Table of Contents

C1: Information Models

Information Systems	To **transform data** into **information** for **generating knowledge** that can be used for **decision making**

Data, Information & Knowledge

Data	**Raw, non-summarized and unanalyzed** facts and figures
Information	Data that have been converted into a **meaningful and useful context** for the receiver
Knowledge	Gained from information for **making decisions**

* Categorization of Information

Source

Primary information	Original source document
Secondary information	Processed primary sources, second-hand versions
Internal information	Come from variety of source **within the company** such as different departments
External information	Gathered **outside the company**, either by interviewing customers or examining published data

Nature

Qualitative data	• Think: looks • A type of data that describes **properties or characteristics** used to identify things
Quantitative data	• Think: numbers • A type of data where the **values have been measured or counted**

Level

Strategic information	• For **long-term decisions** • E.g. identify new markets and products; plan growth; reallocate

	resources across divisions
Tactical战略 information	• Used for **tactical planning and decision-making** within the guidelines set by the strategic plan • E.g. choose suppliers; revise staffing levels; forecast sales, inventory and cash; prepare budgets
Operational information	• For **operational planning** based on the tactical plans • E.g. correct order delays; schedule employees; find production bottlenecks; monitoring resource usage

Time

Historic information	• **Gathered and stored over period of time** • Allows decision makers to draw **comparisons between previous and present activities** • Can be used to **identify trends** over period of time
Present	• Information **created from activities during the current work window**
Future	• Information **created using present and historic information** to try to **predict the future activities, trends and events** relating to the operation of an organization

Information Capturing and Representation

- OCR, Barcode
- RFID
- Document Imaging
- The WWW (unstructured)
    - Twitter
    - Blogs
    - Social Networks
    - Web pages (Requires Big Data Analytics)

Big Data Analytics

Big data	Data that exist in very **large volumes** and many **different varieties** (data types) and that need to be processed at a very **high velocity** (speed)
Analytics	• Systematic analysis and interpretation诠释 of data • Use mathematical, statistical and computational tools to improve our understanding of real-world domain

Metadata / Scheme Association with Data

Metadata	Information about another set of data

* Access through Queries

查询类型	你做的事	数据库做的事	适用场景
过程式 (Procedural)	自己规划步骤，按顺序执行查找	按你的指令执行，不做优化	NoSQL、游标查询
声明式 (Declarative)	直接告诉系统你要什么，系统决定如何找	数据库自动优化查询，提高效率	关系型数据库（SQL）
导航式 (Navigational)	按结构逐步查找，靠路径一步步找到数据	需要按照关系网遍历，逐步找到目标	图数据库（Neo4j）

Procedural queries	• Cumbersome繁琐 and prone to error容易出错 • Programmer must provide the right sequence of instructions • Requires some technical knowledge
Declarative queries	• Only need to state WHAT you need, not HOW to get it • E.g. SQL

Navigational queries	• Searcher knows where he wants to go to find something
	• 2 types:
	  ○ A word, name or brand strongly or uniquely associated with one particular web site (e.g. HP, ebay, hotmail, yahoo)
	  ○ A partial or complete web address (e.g. ebay.com, www.hotmail, yahoo.com)

*Information Security

Confidentiality	**Preserving authorized restrictions保留授权限制 on information access and disclosure泄露**
Integrity	**Guarding against improper information modification or destruction破坏**, including ensuring information **authenticity** and **non-repudiation**
Availability	Ensuring **timely and reliable access** to and **use of information**

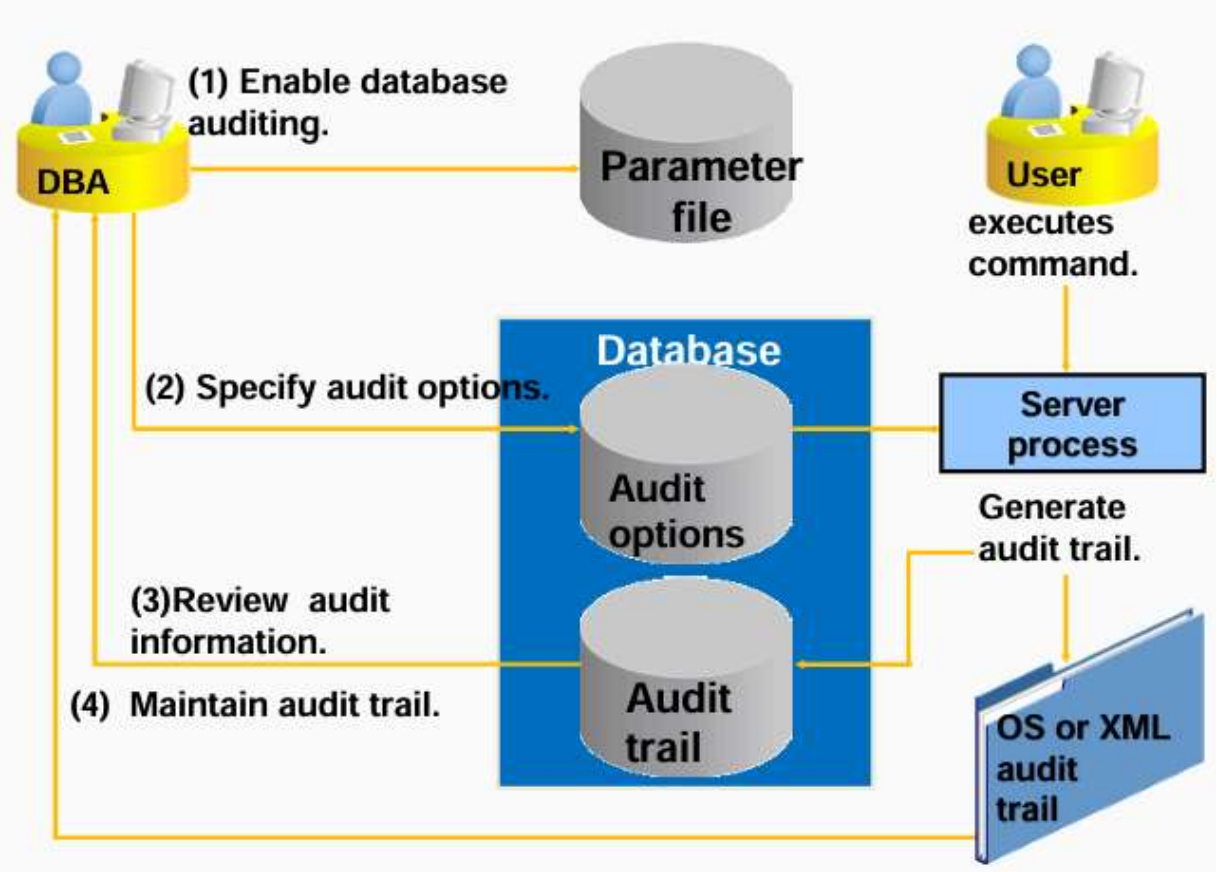
Threats to Information Security

- **Accidental losses** attributable to:
  - Human error
  - Software failure
  - Hardware failure
- **Theft and fraud**
- **Loss of privacy or confidentiality**
  - Loss of privacy (personal data)
  - Loss of confidentiality (corporate data)
- **Loss of data integrity**
- **Loss of availability**
  - Disruption of access to information or use of information

Measures of Information Security

- **Backup and recovery**
- **Physical security**
  - Computerized and manual policies and procedures, fence栅栏, lock and key, security guards
- **Technical security**
  - Encryption, authentication, authorization, audit trace

Database Audit Trace



Information Assurance保障

- Measures that **protect and defend information and information systems** by ensuring their **availability, integrity, authentication, confidentiality and non-repudiation**
- Non-repudiation: Assurance that someone cannot deny something

C2: Database Systems

* Disadvantages of File Processing (Traditional Approach)

Program-Data Dependence	All programs maintain metadata for each file they use
Data Redundancy (Duplication of Data)	Different systems / programs have separate copies of the same data
Limited Data Sharing	No centralized control of data
Lengthy Development Times	Programmers must design their own file formats
Excessive Program Maintenance	80% of the information systems budget

Problems with Data Dependency

- Each application programmer must maintain their own data
- Each application program needs:
    - To **include code for metadata of each file**
    - To **have its own processing routines** for reading, inserting, updating and deleting data
- **Lack of coordination and central control**
- **Non-standard file formats**
- **Waste of space** to have duplicate data
- More **maintenance headaches**
- The biggest problem:
    - **Inconsistencies** when data changes in one file
    - **Compromises data integrity**

Database Management System (DBMS)

- A software system that enables users to define, create, maintain and control access to the database

Functions of DBMS

- **Store, update and retrieve data**
- **Concurrency control services**
    - **Ensure updates and end result are made correctly** when **multiple users make updates to database simultaneously**
    - Solution:
        - Locking & 2-phase locking
        - Time-stamping

