Database Environment and Development Process

Definitions

- **Database**: Organised collection of logically related data
- **Data**: Stored representations of meaningful objects and events.
    - Structured – tables
    - Unstructured – text, images, video, documents
- **Information**: Data processed to increase knowledge for the person using the data
- **Metadata**: Data that describes the properties and context of user data.
    - Descriptions of the properties or characteristics of the data, including data types, field sizes, allowable values and data context.

Disadvantages of File Processing

- *Program-Data Dependence*: all programs maintain metadata for each file they use
- *Duplication of Data*: Different systems/programs have separate copies of the same data
- *Limited Data Sharing*: no centralised control of data
- *Lengthy Development Times*: Programmers must design their own file formats.
- *Excessive Program Maintenance*: 80% of information systems budget

Problems with Data Dependency

- Each application programmer must maintain their own data
- Each application program needs to include code for the metadata of each file
- Each application program must have its own processing routines for reading, inserting, updating and deleting data
- Lack of coordination and central control
- Non-standard file formats

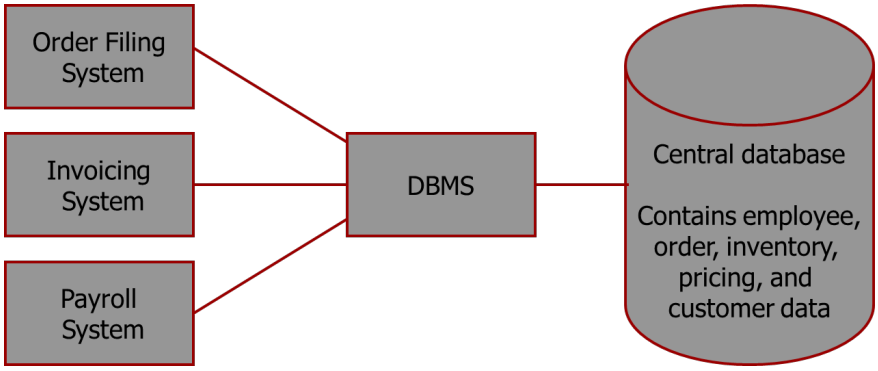
Data Redundancy

- Problems:
    - Waste of space to have duplicate data
    - Causes more maintenance issues
    - Data changes in one file could cause inconsistencies
    - Compromises in data integrity
- Solution: **The database approach**
    - Central repository of shared data
    - Data is managed by a controlling agent
    - Stored in standardised, convenient form
    - Requires a Database Management System (DBMS)

Database Management System (DBMS)

- A software system is used to create, maintain and provide controlled access to use databases
- Since all data is shared in a central database, there is no longer the need for separate systems and programs to maintain they own copy of the data. This reduces duplication and increases integrity.

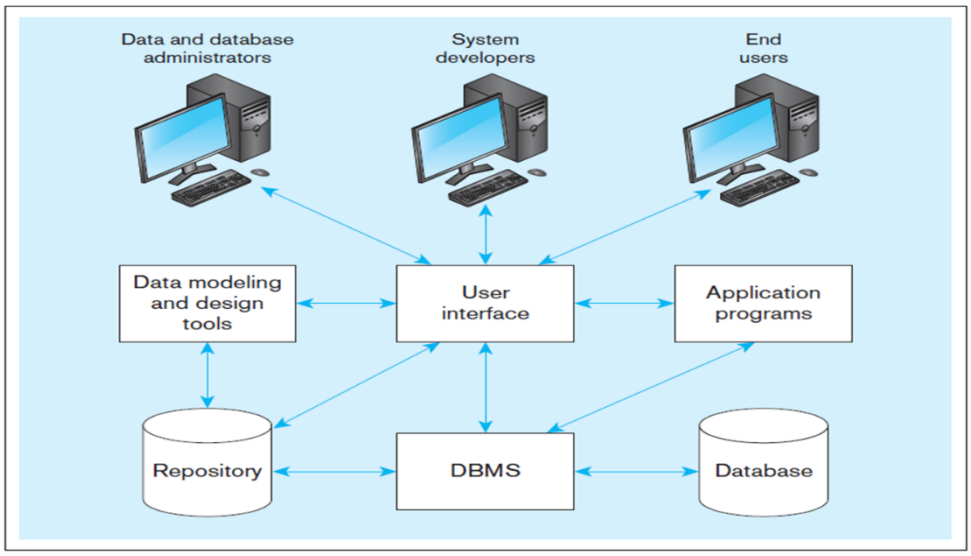
- Role of a DBMS:
    - Enables data to be shared
    - Presents end user with an integrated view of the data
    - Receives and translates application requests into operations required to fulfill the requests
    - Hides database's internal complexity from application programs and users
    - Has data dictionary
    - Helps in performance tuning
    - Helps in security management



Relational Database Approach

- Advantages:
    - Program-data independence
    - Planned data redundancy
    - Improved data consistency
    - Improved data sharing
    - Increased application development productivity
    - Enforcement of organisational standards
    - Improved data quality
    - Improved data accessibility and responsiveness
    - Reduced program maintenance
    - Improved decision support
- Costs and risks:
    - New, specialised personnel
    - Installation and management cost and complexity
    - Conversion costs
    - Need for explicit backup and recovery
    - Organisational conflict

- Components of the database environment:



  - **Data modelling and design tools** – automated tools used to design databases and application programs
  - **Repository** – centralised storehouse of metadata
  - **DBMS** – software for managing the database
  - **Database** – storehouse of the data
  - **Application programs** – software using the data
  - **User Interface** – text, graphical displays, menus, etc
  - **Data/Database Administrators** – personnel responsible for maintaining the database
  - **System Developers** – personnel responsible for designing databases and software
  - **End Users** – people who use applications and databases

Approaches to Database Design and Development

- SDLC – Waterfall
  - System Development Life-cycle
  - Detailed, well-planned process
  - Time consuming but comprehensive
  - Long development cycle
- Prototyping
  - Rapid Application Development (RAD)
  - Cursory attempt at conceptual data modelling
  - Define database during development of initial prototype
  - Repeat implementation and maintenance activities with new prototype versions

Relational Model

- More than 90% of current database applications are built on relational database systems
- It is a simple and uniform data model that relies on a firm mathematical foundation (relational algebra)
- There are 3 major aspects of a relational data model
  - Data structure: consists of two dimensional tables/relations whose attributes are of atomic types

- Data integrity: consists of two general integrity rules, namely entity integrity and referential integrity
- Data manipulation: Consists of a set of algebraic operators for data manipulations.

Entity Relationship Diagrams

Top Down Approach

- We use the Top Down approach for database design
- Three levels in a top down approach
  - *Conceptual Level* – ER/EER diagrams (No PKs, only PIs)
  - *Logical Level* – Transform the ERD into tables (Has PKs and FKs)
  - *Physical Level* – Select the database and write code

Entity Relationship Diagram

- ERD – Entity Relationship Diagram
- Three Components:
  - Entity
  - Attribute
  - Relationship

Entity

- **Entity** – an object/concept that needs to be saved in the database
  - For example – Entities in Macquarie university include: Student, Staff, Units, Department
- Types of Entities:
  - *Strong Entity* – Has its own ID and can stand alone
    - E.g. customer, order, product, season, building
  - *Weak Entity* – Depends on the strong entity to exist. Cannot have its own ID.
    - E.g. Episode (in a season), Room (in a building)
  - *Associative Entity* – Many-many relationship, Ternary relationship

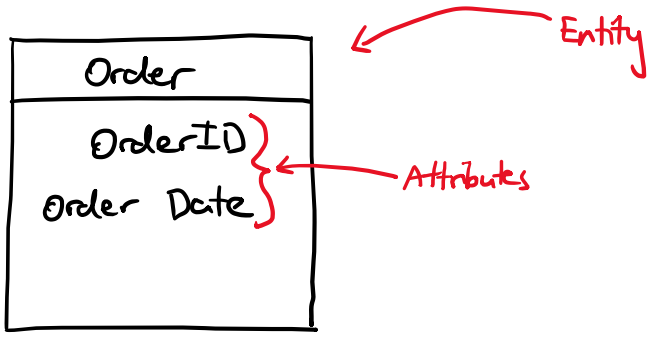
Attributes

- **Attributes** – Properties of entities
  - E.g. For Customer – Customer ID, CustomerName, CustomerAddress, CustomerPhoneNumber. CustomerDOB
- Types of attributes:
  - Identifier Attribute: PI Attribute (E.g. CustomerID)
  - Simple vs Composite Attribute:
    - *Simple*: Cannot be broken down further (E.g. CustomerID, CustomerEmail)
    - *Composite*: Can be broken down further (E.g. CustomerName -> CustomerFirstName, CustomerLastName)
  - Single-Valued vs Multi-Valued Attribute (refers to data):
    - Single-valued: Can have only one value (E.g. CustomerName)
    - Multi-valued: Can have multiple values. I.e. it can have a dropdown menu associated with it. (E.g. CustomerQualifications)
  - Stored/Derived Attribute:
    - E.g. – *Stored*: CustomerDOB -> *Derived*: Age

- Required/Optional Attribute:
  - Required: needs a value and cannot be *null*
  - Optional: Can be *null*

Example

- Order is identified using OrderID. Details of order such as CustomerID and order data are saved. Draw an entity for Order with attributes.

