Contents

[Introduction: 7](#_Toc41576518)

[Characteristics of valuable information: 7](#_Toc41576519)

[Centralization vs Localization: 7](#_Toc41576520)

[Operational and historic data: 7](#_Toc41576521)

[Business Intelligence: 8](#_Toc41576522)

[Definition: 8](#_Toc41576523)

[Data Warehouse: 8](#_Toc41576524)

[Data warehouse defined: 8](#_Toc41576525)

[OLAP: 8](#_Toc41576526)

[Types of data analysis: 9](#_Toc41576527)

[Applications: 9](#_Toc41576528)

[How Amazon works: 9](#_Toc41576529)

[Data Modelling: 10](#_Toc41576530)

[Find your entities: 10](#_Toc41576531)

[Choosing entities: 10](#_Toc41576532)

[Resolving multiple values: 10](#_Toc41576533)

[Finding attributes: 10](#_Toc41576534)

[Identifiers: 10](#_Toc41576535)

[Identifying relationships: 11](#_Toc41576536)

[Entity Grid: 11](#_Toc41576537)

[Cardinality: 11](#_Toc41576538)

[Entity-relationship diagrams: 11](#_Toc41576539)

[Multi-valued attributes: 12](#_Toc41576540)

[Resolving relationships: 12](#_Toc41576541)

[Resolving a many-to-many relationship: 12](#_Toc41576542)

[Why eliminate redundancy? 12](#_Toc41576543)

[Applying constraints to an attribute 13](#_Toc41576544)

[Constraints in SQL Server: 13](#_Toc41576545)

[Data Warehouse Design: 14](#_Toc41576546)

[Knowledge Discovery in Databases (KDD): 14](#_Toc41576547)

[How to get to Business Intelligence: 14](#_Toc41576548)

[Data Integration Component: 14](#_Toc41576549)

[Data Integration Conflicts: 14](#_Toc41576550)

[The Warehousing Approach: 15](#_Toc41576551)

[Dimensional modelling: 15](#_Toc41576552)

[Star schema: 16](#_Toc41576553)

[Star schema essentials: 16](#_Toc41576554)

[Star join – template: 16](#_Toc41576555)

[Star join essentials: 16](#_Toc41576556)

[Why star joins? 16](#_Toc41576557)

[Star Schema development steps: 17](#_Toc41576558)

[Data extractions: 17](#_Toc41576559)

[Online Analytical Processing (OLAP): 18](#_Toc41576560)

[Data extractions 18](#_Toc41576561)

[Data Warehousing and End-User Access Tools: 18](#_Toc41576562)

[Data extractions: 18](#_Toc41576563)

[Introducing OLAP: 18](#_Toc41576564)

[Examples of OLAP Applications in Various Functional Areas: 19](#_Toc41576565)

[The key differences between relational and OLAP databases: 19](#_Toc41576566)

[OLAP key features: 19](#_Toc41576567)

[OLAP Applications -Multi-Dimensional Views of Data: 19](#_Toc41576568)

[OLAP Applications -Support for Complex Calculations: 19](#_Toc41576569)

[OLAP Applications –Time Intelligence: 19](#_Toc41576570)

[OLAP Benefits: 20](#_Toc41576571)

[Representing Multi-Dimensional Data: 20](#_Toc41576572)

[Example of two-dimensional query 20](#_Toc41576573)

[Multi-Dimensional OLAP Servers: 21](#_Toc41576574)

[Categories of OLAP Tools: 21](#_Toc41576575)

[Multi-Dimensional OLAP (MOLAP): 21](#_Toc41576576)

[Typical Architecture for MOLAP Tools: 22](#_Toc41576577)

[Relational OLAP (ROLAP): 22](#_Toc41576578)

[Typical Architecture for ROLAP Tools: 22](#_Toc41576579)

[Building a data warehouse: 23](#_Toc41576580)

[To build a dimensional database: 23](#_Toc41576581)

[Data Mining: 24](#_Toc41576582)

[Data Mining Definition: 24](#_Toc41576583)

[Data mining and data warehousing: 24](#_Toc41576584)

[Examples: What is (not) Data Mining? 24](#_Toc41576585)

[What is not Data Mining? 24](#_Toc41576586)

[What is Data Mining? 24](#_Toc41576587)

[Database Processing vs. Data Mining Processing: 25](#_Toc41576588)

[Query examples: 25](#_Toc41576589)

[Data Mining Process: 25](#_Toc41576590)

[Data Mining Problems: 25](#_Toc41576591)

[Data Mining Techniques - Summary 26](#_Toc41576592)

[Data Mining Algorithms: 27](#_Toc41576593)

[Data Mining Algorithms: 27](#_Toc41576594)

[Data Mining Algorithms: 27](#_Toc41576595)

[The mining model could be any of these: 27](#_Toc41576596)

[Choosing an Algorithm by Type: 28](#_Toc41576597)

[Choosing an Algorithm by Task: 28](#_Toc41576598)

[Data Mining Modeling and Language: 30](#_Toc41576599)

[Data Mining Language: 30](#_Toc41576600)

[SQL Revolution (1970’s): 30](#_Toc41576601)

[DMX Approach: 30](#_Toc41576602)

[Defining a DM Model: 30](#_Toc41576603)

[Training (processing) a DM Model: 31](#_Toc41576604)

[Training a DM Model: Simple 31](#_Toc41576605)

[Data Management DBMS security: 32](#_Toc41576606)

[Security properties: 32](#_Toc41576607)

[Threats: 32](#_Toc41576608)

[Security controls: 32](#_Toc41576609)

[Access control: 32](#_Toc41576610)

[Authorization: 32](#_Toc41576611)

[Integrity: 32](#_Toc41576612)

[Views: 33](#_Toc41576613)

[Backup & Recovery: 33](#_Toc41576614)

[Encryption: 33](#_Toc41576615)

[Raid: 33](#_Toc41576616)

[Creating secure environments: 33](#_Toc41576617)

[Approach to secure thinking: 33](#_Toc41576618)

[Database Risk Management: 34](#_Toc41576619)

[Procedures & Practices: 34](#_Toc41576620)

[Database security strategy: 34](#_Toc41576621)

[Top security myths: 34](#_Toc41576622)

[Database Lifecycle and DBMS Selection: 35](#_Toc41576623)

[Requirements collection and analysis: 35](#_Toc41576624)

[Database Design: 35](#_Toc41576625)

[Database design – Criteria for an Optimal Data Model: 35](#_Toc41576626)

[DBMS Selection: 36](#_Toc41576627)

[Business requirements imply DBMS requirements: 36](#_Toc41576628)

[Essential requirements: 36](#_Toc41576629)

[Implementation: 36](#_Toc41576630)

[Testing: 36](#_Toc41576631)

[Maintenance corrective, perfective, adaptive 36](#_Toc41576632)

[Operational issues: 37](#_Toc41576633)

[Professional, Legal, and Ethical Issues: 38](#_Toc41576634)

[Ethical issues: 38](#_Toc41576635)

[Ethics in the context of information technology: 38](#_Toc41576636)

[Difference between ethical and legal behavior: 38](#_Toc41576637)

[Ethical behavior in information technology: 38](#_Toc41576638)

[British Computer Society: 38](#_Toc41576639)

[Laws: 39](#_Toc41576640)

[Legislation to consider for IT Professionals: 39](#_Toc41576641)

[The United Kingdom’s Data Protection Act 2018 – part of GDPR: 39](#_Toc41576642)

[EU General Data Protection Regulations (GDPR): 39](#_Toc41576643)

[Freedom of Information Act 2000: 39](#_Toc41576644)

[Copyright, Designs and Patent Act 1988: 40](#_Toc41576645)

[Intellectual Property (IP): 40](#_Toc41576646)

[Regulation of Investigatory Powers Act (RIPA) 2000: 41](#_Toc41576647)

[The Telecommunications Regulations 2003: 41](#_Toc41576648)

[Intellectual Property Rights (IPR): 41](#_Toc41576649)

[Establishing a culture of legal and ethical data stewardship: 41](#_Toc41576650)

[Graph Database: 42](#_Toc41576651)

[RDF and the semantic web: 42](#_Toc41576652)

[Querying semantic web: 42](#_Toc41576653)

[Other models: 42](#_Toc41576654)

[Graph data models: 42](#_Toc41576655)

[Attributes: 43](#_Toc41576656)

[A labelled and directed attributed multigraph. 43](#_Toc41576657)

[Demands on a graph database: 43](#_Toc41576658)

[Big Data – why it is becoming important: 44](#_Toc41576659)

[Query Processing: 45](#_Toc41576660)

[Basic Steps in Query Processing: 45](#_Toc41576661)

[Basic Steps: Optimization 45](#_Toc41576662)

[Phases of query processing: 46](#_Toc41576663)

[Query Decomposition: 46](#_Toc41576664)

[Analysis: 46](#_Toc41576665)

[Example RAT: 47](#_Toc41576666)

[Normalization: 47](#_Toc41576667)

[Semantic Analysis: 47](#_Toc41576668)

[Simplification: 47](#_Toc41576669)

[Measures of Query Cost: 48](#_Toc41576670)

[Cost estimation for RA Operations: 48](#_Toc41576671)

[Selection Operation – Linear Search: 48](#_Toc41576672)

[Selection Operation – Binary Search: 49](#_Toc41576673)

[Physical Structure: 50](#_Toc41576674)

[Indexes on tables: 50](#_Toc41576675)

[Why physical storage matters 50](#_Toc41576676)

[How queries make use of physical details: 50](#_Toc41576677)

[Databases and Files in SQL Server: 50](#_Toc41576678)

[Logical and Physical File Names: 50](#_Toc41576679)

[Databases and Files in MySQL: 51](#_Toc41576680)

[The Page Header: 51](#_Toc41576681)

[The Row offset array: 51](#_Toc41576682)

[Classification of Physical Storage Media: 51](#_Toc41576683)

[Physical Storage Media: 53](#_Toc41576684)

[Cache 53](#_Toc41576685)

[Main memory: 53](#_Toc41576686)

[Flash memory: 53](#_Toc41576687)

[Magnetic disk: 53](#_Toc41576688)

[Optical storage 53](#_Toc41576689)

[Tape storage 53](#_Toc41576690)

[Magnetic Disks: 54](#_Toc41576691)

[RAID: 54](#_Toc41576692)

[Storage Access: 54](#_Toc41576693)

[Buffer Manager 54](#_Toc41576694)

[Organization of Records in Files: 55](#_Toc41576695)

[HEAP table behaviour: 55](#_Toc41576696)

[Sequential File Organization: 55](#_Toc41576697)

[Indexes: 56](#_Toc41576698)

[Basic Concepts: 56](#_Toc41576699)

[Ordered Indexes: 56](#_Toc41576700)

[Dense Index Files: 56](#_Toc41576701)

[Sparse Index Files: 56](#_Toc41576702)

[B+-Tree Index Files: 57](#_Toc41576703)

[The insert algorithm for B+ Tree: 57](#_Toc41576704)

[Relational Algebra Symbols: 58](#_Toc41576705)

[Query Optimization: 59](#_Toc41576706)

[Steps of Query Processing: 59](#_Toc41576707)

# Introduction:

## Characteristics of valuable information:

* If information is to be valuable to an organization, it must have the following characteristics, (adapted from Ralph M. Stair's book, Principles of Information Systems):
  + **Accurate** information is free from error.
  + **Complete** information contains all of the important facts.
  + **Economical**. Information should be relatively inexpensive to produce.
  + **Flexible** information can be used for a variety of purposes, not just one.
  + **Reliable** information is dependable information.
  + **Relevant** information is important to the decision-maker.
  + **Simple**. Information should be simple to find and understand.
  + **Timely**. Timely information is readily available when needed.
  + **Verifiable**. Verifiable information can be checked to make sure it is accurate.

## Centralization vs Localization:

* The database approach is one of centralization
* The database is a central resource, accessed when needed
* What problems could this centralization cause?
* Are there other approaches?

## Operational and historic data:

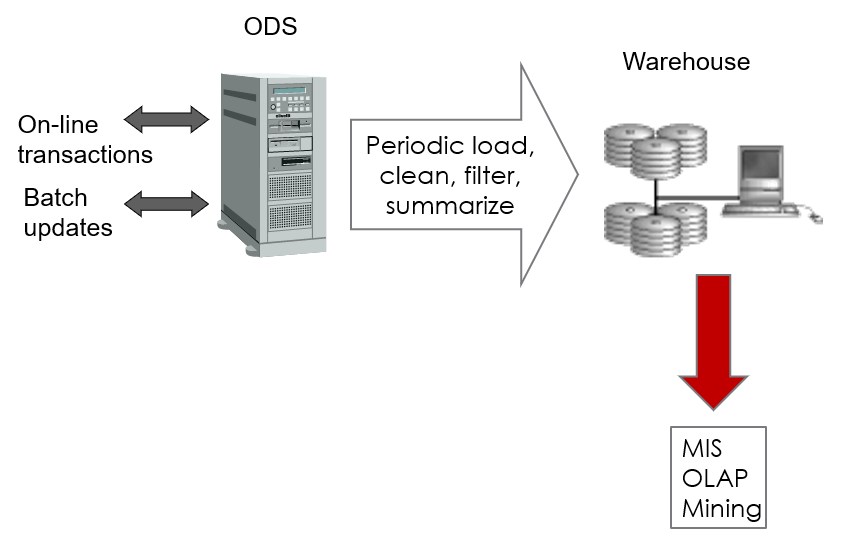
* Businesses are continually using operational databases to support their everyday business processes
* Over time this inevitably causes a build-upof historic data
* This historic data may be
  1. a storage problem
  2. a valuable resource
     + Which is it?

# Business Intelligence:

## Definition:

“Business intelligence (BI) is a broad category of application programs and technologies for gathering, storing, analyzing, and providing access to data to help enterprise users make better business decisions. BI applications include the activities of decision support, query and reporting, online analytical processing (OLAP), statistical analysis, forecasting, and data mining.”  
[www.sauder.ubc.ca/cgs/itm/itm\_glossary.html](http://www.sauder.ubc.ca/cgs/itm/itm_glossary.html)

## Data Warehouse:



## Data warehouse defined:

* A **data** **warehouse** is a database geared towards the **business intelligence** requirements of an organisation.
* The data warehouse integrates data from the **various operational systems** and is typically loaded from these systems at regular intervals.
* Data warehouses contain **historical information** that enables analysis of business performance **over time** (trends).
* Review information pyramid

## OLAP:

* **OLAP** stands for **Online Analytical Processing**. It is an approach to provide the answer to analytical queries that are dimensional in nature – quickly usually via a GUI e.g. Fedscope

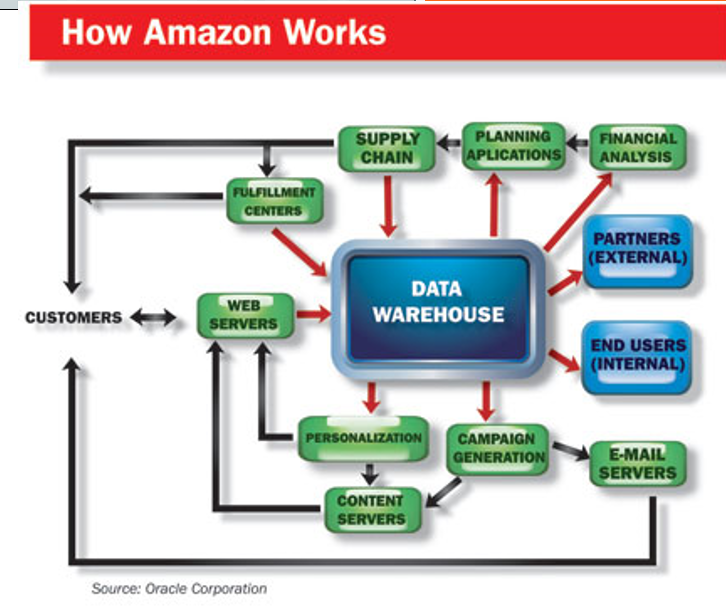
## Types of data analysis:

* **retrospective analysis**
  + What happened and why?
* **predictive analysis**
  + What might happen?
* **identifying patterns over time**
  + Are there trends that can be identified?
* **clustering and segmentation**
  + Can we divide our customers into groups?
* **Associations**
  + Are there significant correlations between entities / events?