        - Versioning
- **Backup and Recovery Services**
    - Crash, fire, natural disaster
    - So data not permanently lost
- Security
    - **Prevent unauthorized access to database**
    - Encryption, authentication, authorization and views
- **Integrity Services**
    - Avoid **incorrect** or **inconsistent** data by complying with a **set of rules** to ensure data integrity is enforced
- **Transaction Support**
    - **Ensure that all the updates are made or that none of them is made**
- A User-Accessible **System Catalogue (Data Dictionary Management)**
    - Repository of information (metadata) describing the data in the database
    - One of the **fundamental components of DBMS**
    - Typically stores:
        - Names, types and sizes of data items
        - Constraints on the data
        - Names of authorized users
        - Data items accessible by a user and the type of access
        - Usage statistics

* Types of Integrity Constraint in Oracle

NOT NULL	Prohibits a database value from being null
UNIQUE	Prohibits multiple rows from having the same value in the same column
PRIMARY KEY	Combines NOT NULL constraint and UNIQUE constraint in a single declaration
FOREIGN KEY	Requires values in one table to match values in another table
CHECK	Requires a value in the database to comply with a specified condition. e.g. • CHECK (supplier_id BETWEEN 100 and 9999) • CHECK (supplier_name IN ('IBM', 'Microsoft', 'NVIDIA')) • CHECK (status NOT IN ('A', 'B', 'C', 'D'))

Components of DBMS

Query processor	Transforms queries into low-level instructions directed to the database manager
Database manager	-

DML preprocessor	Converts DML statements embedded in an application program into standard function calls in the host language
DDL compiler	Converts DDL statements into a set of tables containing metadata. These tables are then stored in the system catalog
Catalog manager	Manages access to and maintains the system catalog
Authorization control	-
Command processor	-
Query optimizer	Determines an optimal strategy for the query execution
Transaction manager	-
Scheduler	Ensures concurrent operations proceed without conflicting with one another
Recovery manager	-
Buffer manager	Responsible for the transfer of data between main memory and secondary storage

*ANSI-SPARC Architecture

- American National Standards Institute, Standards Planning And Requirements Committee)
- An abstract **design standard for Database Management System (DBMS)**

*Three-Level Architecture

External Level	• Users' view of database • Describes that **part of database** that is relevant to a particular user
Conceptual Level	• Community view of database • Describes **what data is stored in database** and relationships among the data
Internal Level	• Physical representation of database on the computer • Describes **how the data is stored in the database**

Objectives of Three-Level Architecture

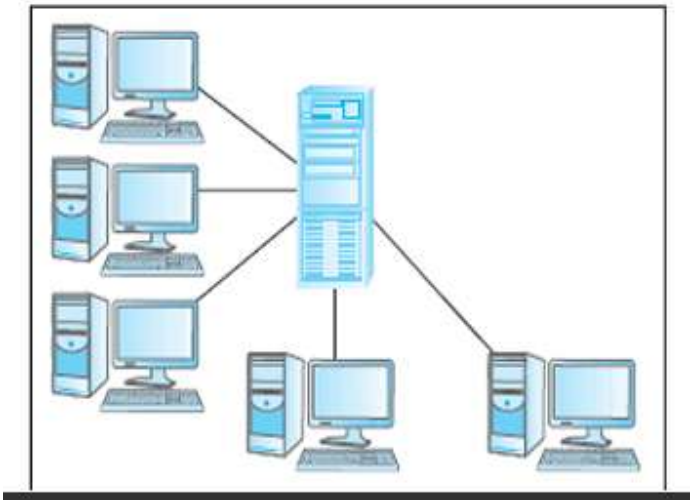
- All users are able to **access the same data**
- A **user's view is immune to changes made in other views**
- Users should not need to know physical database storage details

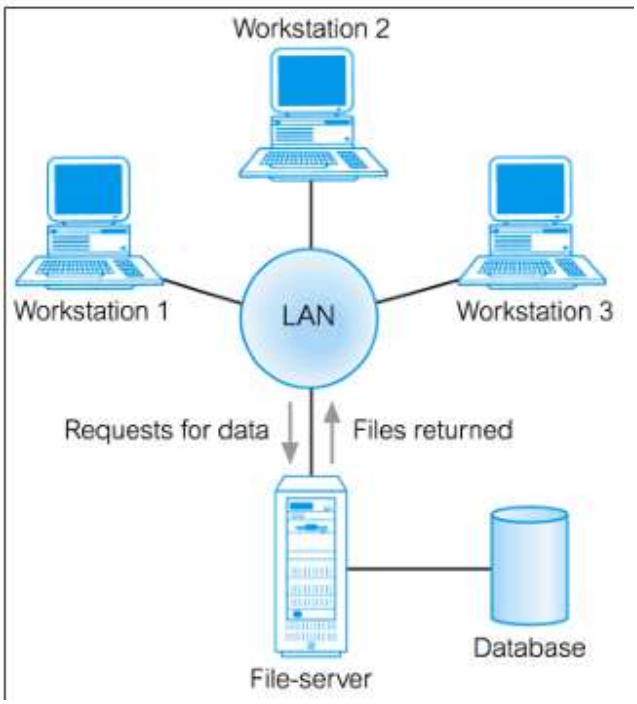
- **DBA should be able to change database storage structures without affecting the users' views**
- The **changes to physical aspects of storage should not affect internal structure of database**
- **DBA should be able to change conceptual structure of database without affecting all users**

* Data Independence

Logical Data Independence	• Immunity of external schemas to changes in conceptual schema • Conceptual schema changes (e.g. addition/removal of entities/attributes) should not require changes to external schema or rewrites of application programs
Physical Data Independence	• Immunity of conceptual schema to changes in the internal schema • Internal scheme changes (e.g. use different file organizations, storage structures/devices) should not require change to conceptual or external schemas

Multi-User DBMS Architectures

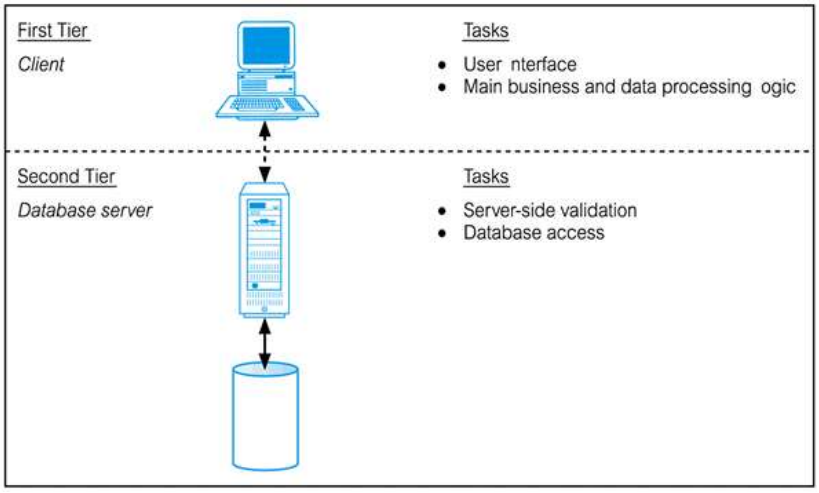
Teleprocessing	• One computer with a single CPU and a number of terminals • **Processing performed within the same physical computer** • User terminals are incapable of functioning on their own, and cabled to the central computer 
File-Server Architecture	• File-server is connected to several workstations across a network

	• Database resides on file-server
	• **DBMS and applications run on each workstation**
	• Disadvantages:
	  ○ Significant network traffic
	  ○ Copy of DBMS on each workstation
	  ○ Concurrency, recovery and integrity control more complex (multiple DBMSs accessing the same files)
	 

Traditional Two-Tier Client-Server	• **Client (tier 1)**
	  ○ Manages user interface and runs applications
	  ○ Accepts and checks syntax of user input
	  ○ Processes application logic
	  ○ Generates database requests and transmits to server
	  ○ Passes response back to user
	• **Server (tier 2)**
	  ○ Holds database and DBMS
	  ○ Accepts and processes database requests from clients
	  ○ Checks authorization
	  ○ Ensures integrity constraints not violated
	  ○ Performs query / update processing and transmits response to client
	  ○ Maintains system catalog, provides concurrent database access and recovery control
	• Disadvantages:

<table>
<tr><td></td><td>

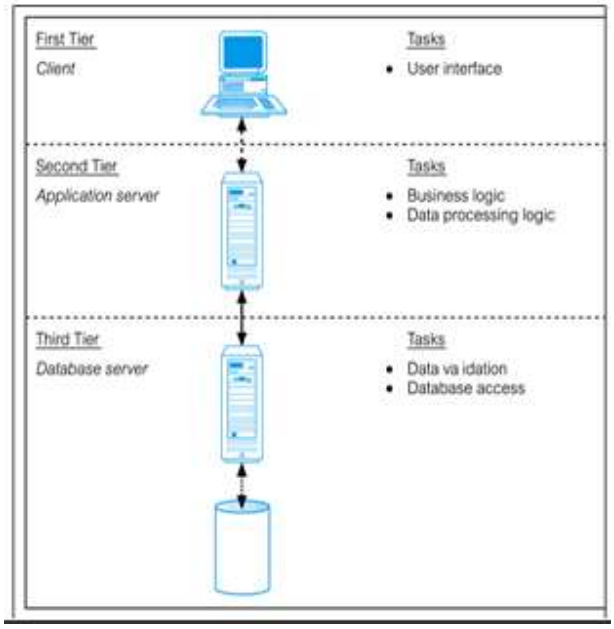
'Fat' client, requiring considerable resources on client's computer to run effectively
Significant client side administration overhead