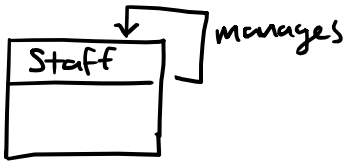


- Students are identified using StudentID. Details of student such as StudentID, name and address are saved. Students can enrol in multiple units. Draw an entity for Student along with attributes.
  - (using *draw.io*)

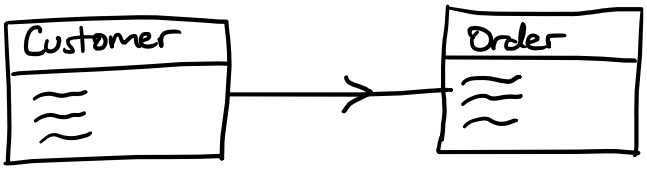
Student	
PK	Student ID
	StudentName StudentAddress

Relationships in an ER Model

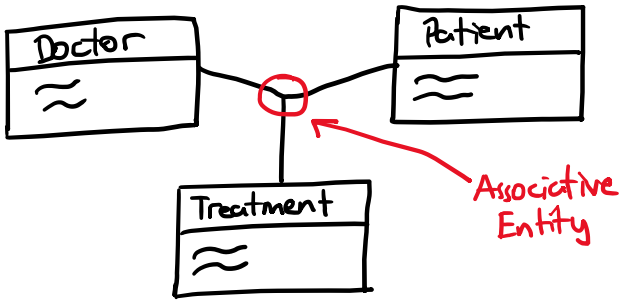
- Relationships connect the entities together.
- Components:
  - Degree
  - Cardinality
  - Constraint
- Degree: defines how many entities are part of the relationship.
  - Unary: Involves one entity (self-relationship)
  - Binary: Involves two entities
  - Ternary: Involves three or more entities
- Unary:



- Binary:



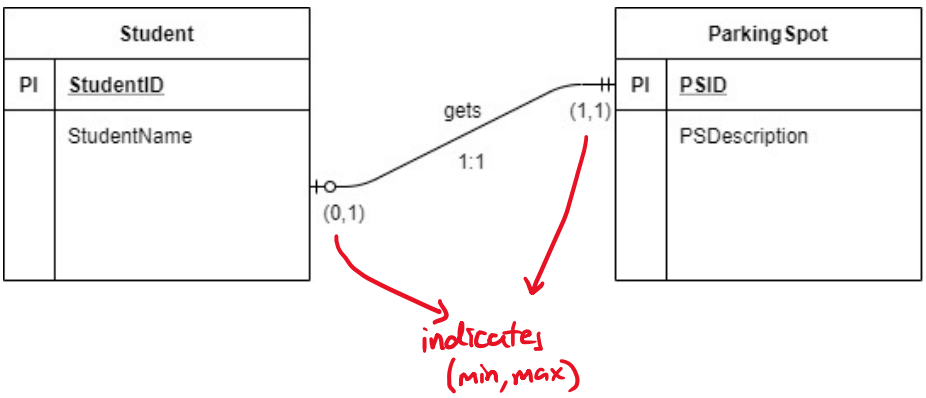
- Ternary:



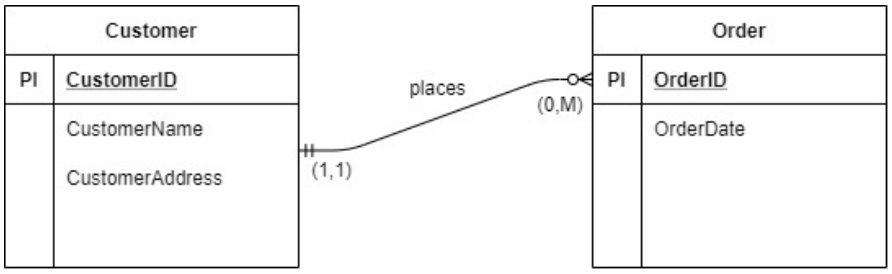
- Cardinality and Constraint:
  - *Cardinality*: Maximum number in a relationship (Can be 1 or many)
  - *Constraint*: Minimum number in a relationship (Can be 0 or 1)
- Terminology:
  - Minimum 0 – Optional
  - Minimum 1 – Mandatory
  - Maximum 1 – One
  - Maximum Many - Many

Examples

- Every student at MQ gets a parking spot. Students are identified using StudentID. Parking spot is identified using an ID. Find the relationship between the two entities.
  - Every Student gets max 1 parking spot and every parking spot belongs to max 1 student. Every student gets minimum 1 parking spot, but every parking spot doesn't necessarily have an allocated student.
  - Optional 1 to Mandatory 1

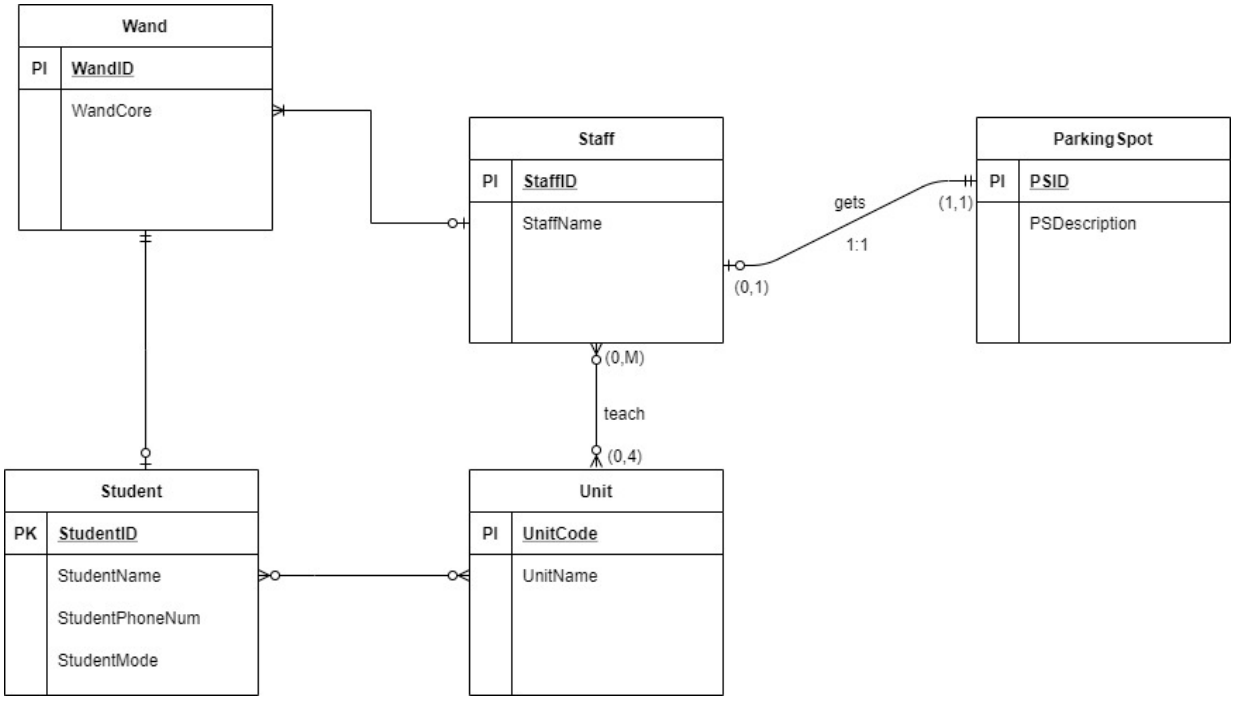


- Customers are identified using an identifier. Name, address gets stored. They place orders. Orders are identified using an identifier. Date of order gets recorded. Find the relationship between the two entities.
    - Mandatory 1 to Optional Many



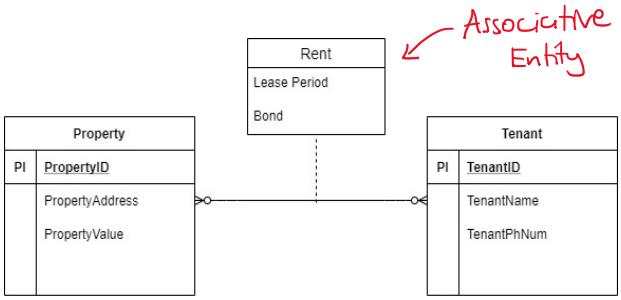
- Order contain multiple products. Find the relationship between the two entities.
    - Optional Many to Mandatory Many

- Staff at Hogwarts are allocated a parking spot. Staff are rostered to teach multiple units. A unit could be taught be multiple staff. Students enrol in units. Every student has a wand. Staff have multiple wands. Dumbledore requests a report on staff details at the end of every week. Draw an ER diagram for this question.