## Applications:

* comparisons with the competition
* marketing
* customer profiling
* cross-selling
* risk assessment
* fraud detection

## How Amazon works:



# Data Modelling:

## Find your entities:

* an entity is anything which is likely to be of interest in the context of your system
* it could be a “real world” entity such as a car
* it could be a business entity such as a rental agreement
* it could be an abstract entity such as a “hire period”

### Choosing entities:

* in order for something to be worth modelling as an entity it should have the following properties:
  + the user is interested in its properties – it is important to the system being studied
  + there are many instances of the same entity type (usually)
  + all instances of the type have the same set of properties
  + it is uniquely identifiable in some way
* Beware - many texts will use the terms “entity” and “entity type” as if they were synonyms
  + use “entity instance” instead of “entity” when  
     you want to be precise

### Resolving multiple values:

* The attributes defined for each entity should not **repeat**
  + For one value of the key attributes there should not be multiple values of other attributes. If this happens the entity is not in 1st Normal Form. A new entity should be created, and the repeating group moved to that entity. The original entity will be master to the new entity’s detail

### Finding attributes:

* attributes are the **properties** which describe the entity
* for example, a car has the attributes: colour, capacity, model …
* more precisely we can say that every entity of type car will have attribute values for each of those attributes
  + e.g. yellow, 1275cc, mini

### Identifiers:

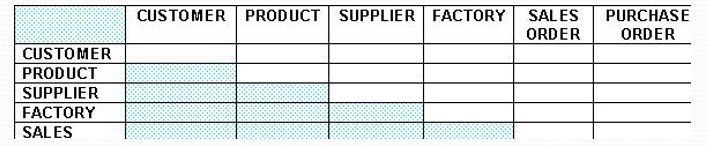
* An identifier is an attribute (or group of attributes) which uniquely identifies an entity instance
* For example, a car is uniquely identified by a registration number
  + (is it always?)
* Some entities are quite difficult to identify
  + e.g. a person - suggest an identifier
* but remember that an entity **must** have  
   an identifier

## Identifying relationships:

* A relationship is an association between two entities
* For example, there is a relationship between a car and the person that owns it
* More precisely, we model a relationship type “owns” between a person type and a car type
* Each combination of entities needs to be examined to see if there is a relationship between them. There are two approaches to this: entity grid (or matrix) and building the diagram directly

### Entity Grid:

* A rigorous approach which uses a grid to force analysis of every pair entities and determine the relationship between them
* Once the entities have been selected, they are drawn up on a grid
* The bottom left corner of the grid is hatched to avoid considering relationships twice

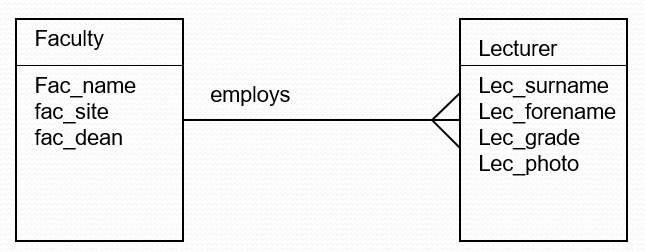


## Cardinality:

* Relationships have a property called cardinality - generally we model cardinality as one-to-one, one-to-many or many-to-many
* for example
  + each module has one and only one syllabus and a syllabus is used on only one module (one-to-one)
  + a lecturer is employed by one faculty; a faculty employs many lecturers (one-to-many)
  + a student studies many modules, a module is studied by   
    many students (many-to-many)

## Entity-relationship diagrams:

* entity types are represented by boxes; relationship types by lines between boxes; a crow’s foot represents the “many” end



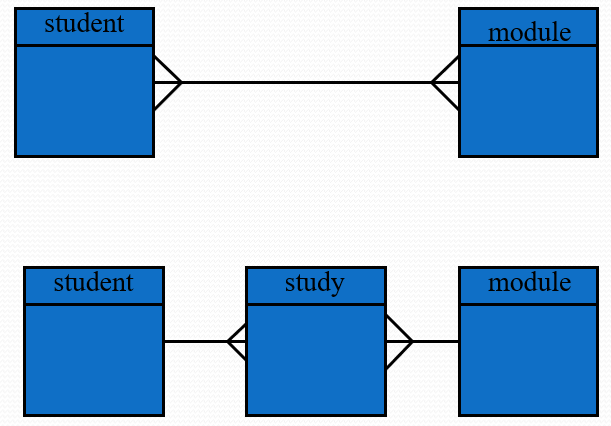
## Multi-valued attributes:

* for sound, practical reasons many practitioners argue against the use of multi-valued attributes
* for example a sales order may have several lines, each with an item code, description, price etc.
* these attributes are therefore multi-valued for a given order
* the solution is to invent a new entity type: “order-line” with a relationship to order

## Resolving relationships:

* the relational model works on the basis of one-many relationships
* eliminate one-to-one relationships by collapsing into a single entity type
* eliminate many-to-many relationships by introducing a junction, or link, entity

## Resolving a many-to-many relationship:



## Why eliminate redundancy?

* eliminate redundancy by integrating the files so that multiple copies are not stored
* repetition of records introduces more possibility of error
* once only storage means updates to values occur only once and the new value is available immediately to all users
* space is maximised & economies of scale
* improved data sharing
* improved data integrity – application of constraints
* improved security – access restricted to operation type
* improved backup and recovery

## Applying constraints to an attribute

* constraints are rules that determine what values the table attributes can assume
* you can prevent people from entering invalid data
* every value in the domain must satisfy all the constraints
* constraints are a method of applying business rules
* in SQL a **check constraint** (also known as **table check constraint**) is a condition that defines valid data when adding or updating an entry
* Can be used to insert a default value

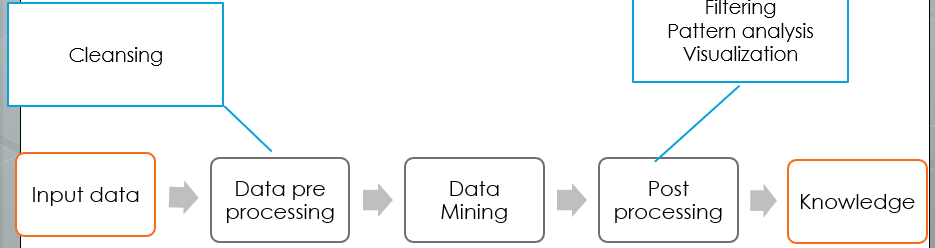
## Constraints in SQL Server:

SQL Server supports the following classes of constraints:

* NOT NULL specifies that the column does not accept NULL values
* CHECK constraints enforce domain integrity by limiting the values that can be put in a column
  + A CHECK constraint specifies a Boolean (evaluates to TRUE, FALSE, or unknown) search condition that is applied to all values that are entered for the column. All values that evaluate to FALSE are rejected. You can specify multiple CHECK constraints for each column.
* UNIQUE constraints enforce the uniqueness of the values in a set of columns.
  + In a UNIQUE constraint, no two rows in the table can have the same value for the columns. Primary keys also enforce uniqueness, but primary keys do not allow for NULL as one of the unique values.
* PRIMARY KEY constraints identify the column or set of columns that have values that uniquely identify a row in a table
  + No two rows in a table can have the same primary key value. You cannot enter NULL for any column in a primary key.

# Data Warehouse Design:

## Knowledge Discovery in Databases (KDD):



## How to get to Business Intelligence:

* Data mining techniques, technologies and algorithms are being developed to provide BI
* Before we get to data mining we must first pre process the data to get it into the right form
* This will often involve the design of a data warehouse:
  + Design the data warehouse architecture
  + Extract and cleanse the data
  + Integrate the data into a common data mart schema

## Data Integration Component:

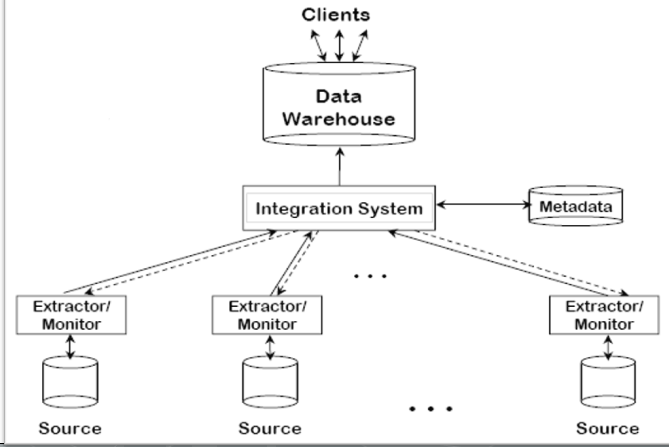
* Data Type Differences
* Value differences   
   (Colour:Black (*0*, or *BL*, or *Black*))
* Semantic Differences  
   ( Terms -> Different interpretations)
* Missing Values (NULL VALUES)
* Different Schemata

## Data Integration Conflicts:

* Schema level conflicts are due to
  + different perceptions
  + different focus
* Value level conflicts are due to
  + different representations, coding, etc.
  + different precision
  + incorrect information
  + data entry errors
  + Data cleaning

## The Warehousing Approach:

* Information integrated in advance
* Stored in WH for direct querying and analysis



## Dimensional modelling:

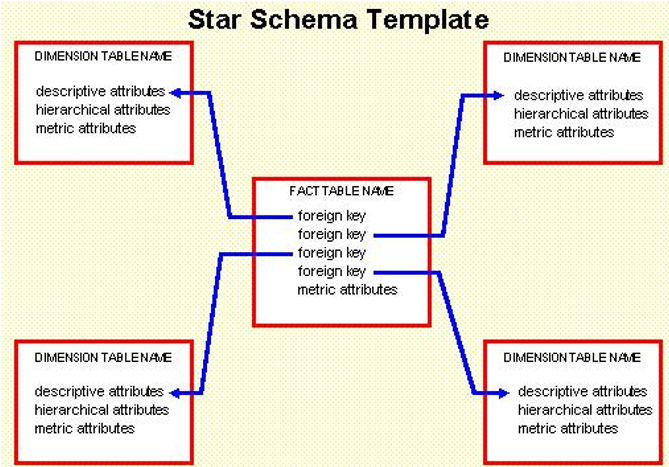
* Dimensional modelling - presenting the data in a standard form
* Each Dimensional Model
  + One fact table
  + Dimension tables
  + Star schema or Star join

## Star schema:

### Star schema essentials:

* One fact table
* Multiple dimension tables
* Each dimension gives a different *focus* to the data
* Different functional areas of the business might be interested in different aspects of the same data so would have a different star schema that suited them

## Star join – template:



## Star join essentials:

* Again - **fact** tables and **dimension** tables.
* The **fact table (1)** represents the structure that holds the majority of the *occurrences* of the data. Fact tables typically combine data and cross reference keys from a variety of other tables.
* **Dimension tables (\*)** contain data which is not terribly voluminous. Dimension tables are related to fact tables by means of a foreign key relationship.
* Now for each *sales fact* you can find out wh*a*t was sold to *whom*, *where* and *when*.

## Why star joins?

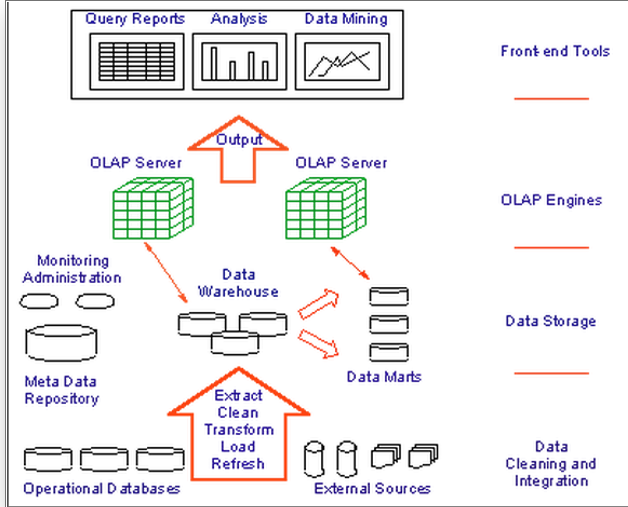
* By building star joins, the designer has created a structure for efficient access of large volumes of data and natural end-user viewing.
* Problem with star joins.
  + In order to know how to create the star join, the designer must make assumptions about the usage of the data.
  + One department will look at data very differently from another department. The star join for finance will be very different than the star join for production, for example.

## Star Schema development steps:

1. Choose the process
2. Choose the grain
3. Identify the dimensions
4. Choose the facts

## Data extractions:

* Data mart software will often include extraction and visualization tools
* These are given the umbrella term OLAP
* **O**n-**L**ine **A**nalytical **P**rocess



# Online Analytical Processing (OLAP):

## Data extractions

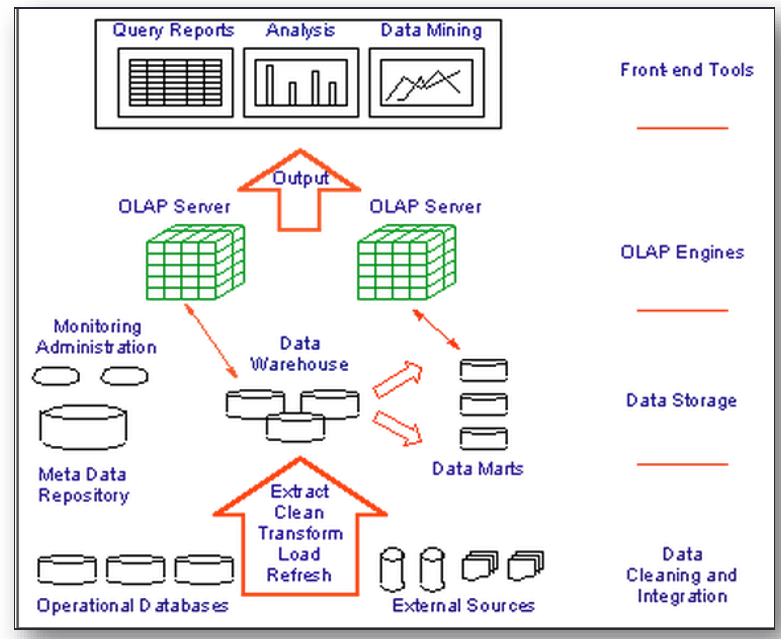
* Data warehouse software will often include extraction and visualization tools .
* These are given the umbrella term OLAP .
* On-Line Analytical Processing.