</td></tr>
<tr><td>Three-Tier Client-Server</td><td>

Advantages

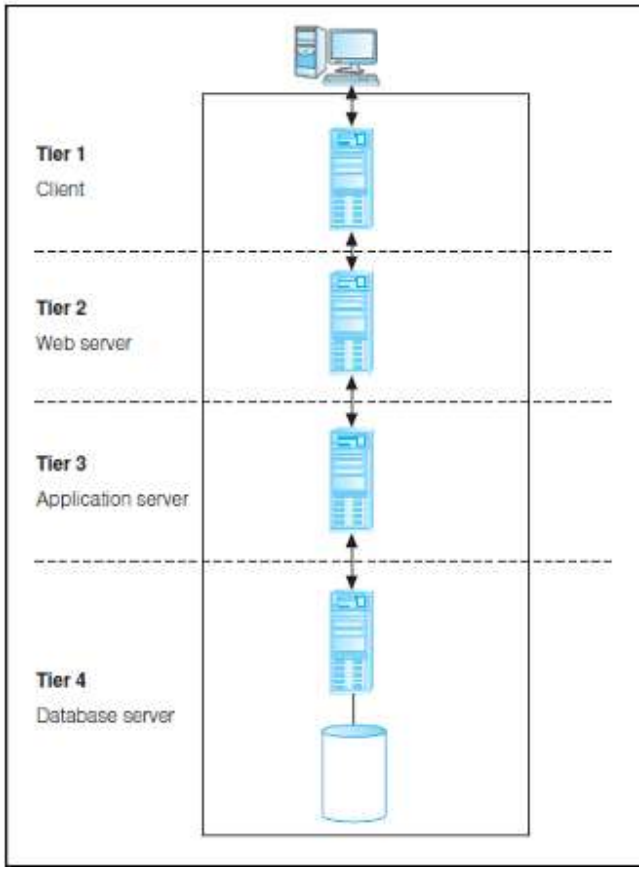
'Thin' client, requiring less expensive hardware
Application maintenance centralized
Easier to modify or replace on tier without affecting others
Separating business logic from database functions makes it easier to implement load balancing
Maps quite naturally to Web environment



</td></tr>
<tr><td>N-Tier Client-Server</td><td>

Can be expanded to n tiers, with additional tiers

</td></tr>
</table>

Architectures	**providing more flexibility and scalability** • Applications servers host API to expose business logic and business processes for use by other applications 

* Transaction Processing Monitors

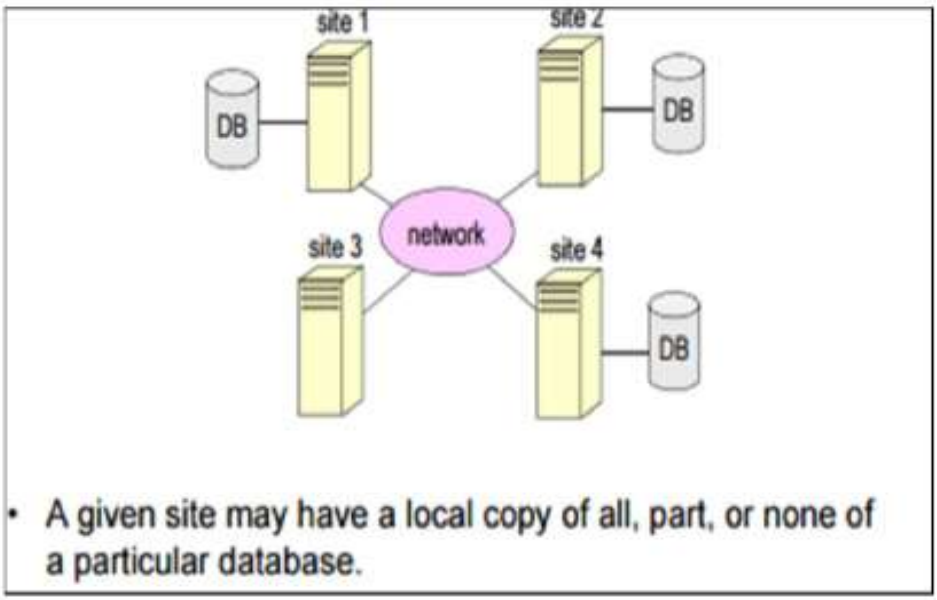
- Forms the middle tier of a three-tier architecture
- TP monitor is a program that **controls data transfer between clients and servers** for providing a **consistent environment**, particularly for Online Transaction Processing (OLTP)

How TP Monitor provides for consistent environment?

Transaction routing	Increase scalability by directing transactions to specific DBMS
Managing distributed transactions	Manage transactions that require access data held in multiple DBMS
Load balancing	Balance client request across multiple DBMS on one or more computer
Increased reliability	• Acts as transaction manager

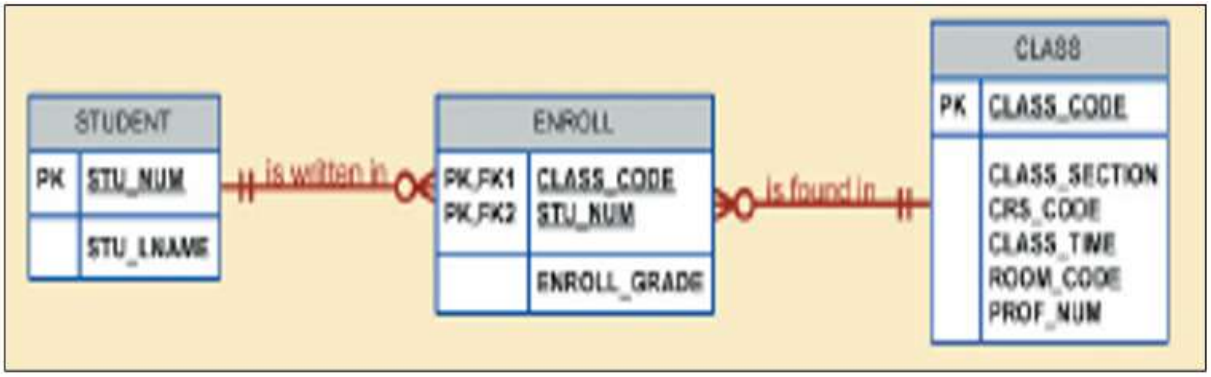
	• Perform necessary actions to maintain the consistency of database • If database fails, TP Monitor will resubmit the transaction to another DBMS or hold the transaction until the DBMS become available again
Funneling	• Establish connection with DBMS when required • Funnel user requests through these connections

Distributed DBMSs

- A logically interrelated collection of shared data, physically distributed over a computer network
- Distributed DBMS permits the management of the distributed database and makes the distribution transparent to users.
- 

C3: Data Modeling

Data Modeling

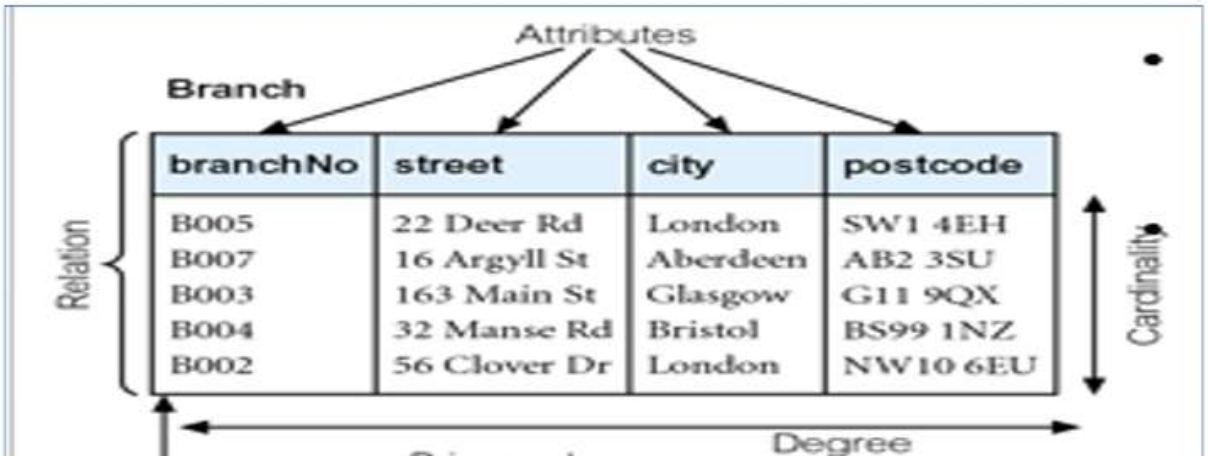
- **First step in designing a database**
- **Process of creating a specific data model** for an information system
- **Iterative and progressive** process
- 

Importance of Data Models

- **Facilitate interaction among designer, applications programmer and end user**
- Well-developed data model can **foster improved understanding** of the organization for which the database design is developed
    - **Provide overall view** of database
    - **Organize data** for various users
    - Abstraction for **creation of good database**

Data Model Concepts

- Data model is a collection of concepts for describing:
    - A **structural part**
    - A **manipulative part** (types of operation that are allowed on the data)
    - Possibly **a set of integrity rules** which ensures that the data is accurate