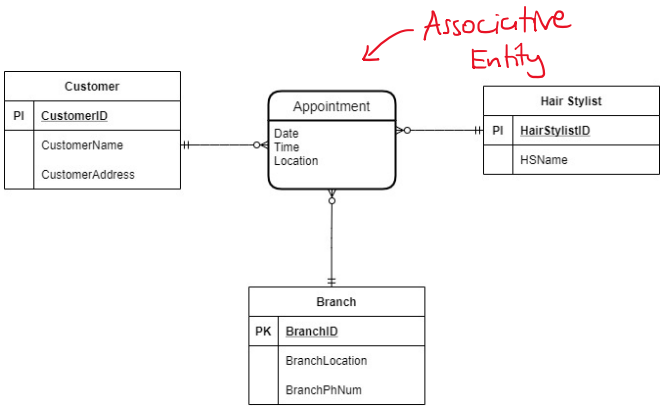


Associative Entity

- An Associative Entity (AE) is a type of entity.
- It is created during either a:
    - Many-Many Relationship, or a



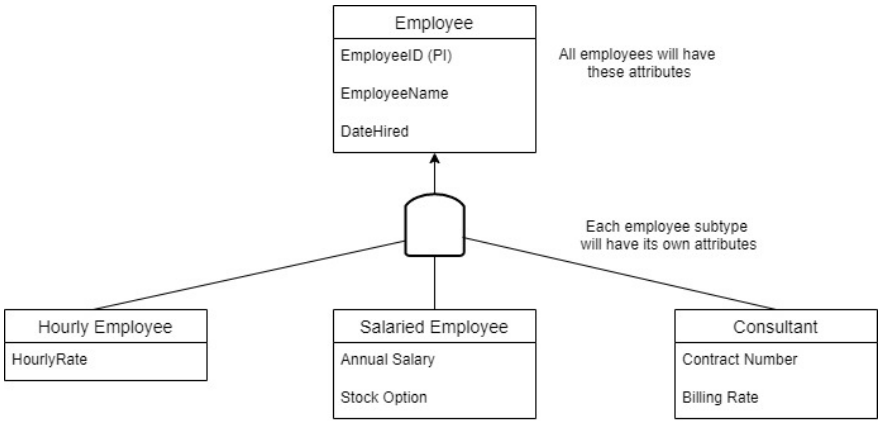
    - Ternary Relationship



Enhanced Entity Relationship Diagram

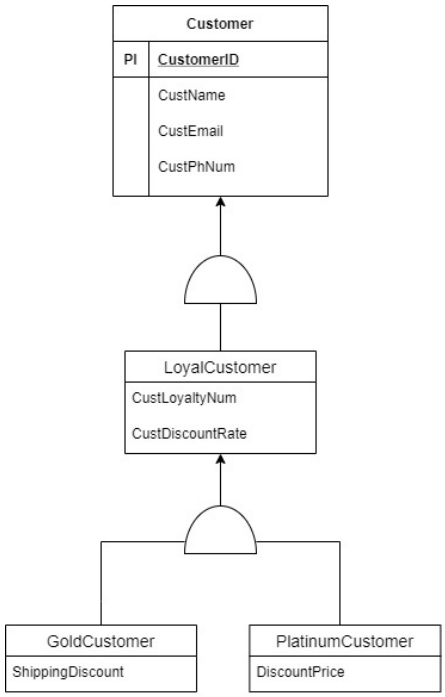
Supertypes and Subtypes

- Enhanced ER Model: Extends original ER model with new modelling constructs
- Subtype: A subgrouping of the entities in an entity type that has attributes distinct from those in other subgroupings.
- Supertype: A generic entity type that has a relationship with one or more subtypes.
- Attribute Inheritance:
    - Subtype entities inherit values of all attributes of the supertype
    - An instance of a subtype is also an instance of the supertype.



Multi-Level EER

- Following is an example of a Multi-level EER.

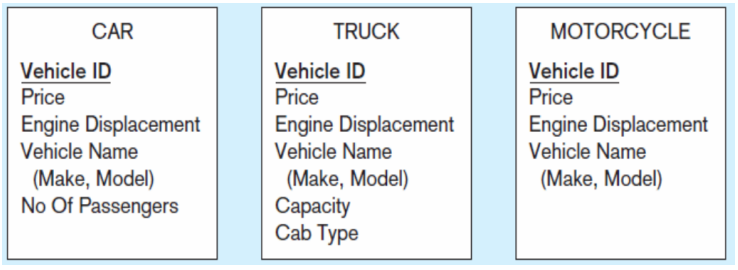


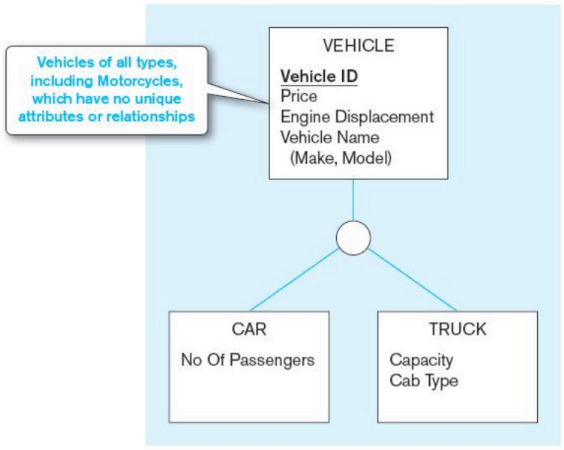
Overlap/Disjoint Constraints

- Disjointness constraints refer to whether an instance of a supertype may simultaneously be a member of two (or more) subtypes.
    - *Disjoint*: an instance of the supertype can be only ONE of the subtypes (indicated with an "o" in the semi-circle)
    - *Overlap*: An instance of the supertype could be more than one of the subtypes (indicated with an "d" in the semi-circle)

Generalization and Specialisation

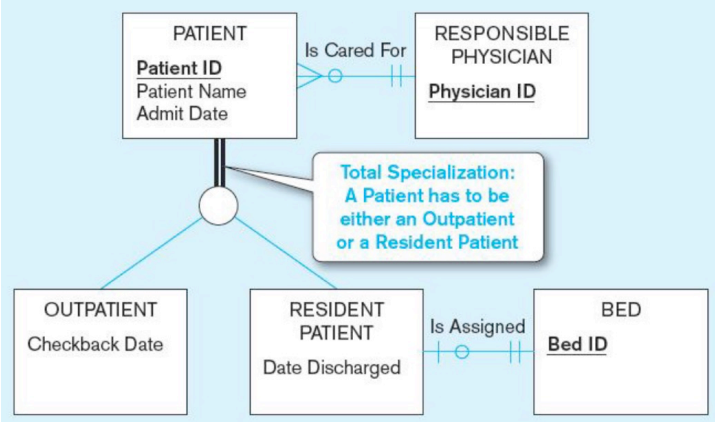
- Generalisation: The process of defining a more general entity type from a set of more specialised entity types (bottom-up)
    - For example:





So we put the shared attributes in a supertype

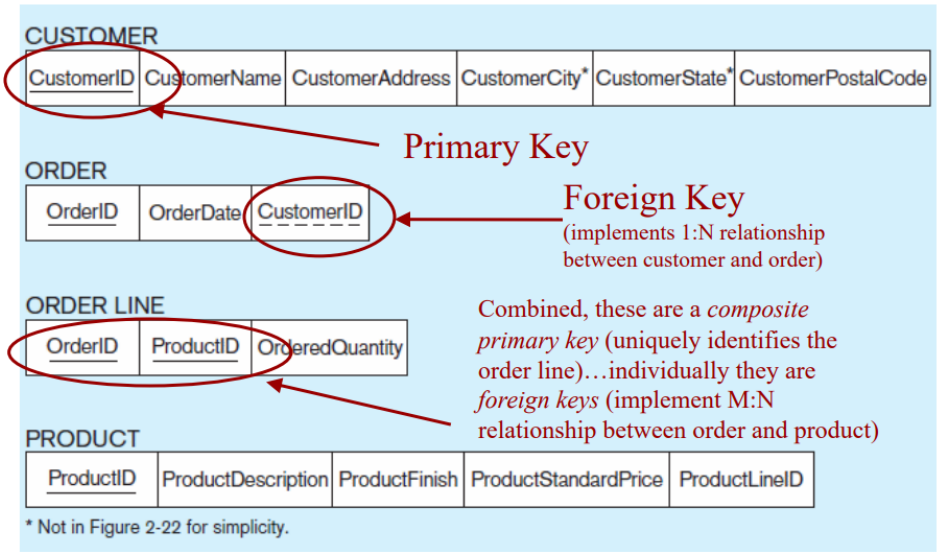
- Specialisation: The process of defining one of more subtypes of the supertype and forming supertype/subtype relationships (top-down)
    - **Total Specialisation Rule**: An instance of a supertype MUST also be a member of at least one subtype (indicated with a double line)



    - **Partial Specialisation Rule**: An instance of a supertype doesn't have to be a member of any of the subtypes (indicated with a single line)

Key Fields

- Keys are special fields that serve two main purposes.
    - **Primary Keys**: unique identifiers of the relation. Examples include employee numbers, social security numbers, etc. This guarantees that all rows are unique. Cannot have null values.
    - **Composite Primary Key**: When more than one column creates a primary key, that becomes a composite primary key.
    - **Foreign Keys**: identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship). When a PK is being used in another table, the referencing column is known as a FK.
- Keys are used as indexes to speed up the response to user queries.

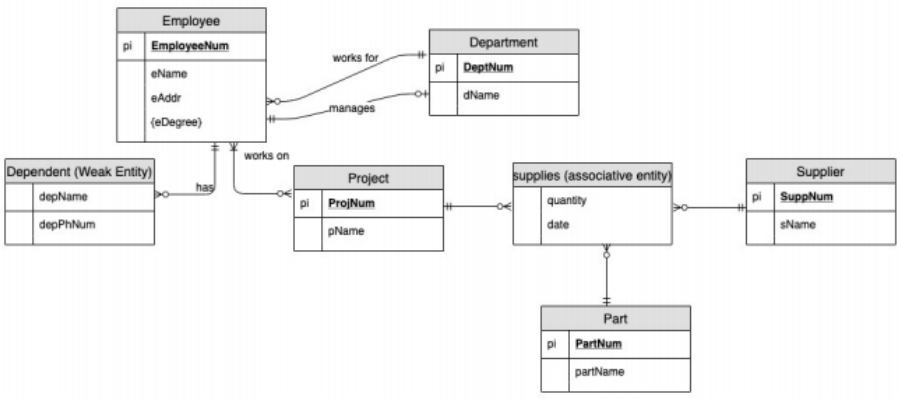


Logical Design

- Logical Design involves transformation.
    - We transform entities into tables
    - Attributes into columns
    - Relationships into Primary and Foreign Keys

Transforming EER Diagrams into Relations

- There is an 8 step process to transform EER diagrams into tables/relations.
- We will be using the following ER diagram as an example:



Step 1: Strong Entities

- For each entity in the ER model, create a relation (i.e. a table that includes all the simple attributes). Make sure to identify the primary key for the relation. If there is a composite attribute, you can expand them. Leave multi-valued attributes out.
- Strong entity becomes a table.
- PI becomes PK
- In our example, strong entities are:
    - Employee (EmployeeNum(pk), eName, eAddress)

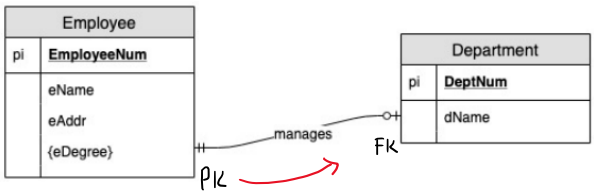
- Project (ProjNum, pName)
- Department (DeptNum, dName)
- Supplier(SuppName, sName)
- Part(PartNum, partName)

Step 2: Weak Entities

- For each weak entity in the ER model, create a relation which includes all the simple attributes. The primary key of the relation is the combination of the primary keys of the 'owner' and the main attribute of the weak entity itself.
- Strong entity's PK becomes a composite PK in the weak entity.
- Example:
  - Dependent (EmployeeNum(pk, fk), depName(pk), depPhNum)

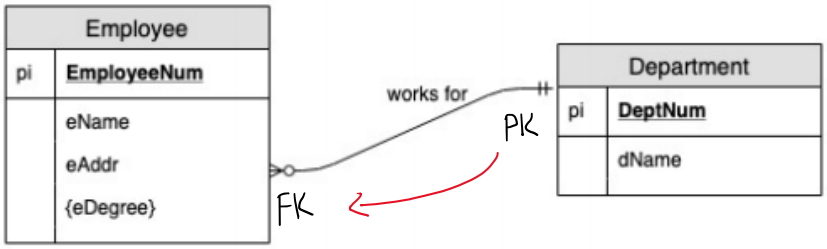
Step 3: 1:1 Relationship

- For each 1 to 1 relationship identify the two relations that correspond to the entities participating in the relationship. Choose the PK of the relation (usually the one with the mandatory constraint) and it a foreign key of the other relation.
- Example:
  - Department (DeptNum(pk), dName, *managerEmpNum(fk)*)



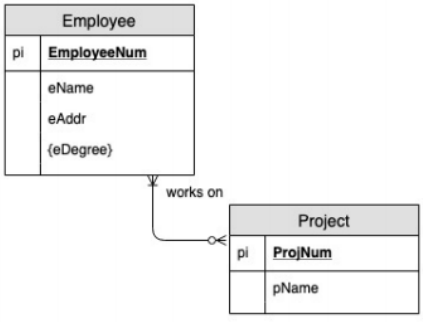
Step 4: 1:N Relationship

- For each binary 1 to N relationship, identify the relations that represent the participating entity at the N side of the relationship. Include as foreign key in the relation that holds the N side, the primary key of the other entity.
- Example:
  - Employee (EmployeeNum(pk), eName, eAddress, DeptNum(fk))



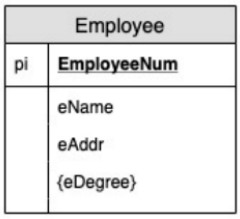
Step 5: M:M Relationship

- For each binary M to M relationship, create a new relation to represent the relationship. The primary key of the new relation is the combination of the primary keys of the two connected entities. This is an associative entity.
- Example:
  - Works on (EmployeeNum(pk, fk), ProjNum(pk, fk))



Step 6: Multi-valued attributes

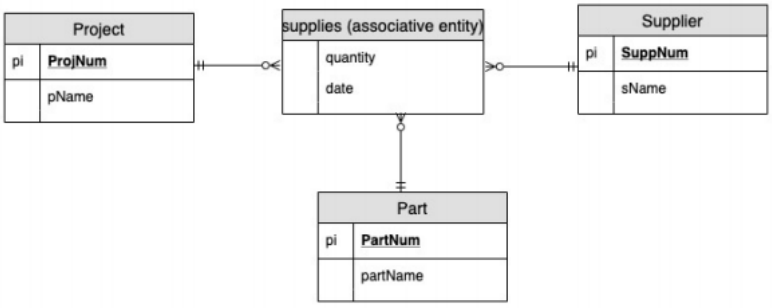
- For each multi-valued attribute, create a new relation that includes the multivalued attribute and the primary key of the entity where the multivalued attribute is attached.
- Example:



  - Employee(EmployeeNum, eName, eAddress)
  - EDegree(EmployeeNum(pk, fk), degreeName(pk))

Step 7: Associative Entities

- For each ternary relationship, create a new relation to represent the relationship. The primary key of the new relation is the combination of the primary keys of the participating entities that hold the many side. In more cases of a ternary relationship, all the participating entities hold a many side.
- Example:
  - Supplies(SuppNum, ProjNum, PartNum, date, quantity)



Final Tables

- Use the updated version of all tables and combine them to get the result:
  - Employee(EmployeeNum(pk), eName, eAddress, DeptNum(fk))
  - Department(DeptName(pk), dName, managerEmployeeNum(fk))
  - Project(ProjNum(pk), pName)
  - Supplier(SuppNum(pk), sName)

- Part(PartNum(pk), partName)
- Dependent(EmployeeNum(pk, fk), depName(pk), depPhNum)
- Work On(EmployeeNum(pk, fk), ProjNum(pk, fk))
- EDegree(EmployeeNum(pk, fk), degreeName(pk))
- Supplies(SuppNum(pk, fk), ProjNum(pk, fk), PartNum(pk, fk), date, quantity)

Transforming an EER Diagram

- ER Diagram (Steps 1-7)
- EER Diagram (Steps 1-7, 8a/8b/8c/8d, Repeat Steps 2-7)
- Using the following example:
  - The university keeps information of students. Students can be grouped into undergraduate (UG), coursework postgraduate (CPG), and research postgraduate (RPG). For UG, the degree major is recorded. For CPG, all their previous degrees are required. For all students, their student ID and personal details are required.
  - RPG can work in a specific research area. The information of research area is stored, which includes the research area id and description.
- Step 1: Student(StudID(pk), StName); ResearchArea(ResID(pk), Rdesc)
- Step 2: No weak entities
- Step 3: No 1:1 relationship
- Step 4: No 1:M relationship
- Step 5: No M:M relationship
- Step 6: No Multi-valued attributes
- Step 7: No associative entities

Step 8a

- Applies for total/partial, overlap/disjoint
- Supertype's PK becomes PK in all the subtypes (just the PK gets inherited)
- Example:
  - UG(StudID(pk, fk), Major)
  - CPG(StudID(pk, fk))
  - RPG(StudID(pk, fk))
- Repeat Step 2-7:
  - Step 4: 1:M Relationship
    - RPG(StudID(pk, fk), ResID(fk))
  - Step 6: Multi-valued attribute
    - SDegree (StudID(pk, fk), DegreeName(pk))

Final Tables

- Student(StudID(pk), StName)
- ResearchArea(ResID(pk), Rdesc)
- UG(StudID(pk, fk), Major)
- CPG(StudID(pk, fk))
- RPG(StudID(pk, fk), ResID(fk))
- SDegree (StudID(pk, fk), DegreeName(pk))

Step 8b

- Only applies for total constraint, overlap/disjoint
- Supertype is disintegrated and all the attributes would be inherited by the subtypes
- Example:
    - UG(StudID(pk), SName, Major)
    - CPG(StudID(pk), SName)
    - RPG(StudID(pk), SName)

Step 8c

- Applies for total/partial, Only for Disjoint
- Subtypes are disintegrated and supertypes take all of their attributes.
- But to differentiate between subtypes, an extra attribute is added.
- Example:
    - Student (StudID(pk), StName, Major, *StudType*)

Step 8d

- Applies for total/partial, Only for Overlap
- Subtypes are disintegrated and supertypes take all of their attributes.
- But to differentiate between subtypes, an extra attribute for every type is added.
- Example:
    - 8d will not apply

Data Normalisation

- Data normalisation is a tool to validate and improve a logical design so that it satisfies constraints that avoid unnecessary duplication of data.
- The process of decomposing relations with anomalies to produce smaller, well-structured relations.

Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete and update rows without causing inconsistencies.
- The goal is to avoid these anomalies:
    - **Insertion anomaly** – adding new rows forces user to create duplication data
    - **Deletion anomaly** – deleting rows may cause a loss of data that would be needed for future rows
    - **Modification anomaly** – changing data in a row forces changes in other rows because of duplication

Functional Dependencies and Keys

- Functional Dependency – The value of one attribute (the determinant) determines the value of another attribute.
- Candidate key:
    - A unique identifier. One of the candidate keys will become the primary key
    - Each non-key field is functionally dependent on every candidate key

Un-Normalised Form (UNF)

- This is a form of a data table where there is at least one multi-valued attribute

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

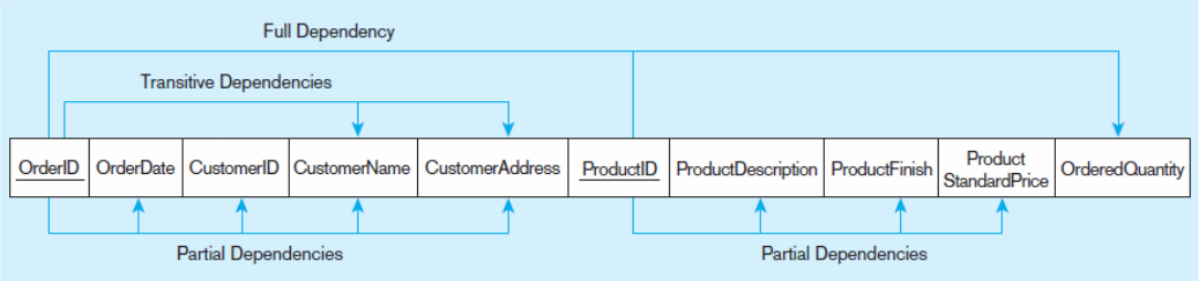
First Normal Form (1NF)