## Data Warehousing and End-User Access Tools:

Accompanying growth in data warehouses is increasing demand for more powerful access tools providing advanced analytical capabilities

* Key developments include:
  + Online analytical processing (OLAP)
  + SQL extensions for complex data analysis
  + Data mining tools

## Data extractions:



## Introducing OLAP:

The dynamic synthesis, analysis, and consolidation of large volumes of multi-dimensional data, Codd (1993).

Describes a technology that uses a multi-dimensional view of aggregate data to provide:

* quick access to strategic information.
* for purposes of advanced analysis.

## Examples of OLAP Applications in Various Functional Areas:

|  |  |
| --- | --- |
| Functional area: | Examples of OLAP applications: |
| Finance | Budgeting, activity-based costing, financial performance analysis and financial modelling. |
| Sales | Sales analysis and sales forecasting. |
| Marketing | Market research analysis, sales forecasting, promotions analysis, customer analysis and market / customer segmentation |
| Manufacturing | Production planning & Defect analysis. |

## The key differences between relational and OLAP databases:

|  |  |
| --- | --- |
| **Relational Database (OLTP)** | **Dimensional Database (OLAP)** |
| Data is atomized | Data is summarized |
| Data is current | Data is historical |
| Processes one record at a time | Processes many records at a time |
| Process Oriented | Subject Oriented |
| Designed for Highly structured repetitive processing | Designed for highly unstructured analytical processing. |

## OLAP key features:

Although OLAP applications are found in widely divergent functional areas, all have following key features:

* multi-dimensional views of data;
* support for complex calculations;
* time intelligence (changes over time -> trend identification)

## OLAP Applications -Multi-Dimensional Views of Data:

Core requirement of building a ‘realistic’ business model

* Provides basis for analytical processing through flexible access to corporate data
* A dimensional data model is harder to maintain for very large data warehouses than a relational data model.
* For this reason, data warehouses typically are based on a relational data model. However, a dimensional data model is particularly well-suited for building data marts (a subset of data warehouse).

## OLAP Applications -Support for Complex Calculations:

* Must provide a range of powerful computational methods such as that required by sales forecasting.
* Mechanisms for implementing computational methods should be clear.

## OLAP Applications –Time Intelligence:

* Key feature of almost any analytical application as performance is almost always judged over time.
* Time hierarchy is not always used in same manner as other hierarchies.
* Concepts such as year-to-date and period-over-period comparisons should be easily defined.

## OLAP Benefits:

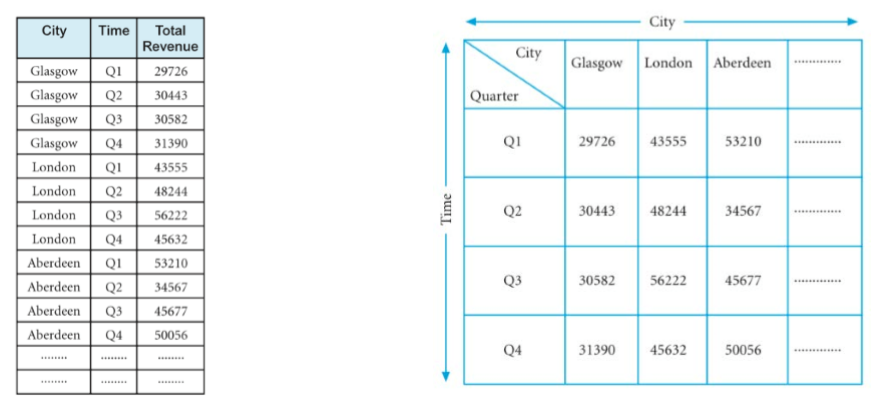
* Increased productivity of end-users – because end users can query them with IT expertise.
  + Therefore reduced backlog of applications development for IT staff.
* Retention of organizational control over the integrity of corporate data.
  + Reduced query drag and network traffic on OLTP systems.
  + Improved potential revenue and profitability.

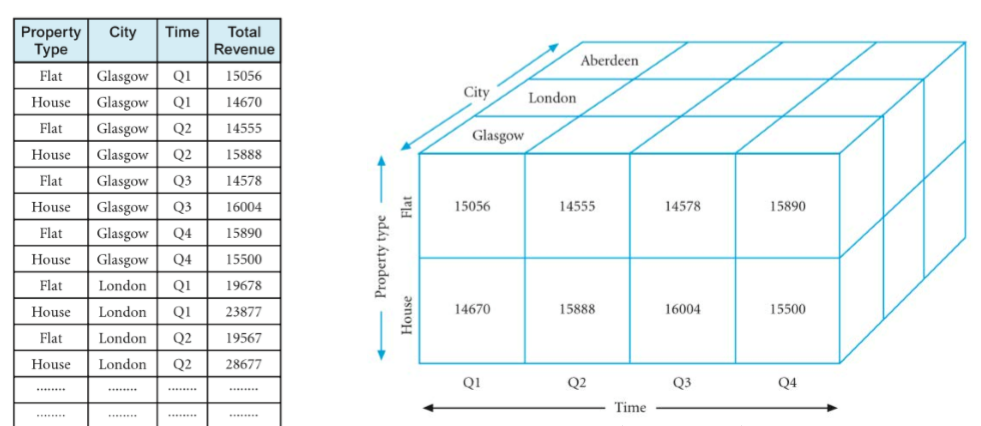
## Representing Multi-Dimensional Data:

### Example of two-dimensional query

What is the total revenue generated by sales in each region, in each quarter of 20xx?’

* Choice of representation is based on types of queries end-user may ask .
* Compare representation - three-field relational table versus two-dimensional matrix.





## Multi-Dimensional OLAP Servers:

* Use multi-dimensional structures to store data and relationships between data.
* Multi-dimensional structures are best visualized as cubes of data, and cubes within cubes of data. Each side of cube is a dimension .
* A cube supports matrix arithmetic .
* Multi-dimensional query response time depends on how many cells have to be added ‘on the fly’.
* Majority of multi-dimensional queries use summarized, high-level data
* Solution is to pre-aggregate (consolidate) all logical subtotals and totals along all dimensions
* Pre-aggregation is valuable, as typical dimensions are hierarchical in nature
  + (e.g. Time dimension hierarchy - years, quarters, months, weeks, and days)
* Predefined hierarchy allows logical pre-aggregation and, conversely, allows for a logical ‘drill-down’.
* Supports common analytical operations
  + Consolidation
  + Drill-down
  + Slicing and dicing

**Consolidation**: aggregation of data such as simple ‘rollups’ or complex expressions involving inter-related data

**Drill-Down**: is reverse of consolidation and involves displaying the detailed data that comprises the consolidated data

**Slicing & Dicing:** refers to the ability to look at the data from different viewpoints

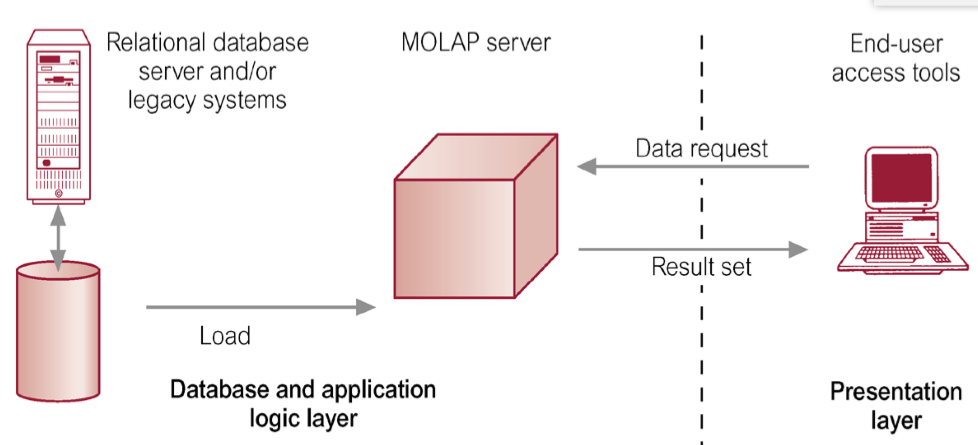
## Categories of OLAP Tools:

* OLAP tools are categorized according to the architecture of the underlying database
* Main categories of OLAP tools include
  + Multi-dimensional OLAP (MOLAP or MD-OLAP)
  + Relational OLAP (ROLAP), also called multi-relational OLAP

## Multi-Dimensional OLAP (MOLAP):

* Uses specialized data structures and multi-dimensional Database Management Systems (MDDBMSs) to organize, navigate, and analyse data.
* Data is typically aggregated and stored according to predicted usage to enhance query performance.
* Use array technology and efficient storage techniques that minimize the disk space requirements through sparse data management
* Provides excellent performance when data is used as designed, and the focus is on data for a specific decision support application

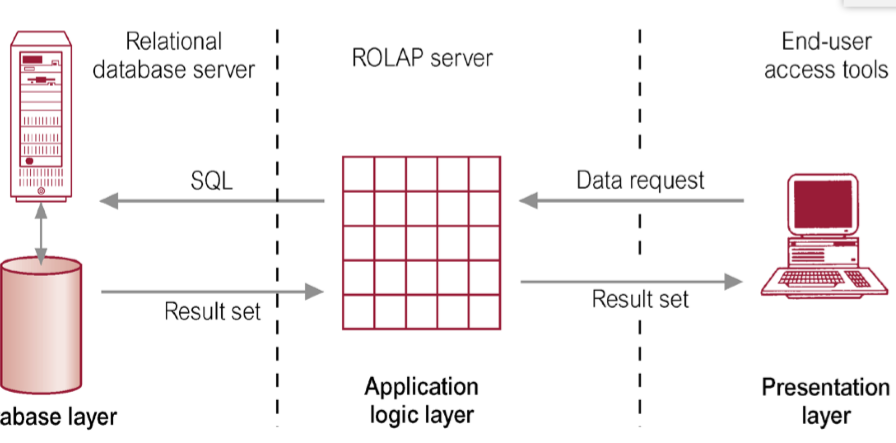
## Typical Architecture for MOLAP Tools:



## Relational OLAP (ROLAP):

* Fastest growing style of OLAP technology
* Supports RDBMS products using a metadata layer avoids need to create a static multi-dimensional data structure - facilitates the creation of multiple multidimensional views of the two-dimensional relation.
* To improve performance, some products use SQL engines to support complexity of multi-dimensional analysis, while others recommend, or require, the use of highly denormalized database designs such as the star schema.

## Typical Architecture for ROLAP Tools:



## Building a data warehouse:

* To build a dimensional data model, you need a methodology that outlines the decisions you need to make to complete the database design.
* This methodology uses a top-down approach because it first identifies the major processes in your organization where data is collected.
* An important task of the database designer is to start with the existing sources of data that your organization uses. After the processes are identified, one or more fact tables are built from each business process.

## To build a dimensional database:

* Choose the business processes that you want to use to analyse the subject area to be modelled.
* Determine the granularity of the fact tables.
* Identify dimensions and hierarchies for each fact table.
* Identify measures for the fact tables.
* Determine the attributes for each dimension table.
* Get users to verify the data model.

# Data Mining:

## Data Mining Definition:

* Mining
  + Act of excavation in the earth from which ore or minerals can be extracted
* Data Mining
  + Act of excavation in the data from which patterns can be extracted
  + The process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions
* Alternative name: Knowledge discovery in databases (KDD)
* Multiple disciplines: database, statistics, artificial intelligence
* Fast maturing technology
* Unlimited applicability
  + Retail/Marketing
    - Identifying buying patterns
    - Predicting response to mailing campaigns
  + Banking
    - Detecting patterns of fraudulent card use
    - Determining credit card spending by customer groups

## Data mining and data warehousing:

* Data mining requires a single, separate, clean, and integrated source of data
* A data warehouse is suitable because:
  + They are populated with clean and consistent data
  + They contain data from different sources
  + Its query capabilities are required

## Examples: What is (not) Data Mining?

### What is not Data Mining?

* + Look up phone number in phone directory

* + Query a Web search engine for information about “Amazon”

### What is Data Mining?

* + Certain names are more prevalent in certain US locations (O’Brien, O’Rurke, O’Reilly… in Boston area)
  + Group together similar documents returned by search engine according to their context (e.g. Amazon rainforest, Amazon.com,)

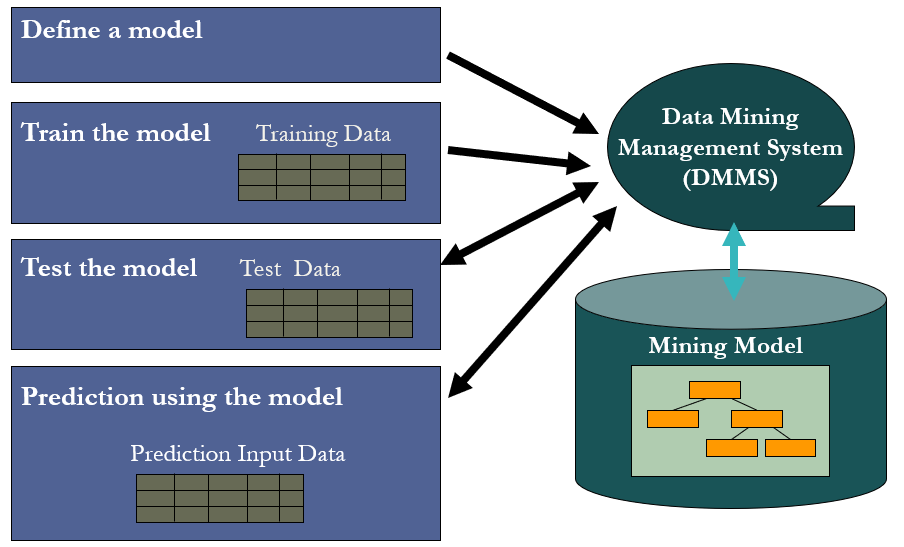
## Database Processing vs. Data Mining Processing:

|  |  |
| --- | --- |
| * Query   + Well defined   + SQL | * Query   + Poorly defined   + No precise query language |
| * Output   + Precise   + Subset of database | * Output   + Fuzzy   + Not a subset of database |

## Query examples:

* Database
  + Find all *credit applicants* with last name of *Smith*
  + Identify *customers* who have purchased *more than $10,000 in the last month*
  + Find all *customers* who have *purchased* *milk*
* Data mining
  + Find all *credit applicants* with *high credit risks*
  + Identify *customers* with *similar buying habits*
  + Find all *items* which are *frequently purchased with milk*

