
  - **Issues and the ETL Process**
- **OLAP in Relational DBMS**
  - **Star Schema**
  - **Data cube representation (for display)**
  - **CUBE Operator**
  - **WINDOW clause**
- **Schema-late approaches (“Data Lakes” and “Lakehouses”)**
- **Data Governance**

Modern Trends in Analytics

- Data warehouse approach is limited
  - Deals with relational data
  - Decide in advance on a relational schema for the combined data, so ETL can get data into the warehouse
- important modern trends
  - Use of machine learning for finding patterns and making predictions, in data of many kinds
  - capture data for exploration, without first defining the schema

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?
  - See comp3008, comp4318

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)
  - Not well queries by SQL, but very suitable for machine learning

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 (do no work to choose common schema, integrate carefully, etc)
  - store across lots of cheap CPUs and storage
  - run programs in Hadoop or similar analysis frameworks to explore this data
- The programs ("scripts") 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 relevant information and analysis scripts
- The data lake usually lacks data management features such as access control, transactional updates, enforcing integrity constraints, etc
  - It may be hard to ensure data governance for the enterprise
  - Sometimes it is called a "data swamp"

Lakehouse

- A recent trend is for systems called “lakehouses” that support both structured and unstructured data of various types (such as tuples, graphs, text and images), with some level of data management support
  - combine features of data lake and data warehouse
- Metadata extraction
  - System may use machine learning techniques to infer structured data from the unstructured data [eg identify people who appear in certain images]
  - System may use machine learning techniques to infer “approximate schema” information from the unstructured data [eg identify that certain values are likely addresses]

Agenda

- **Motivation, define OLAP**
- **Data Warehousing**
  - **Issues and the ETL Process**
- **OLAP in Relational DBMS**
  - **Star Schema**
  - **Data cube representation (for display)**
  - **CUBE Operator**
  - **WINDOW clause**
- **Schema-late approaches (“Data Lakes” and “Lakehouses”)**
- **Data Governance**

Data Governance

- The enterprise needs to put considerable effort to tracking where its data is kept, and what is done with it
- The data can produce a lot of value, but, doing things badly can have serious consequences (poor decisions, violating laws, etc)
- Definitions
  - (rom https://en.wikipedia.org/wiki/Data_governance) "Data governance encompasses the people, processes, and information technology required to create a consistent and proper handling of an organization's data across the business enterprise."
  - (from https://datagovernance.com/the-data-governance-basics/definitions-of-data-governance/) "*Data Governance is a system of decision rights and accountabilities for information-related processes, executed according to agreed-upon models which describe who can take what actions with what information, and when, under what circumstances, using what methods.*"

DGI Data Governance Framework

From https://datagovernance.com/data-governance-framework-components/

- Mission and Values
- Beneficiaries
- Data products
  - Eg data catalogues, dashboards, “source of truth” datasets, etc
- Controls
- Accountabilities
- Decision Rights
- Policies and Rules
- Processes, Tools and Communications
- Work Program
- Participants

Aspects of Data Governance

- 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

References

- Kifer/Bernstein/Lewis(complete version, 2ed)
  - Chapter 17.1-17.4, 17.6
- Silberschatz/Korth/Sudarshan(7ed)
  - Chapters 11.1-11.3
- Ramakrishnam/Gehrke(3ed)
  - Ch 25.1-25.5, 25.7-25.8
- Garcia-Molina/Ullman/Widom(complete book, 2ed)
  - Chapter 10.6-10.7, 21.1-21.3
- *All these books also have some technical details on internals, eg data layouts so analytics can be calculated efficiently*

Summary

- Decision support is an emerging, rapidly growing subarea of databases
  - A variety of tools, user interfaces, that need to work across data from many sources
- Often involves the creation of large, consolidated data repositories called *data warehouses*
  - Populating such warehouses is non-trivial (ETL, data integration etc.)
- Warehouses exploited using 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 operator, as well as new WINDOW clause