- No multi-valued attributes

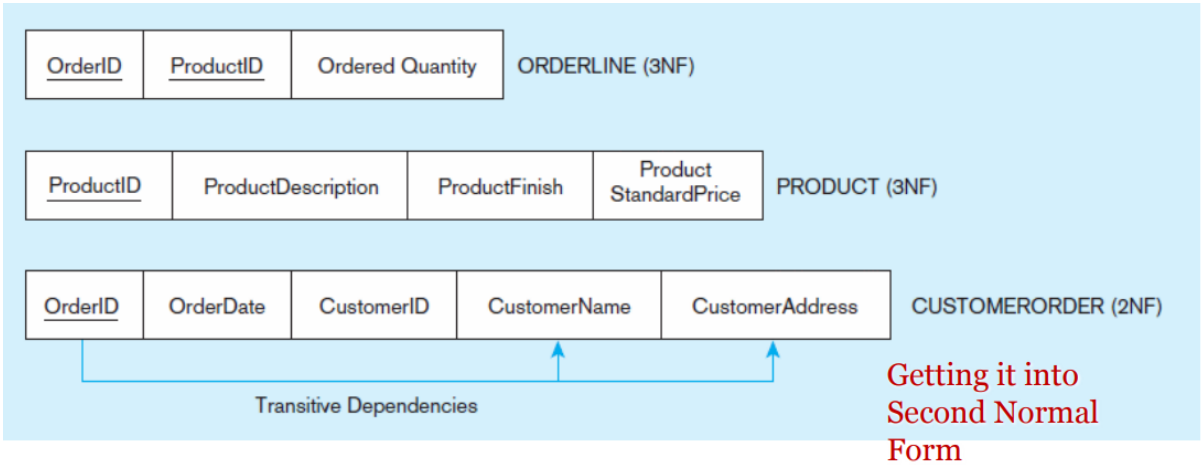
OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Second Normal Form (2NF)

- This is when the table is in 1NF and there is NO partial dependency
- Every non-key attribute is fully functionally dependent on the entire primary key



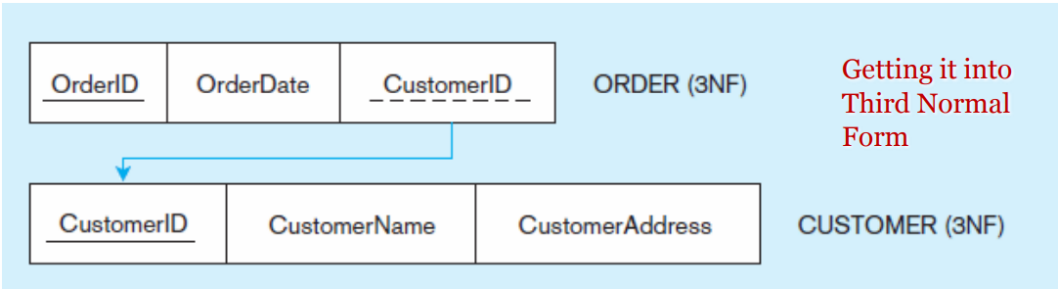
- **OrderID** → OrderDate, CustomerID, CustomerName, CustomerAddress
- **CustomerID** → CustomerName, CustomerAddress
- **ProductID** → ProductDesciption, ProductFinish, ProductStandardPrice
- **OrderID**, **ProductID** → OrderQuantity



- Partial Dependencies are removed, but there are still transitive dependencies

Third Normal Form (3NF)

- This is when the table is in 2NF and has no transitive dependencies (functional dependencies on non-primary key attributes)
- Non-key determinant with transitive dependencies go into a new table; non key determinant becomes primary key in the new table and stays as a foreign key in the old table.



Database Application Development

Examples of Database Application

- Banking database: withdraw, deposit, check balance
- Library database: search by title, search by author
- Student database: search for unit info, enrol in a unit, print transcript

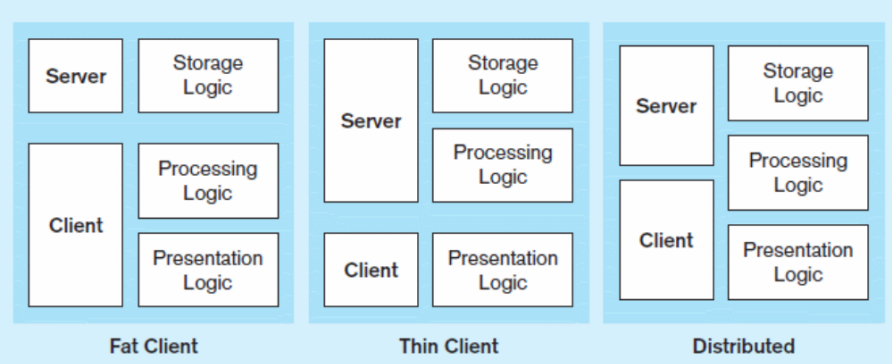
Client-Server Architectures

- Processes distributed between clients and servers
- Client (usually a PC) that requests and uses a service
- Server (PC/mini/mainframe) that provides a service
- For DBMS, the server is a database server

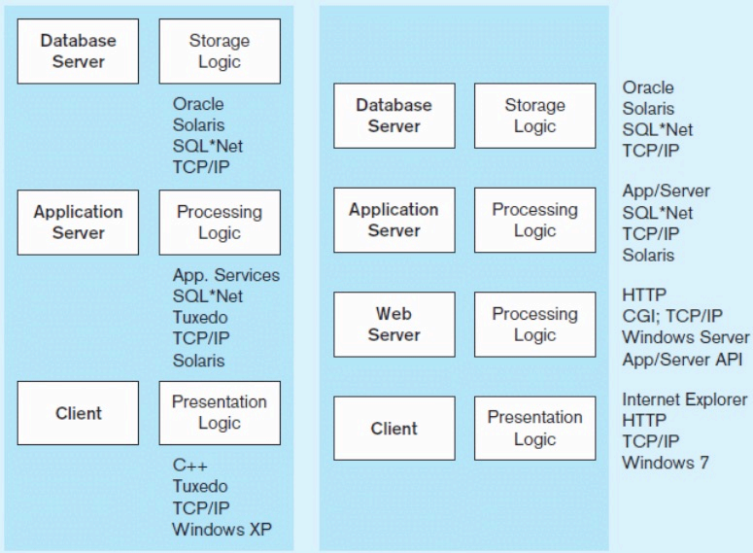
Application Logic in Client-Server Systems

- **Presentation Logic** (GUI Interface):
  - Input – keyboard, mouse
  - Output – monitor, printer
- **Processing Logic** (Procedures, functions, programs):
  - I/O processing
  - Business rules
  - Data management
- **Storage Logic** (DBMS activities):
  - Data storage/retrieval

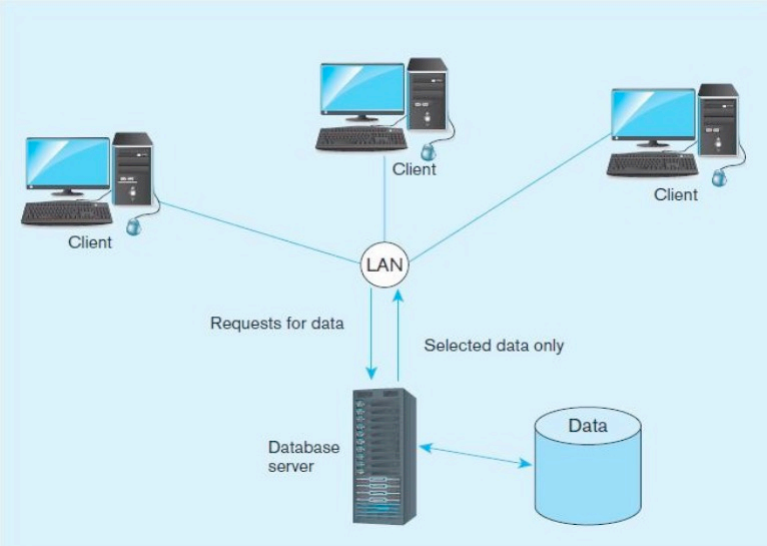
- Application Partitioning: Placing portions of the application code in different locations (client or server) after it is written
- Advantages:
  - Improved performance
  - Improved interoperability
  - Balanced workloads
- Common Logic Distributions:
  - Two-tier client-server environments



- Three-tier and *n*-tier client-server environments



Two-tier Architectures



- Departmental in scope (few users)
- Not mission-critical
- Low transaction volumes
- Common programming languages
  - Java, VB.Net, C#
- Interface database via middleware, API's