## Data Mining Process:



## Data Mining Problems:

* *Is this student going to go to a college?*
  + Based on Gender, ParentIncome, ParentEncouragement, IQ, etc.
  + E.g., if ParentEncouragement=Yes and IQ>100, College=Yes
  + **Classification (prediction) ) – what might you do with THIS information?**
* Similar questions:
  + Is this a spam email? (spam filtering)
  + How good/bad is your credit? (credit scoring)
  + Recognition of hand-written letters (pen recognition)
  + What is this gene like? (bioinformatics)
  + Does this person behave like a terrorist? (TIA)
* *What is the age of a person?*
  + Based on Hobby, MaritalStatus, NumberOfChildren, Income, HouseOwnership, NumberOfCars, …
  + e.g., If MaritalStatus=Yes, Age = 20+4\*NumberOfChildren+0.0001\*Income+…

**🡺 Regression (prediction) ) – what might you do with THIS information?**

* Similar questions:
  + What’s the sales amount of ice cream next month? (sales prediction)
  + What’s the stock price of MSFT next week? (stock prediction)
  + What’s the income of a customer? (marketing)
  + What’s the life-time of a software bug? (bug tracking)
* *Who are my Web visitors?*
  + Identify *similar groups* based on *demographics, visiting patterns*
  + E.g., Daily news readers, email users, shoppers, short-stayers, etc
  + **Segmentation (clustering) – what might you do with THIS information?**
* Similar questions:
  + Identify groups of genes (bioinformatics)
  + Identify groups of locations of Cholera incidents in London (spatial data mining)
  + Identify group of customers in merchants (Amazon, E-Bay, MSN, WalMart, BlockBuster, etc) (target marketing)
  + Identify groups of documents. (text categorization)
* *What other products are purchased together with a digital camera?*
  + Based on previous purchases (shopping cart)
  + E.g., If a digital camera is purchased, flash memory, battery, printer are also purchased.
  + **Association Analysis (recommendation, market basket analysis) ) – what might you do with THIS information?**
* Similar questions:
  + What products to recommend in on-line stores such as Amazon.com, Barnes & Nobles, movie rental, wireless themes, etc.
  + What items should be displayed together in merchant.
  + What genes appear together in toxic mushrooms.
* *Could this network packet be from a virus attack?*
  + Predict likelihood of the network packet pattern
  + **Anomaly detection (outlier detection) ) – what might you do with THIS information?**
* Similar questions:
  + Are the hospital lab results normal (Adverse drug effect detection)
  + Is this credit transaction fraudulent? (fraud detection)
  + Does this person behave unusual, maybe worth high-level of security clearance? (TIA)

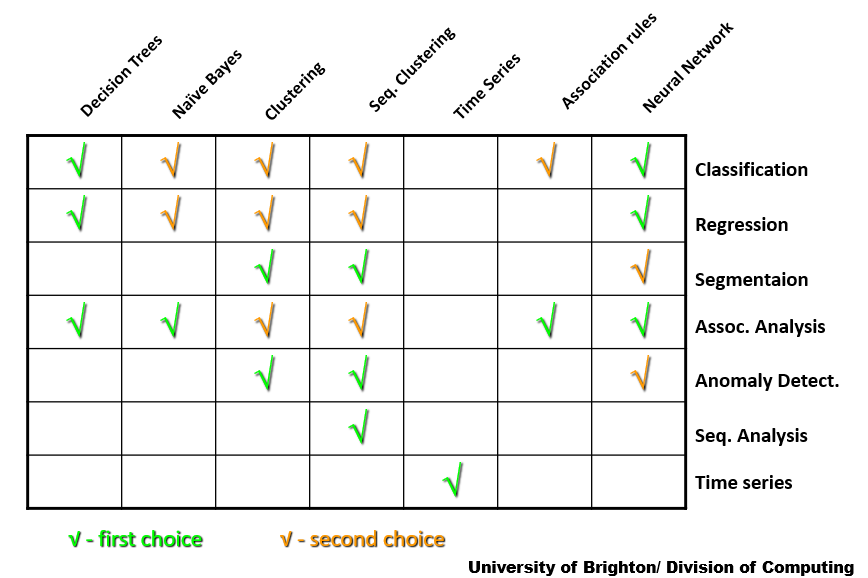
## Data Mining Techniques - Summary

* Classification
* Regression
* Segmentation
* Association Analysis
* Anomaly detection
* Sequence Analysis
* Time-series Analysis
* Text categorization
* Advanced insights discovery
* Others

## Data Mining Algorithms:

* Decision Trees
* Naïve Bayesian
* Clustering
* Sequence Clustering
* Association Rules
* Neural Network
* Time Series
* Support Vector Machines

## Data Mining Algorithms:



## Data Mining Algorithms:

* A *data mining algorithm is*
  + a set of heuristics and calculations that creates a data mining model from data.
* Analyzes the data you provide, looking for specific types of patterns.
* Uses the results of this analysis to define the best parameters for creating the mining model (training).
* These parameters are then applied across the entire data set to extract patterns and statistics.

## The mining model could be any of these:

* A set of clusters that describe how the cases in a dataset are related.
* A decision tree that predicts an outcome.
* A mathematical model that forecasts sales.
* A set of rules that describe how products are grouped together in a transaction.

## Choosing an Algorithm by Type:

* **Classification algorithms** predict one or more discrete variables, based on the other attributes in the dataset.
* **Regression algorithms** predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset.
* **Segmentation algorithms** divide data into groups, or clusters, of items that have similar properties.
* **Association algorithms** find correlations between different attributes in a dataset. The most common application of this kind of algorithm is for creating association rules, which can be used in a market basket analysis.
* **Sequence analysis algorithms** summarize frequent sequences in data, such as a Web path flow.

## Choosing an Algorithm by Task:

|  |  |
| --- | --- |
| **Predicting a discrete attribute example** | **Microsoft algorithms to use** |
| Flag the customers in a prospective buyer list as good or poor prospects.  Calculate the probability that a server will fail within the next 6 months.  Categorize patient outcomes and explore related factors. | [Microsoft Decision Trees Algorithm](http://technet.microsoft.com/en-us/library/ms175312.aspx)  [Microsoft Naive Bayes Algorithm](http://technet.microsoft.com/en-us/library/ms174806.aspx)  [Microsoft Clustering Algorithm](http://technet.microsoft.com/en-us/library/ms174879.aspx)  [Microsoft Neural Network Algorithm](http://technet.microsoft.com/en-us/library/ms174941.aspx) |

|  |  |
| --- | --- |
| **Predicting a continuous attribute example** | **Microsoft algorithms to use** |
| Forecast next year's sales.  Predict site visitors given past historical and seasonal trends.  Generate a risk score given demographics. | [Microsoft Decision Trees Algorithm](http://technet.microsoft.com/en-us/library/ms175312.aspx)  [Microsoft Time Series Algorithm](http://technet.microsoft.com/en-us/library/ms174923.aspx)  [Microsoft Linear Regression Algorithm](http://technet.microsoft.com/en-us/library/ms174824.aspx) |

|  |  |
| --- | --- |
| **Predicting a sequence example** | **Microsoft algorithms to use** |
| Perform clickstream analysis of a company's Web site.  Analyse the factors leading to server failure.  Capture and analyse sequences of activities during outpatient visits, to formulate best practices around common activities. | [Microsoft Sequence Clustering Algorithm](http://technet.microsoft.com/en-us/library/ms175462.aspx) |

|  |  |
| --- | --- |
| **Finding groups of common items in transactions examples** | **Microsoft algorithms to use** |
| Use market basket analysis to determine product placement.  Suggest additional products to a customer for purchase.  Analyse survey data from visitors to an event, to find which activities or booths were correlated, to plan future activities. | [Microsoft Association Algorithm](http://technet.microsoft.com/en-us/library/ms174916.aspx)  [Microsoft Decision Trees Algorithm](http://technet.microsoft.com/en-us/library/ms175312.aspx) |

|  |  |
| --- | --- |
| **Finding groups of similar items examples** | **Microsoft algorithms to use** |
| Create patient risk profiles groups based on attributes such as demographics and behaviours.  Analyse users by browsing and buying patterns.  Identify servers that have similar usage characteristics.  Analyse DNA, group by similar features | [Microsoft Clustering Algorithm](http://technet.microsoft.com/en-us/library/ms174879.aspx)  [Microsoft Sequence Clustering Algorithm](http://technet.microsoft.com/en-us/library/ms175462.aspx) |

# Data Mining Modeling and Language:

## Data Mining Language:

* New challenges in data mining API
  + Large spectrum of applications: embedded to interactive BI
  + Interoperability between different DM providers (engine) and DM consumers (tools)
  + Data independence between content representation (trees, attributes, networks, etc) and data mining task (prediction, scoring, etc)
* Requirements:
  + Algorithm-neutral
  + Task-oriented (specification of what we need, rather than how to)
  + Vendor-neutral
  + Flexible, extensible, declarative/self-contained
* Sound familiar?
* Yes, SQL

## SQL Revolution (1970’s):

|  |  |  |
| --- | --- | --- |
|  | Before | After |
| Architecture | File system,  Hierarchical/network DB | Relational DB |
| API | Proprietary ISAM,  X/OPEN CLI, etc | SQL |
| Data independence | Physical model tied to logical model (appl logic)  🡪 Physical model change requires re-develop the apps. | Clear separation between physical/logical model  🡪 No more app changes due to physical model update |
| Appl dev tools | Not many.  Custom dev with consulting services | Commodity.  Product services than consulting |

## DMX Approach:

* Data Mining Extensions (DMX) to SQL
* Table vs. Mining Model

|  |  |  |
| --- | --- | --- |
|  | TABLE | MINING MODEL |
| schema | Column definition | Attribute (variable) definition |
| contains | Rows | Patterns, knowledge, cases |
| operations | DDL (create, drop, alter)  DML (insert, delete)  Query (select) | Create/drop/alter a model  Train (populate) a model  Prediction/browsing a model |

## Defining a DM Model:

* Defines
  + Shape of “training cases” (top-level entity being modeled)
  + Input/output attributes (variables): type, distribution
  + Algorithms and parameters

## Training (processing) a DM Model:

* Simply issue INSERT with training data
* DMMS takes care of everything:
  + Accessing the training data possibly outside the system
  + Transformation (e.g., discretization, normalization)
  + Tokenization, numeric conversion, feature selection, etc.
  + Learn the algorithm
  + Persistency of patterns discovered
* Multiple ways to specify training data
  + SELECT, OPENROWSET, SHAPE, etc.

## Training a DM Model: Simple

**INSERT INTO CollegePlanModel**

**(StudentID, Gender, ParentIncome,**

**Encouragement, CollegePlans)**

**OPENROWSET(‘<provider>’, ‘<connection>’,**

**‘SELECT StudentID,**

**Gender,**

**ParentIncome,**

**Encouragement,**

**CollegePlans**

**FROM CollegePlansTrainData’)**

# Data Management DBMS security:

## Security properties:

Confidentiality: prevent unauthorized users to access data

Integrity: prevent the unauthorized modification of data

Availability: data must be available when needed

Protecting the database/system from intentional or unintentional threats

Database security encompasses:

* + software
  + hardware
  + humans

## Threats:

* Threat is any situation or event, whether intentional or unintentional, that may adversely affect the database a system and consequently the organization.
* A threat may include a person, an action, or a circumstance that is likely to bring harm to an organization.
* Harm may be:
  + Tangible
  + Intangible

## Security controls:

### Access control:

* A mechanism for granting and revoking privileges
* A privilege allow a user to read, write, or modify a database object
* Privileges are granted to users to allow them to accomplish their jobs
* *Excessive granting of unnecessary privileges can compromise security: a privilege should only be granted to a user if that user cannot accomplish his or her work without that privilege*

### Authorization:

* All about checking the ‘access request’
  + Object + operation + user
* User identification and authentication
* At both a user and role level

### Integrity:

* Integrity enforced at the DBMS level
* Based on accurate and consistent data models
  + Entity integrity
  + Referential integrity
* Additional business rules specified by the users or database administrators of a database that define or constrain some aspect of the enterprise.

### Views:

* Integrity enforced at the DBMS level
* Based on accurate and consistent data models
  + Entity integrity
  + Referential integrity
* Additional business rules specified by the users or database administrators of a database that define or constrain some aspect of the enterprise.

### Backup & Recovery:

* Backup is the process of periodically taking a copy of the database and log file on to offline storage media
* Backup & Recovery can be handled at the DBMS level and the operator level
* Ensure copies that are stored in a secure location

### Encryption:

* Encryption is the encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key
* Any negative impact of encryption?

### Raid:

* Redundant Array of Independent Disks (RAID) is is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy
* Ensures fault tolerance of the database
* <https://datapacket.com/blog/advantages-disadvantages-various-raid-levels/>

## Creating secure environments:

* As much a frame of mind as a set of technological controls
* More can be achieved through good housekeeping than sophisticated technology
  + >50% of cases of breaches of security occur internally rather than externally

## Approach to secure thinking:

* Identification of possible threats
* Assessment of potential risk i.e.
  + the possibility of loss or inaccuracy of the data
  + the difficulties/costs that the organisation would face if the loss occurred
  + Establishment of appropriate controls based on
  + the cost of implementing the control

vs

* + the cost of NOT implementing the control

## Database Risk Management:

* Identification of possible threats
* Assessment of potential risk i.e.
  + the possibility of loss or inaccuracy of the data
  + the difficulties/costs that the organisation would face if the loss occurred
  + Establishment of appropriate controls based on
  + the cost of implementing the control

vs

* + the cost of NOT implementing the control

## Procedures & Practices:

* No level of security will guard against poor manual practices & slack controls e.g.
  + passwords infrequently changed
  + existence of default users and accounts
  + poorly protected development environment
  + infrequent backups
  + poor protection of backups
  + lack of audit trails
  + loose supervisory structures

## Database security strategy:

* At the client
  + *Very difficult, how do you ensure everyone uses same tool set for data access?*
* Within the database
  + *No standards, each DBMS may offer something different*
  + *In mixed database environment may be inconsistencies e.g logging*
  + *Performance Issues*
* In between
  + *Special tools*
  + *Now problems of* 
    - How to capture the information
    - Dealing with the complexity of the traffic

## Top security myths:

* The field of data security is rife with mistaken beliefs that cause people to design ineffective security solutions. Here are some of the most prevalent security myths:
  + Myth: Hackers cause most security breaches.
    - In fact, 80% of data loss is caused by insiders.
  + Myth: Encryption makes your data secure.
    - In fact, encryption is only one approach to securing data. Security also requires access control, data integrity, system availability, and auditing.
  + Myth: Firewalls make your data secure.
    - In fact, 40% of Internet break-ins occur in spite of a firewall being in place.
* To design a security solution that truly protects your data, you must understand the security requirements relevant to your site, and the scope of current threats to your data.

# Database Lifecycle and DBMS Selection:

## Requirements collection and analysis:

* The process of collecting and analysing information about the part of the organization that is to be supported by the database system, and using this information to identify the requirements for the new system.
  + a description of the data used;
  + the details of how data is to be used;
  + any additional requirements for the new database system.