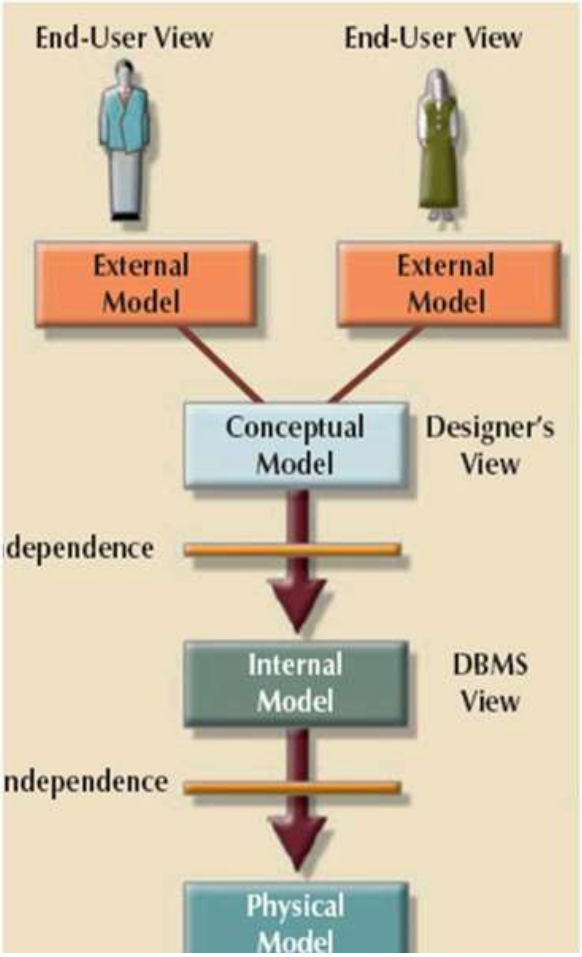
- ANSI-SPARC (American National Standards Institute, Standards Planning And Requirements Committee) defined a framework for data modeling based on degrees of data abstraction

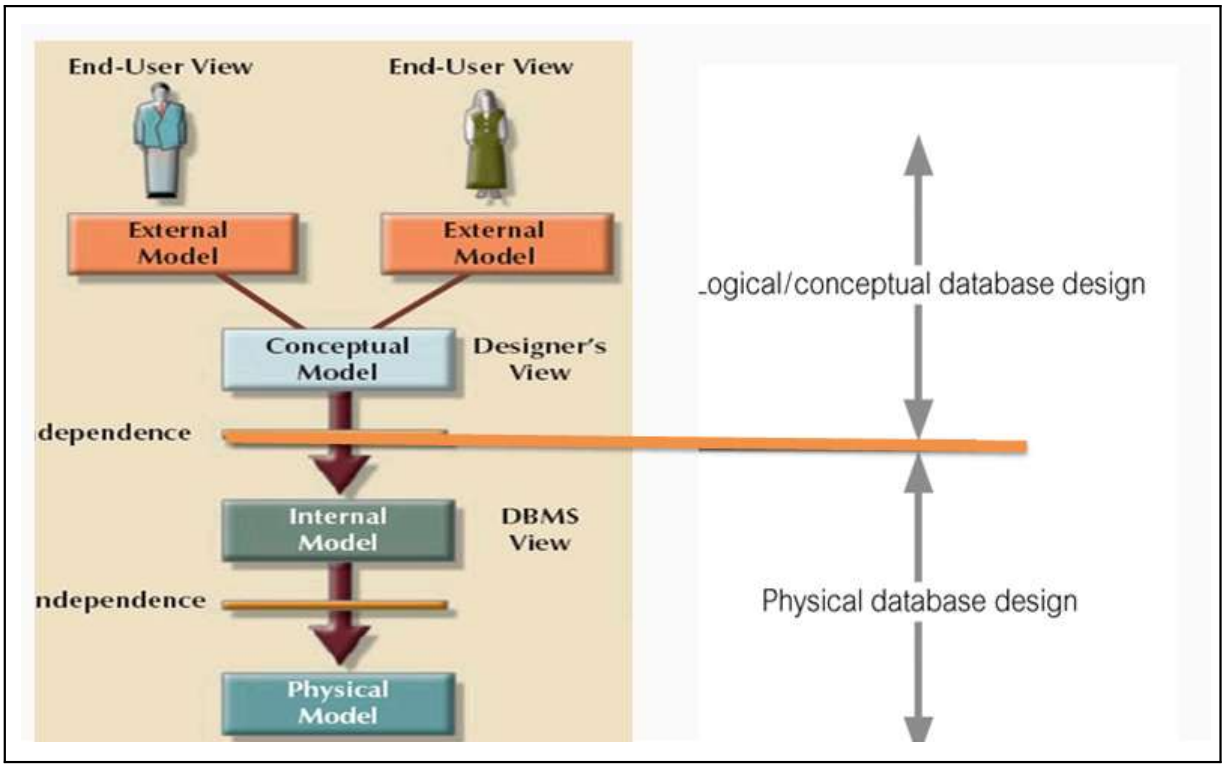
* Mentioned in previous chapter

External model
- End-user view of data environment
- Can be many different models

Conceptual model
- Describes what data to be stored used in an enterprise, independent of all physical considerations
- Describes the relationships among data

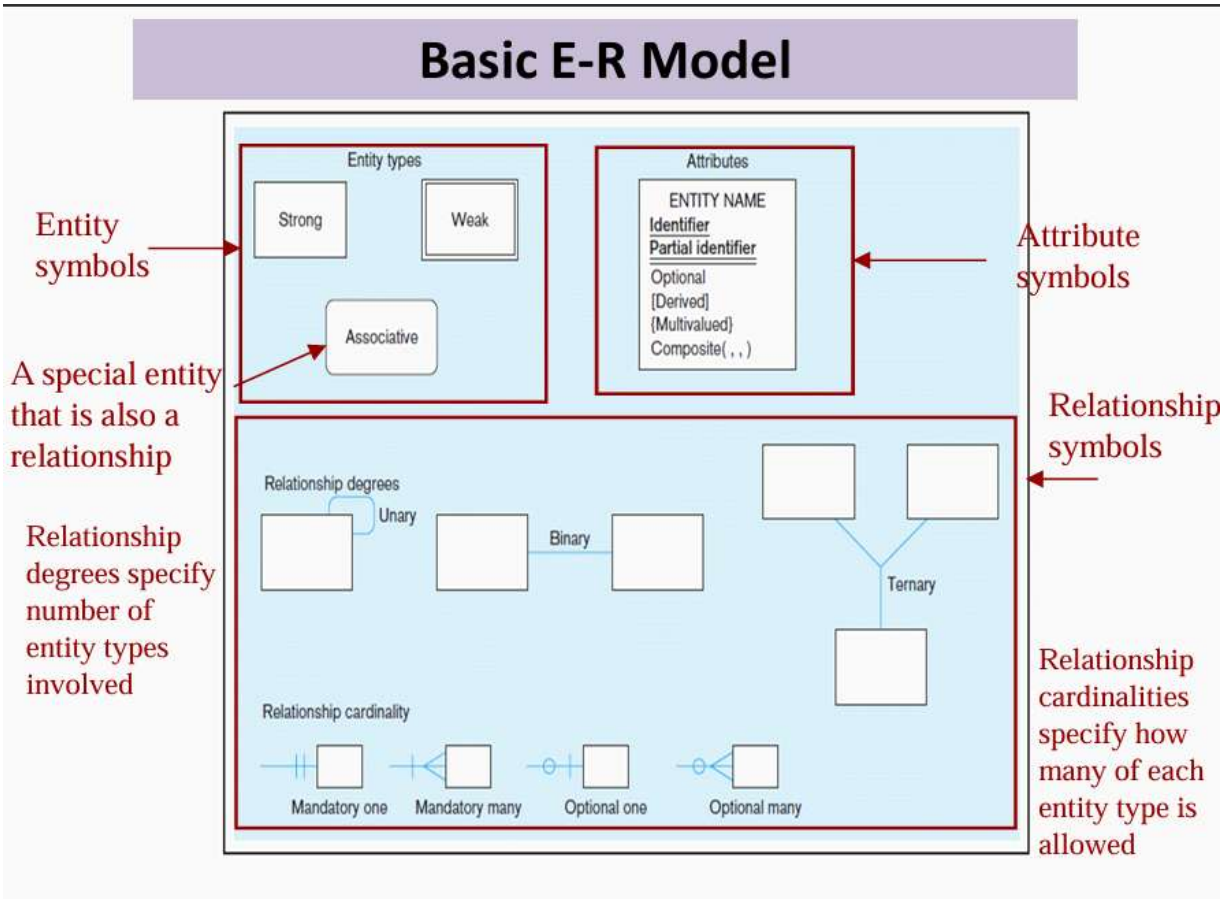
Internal model
- How the data are stored
- Complex low-level structures described in detail