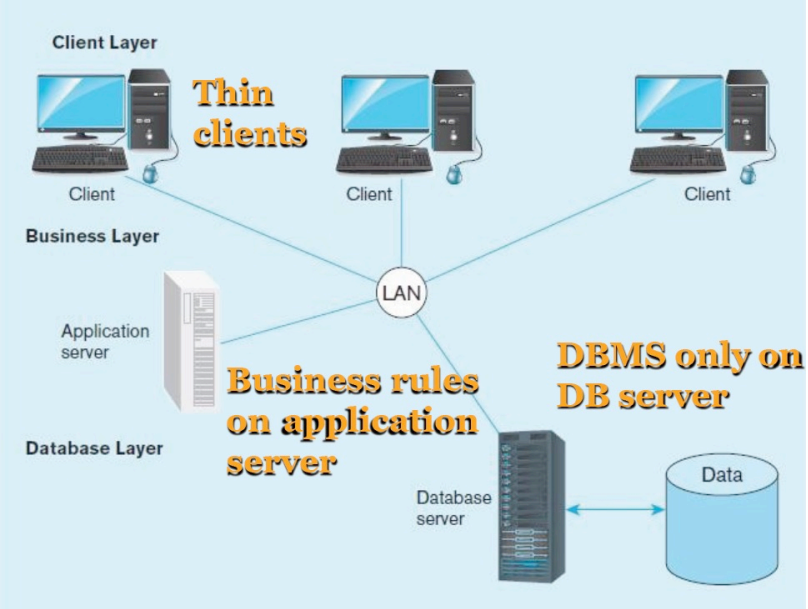
Middleware and API's

- **Middleware** – software that allows an application to interpolate with other software without requiring users to understand and code low-level operations
- **Application Program Interface (API)** – routines that an application uses to direct the performance of procedures by the computer's operating system
  - Common database API's – ODBC, ADO.Net, JDBC

- Steps for using databases via Middleware API's
    - Identify and register a database driver
    - Open a connection to a database
    - Execute a query against the database
    - Process the results of the query
    - Repeat steps 3-4 as necessary
    - Close the connection to the database

Three-tier Architectures

- Client (Browser):
    - GUI Interface (I/O processing)
- Application Server (Web Server):
    - Business Rules
- Database Server (DBMS):
    - Data storage



Thin Client

- An application where the client (PC) accessing the application primarily provides the user interfaces and some application processing, usually with no or limited local data storage
- Usually, thin client application is a Web browser, and the three-tier architecture involves a Web application

Web Application Components

- Database Server – hosts the DBMS
    - E.g. Oracle, SQL Server
- Web Server – receives and responds to browser requests using HTTP Protocol
    - E.g. Apache, Internet Information Services (IIS)
- Application Server – software building blocks for creating dynamic web sites
    - E.g. MS ASP .NET framework, Java EE, ColdFusion, PHP

- Web Browser – client program that sends web requests and receives web pages
    - E.g. Internet Explorer, Safari
- Languages for creating web pages:
    - HTML, XML, JavaScript/VBScript, CSS

Processing in 3-tier Applications

- Static page requests:
    - .htm or .html requests handled by the Web Server
- Dynamic page requests:
    - .jsp, .aspx, and .php requests are routed to the application server
    - Server-side processing by JSP servlet
    - Database access via database middleware

Considerations in 3-tier Applications

- Stored procedures:
    - Code logic embedded in DBMS
    - Improve performance, but proprietary
- Transactions:
    - Involve many database updates
    - Either all must succeed, or none should occur
- Database connections:
    - Maintaining an open connection is resource-intensive
    - Use of connection pooling

Benefits of Stored Procedures

- Performance improves for compiled SQL statements
- Reduced network traffic
- Improve security
- Improved data integrity
- Thinner Clients

Benefits of Three-Tier Architectures

- Scalability
- Technological flexibility
- Long-term cost reduction
- Better match of systems to business needs
- Improved customer service
- Competitive advantage
- Reduced risk

Data Warehousing

Data Warehouse

- A subject-oriented, integrated, time-variant, non-updatable collection of data used in support of management decision-making processes.
    - Subject Oriented: e.g. customers, patients, students, products
    - Integrated: consistent naming conventions, formats, encoding structures
    - Time variant: can study trends and changes
    - Non-updatable: read-only, periodically refreshed
- **Data Mart**: A data warehouse that is limited in scope
- Need for data warehousing:
    - Integrated, company-wide view of high quality information (from disparate databases)
    - Separation of operational informational systems and data (for improved performance)
- Organisational trends motivating data warehouses:
    - No single system of records
    - Multiple systems not synchronised
    - Organisational need to analyse activities in a balanced way
    - Customer relationship management
    - Supplier relationship management

Issues with Company-Wide View

- Inconsistent key structures
- Synonyms
- Free-form vs. Structured fields
- Inconsistent data values
- Missing data

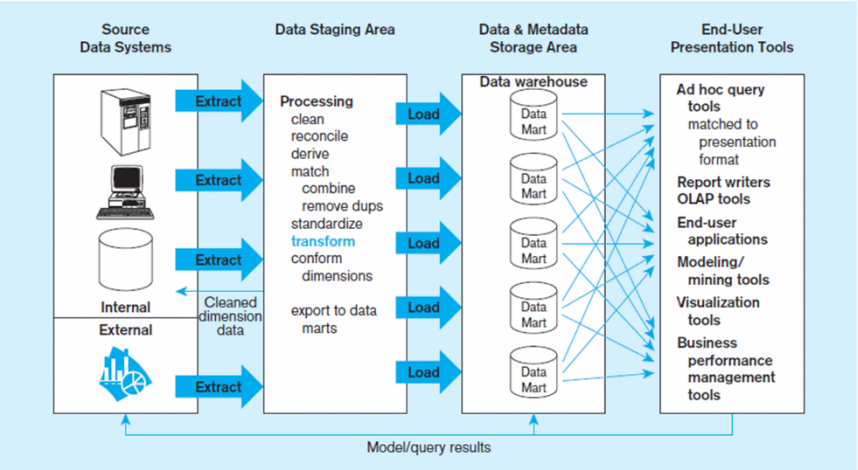
Operational and Informational Systems

- **Operational System** – a system that is used to run a business in real time, based on current data; also called a system of record
- **Informational system** – a system designed to support decision making based on historical point-in-time and prediction data for complex queries or data-mining applications.

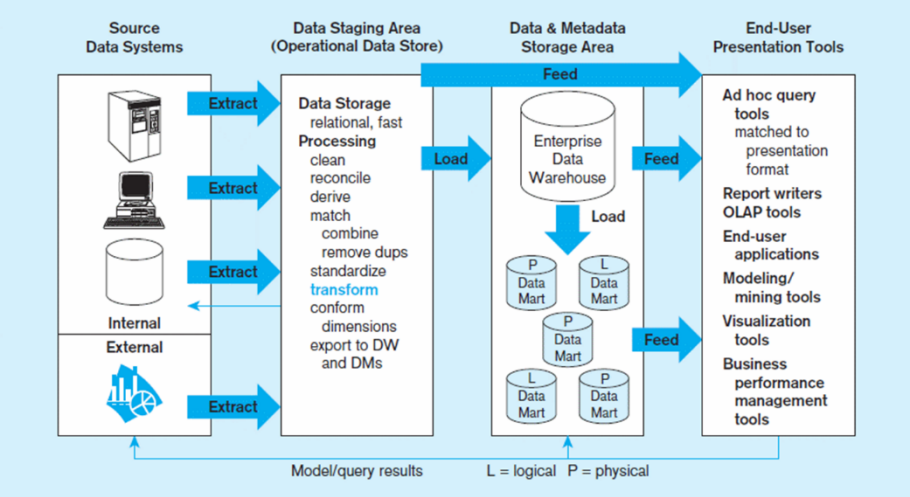
Characteristic	Operational Systems	Informational Systems
Primary purpose	Run the business on a current basis	Support managerial decision making
Type of data	Current representation of state of the business	Historical point-in-time (snapshots) and predictions
Primary users	Clerks, salespersons, administrators	Managers, business analysts, customers
Scope of usage	Narrow, planned, and simple updates and queries	Broad, ad hoc, complex queries and analysis
Design goal	Performance: throughput, availability	Ease of flexible access and use
Volume	Many constant updates and queries on one or a few table rows	Periodic batch updates and queries requiring many or all rows

Data Warehouse Architectures

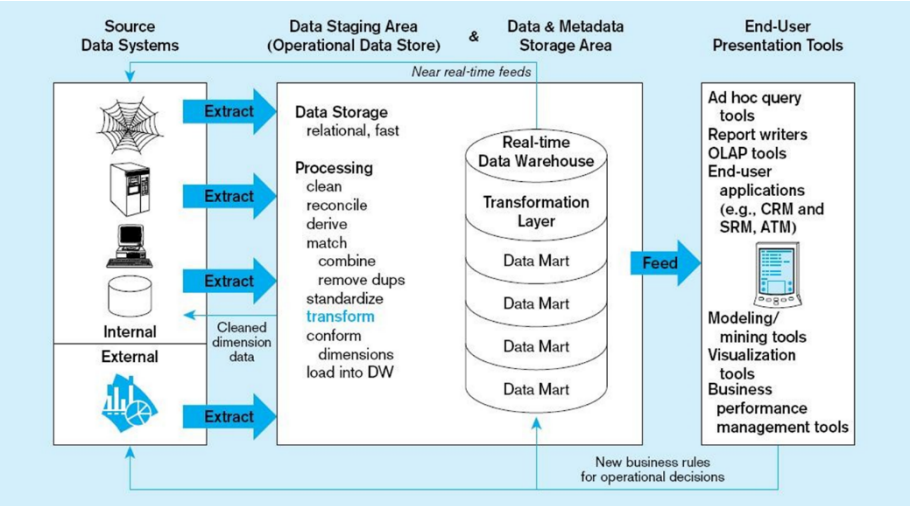
- *All of these involve some form of extract, transform and load (ETL)*
  - Independent Data Mart