## Database Design:

* The process of creating a design that will support
  + all major application areas and user groups
  + any transactions required on the data
  + performance requirements for the system
* Three phases of database design:
  + Conceptual database design
  + Logical database design
  + Physical database design.

## Database design – Criteria for an Optimal Data Model:

|  |  |
| --- | --- |
| Structural validity | consistency with the way the enterprise deigns and organizes information. |
| simplicity | ease of understanding by is professionals and no-technical users. |
| expressibility | ability to distinguish between different data, relationships between data and constraints. |
| non redundancy | exclusion of extraneous information, in particular the representation of any one piece of information exactly once. |
| shareability | not specific to any particular application or technology and thereby usable by many. |
| extensibility | ability to evolve to support new requirements with minimal effect on existing users. |
| integrity | consistency with the way the enterprise uses and manages information. |
| diagrammatic representation | ability to represent a model using an easily understood diagrammatic notation. |

## DBMS Selection:

* The selection of an appropriate DBMS to support the database system.
* Undertaken at any time prior to logical design provided sufficient information is available regarding system requirements.
* Main steps to selecting a DBMS:
  + Determine DBMS requirements
  + shortlist two or three products;
  + evaluate products;
  + recommend selection and produce report.

## Business requirements imply DBMS requirements:

* Business requirements
  + strategic
  + tactical
  + resource constrained
  + external factors
  + innovation i.e.technology driven

## Essential requirements:

* Cost
* Availability
* Vendor support
* Compatibility with existing hardware/software
* Political

## Implementation:

* Build the database (and applications).
* Changeover
  + Data conversion and loading
  + May be possible to convert and use application programs from old system for use by new system.

## Testing:

Database tested as part of application testing

Verify database against the **requirements** e.g. response time, throughput (transactions/second)

Verify **non-functional requirements** e.g. backup, security, response time, throughput (transactions/second)

## Maintenance corrective, perfective, adaptive

Process of monitoring and maintaining system following installation.

Monitoring performance of system.

* + if performance falls, may require tuning or reorganization of the database.

Maintaining and upgrading database application (when required).

Incorporating new requirements into database application.

## Operational issues:

* Process of operating the new system
* Technical
  + Interfacing to legacy systems i.e. *ones that already exist in the organisation* e.g. existing stock management system from new sales system. [Why Enterprise-wide approach is beneficial]
  + Interfacing to external systems i.e. *other organisation’s systems* e.g. payment services
* Human resources
  + Right mix of people with right skills
  + Training/career paths etc.
* Organisational
  + Fault reporting/fixing procedures

# Professional, Legal, and Ethical Issues:

PROFESSIONAL: Responsible for your own actions – likely to belong to a professional body which sets standards

LEGAL: Organizations increasingly find themselves having to answer tough questions about the conduct and character of their employees and the manner in which their activities are carried out.

ETHICAL: what constitutes professional and non-professional behavior?

## Ethical issues:

### Ethics in the context of information technology:

A set of principles of right conduct or a theory or a system of moral values.

Can consider ethical behavior as “doing what is right” according to the standards of society. This, of course, begs the question “of whose society” as what might be considered ethical behavior in one culture (country, religion, and ethnicity) might not be so in another.

### Difference between ethical and legal behavior:

* Laws can be considered as simply enforcing certain ethical behaviors. This leads to two familiar ideas: what is ethical is legal and what is unethical is illegal.
* Consider –
  + Is all unethical behavior illegal?
  + Is all ethical behavior legal?
* Ethical codes of practice help determine whether specific laws should be introduced. Ethics fills the gap between the time when technology creates new problems and the time when laws are introduced.

### Ethical behavior in information technology:

(Old) survey by TechRepublic reported that 59% of the IT workers polled indicated they had been asked to do something ‘unethical’ by their supervisors   
<https://www.techrepublic.com/article/it-managers-face-ethical-issues-from-piracy-to-privacy/>

Examples include installing unlicensed software, accessing personal information, and divulging trade secrets.

Data Mining has its own ethical issues <https://www.liebertpub.com/doi/full/10.1089/big.2018.0083>

### British Computer Society:

* …is the professional body for IT professionals in this country
* They have a Code of Conduct covering:
  + THE PUBLIC INTEREST
  + DUTY TO RELEVANT AUTHORITY
  + DUTY TO THE PROFESSION
  + PROFESSIONAL COMPETENCE AND INTEGRITY

BCS, 2015, *Code of Conduct*, [online] available at: <https://www.bcs.org/category/6030> [accessed Dec 2019]

## Laws:

### Legislation to consider for IT Professionals:

* The EU General Data Protection Regulations (new from May 2018) *part of which is met by the:*
* The United Kingdom’s Data Protection Act (1998, 2018)
* Freedom of Information Act 2000
* Copyright, Designs and Patent Act 1988
* Regulation of Investigatory Powers Act 2000
* The Telecommunications Regulations 2003

### The United Kingdom’s Data Protection Act 2018 – part of GDPR:

* 7 Key Principles:
  + Lawfulness, fairness and transparency
  + Purpose limitation
  + Data minimisation
  + Accuracy
  + Storage limitation
  + Integrity and confidentiality (security)
  + \*Accountability\* new principle

PLUS

* + Individual Rights - Chap III GDPR
  + International Transfers – Chap V GPDR

For more information: [ICO website](https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/principles/)

### EU General Data Protection Regulations (GDPR):

* Builds on existing DPA
  + Fines increased to EUR 20M or 4% turnover
  + More evidence of active compliance required
  + Requires breach notification
  + Increased ‘right to be forgotten’
  + Right to port your own data

*What happens after Brexit?*

Initially will have this law enshrined

in UK law. But over time….

Information Commissioner’s website will be the place to  
find out: [ICO Website - Brexit](https://ico.org.uk/for-organisations/data-protection-and-brexit/)

### Freedom of Information Act 2000:

* Only applies to public bodies e.g Police, NHS, state schools (including academies) etc. and (since 2013 amendment) organisations which are largely publicly fund e.g Kensington and Chelsea Tenant Management Organisation (KCTMO)
* Requires:
  + Publication of information: see more: [Model Publication Scheme](https://ico.org.uk/media/for-organisations/documents/1153/model-publication-scheme.pdf)
  + Response to information requests: see more: [Request Flowchart](https://ico.org.uk/media/for-organisations/documents/1167/flowchart_of_request_handling_under_foia.pdf)

### Copyright, Designs and Patent Act 1988:

* Type of intellectual property (something unique that you physically create)
* Copyrights / design rights can be bought and sold e.g Michael Jackson bought the copyright to 50% of Beatles back catalogue, then Sony did, now [Paul McCartney has reached a deal with them under the clawback clause](https://www.forbes.com/sites/legalentertainment/2017/06/30/paul-mccartney-settles-with-sonyatv-over-beatles-catalogue/)
* If you create software at work, [the company usually owns the copyright](https://www.gov.uk/guidance/ownership-of-copyright-works)
* Making a backup copy (as a lawful user) does not breach copyright – [software has its own section](https://www.legislation.gov.uk/ukpga/1988/48/section/50A) in the legislation
* ‘[Fair dealing](https://www.smashingmagazine.com/2018/03/copyright-law-basics-for-uk-software-developers/)’ is an important concept in the legislation
* [Exceptions](https://www.gov.uk/guidance/exceptions-to-copyright) are available

### Intellectual Property (IP):

* Important that data and database administrators as well as business analysts and software developers recognize and understand the issues surrounding IP both to ensure that their ideas can be protected and to ensure that other people’s rights are not infringed.
* IP is the product of human creativity in the industrial, scientific, literary, and artistic fields.
* Important that data and database administrators as well as business analysts and software developers recognize and understand the issues surrounding IP both to ensure that their ideas can be protected and to ensure that other people’s rights are not infringed.
* IP is the product of human creativity in the industrial, scientific, literary, and artistic fields.
* **Copyright** - provides an exclusive (legal) right for a set period of time to reproduce and distribute a literary, musical, audiovisual, or other ‘work’ of authorship.
* **Trademark** - provides an exclusive (legal) right to use a word, symbol, image, sound, or some other distinction element that identifies the source of origin in connection with certain goods or services another make, use, sell, or import an invention.
* Why important to consider?
  + To understand your own right or your organization’s right as a producer of original ideas and works;
  + To recognize the value of original works;
  + To understand the procedures for protecting and exploiting such work;
  + To know the legal measures that can be used to defend against the illegal use of such work;
* Issues related specifically to IPR and software include –
  + Software and patentability
  + Software and copyright
    - Commercial software (perpetual use),
    - Commercial software (annual fee),
    - Shareware,
    - Freeware.
* Patents - provides an exclusive (legal) right for a set period of time to make, use, sell or import an invention.
* Patents are granted by a government when an individual or organization can demonstrate:
  + the invention is *new*;
  + the invention is in some way *useful*;

### Regulation of Investigatory Powers Act (RIPA) 2000:

* Covers what is appropriate covert behaviour
* Refers to codes of practice covering:
  + [**Interception of communications: code of practice 2016**](https://www.gov.uk/government/publications/interception-of-communications-code-of-practice-2016)
  + [**Equipment interference: code of practice**](https://www.gov.uk/government/publications/equipment-interference-code-of-practice)
  + [**Covert surveillance and covert human intelligence sources codes of practice**](https://www.gov.uk/government/publications/covert-surveillance-and-covert-human-intelligence-sources-codes-of-practice)
  + [**Code of practice for investigation of protected electronic information**](https://www.gov.uk/government/publications/code-of-practice-for-investigation-of-protected-electronic-information)

Designed with terrorism – some uses have been less … obvious

[Potential Misuses of RIPA](https://www.theguardian.com/world/2016/dec/25/british-councils-used-investigatory-powers-ripa-to-secretly-spy-on-public)

[The Grim Ripa](https://www.bigbrotherwatch.org.uk/TheGrimRIPA.pdf)

### The Telecommunications Regulations 2003:

Communication can be lawfully monitored for:

* Record-keeping
* National security
* Crime detection

There must be:

* Reasonable efforts to inform people that monitoring is taking place (crime exception)
* Monitor is authorised either by system owner

### Intellectual Property Rights (IPR):

* Consideration must also be paid to data that an organization collects, processes, and possibly shares with its trading partners.
* In conjunction with senior management and legal counsel, data administrators must define and enforce policies that govern when data can be shared and in what ways it can be used within the organization

### Establishing a culture of legal and ethical data stewardship:

* + Senior managers such as board members, presidents, Chief Information Officers (CIOs), and data administrators are increasingly finding themselves liable for any violations of these laws.
  + Steps to consider include:
    - Develop an organization-wide policy for legal and ethical behavior.
    - Professional organizations and codes of ethics.

# Graph Database:

## RDF and the semantic web:

* Resource Descriptor Framework (RDF) a model for encoding semantic relationships between items of data so that these relationships can be interpreted computationally:
* "RDF is the primary foundation for the Semantic Web"
* RDF, whilst the foundation of defining data structures for the semantic web, does not in itself describe the *semantics*, or meaning, behind the data.
  + This comes with RDFS (RDF Schema) and OWL (Web Ontology Language). <https://www.w3.org/OWL/>
* RDF structures data and relationships in a way that is different - a *paradigm shift* from the relational or hierarchical means of modelling data to a graph model.

## Querying semantic web:

* SPARQL language has similarities to SQL
* SPARQL can query RDF datasets as SQL queries a relational database
* What does it look like?

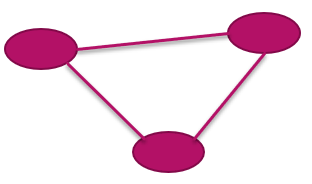
## Other models:

* The semantic web (RDF) approach *extends* XML for relationship data
* The object-relational approach *extends* the relational model with object-like features
* Two other approaches create another approach of storing data in a different way
  + Graph databases
  + Object-oriented databases

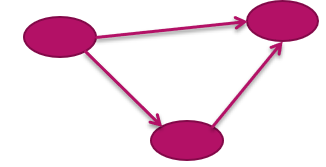
## Graph data models:

* Now becoming increasingly popular as a way to store data, particularly from social networking sites
* There can be many types of graph model

1. Undirected graph
   * nodes or vertices and edges
   * The edges have no orientation



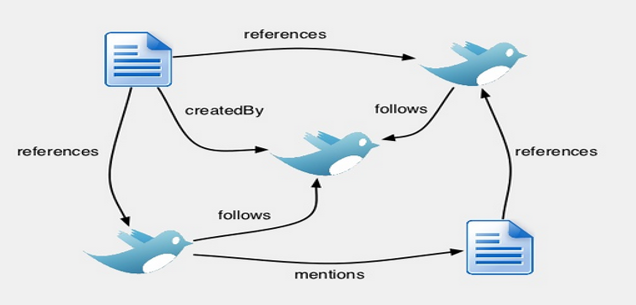
1. Directed graphs – edges have a direction



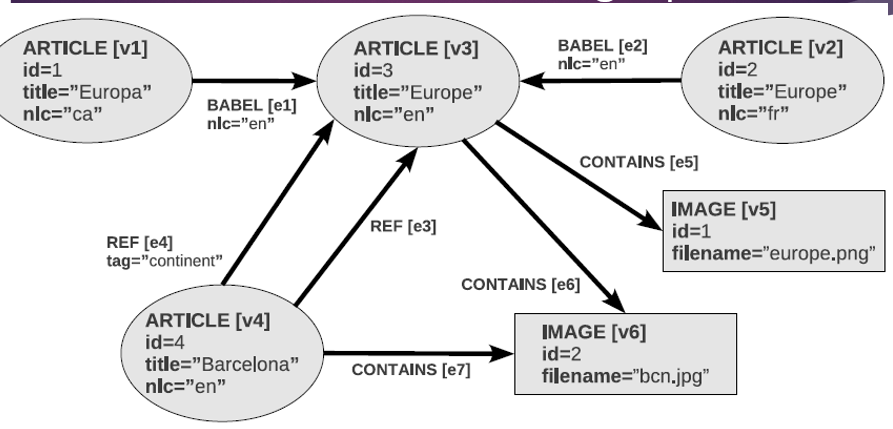
1. Multigraphs
   * Multiple edges and loops are allowed

## Attributes:

* Attributes may be added to both vertices and edges



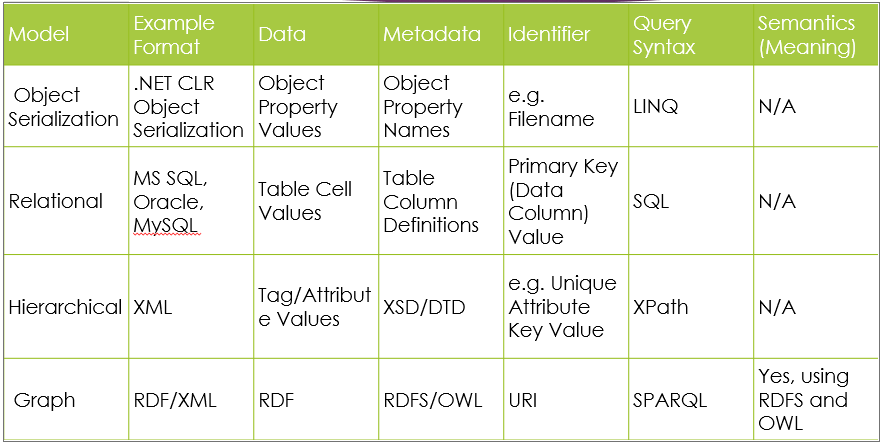
## A labelled and directed attributed multigraph.