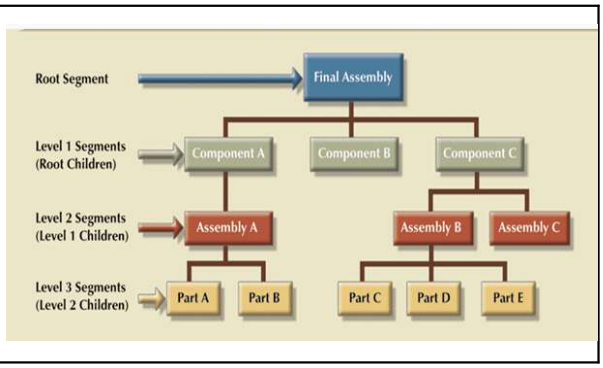


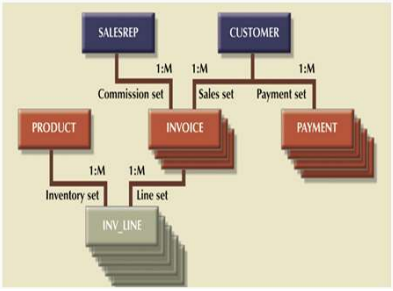
The Entity-Relationship Model

Entity	Distinguishable object that exists
Attributes	• Properties or characteristics of an entity • Each entity contains a set of attributes
Relationship	Association among several entities

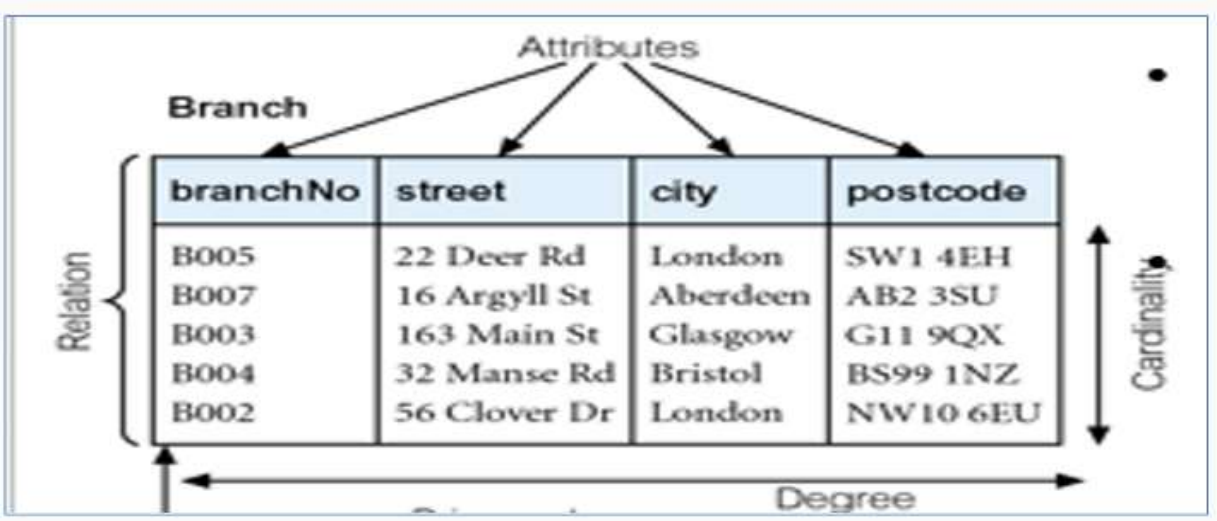


Early Data Models

Hierarchical model • Represented by an upside-down "tree" • Contains levels or segments • Depicts a set of one-to-many (1:M) relationships between parent and its children segments	

Network model • Represent complex data relationships more effectively than hierarchical model • Improve database performance and to impose database standard	 **supports many-to-many relationship**
Limitations of hierarchical and network data models • Data are stored in **rigid**, predetermined relationships • **No DDL** existed, changing the structure of the data was difficult • **Lacked a simple query language**, which hindered application development	

Relational Model

- Data and relationships are represented by a **collection of tables**
- Each table has various **columns with unique names**, called as **attributes**
- **Degree** is the number of attributes in a relation
- **Cardinality** is the number of tuples in a relation
- 

Relational Model - Integrity Constraints

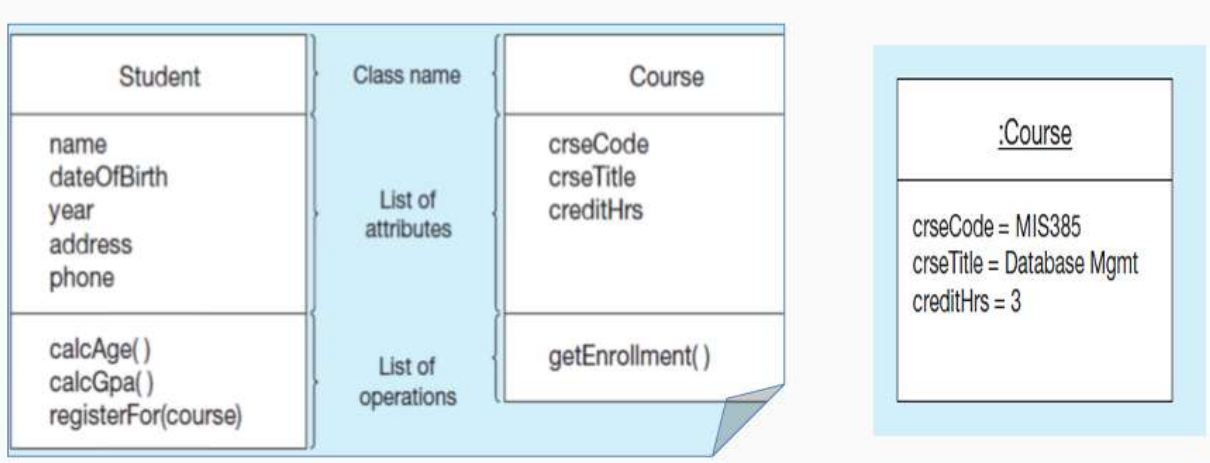
Entity integrity	• Not allow multiple rows to have the same identity within a table • e.g. Primary key
Domain integrity	• Restricting data • Predefined data types, e.g. dates • Set allowable values, e.g. CREATE

	DOMAIN PriceChange AS DECIMAL CHECK (VALUE BETWEEN .001 and .15)
Referential integrity	• Require the existence of a related row in another table • E.g. A customer for a given customer ID, Foreign key

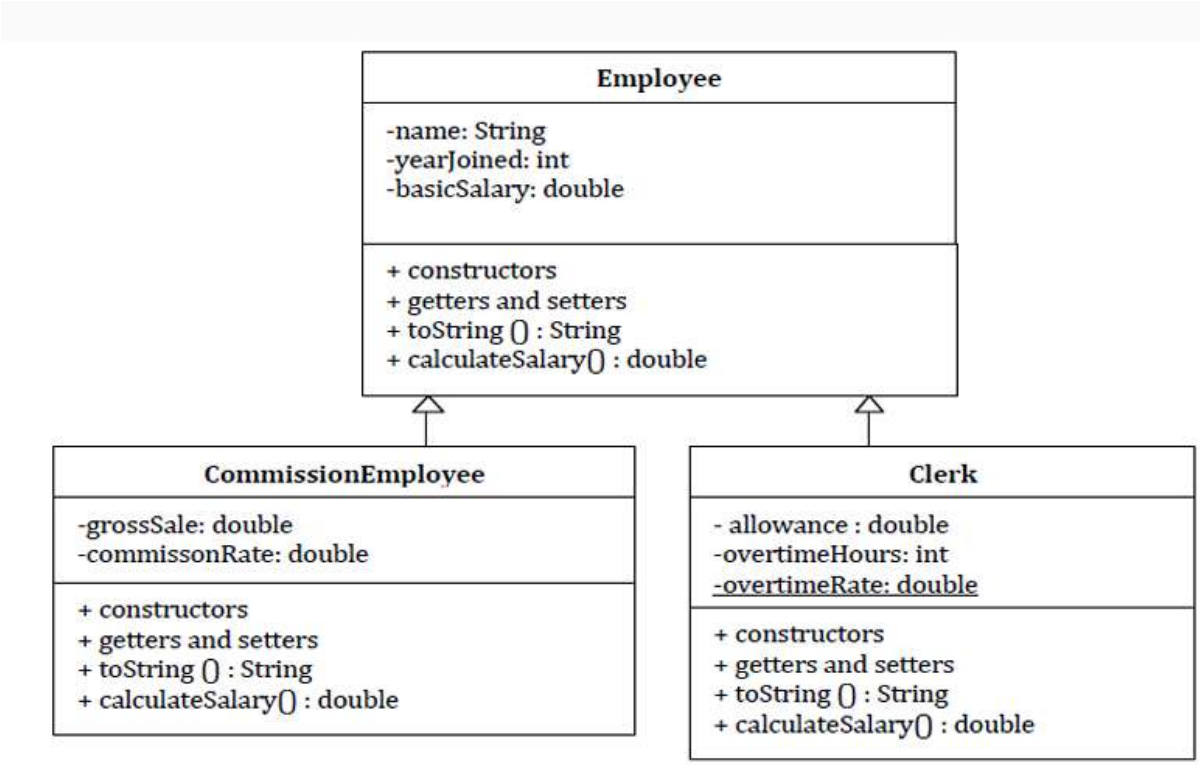
Object-Oriented Model

Object	Abstraction of real-world entity
Attributes/state	Describe the properties of an object
Behavior/operation	Specify how an object acts and reacts