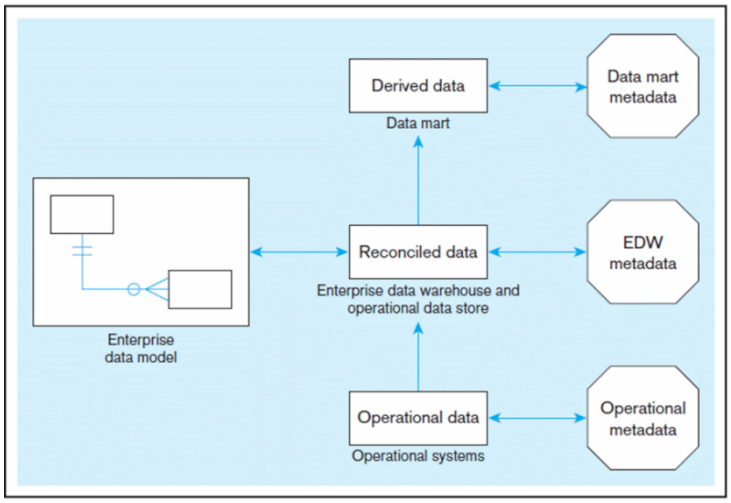
  - Dependent data mart and Operational data store



  - Logical data mart and Real-time data warehouse



- Three-layer architecture

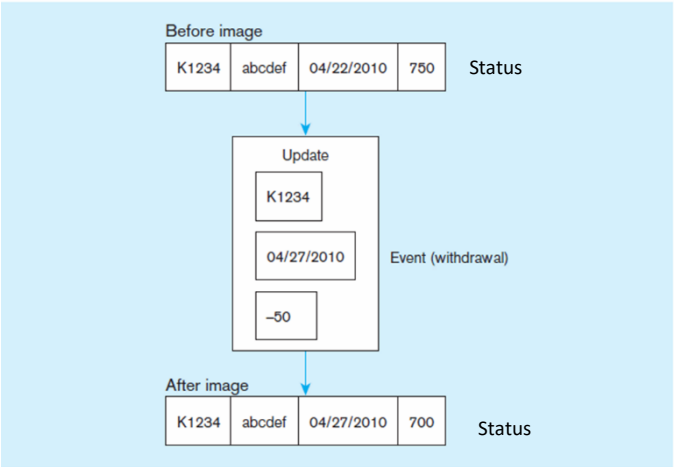


Data Warehouse vs Data Mart

Data Warehouse	Data Mart
Scope	**Scope**
• Application independent • Centralized, possibly enterprise-wide • Planned	• Specific DSS application • Decentralized by user area • Organic, possibly not planned
Data	**Data**
• Historical, detailed, and summarized • Lightly denormalized	• Some history, detailed, and summarized • Highly denormalized
Subjects	**Subjects**
• Multiple subjects	• One central subject of concern to users
Sources	**Sources**
• Many internal and external sources	• Few internal and external sources
Other Characteristics	**Other Characteristics**
• Flexible • Data oriented • Long life • Large • Single complex structure	• Restrictive • Project oriented • Short life • Start small, becomes large • Multi, semi-complex structures, together complex

Data Characteristics Status vs Even Data

- Event – a database action (create/update/delete) that results from a transaction



- Transient Operational data – changes to existing records are written over previous records
- Periodic warehouse data – periodic data are never physically altered or deleted once they have been added to the store

Derived Data

- Objectives:
    - Ease of use for decision support applications
    - Fast response to predefined user queries
    - Customised data for particular target audiences
    - Ad-hoc query support
    - Data mining capabilities
- Characteristics:
    - Detailed (mostly periodic) data
    - Aggregate (for summary)
    - Distributed (to departmental servers)

Surrogate Keys

- Dimension table keys should be surrogate (non-intelligent and non-business related), because:
    - Business keys may change over time
    - Helps keep track of non-key attribute values for a given production key
    - Surrogate keys are simpler and shorter
    - Surrogate keys can be the same length and format for all keys

Grain of the Fact Table

- Granularity of the fact table refers to the level of detail wanted:
    - **Transactional grain** – finest level
    - **Aggregated grain** – more summarised
    - Finer grains – better market basket analysis capability
    - Finer grain – more dimension tables, more rows in fact table
    - In Web-based commerce, finest granularity is a click

Duration of the Database

- Natural duration – 13 months or 5 quarters
- Financial institutions may need longer duration
- Older data is more difficult to source and cleanse

Size of the Fact Table

- Depends on the number of dimensions and the grain of the fact table
- Number of rows – product of number of possible values for each dimension associated with the fact table
    - Example:

Total number of stores = 1,000
Total number of products = 10,000
Total number of periods = 24 (2 years' worth of monthly data)
Total rows = 1,000 stores x 5,000 active products x 24 months
= 120,000,000 rows (!)

Variations of the Star Schema

- Multiple Fact Tables:
    - Can improve performance
    - Often used to store facts for different combinations of dimensions
    - Conformed dimensions
- Fact-less Fact Tables:
    - No non-key data, but foreign keys for associated dimensions
    - Used for: tracking events and inventory coverage

Normalising Dimension Tables

- Multivalued Dimensions:
    - Facts qualified by a set of values for the same business subject
    - Normalisation involves creating a table for an associative entity between dimensions
- Hierarchies
    - Sometimes a dimension forms a natural, fixed depth hierarchy
    - Design options:
        - Include all information for each level in a single de-normalised table
        - Normalise the dimension into a nested set of 1:M table relationships

Slowly Changing Dimensions (SCD)

- To maintain knowledge of the past:
- Kimble's Approach:
    - Type 1: replace old data with new (lose historical data)
    - Type 2: for each changing attribute, create a current value field and several old-valued fields (multivalued)
    - Type 3: create a new dimension table row each time the dimension object changes, with all dimension characteristics at the time of change (most common approach)

10 Essential Rules for Dimensional Modelling

1. Use atomic facts
2. Create single-process fact tables
3. Include a data dimension for each fact table
4. Enforce consistent grain
5. Disallow null keys in fact tables
6. Honour hierarchies
7. Decode dimension tables
8. Use surrogate keys
9. Conform dimensions
10. Balance requirements with actual data

Data Warehouse Advances

- Columnar databases:
    - Issue of Big Data (huge volume, often unstructured)
    - Columnar databases optimize storage for summary data of few columns
    - Data compressions
- NoSQL
    - "Not only SQL"

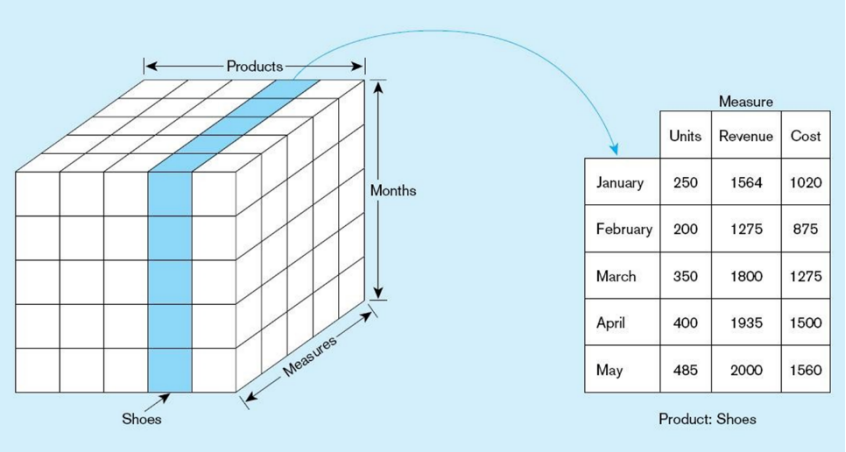
- Deals with unstructured data

The User Interface Metadata

- Identify subjects of the data mart
- Identify dimensions and facts
- Indicate how data is derived from enterprise data warehouses, including derivation rules
- Identify available reports and predefined queries
- Identify data analysis techniques
- Identify responsible people

Online Analytical Processing (OLAP) Tools

- The use of a set of graphical tools that provides users with multidimensional views of their data and allows them to analyse the data using simple windowing techniques
- Relational OLAP (ROLAP)
  - Traditional relational representation
- Multidimensional OLAP (MOLAP)
  - Cube structure
- OLAP Operations
  - Cube slicing – come up with 2D view of data
  - Drill-down – going from summary to more detailed views



Data Mining

- Knowledge discovery using a blend of statistics, AI and computer generated graphics
- Goals:
  - Explanatory – explain observed events or conditions
  - Confirmatory – confirm hypotheses
  - Exploratory – analyse data for new or unexpected relationships

- Data mining techniques:

Technique	Function
Regression	Test or discover relationships from historical data
Decision tree induction	Test or discover if . . . then rules for decision propensity
Clustering and signal processing	Discover subgroups or segments
Affinity	Discover strong mutual relationships
Sequence association	Discover cycles of events and behaviors
Case-based reasoning	Derive rules from real-world case examples
Rule discovery	Search for patterns and correlations in large data sets
Fractals	Compress large databases without losing information
Neural nets	Develop predictive models based on principles modeled after the human brain

- Text Mining – Discovering meaningful information algorithmically based on computational analysis of unstructured textual information