## Demands on a graph database:

* Graph queries often combine graph traversals with intensive attribute accesses
* For example:
  + The best route in a network mapping a country roadmap can be constrained by several factors such as the road type or the existence of a petrol station on that road.
* Many applications demand multigraphs (rather than simpler forms)
* For example:
  + Two authors might have collaborated coauthoring a paper more than once
  + Two cities will be connected by more than two roads

Comparing the features of the mainstream ways of modelling data versus the semantic web model:



## Big Data – why it is becoming important:

“A new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-speed capture, discovery and/or analysis”

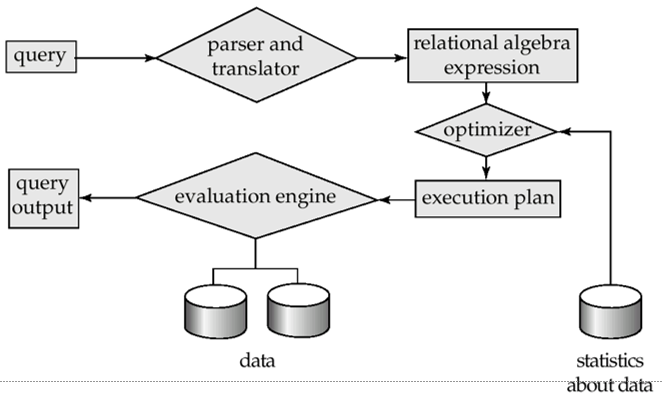
New investment initiatives are coming:

“more than $200 million in new funding through six agencies and departments to improve the nation’s  ability to extract knowledge and insights from large and complex collections of digital data”

# Query Processing:

## Basic Steps in Query Processing:

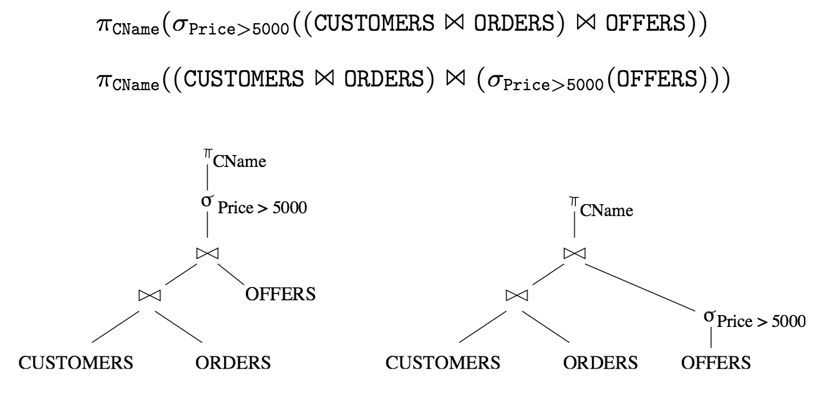
* Parsing and translation
* Optimization
* Evaluation



* Parsing and translation
* Translate the query into its internal form.
* This is then translated into relational algebra.
* Parser checks syntax, verifies relations
* Evaluation
  + The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.

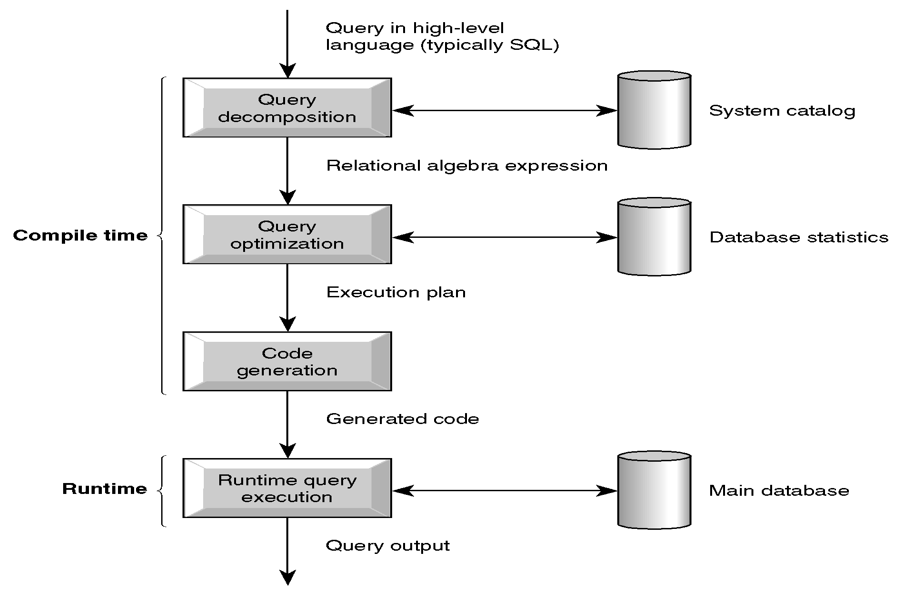
## Basic Steps: Optimization

* Parsing and translation
  + Translate the query into its internal form.
  + This is then translated into relational algebra.
* Parser checks syntax, verifies relations
* Evaluation
  + The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.



* Query Optimization: Amongst all equivalent evaluation plans choose the one with lowest cost.
  + Cost is estimated using statistical information from the  
     database catalog
    - e.g. number of tuples in each relation, size of tuples, etc.
  + queries, that is, how to find an evaluation plan with lowest estimated cost

## Phases of query processing:



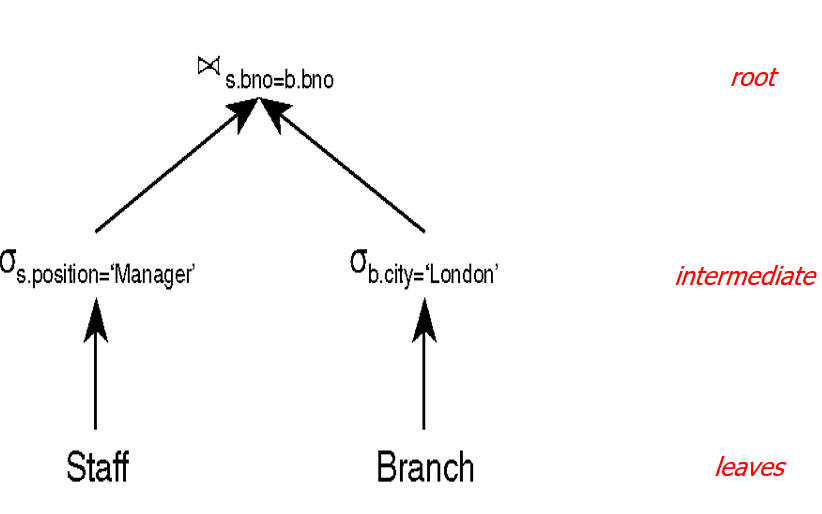
## Query Decomposition:

* Aim is to transform high-level query into RA query and check that query is syntactically and semantically correct.
* Typical stages are:
  + Analysis
  + Normalization
  + Semantic analysis
  + Simplification
  + Query restructuring

## Analysis:

* Analyze query lexically and syntactically using compiler techniques.
* Verify that relations and attributes exist.
* Verify operations are appropriate for object type.
* Query transformed into some internal representation more suitable for processing.
* Some kind of query tree is typically chosen, constructed as follows:
  + Leaf node created for each base relation.
  + Non-leaf node created for each intermediate relation produced by RA operation.
  + Root of tree represents query result.
  + Sequence is directed from leaves to root.
  + are appropriate for object type.

## Example RAT:



## Normalization:

* Converts query into a normalized form for easier manipulation.
* Predicate can be converted into one of two forms:
  + **Conjunctive** normal form (**and**):
    - (position = 'Manager' Ú salary > 20000) **Ù** (bno = 'B3')
  + **Disjunctive** normal form (**or**):
    - (position = 'Manager' Ù bno = 'B3' ) **Ú**

(salary > 20000 Ù bno = 'B3')

## Semantic Analysis:

* Rejects normalized queries that are incorrectly formulated or contradictory.
* Query is incorrectly formulated *if components do not contribute to generation of result*.
* Query is contradictory if its *predicate cannot be satisfied by any tuple*.

## Simplification:

* + Detects redundant qualifications
  + Eliminates common sub-expressions
  + Transforms query to semantically equivalent but more easily and efficiently computed form.

## Measures of Query Cost:

* Cost is generally measured as total elapsed time for answering query
* Many factors contribute to time cost
  + disk accesses, CPU, or even network communication
* Typically, disk access is the predominant cost, and is also relatively easy to estimate. Measured by taking into account
  + Number of seeks \* average-seek-cost
  + Number of blocks read \* average-block-read-cost
  + Number of blocks written \* average-block-write-cost
    - Cost to write a block is greater than cost to read a block
      * data is read back after being written to ensure that the write was successful
* Number of block transfers from disk as the cost measure
  + We ignore the difference in cost between sequential and random I/O for simplicity
  + We also ignore CPU costs for simplicity
* Costs depends on the size of the buffer in main memory
  + Having more memory reduces need for disk access
  + Amount of real memory available to buffer depends on other concurrent OS processes, and hard to determine ahead of actual execution
  + We often use worst case estimates, assuming only the minimum amount of memory needed for the operation is available
* We do not include cost of writing output to disk in our cost   
  formulae

## Cost estimation for RA Operations:

* Many different ways of implementing RA operations.
* Use formulae that estimate costs for a number of options, and select one with lowest cost.
* Many estimates are based on cardinality of the relation, so need to be able to estimate this.

## Selection Operation – Linear Search:

* File scan – search algorithms that locate and retrieve records that fulfill a selection condition.
* Algorithm (linear search). Scan each file block and test all records to see whether they satisfy the selection condition.
  + Cost estimate (number of disk blocks scanned) = br
    - br denotes number of blocks containing records from relation r
  + If selection is on a key attribute, cost = (br /2)
    - stop on finding record
  + Linear search can be applied regardless of
    - selection condition or
    - ordering of records in the file, or
    - availability of indices

## Selection Operation – Binary Search:

* (Binary search). Applicable if selection is an equality comparison on the attribute on which file is ordered.
  + Assume that the blocks of a relation are stored contiguously
  + Cost estimate (number of disk blocks to be scanned):
    - ⎡log2(br)⎤ — cost of locating the first tuple by a binary search on the blocks
    - Plus number of extra blocks containing records that satisfy selection condition if selection on non-key attribute.

# Physical Structure:

## Indexes on tables:

* Purpose – to improve data *reading* efficiency
* (note: always slows down *writing -* why*?*)
* Important when db large
* Index allows you to (almost) directly access data, without looking through whole table (c.f. book)
* Review this example and consider it when we discuss physical storage <https://www.essentialsql.com/what-is-a-database-index/>

### Why physical storage matters

* Relational databases hide physical storage details from the users
  + Physical data independence provided
  + DBMS makes use of physical storage structures to access, manage and maintain the data (and metadata)
* Query Optimizer needs to know about the physical structures to optimize its Query Plans

### How queries make use of physical details:

* SQL query sent to DBMS
* DBMS accesses metadata (system catalogs) to determine physical details of underlying tables, indexes etc.
* Query optimizer uses this information to plan query
* DBMS exploits physical structures to execute efficient query

## Databases and Files in SQL Server:

* One *database* maps to
  + One primary (data) file (.mdf extension)
  + *N* secondary files (.ndf extension)
  + At least one log file (.ldf extension)
* *Disk space* in SQL Server is divided into pages
  + Each file is split into a number of 8K pages
    - 128 pages per one Megabyte
* Log files do not contain pages
  + They contain a series of log records

For more info: <https://docs.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-2017>

## Logical and Physical File Names:

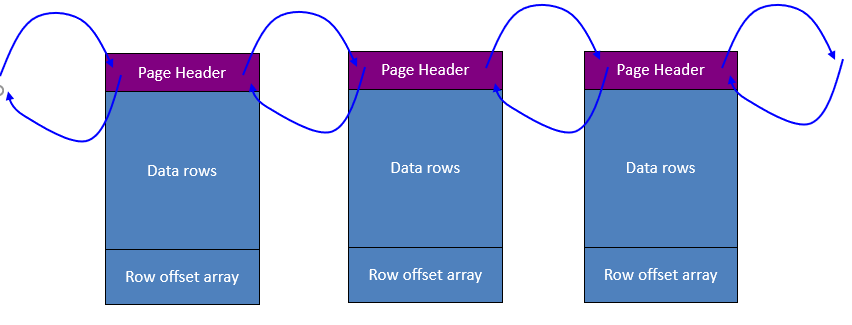
* SQL Server files have two names
  + **logical\_file\_name**
    - The name used to refer to the physical file in all SQL statements
* e.g. tCust
  + **os\_file\_name**
    - The name of the physical file including the directory path.
    - It must follow the rules for the operating system file names (for whichever OS you are using)

## Databases and Files in MySQL:

* One *database* maps to:
  + A directory (variable name: *datadir)* which contains ….
  + Form file for each table (.*frm* extension)
* *Disk space* in MySQL is divided into pages
  + Each file is split into a number of (default size) 16K *pages*
  + Grouped into 1MB *extents (*e.g. 64 consecutive 16K pages)
  + Exact structure depends on stored engine chosen – innoDB is default (supports B-Tree indexes, see here for comparison: <https://stackoverflow.com/questions/870218/differences-between-b-trees-and-b-trees>)
  + <https://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html>
* Log files In MySQL
  + <https://dev.mysql.com/doc/refman/5.7/en/server-logs.html>

## The Page Header:

* Chains the pages together using pointers
* Stores a variety of housekeeping information
* 96 bytes in total



## The Row offset array:

* An array of 2-byte slots
* One slot for each data row
* Each slot holds offset of a data row
* Order of slots determines logical order of rows
* More info:
* <https://sqlity.net/en/2134/row-offset-array/>

## Classification of Physical Storage Media:

* Speed with which data can be accessed
* Cost per unit of data
* Reliability
  + data loss on power failure or system crash
  + physical failure of the storage device
* Can differentiate storage into:
  + **volatile storage:** loses contents when power is switched off
  + **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter-backed up main-memory.

## Physical Storage Media:

### Cache

* fastest and most costly form of storage; volatile; managed by the computer system hardware.

### Main memory:

* + fast access (10s to 100s of nanoseconds; 1 nanosecond = 10–9 seconds)
  + generally too small (or too expensive) to store the entire database – but changing all the time (see [this](https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/)) – and whole pricing field is more complex (see [this](https://www.clearskydata.com/blog/stop-measuring-storage-costs-in-terms-of-dollars-per-gb))
    - capacities to many Gigabytes widely used currently
    - capacities have gone up and per-byte costs have decreased steadily and rapidly (roughly factor of 2 every 2 to 3 years)
  + **Volatile** — contents of main memory are usually lost if a power failure or system crash occurs.