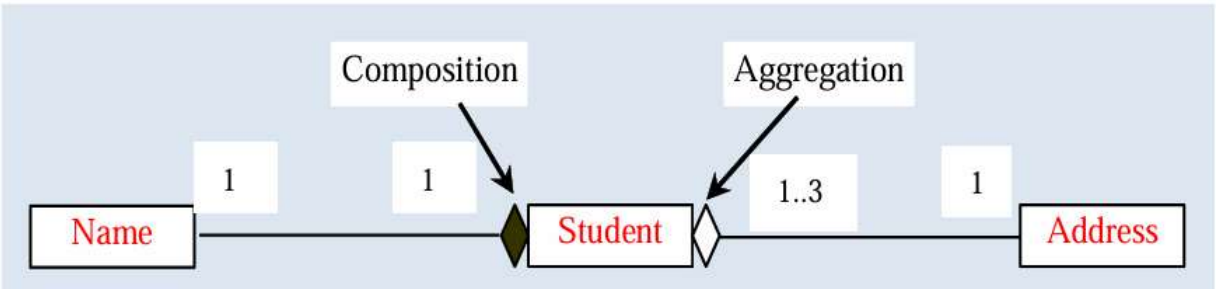
- Objects that share similar characteristics are grouped in classes



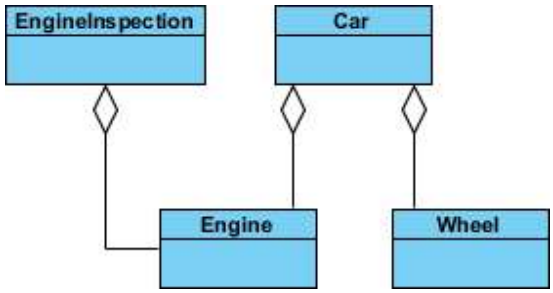
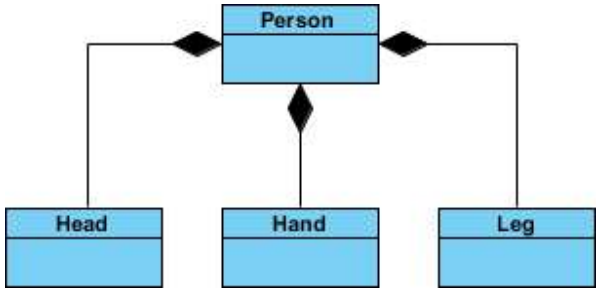
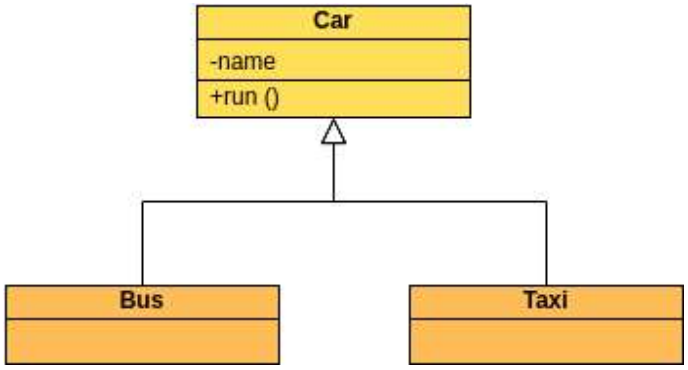
- Classes are organized in class hierarchy
- Inheritance: object inherits methods and attributes of parent class



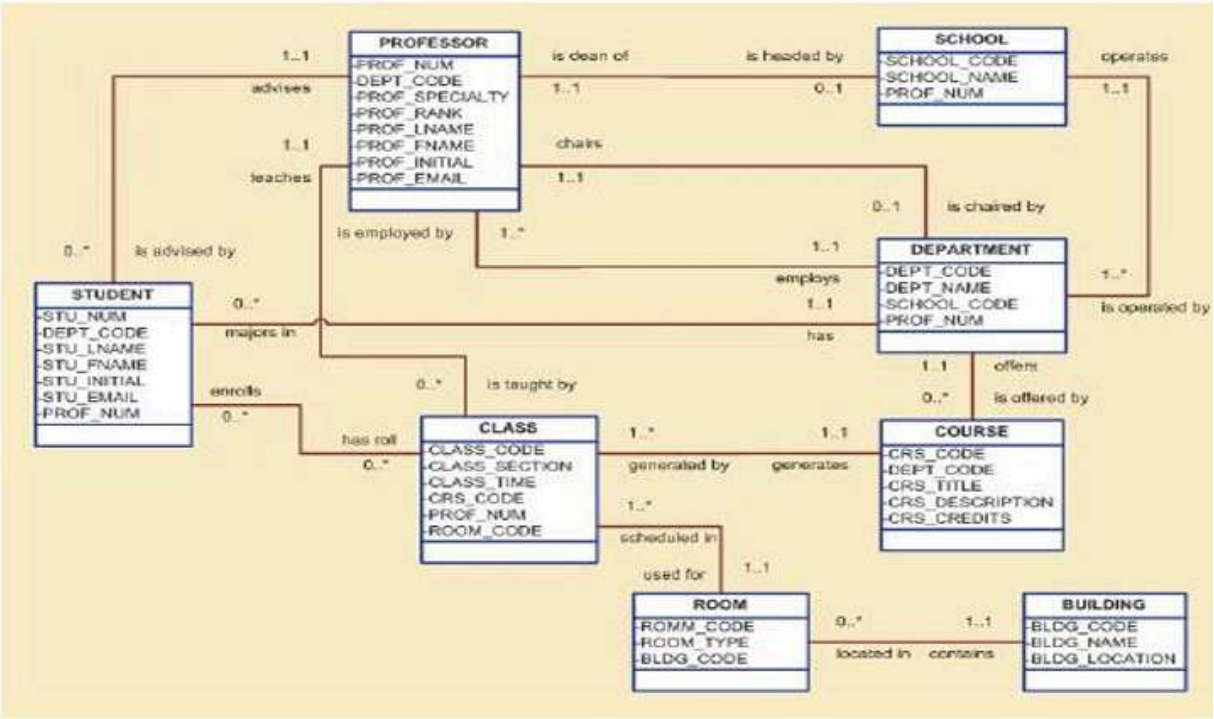
Aggregation and Composition



Association	• Describes the relationship between two independent classes
	• Viewed as describing a "uses-a" relationship where an object uses, or in any way interacts with, another object,
	• It contains many forms, such as:
	  ○ One-to-one
	 
	
	  ○ Many-to-one
	Teacher 1----------* Student

	○ Many-to-many Student *----------* Course
Aggregation	• Implies relationship where the child can exist independently of the parent • Example 1: Class (parent) and Student (child). Delete the Class and the Students still exist. • Example 2: ○ 
Composition	• Implies a relationship where the child cannot exist independent of the parent. • Example 1: House (parent) and Room (child). Rooms don't exist separate from a House. • Example 2: ○ 
Generalization / Inheritance	• Describes the relationship between parent and child classes. • Subclass inherits all the functions of the parent class, and the parent class has all the attributes, methods, and subclasses. • E.g. ○ 

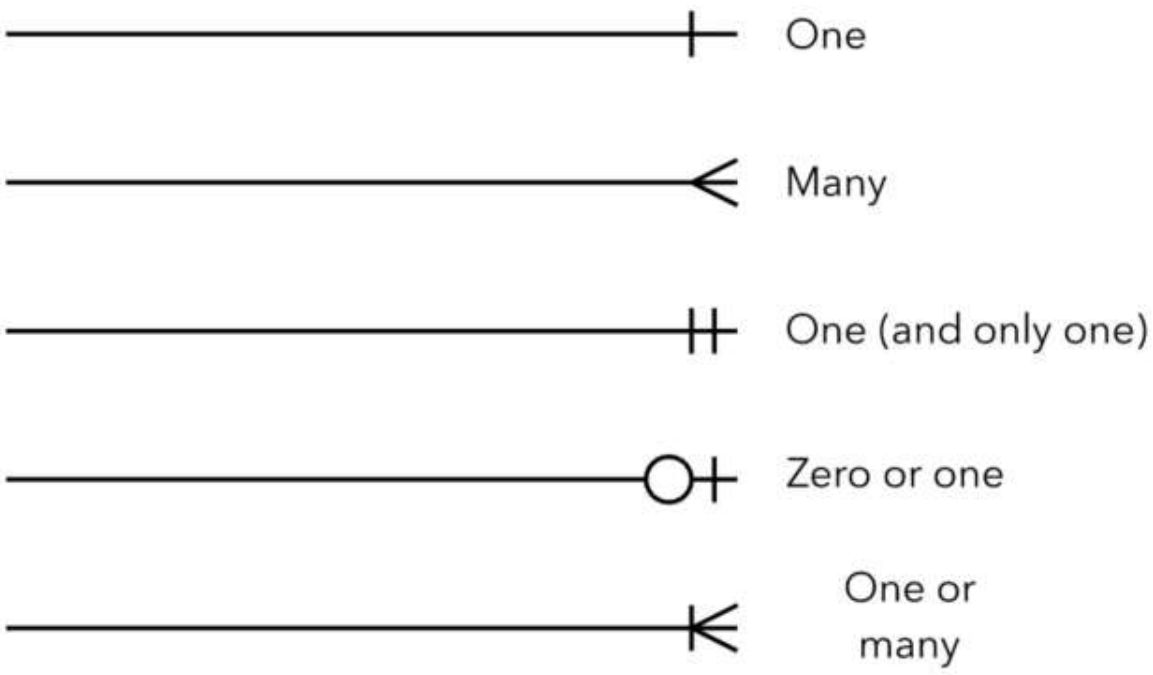
UML



Why OO?