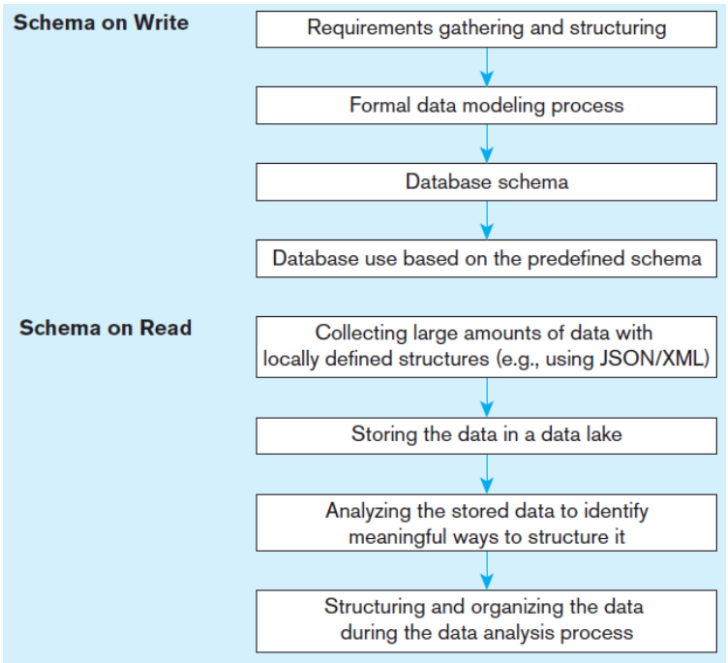
Big Data

Introduction

- Big Data – Data that exists in very large volumes and many different varieties (data types), that needs to be processed at a very high speed
- Analytics – Systematic analysis and interpretation of data. Typically using mathematical, statistical and computational tools – to improve our understanding of a real-world domain

Characteristics of Big Data

- The 5 V's of Big Data:
  - Volume – much larger quantity of data than typical for relational databases
  - Variety – lots of different data types and formats
  - Velocity – data comes at a very fast rate
  - Veracity – traditional data quality methods don't apply. Judging the data's accuracy and relevance
  - Value – big data is valuable to the bottom line, and for fostering good organisational decisions
- **Big data is schema on read**, rather than schema on write:
  - *Schema on Write* – Pre-existing data model, how traditional databases are designed (relational databases)
  - *Schema on Read* – data model determined later, depends on how you want to use it. Capture and store the data, and worry about how you want to use it later

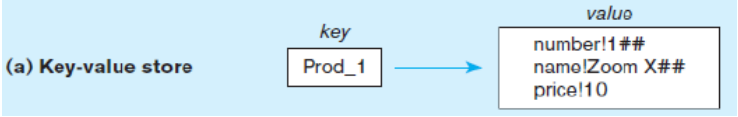


NoSQL

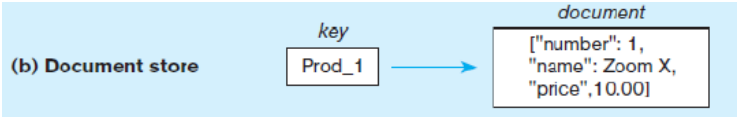
- NoSQL – “Not only SQL”
- A category of recently introduced data storage and retrieval technologies not based on the relational model
- Scaling out rather than scaling up
- Natural for a cloud environment
- Supports schema on read
- Largely open source
- Not ACID compliant
- BASE – basically available, soft state, eventually consistent

NoSQL Classifications

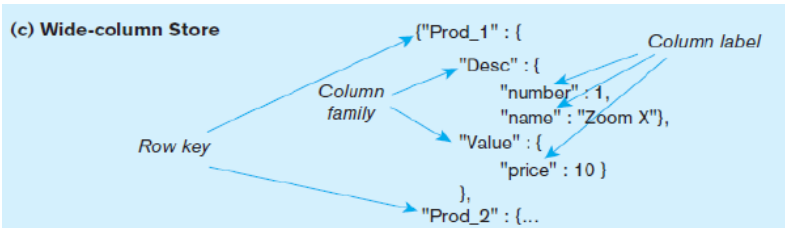
- Key-value stores:
  - A simple pair of a key and an associated collection of values. Key is usually a string. Database has no knowledge of the structure or measuring of the values.



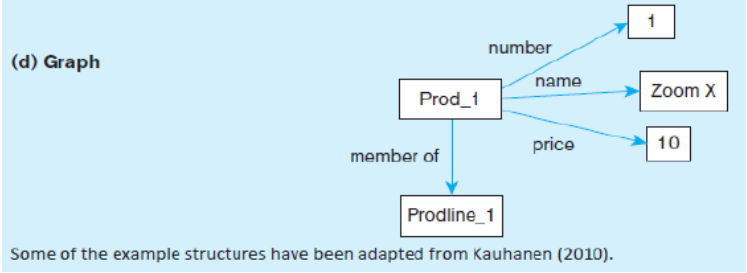
- Document stores:
  - Like a key-value store, but “document” goes further than “value”. Document is structured so specific elements can be manipulated separately.



- Wide-column stores:
  - Rows and columns. Distribution of data based on both key values (records) and columns, using “column groups/families”.



- Graph-oriented database:
  - Maintain information regarding the relationships between data items. Nodes with properties, connections between nodes (relationships) can also have properties.



Some of the example structures have been adapted from Kauhanen (2010).

Analytics

- Historical precedents to analytics:
    - Management information systems (MIS) -> Decision Support Systems (DSS) -> Executive Information Systems (EIS)
    - DSS evolved into Business Intelligence (BI)
- Business intelligence – a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information
- Analytics encompasses more than BI:
    - Umbrella term that includes BI
    - Transform data into useful information
    - Infrastructure for analysis
    - Data clean up processes
    - User interfaces

Types of Analytics

- **Descriptive analytics** – describes the past status of the domain of interest using a variety of tools through techniques such as reporting, data visualisation, dashboards and scorecards
- **Predictive analytics** – applies statistical and computational methods to data regarding past and current events to predict future events
- **Prescriptive analytics** – uses results of predictive analytics along with optimisation and simulation tools to recommend actions that will lead to a desired outcome

Use of Descriptive Analytics

- Descriptive analytics was the original emphasis of BI
- Reporting of aggregate quantitative query results
- Tabular or data visualisation displays
- Dashboard – a few key indicators
- Scorecard – dashboard with broader range
- OLAP – online analytical processing

SQL OLAP Querying

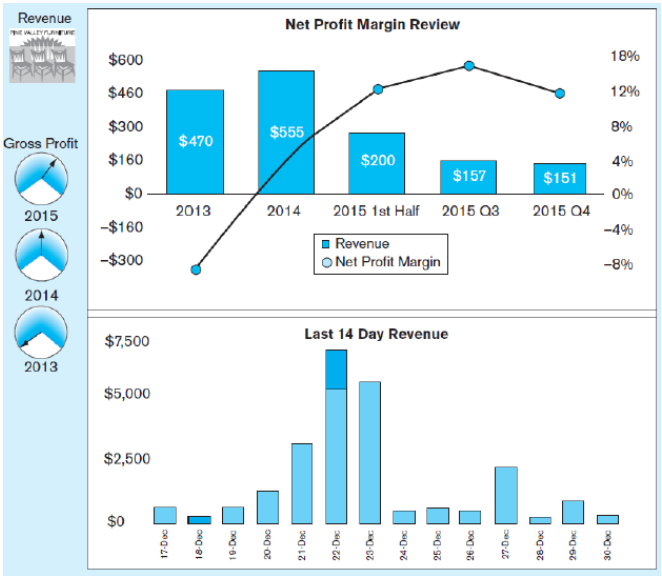
- SQL is generally not an analytic language, but it can be used for analysis
- However, OLAP extensions to SQL make this easier.
- OLAP queries should support:
    - Categorisation – e.g. group data by dimension characteristics
    - Aggregation – e.g. creates averages per category
    - Ranking – e.g. find customer in some category with highest average monthly sales

Online Analytical Processing (OLAP) Tools

- OLAP – the use of a set of graphical tolls that provides users with multidimensional views of their data and allows them to analyse the data using simple windowing techniques
- Relational OLAP (ROLAP) – OLAP tools that view the database as a traditional relational database in either a star schema or other normalised/de-normalised sets of tables
- Multidimensional OLAP (MOLAP) – OLAP tools that load data into an intermediate structure, usually three- or higher-dimensional array

Data Visualisation

- Representation of data in graphical and multimedia formats
- Can depict relationships in the data
- Often used in dashboards
- Business Performance Management (BPM):
    - BPM systems allow managers to measure, monitor and manage key activities and processes to achieve organisational goals. Dashboards are often used.



    - Charts like these are example of data visualisation

Big Data and Analytics Impact

- Applications:
    - Business
    - E-government and politics
    - Science and technology
    - Smart health and well-being
    - Security and public safety
- Social Implications
    - Personal Privacy vs Collective benefit
    - Ownership and access
    - Data/algorithm quality and reuse
    - Transparency and validation
    - Demands for workforce capabilities and education

Green IT and IS

The Need for Green IS and IT

- The IT industry is an active player in supporting sustainable economic development
    - IT – Information Technology transmits, process or stores information
    - IS – An information system is an integrated and cooperating set of software using IT to support individuals/groups.

Green IT

- Green IT is mainly focused on energy efficiency and equipment utilisation. It addresses issues such as:
    - Designing energy efficient chips and disk drives
    - Relacing personal computers with energy efficient thin clients
    - Use of virtualisation software to run multiple operation systems on one server
    - Reducing the energy consumption of data centres
    - Using renewable energy sources to power data centres
    - Reducing electronic waste from obsolete computing equipment
    - Promoting telecommuting and remote computer administration to reduce transportation emissions.

Green IS

- Green IS, in contrast, refers to the design and implementation of information systems that contribute to sustainable business processes.
    - Reduce transportation costs
    - Support global teamwork and meetings
    - Track environmental information
    - Monitor a firm's operational emissions and waste products
    - Provides information to consumers to help them make green choices

Organisational Perspective

- Organisations strive to sustain their existence, and the notion of corporate sustainability incorporates ecological thinking and three different approaches to it:
    - *Eco-efficiency* – The delivery of competitively-priced goods and services that satisfy human needs and bring quality of life, while progressively reducing ecological impacts
    - *Eco-equity* – aims for the fair distribution of natural resources to future generations
    - *Eco-effectiveness* – aim to end current practices that result in ecological degradation