### Flash memory:

* Data survives power failure.
* Data can be written at a location only once, but location can be erased and written to again.
  + Can support only a limited number of write/erase cycles.
  + Erasing of memory has to be done to an entire bank of memory
  + Reads are roughly as fast as main memory
  + But writes are slow (few microseconds), erase is slower
  + Cost per unit of storage roughly similar to main memory
  + Widely used in embedded devices such as digital cameras
  + also known as EEPROM (Electrically Erasable Programmable Read-Only Memory)

### Magnetic disk:

* + Data is stored on spinning disk, and read/written magnetically
  + Primary medium for the long-term storage of data; typically stores entire database
  + Data must be moved from disk to main memory for access, and written back for storage
    - Much slower access than main memory

### Optical storage

* + non-volatile, data is read optically from a spinning disk using a laser
  + CD-ROM (640 MB) and DVD (4.7 to 17 GB) most popular forms
  + Write-one, read-many (WORM) optical disks used for archival storage (CD-R and DVD-R)
  + Multiple write versions also available (CD-RW, DVD-RW, and DVD-RAM)
  + Reads and writes are slower than with magnetic disk

### Tape storage

* + non-volatile, used primarily for backup (to recover from disk failure), and for archival data
  + **sequential-access** – much slower than disk
  + very high capacity (10 TB tapes available)
  + tape can be removed from drive ⇒ storage costs much cheaper than disk, but drives are expensive
  + Around £500 for 10TB drive -see this article from [ITPro Today](https://www.itprotoday.com/backup/tape-storage-still-here) for current thoughts

### Magnetic Disks:

* **Read-write head**
  + Positioned very close to the platter surface (almost touching it)
  + Reads or writes magnetically encoded information.
* Surface of platter divided into circular **tracks**
  + Over 16,000 tracks per platter on typical hard disks
* Each track is divided into **sectors.**
  + A sector is the smallest unit of data that can be read or written.
  + Sector size typically 512 bytes
  + Typical sectors per track: 200 (on inner tracks) to 400 (on outer tracks)
* To read/write a sector
  + disk arm swings to position head on right track
  + platter spins continually; data is read/written as sector passes under head
* Head-disk assemblies
  + multiple disk platters on a single spindle (typically 2 to 4)
  + one head per platter, mounted on a common arm.
* **Cylinder** *i*consists of *i*th track of all the platters

## RAID:

* **RAID:** Redundant Arrays of Independent Disks
  + disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
    - high capacity and high speed by using multiple disks in parallel, and
    - high reliability by storing data redundantly, so that data can be recovered even if a disk fails
* The chance that some disk out of a set of *N* disks will fail is much higher than the chance that a specific single disk will fail.
  + Techniques for using redundancy to avoid data loss are critical with large numbers of disks

## Storage Access:

* A database file is partitioned into fixed-length storage units called **blocks**. Blocks are units of both storage allocation and data transfer.
* Database system seeks to minimize the number of block transfers between the disk and memory. We can reduce the number of disk accesses by keeping as many blocks as possible in main memory.
* **Buffer** – portion of main memory available to store copies of disk blocks.
* **Buffer manager** – subsystem responsible for allocating buffer space in main memory.

### Buffer Manager

* Programs call on the buffer manager when they need a block from disk.
  1. If the block is already in the buffer, the requesting program is given the address of the block in main memory
  2. If the block is not in the buffer,
     1. the buffer manager allocates space in the buffer for the block, replacing (throwing out) some other block, if required, to make space for the new block.
     2. The block that is thrown out is written back to disk only if it was modified since the most recent time that it was written to/fetched from the disk.
     3. Once space is allocated in the buffer, the buffer manager reads the block from the disk to the buffer, and passes the address of the block in main memory to requester.

## Organization of Records in Files:

**Heap:**  a record can be placed anywhere in the file where there is space

**Sequential**  store records in sequential order, based on the value of the search key of each record.

**Hashing** a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed.

**Indexed Sequential**  Combines Indexed and Sequential file organization.

Records of each relation may be stored in a separate file. In a clustering **file organization** records of several different relations can be stored in the same file.

## HEAP table behaviour:

* Default structure – the HEAP
  + Collection of pages filled with rows
  + A record can be placed anywhere in the file where there is space
  + New set of pages (an extent) added as required
  + A HEAP table has no primary key
  + Rows can be accessed in two ways
    - Serial scan of all pages
    - Through a type of bookmark known as a RID comprising a file number, page number and slot number
  + If a row grows in size, the extra data is put on a new page and a forwarding pointer is used to connect

## Sequential File Organization:

* Suitable for applications that require sequential processing of the entire file
* The records in the file are ordered by a search-key
* Deletion – use pointer chains
* Insertion –locate the position where the record is to be inserted
  + if there is free space insert there
  + if no free space, insert the record in an overflow block
  + In either case, pointer chain

must be updated

* Need to reorganize the file  
   from time to time to restore  
   sequential order

## Indexes:

### Basic Concepts:

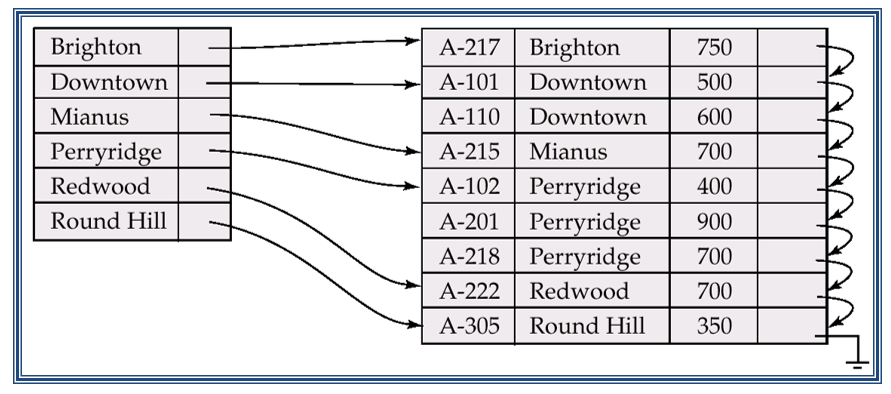
* Indexing mechanisms are used to speed up access to desired data.
* **Search Key** - attribute to set of attributes used to look up records in a file.
* An **index file** consists of records (called **index entries**) of the form
* Index files are typically much smaller than the original file
  + **In ordered indices** search keys are stored in sorted order

### Ordered Indexes:

* In an **ordered index,** index entries are stored sorted on the search key value. E.g., author catalog in library.
* **Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file.
  + Also called **clustering index**
  + The search key of a primary index is usually but not necessarily the primary key.
* **Secondary index**:an index whose search key specifies an order different from the sequential order of the file. Also called   
  non-clustering index**.**
* Index-sequential file**:** ordered sequential file with a primary index.

### Dense Index Files:

* **Dense index** — Index record appears for every search-key value in the file.



### Sparse Index Files:

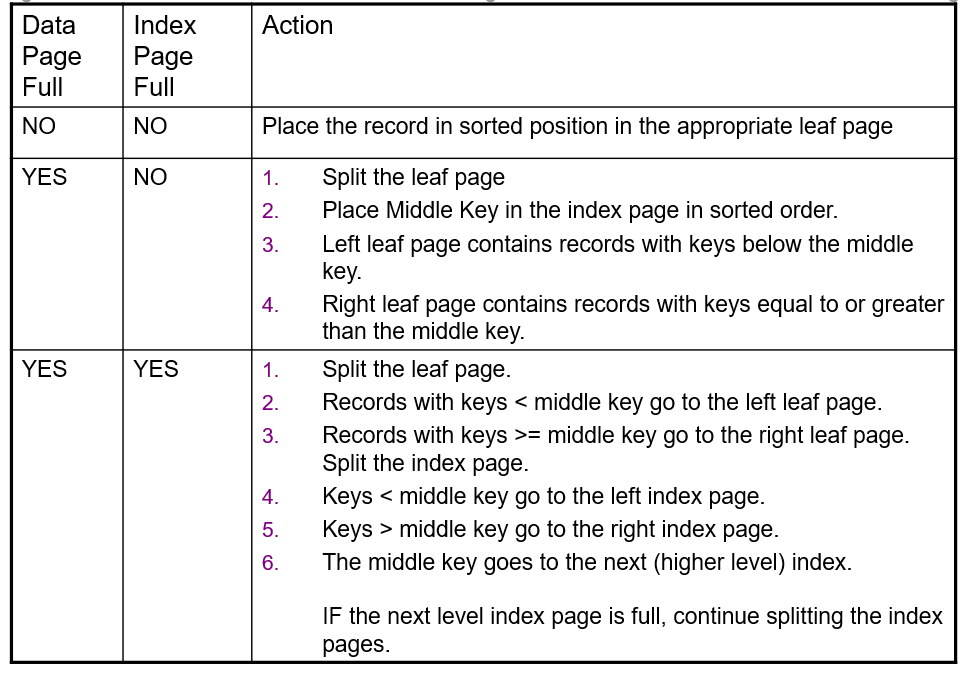
* **Sparse Index**: contains index records for only some search-key values.
  + Applicable when records are sequentially ordered on search-key
* To locate a record with search-key value *K* we:
  + Find index record with largest search-key value < *K*
  + Search file sequentially starting at the record to which the index record points
* Less space and less maintenance overhead for insertions and deletions.
* Generally slower than dense index for locating records.
* Good tradeoff: sparse index with an index entry for every block in file, corresponding to least search-key value in the block.

### B+-Tree Index Files:

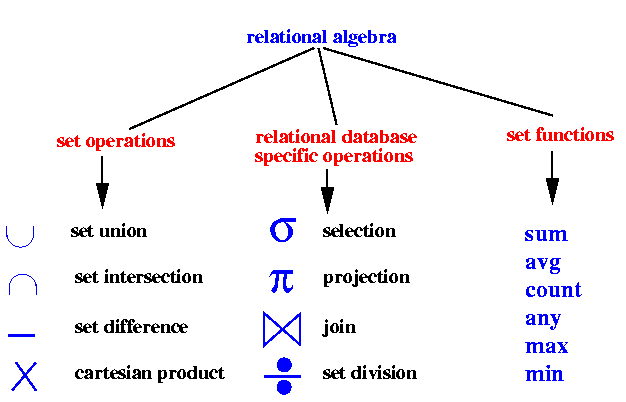
* Disadvantage of indexed-sequential files: performance degrades as file grows, since many overflow blocks get created. Periodic reorganization of entire file is required.
* Advantage of B+-tree index files: automatically reorganizes itself with small, local, changes, in the face of insertions and deletions. Reorganization of entire file is not required to maintain performance.
* Disadvantage of B+-trees: extra insertion and deletion overhead, space overhead.
* Advantages of B+-trees outweigh disadvantages, and they are used extensively.
* A B+tree is a balanced tree in which every path from the root of the tree to a leaf is of the same length
* Each nonleaf node of the tree has between [n/2] and [n] children, (where n is fixed for a particular tree).
* It contains index pages and data pages. The capacity of a leaf has to be 50% or more. For example: if n = 4, then the key for each node is between 2 to 4. The index page will be 4 + 1 = 5 pointers.

Example of a B+ tree with four keys (n = 4) is on the next slide

### The insert algorithm for B+ Tree:



# Relational Algebra Symbols:



# Query Optimization:

## Steps of Query Processing:

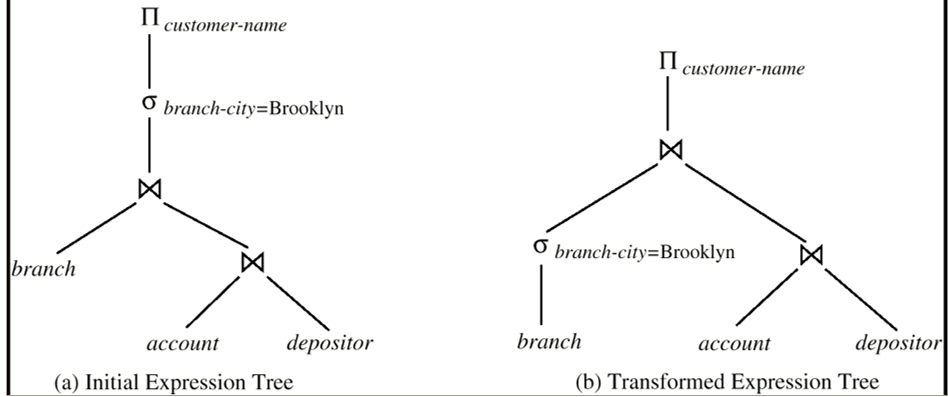
* Parsing and translation
* Optimization
* Evaluation

## Introduction to query optimization

* Alternative ways of evaluating a given query
  + Equivalent expressions
  + Different algorithms for each operation
* Cost difference between a good and a bad way of evaluating a query can be enormous
  + Example: performing a *r* X *s*  followed by a selection *r.A = s.B* is much slower than performing a join on the same condition
* Need to estimate the cost of operations
  + Depends critically on statistical information about relations which the database must maintain
    - E.g. number of tuples
  + Need to estimate statistics for intermediate results to compute cost of complex expressions

## Equivalence in expressions:

Relations generated by two equivalent expressions have the same set of attributes and contain the same set of tuples, although their attributes may be ordered differently.



## Query evaluation plan:

* Generation of query-evaluation plans steps:
  1. Generating logically equivalent expressions
     + Use **equivalence rules** to transform an expression into an equivalent one.
  2. Annotating resultant expressions to get alternative query plans
  3. Choosing the cheapest plan based on **estimated cost**
* The overall process is called **cost based optimization.**

## Statistical Information for Cost Estimation:

* *nr:* number of tuples in a relation *r.*
* *br*: number of blocks containing tuples of *r.*
* *sr*: size of a tuple of *r.*
* *fr:* blocking factor of *r* — i.e., the number of tuples of *r* that fit into one block.
* *V(A, r):* number of distinct values that appear in *r* for attribute *A.*
* *SC*(*A, r*): selection cardinality of attribute *A* of relation *r*; average number of records that satisfy equality on *A*.

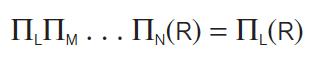
## Query Equivalence Rules:

* Two relational algebra expressions are said to be equivalent if on every legal database instance the two expressions generate the same set of tuples
  + Note: order of tuples is irrelevant
  + Conjunctive selection operations can be deconstructed into a sequence of individual selections.