- Conventional data models are inadequate
    - Cannot model complex data and unstructured data
    - Cannot model processes (dynamic behavior)
    - Does not support reuse
- OO model is a more 'natural' way to represent the real world

ERD Relationship



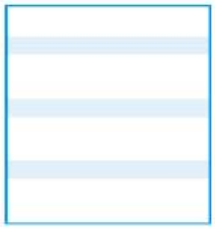
Semi-Structured Data Model

- Has a certain structure
- Structure may **not be rigid, regular or complete**
- E.g. scientific data, E-Catalogs, IMDb
- JSON, XML and other markup languages, email are all forms of semi-structured data
- Advantages:
    - Can **represent information of data sources that cannot be constrained by schema**
    - Provides flexible format for **data exchange between different types of databases**
    - Schema can **easily be changed**
- Comparison with relational data:
    - **Inefficient**: tags which represent schema information are repeated
    - Better than relational tuples as data-exchange format
        - XML/JSON data is **self-documenting** due to use of tags
        - **Non-rigid format**: tags can be added / changed into schema easily
        - Allows **nested structure**
    - **Wide acceptance** (database systems, browsers, tools, applications)

C4: Relational Model

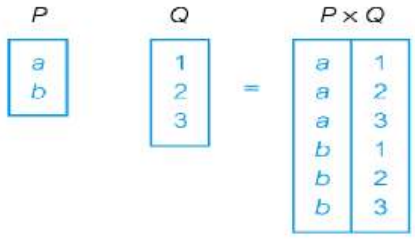
Relational Algebra Operations

* All these operations will be applied in the later contents

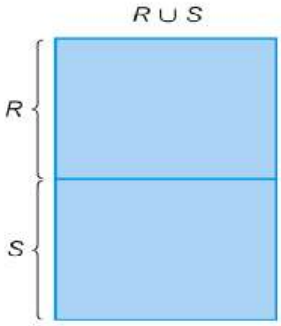


(a) Selection

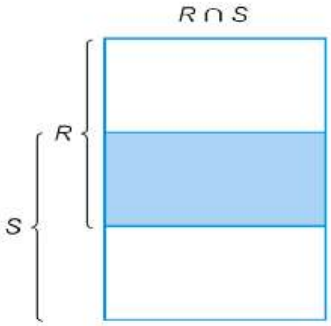
(b) Projection



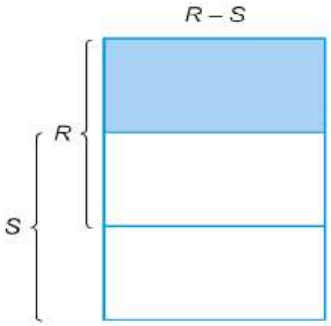
(c) Cartesian product



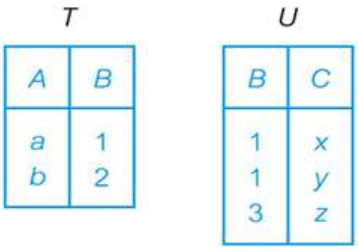
(d) Union

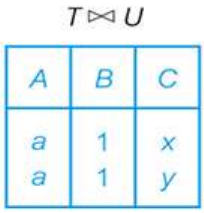


(e) Intersection

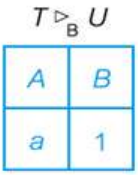


(f) Set difference

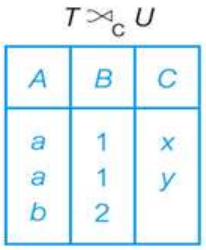




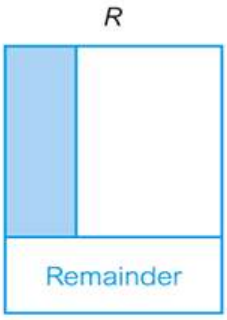
(g) Natural join



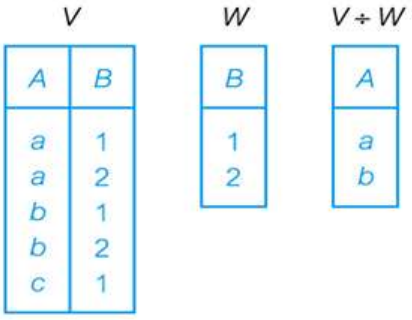
(h) Semijoin



(i) Left Outer join

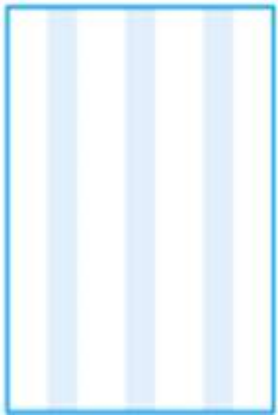
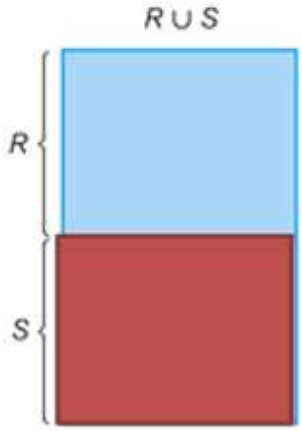


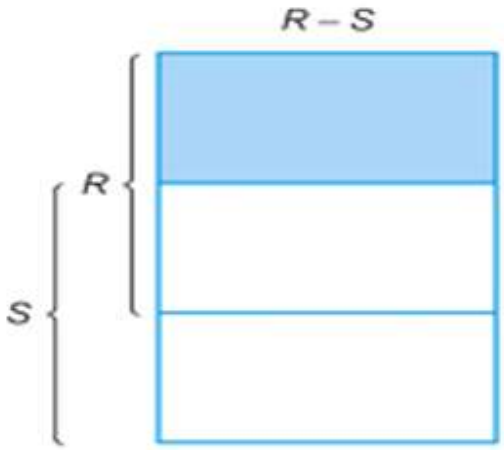
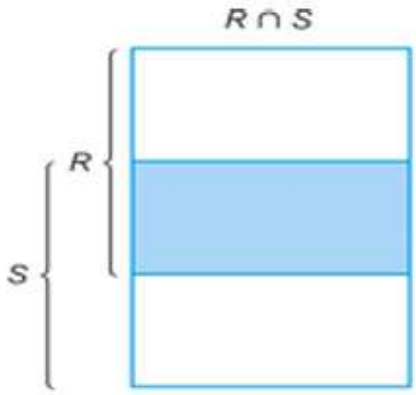
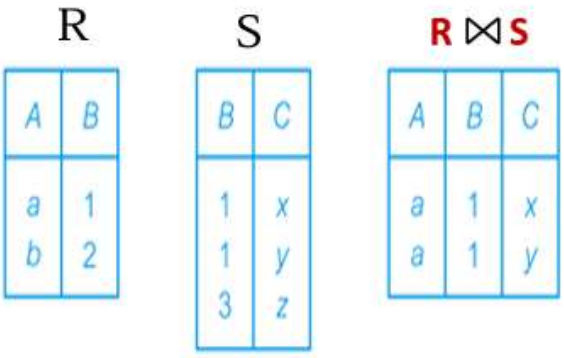
(j) Divis on (shaded area)

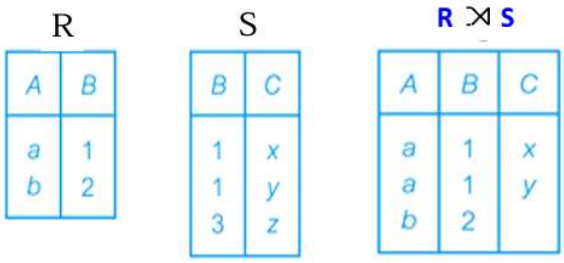


Example of division

Relational Algebra Symbols

Selection (σ)  How to write: σ predicate (R)	• R = table / relation • Predicate = condition that the rows of R should satisfy • Result: Display all attributes (rows) which has satisfied the condition • E.g. ○ σ salary > 10000 (Staff) → List all staff with a salary greater than 10,000
Projection (π)  How to write: π col1, ..., coln (R)	• R = table / relation • Coln = Column that are selected to display • Looks for the vertical column of R to obtain the specified columns' value • Result: Project yields values for selected attributes • E.g. ○ π staffNo, fName, lName, salary (Staff) → Display a list of salaries for all staff, showing only staffNo, fName, lName and salary details
Union  How to write: R U S	• Contains all tuples of R or S or both R and S, duplicate tuples being eliminated • R and S must be union compatible (have same attributes) • E.g. ○ π city (Branch) U π city (PropertyForRent) → List all cities where there is either a branch office or a property for rent