1. Selection operations are commutative\*.



1. Only the last in a sequence of projection operations is needed, the others can be omitted.



1. Selections can be combined with Cartesian products and theta joins.
   1. σθ(E1 X E2) = E1 θ E2
   2. σθ1(E1 θ2 E2) = E1 θ1∧ θ2 E2
2. Theta-join operations are commutative.  
    *E*1 θ *E*2 = *E*2 θ *E*1
3. Natural join operations are \* associative:

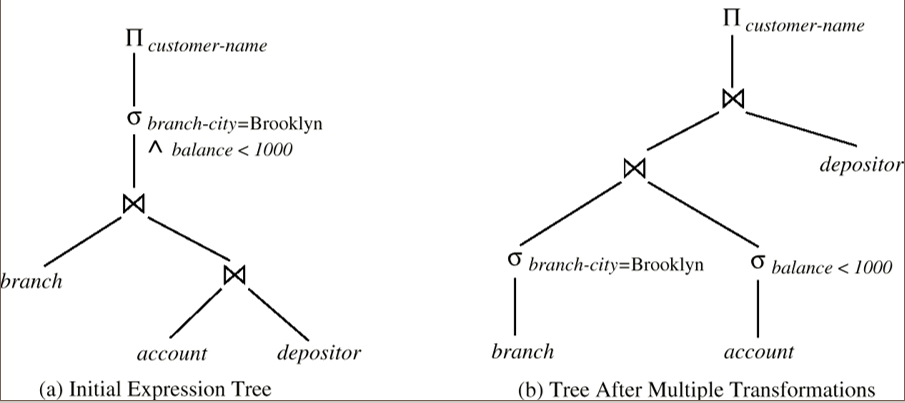
(*E*1 *E2*) *E3 = E*1 (*E*2 *E*3)

\* \* **associative** *grouping* doesn’t matter

## Transformation Example

* Query: Find the names of all customers who have an account at some branch located in Brooklyn.  
  Π*customer-name*(σ*branch-city = “*Brooklyn” (*branch* (*account depositor)))*
* Transformation:  
   Π*customer-name* ((σ*branch-city =“*Brooklyn” (*branch*))  
   (*account*  *depositor*))
* Performing the selection as early as possible reduces the size of the relation to be joined.

## Multiple Transformations:



## Heuristic Optimization:

* Heuristic optimization transforms the query-tree by using a set of rules that typically improve execution performance:
  + Perform selection early (*reduces the number of tuples*)
  + Perform projection early (*reduces the number of attributes*)
  + Perform most restrictive *selection* and *join* operations before other similar operations.
  + Some systems use only heuristics, others combine heuristics with partial cost-based optimization.

## Steps in Typical Heuristic Optimization:

1. Deconstruct conjunctive selections into a sequence of single selection operations.
2. Move selection operations down the query tree for the earliest possible execution.
3. Execute first those selection and join operations that will produce the smallest relations.
4. Replace Cartesian product operations that are followed by a selection condition by join operations.
5. Deconstruct and move as far down the tree as possible lists of projection attributes.
6. Identify those subtrees whose operations can be pipelined, and execute them using pipelining.

## Steps in optimization:

1. Write the relational algebra expression to satisfy the query
2. Create a R.A.T. To represent the query
3. Optimize the tree
4. Annotate the tree with physical file information to create an execution plan

## Create a R.A.T. To represent the query:

* Leaf node created for each base relation.
* Non-leaf node created for each intermediate relation produced by RA operation.
* Root of tree represents query result.
* Sequence is directed from leaves to root.

## Optimize the tree:

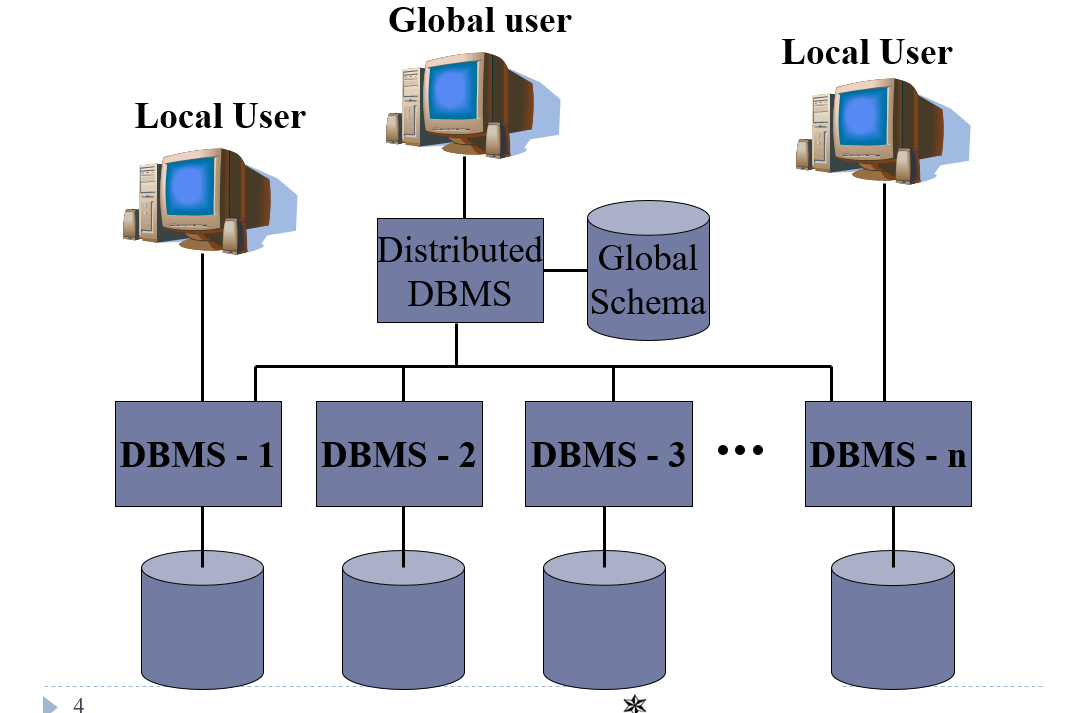
* Move selection operations down the query tree for the earliest possible execution.
* Execute first those selection and join operations that will produce the smallest relations.
* Replace Cartesian product operations that are followed by a selection condition by join operations.
* Deconstruct and move as far down the tree as possible lists of projection attributes.

# Distributed Databases:

## DDBMS characteristics:

* Collection of logically related shared data.
* Data split into fragments.
* Fragments may be replicated.
* Fragments/replicas allocated to sites.
* Sites linked by a communications network.
* Data at each site is under control of a DBMS.
* DBMSs handle local applications autonomously.
* Each DBMS participates in at least one global application.

## Heterogeneous Distributed Database Environment



## Functions of a DDBMS:

* Expect DDBMS to have at least the functionality of a DBMS.
* Also to have following functionality:
  + Extended communication services.
  + Extended system catalog.
  + Distributed query processing.
  + Extended security control.
  + Extended concurrency control.
  + Extended recovery services.

## Distributed Database Design:

**Three key issues:**

Fragmentation: Relation may be divided into a number of sub-relations, which are then distributed.

Allocation: Each fragment is stored at site with "optimal" distribution.

Replication: Copy of fragment may be maintained at several sites.

### Fragmentation:

* Definition and allocation of fragments carried out strategically to achieve:
  + Locality of Reference
  + Improved Reliability and Availability
  + Improved Performance
  + Balanced Storage Capacities and Costs
  + Minimal Communication Costs.
* Involves analyzing most important applications (uses), based on quantitative/qualitative information.
* Quantitative information may include:
  + frequency with which an application is run;
  + site from which an application is run;
  + performance criteria for transactions and applications.
* Qualitative information may include transactions that are executed by application, type of access (read or write), and predicates of read operations.

#### Data Allocation:

Four alternative strategies regarding placement of data:

* Centralized
  + Consists of single database and DBMS stored at one site with users distributed across the network.
* Partitioned
  + Database partitioned into disjoint fragments; each fragment assigned to one site
* Complete Replication
  + Consists of maintaining complete copy of database at each site.
* Selective Replication
  + Combination of partitioning, replication, and centralization.

#### Why Fragment?

* Usage
  + Applications work with views rather than entire relations.
* Efficiency
  + Data is stored close to where it is most frequently used.
  + Data that is not needed by local applications is not stored
* Parallelism
  + With fragments as unit of distribution, transaction can be divided into several sub-queries that operate on fragments.
* Security
  + Data not required by local applications is not stored and so not available to unauthorized users.
* Disadvantages
  + Performance
  + Integrity.

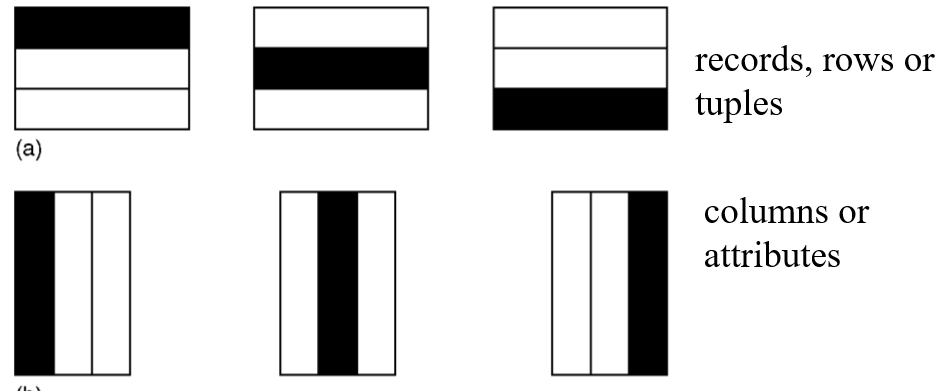
#### Correctness of Fragmentation:

* Completeness
  + If relation (TABLE) R is decomposed into fragments R1, R2, ... Rn, each data item that can be found in R must appear in at least one fragment.
* Reconstruction
  + Must be possible to define a relational operation that will reconstruct R from the fragments.
  + Reconstruction for horizontal fragmentation is UNION operation and JOIN for vertical.
* Disjointness
  + If data item disappears in fragment Ri, then it should not appear in any other fragment.
  + Exception: vertical fragmentation, where primary key attributes must be repeated to allow reconstruction.
  + For horizontal fragmentation, data item is a tuple (record or row)
  + For vertical fragmentation, data item is an attribute.

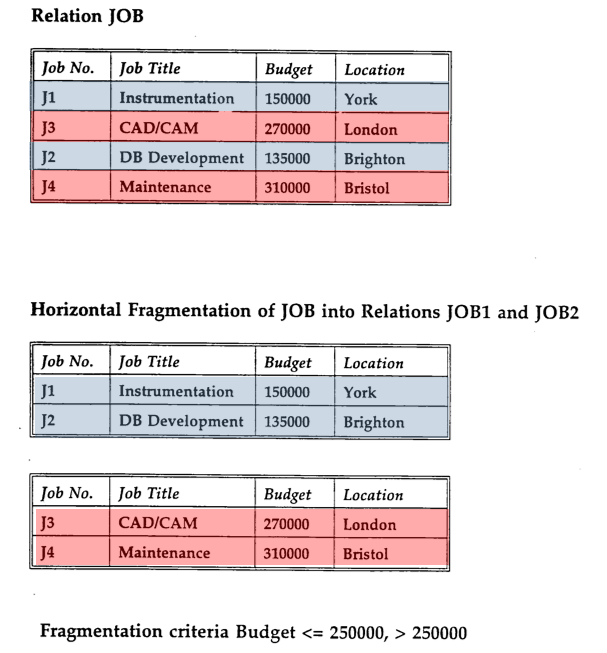
#### Types of Fragmentation:

* Four types of fragmentation:
  + Horizontal
  + Vertical
  + Mixed
  + Derived
  + Other possibility is no fragmentation:
  + If relation (table) is small and not updated frequently, may be better not to fragment relation.

#### Horizontal and Vertical Fragmentation:



##### (Primary) horizontal with tuples allocated to different fragments:



#### Primary Horizontal Fragmentation:

JOB1

select \* from JOB where Budget <= 250000

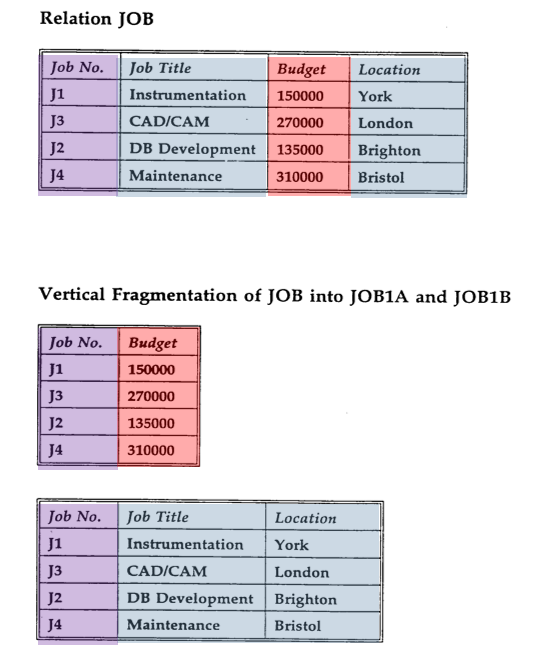
JOB2

select \* from JOB where Budget > 250000

Reconstruction of JOB

(select \* from JOB1) UNION (select \* from JOB2)

##### Vertical with attributes allocated to different fragments:



## Derived Horizontal Fragmentation (cont.) SQL:

* Fragment Region
  + REGION1:

Select \*

from REGION where Region = ‘South East’;

* + REGION2:

Select \*

from REGION where Region <> ‘South East’;

* JOB1A

Select [Job no], [Job title], Budget, J.Location

From JOB as J inner join REGION1 as R1

on J.Location = R1.Location

* JOB2A

Select [Job no], [Job title], Budget, J.Location

From JOB as J inner join REGION2 as R2

on J.Location = R2.Location

## Transparencies in a DDBMS:

* Distribution Transparency
  + Fragmentation Transparency
  + Location Transparency
  + Replication Transparency
  + Local Mapping Transparency
  + Naming Transparency
* Distribution Transparency
  + Fragmentation Transparency
  + If Fragmentation transparency is provided by DDBMS, then the user does not need to know that the data is fragmented.
  + As the result, database accesses are based on global schema, so that the user does not need to specify fragment name or location.
* Distribution Transparency
  + Location Transparency

With location transparency, the user must know how the data has been fragmented but still does not have to know the location of the data.

* Distribution Transparency
  + Replication Transparency
  + The user is unaware of the replication of fragments.
  + Replication transparency is implied by location transparency. However, to remember, it is possible for a system not to have location transparency but to have replication transparency.
* Distribution Transparency
  + Replication Transparency
  + The user is unaware of the replication of fragments.
  + Replication transparency is implied by location transparency. However, to remember, it is possible for a system not to have location transparency but to have replication transparency.
* Distribution Transparency
  + Naming Transparency
  + As in centralise database, each item in a distributed database must have a unique name.
  + Therefore DDBMS must ensure that no two sites create a database object with the same name.
  + One common solution to this problem is to create a central Name Server, which has the responsibility for ensuring uniqueness of all names in the system. However, this approach may result in:
    - Loss of some local autonomy
    - Performance problems, if the central site becomes a bottleneck
    - Low availability, if the central site fails, the remaining sites cannot create any new database objects.
* Objectives do not change just because the database is distributed
* Our distributed system still has a local DBMS with all the functionality of previously
* Added components to handle the global issues