Set Difference  How to write: R - S	• Defines a relation consisting of tuples that are in relation R, but not in S • R and S must be union-compatible • E.g. ○ π city (Branch) - π city (PropertyForRent) → List all cities where there is a branch office but no properties for rent
Intersection  How to write: R ∩ S	• Defines a relation consisting of the set of all tuples that are in both R and S • R and S must be union-compatible • E.g. ○ π city (Branch) ∩ π city (PropertyForRent) → List all cities where there is both a branch office and at least one property for rent
Natural Join  How to write: R ⋈ S	• Join table R and S together • Result: Set of all combinations of tuples in R and S that are equal on their common attribute names (If table R has attribute B and table S also has attribute B, they can be joined together on attribute B.) • E.g. ○ (π clientNo, fName, lName (Client)) ⋈ (π clientNo, propertyNo, comment (Viewing)) → List the clientNo, names and comments of all clients who have viewed a property for rent
Left Outer Join (not covered in midterm)	• Join tuples from R that do not have

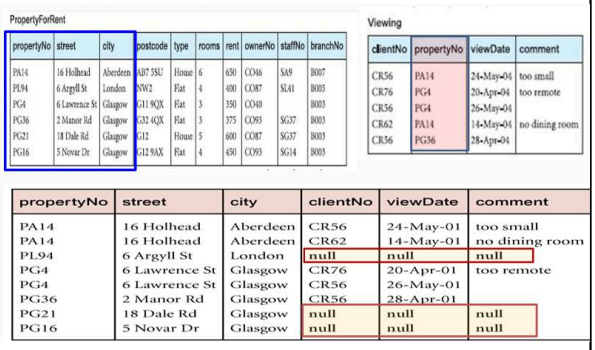
<table>
<tr>
<td>


<table>
<tr><th>A</th><th>B</th></tr>
<tr><td>a</td><td>1</td></tr>
<tr><td>b</td><td>2</td></tr>
</table>
<table>
<tr><th>B</th><th>C</th></tr>
<tr><td>1</td><td>x</td></tr>
<tr><td>1</td><td>y</td></tr>
<tr><td>3</td><td>z</td></tr>
</table>
<table>
<tr><th>A</th><th>B</th><th>C</th></tr>
<tr><td>a</td><td>1</td><td>x</td></tr>
<tr><td>a</td><td>1</td><td>y</td></tr>
<tr><td>b</td><td>2</td><td></td></tr>
</table>
How to write: R \left_outer_join S

</td>
<td>
matching values in common columns of S are also included in the result relation (R joins S, but it will focus more on the R. All the relevant rows of R must be included even though S does not have the relevant rows for it)

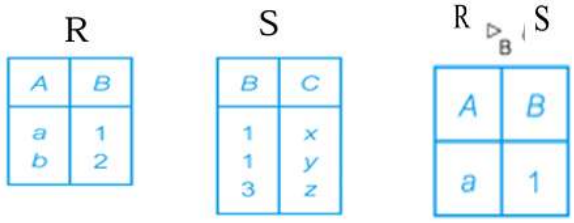
E.g.

π propertyNo, street, city (PropertyForRent) \left_outer_join Viewing → Produce a status report on property viewings


PropertyForRent
<table>
<tr><th>propertyNo</th><th>street</th><th>city</th><th>postcode</th><th>type</th><th>rooms</th><th>rent</th><th>ownerNo</th><th>staffNo</th><th>branchNo</th></tr>
<tr><td>PA14</td><td>16 Holhead</td><td>Aberdeen</td><td>AB7 5SU</td><td>House</td><td>6</td><td>650</td><td>CO46</td><td>SA9</td><td>B007</td></tr>
<tr><td>PL94</td><td>6 Argyll St</td><td>London</td><td>NW2</td><td>Flat</td><td>4</td><td>400</td><td>CO87</td><td>SL41</td><td>B005</td></tr>
<tr><td>PG4</td><td>6 Lawrence St</td><td>Glasgow</td><td>G11 9QX</td><td>Flat</td><td>3</td><td>350</td><td>CO40</td><td></td><td>B003</td></tr>
<tr><td>PG36</td><td>2 Manor Rd</td><td>Glasgow</td><td>G32 4QX</td><td>Flat</td><td>3</td><td>375</td><td>CO93</td><td>SG37</td><td>B003</td></tr>
<tr><td>PG21</td><td>18 Dale Rd</td><td>Glasgow</td><td>G12</td><td>House</td><td>5</td><td>600</td><td>CO87</td><td>SG37</td><td>B003</td></tr>
<tr><td>PG16</td><td>5 Novar Dr</td><td>Glasgow</td><td>G12 9AX</td><td>Flat</td><td>4</td><td>450</td><td>CO93</td><td>SG14</td><td>B003</td></tr>
</table>
Viewing
<table>
<tr><th>clientNo</th><th>propertyNo</th><th>viewDate</th><th>comment</th></tr>
<tr><td>CR56</td><td>PA14</td><td>24-May-04</td><td>too small</td></tr>
<tr><td>CR76</td><td>PG4</td><td>20-Apr-04</td><td>too remote</td></tr>
<tr><td>CR56</td><td>PG4</td><td>26-May-04</td><td></td></tr>
<tr><td>CR62</td><td>PA14</td><td>14-May-04</td><td>no dining room</td></tr>
<tr><td>CR56</td><td>PG36</td><td>28-Apr-04</td><td></td></tr>
</table>
<table>
<tr><th>propertyNo</th><th>street</th><th>city</th><th>clientNo</th><th>viewDate</th><th>comment</th></tr>
<tr><td>PA14</td><td>16 Holhead</td><td>Aberdeen</td><td>CR56</td><td>24-May-01</td><td>too small</td></tr>
<tr><td>PA14</td><td>16 Holhead</td><td>Aberdeen</td><td>CR62</td><td>14-May-01</td><td>no dining room</td></tr>
<tr><td>PL94</td><td>6 Argyll St</td><td>London</td><td>null</td><td>null</td><td>null</td></tr>
<tr><td>PG4</td><td>6 Lawrence St</td><td>Glasgow</td><td>CR76</td><td>20-Apr-01</td><td>too remote</td></tr>
<tr><td>PG4</td><td>6 Lawrence St</td><td>Glasgow</td><td>CR56</td><td>26-May-01</td><td></td></tr>
<tr><td>PG36</td><td>2 Manor Rd</td><td>Glasgow</td><td>CR56</td><td>28-Apr-01</td><td></td></tr>
<tr><td>PG21</td><td>18 Dale Rd</td><td>Glasgow</td><td>null</td><td>null</td><td>null</td></tr>
<tr><td>PG16</td><td>5 Novar Dr</td><td>Glasgow</td><td>null</td><td>null</td><td>null</td></tr>
</table>
</td>
</tr>
<tr>
<td>
Right Outer Join (not covered in midterm)
Employee
<table>
<tr><th>Name</th><th>EmpId</th><th>DeptName</th></tr>
<tr><td>Harry</td><td>3415</td><td>Finance</td></tr>
<tr><td>Sally</td><td>2241</td><td>Sales</td></tr>
<tr><td>George</td><td>3401</td><td>Finance</td></tr>
<tr><td>Harriet</td><td>2202</td><td>Sales</td></tr>
<tr><td>Tim</td><td>1123</td><td>Executive</td></tr>
</table>
Dept
<table>
<tr><th>DeptName</th><th>Manager</th></tr>
<tr><td>Sales</td><td>Harriet</td></tr>
<tr><td>Production</td><td>Charles</td></tr>
</table>
Employee ⟖ Dept
<table>
<tr><th>Name</th><th>EmpId</th><th>DeptName</th><th>Manager</th></tr>
<tr><td>Sally</td><td>2241</td><td>Sales</td><td>Harriet</td></tr>
<tr><td>Harriet</td><td>2202</td><td>Sales</td><td>Harriet</td></tr>
<tr><td>ω</td><td>ω</td><td>Production</td><td>Charles</td></tr>
</table>
How to write: R \right_outer_join S

</td>
<td>

Display set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R. (R joins S, but it will focus more on the S. All the relevant rows of S must be included even though R does not have the relevant rows for it)

</td>
</tr>
<tr>
<td>
Semijoin (not covered in midterm)


<table>
<tr><th>A</th><th>B</th></tr>
<tr><td>a</td><td>1</td></tr>
<tr><td>b</td><td>2</td></tr>
</table>
<table>
<tr><th>B</th><th>C</th></tr>
<tr><td>1</td><td>x</td></tr>
<tr><td>1</td><td>y</td></tr>
<tr><td>3</td><td>z</td></tr>
</table>
<table>
<tr><th>A</th><th>B</th></tr>
<tr><td>a</td><td>1</td></tr>
</table>
How to write: R \semijoin S
</td>
<td>

Defines a relation that contains the tuples of R that participate in the join of R with S.
E.g.

</td>
</tr>
</table>

$R \triangleright_F S$

$Staff \triangleright_{Staff.branchNo=Branch.branchNo}(\sigma_{city='Glasgow'}(Branch))$

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

39

Division (Not covered in midterm)

R

A	B
a	1
a	2
b	1
b	2
c	1

S

B
1
2

R ÷ S

A
a
b

How to write: R ÷ S

- Defines a relation over the attributes B that consist of set of tuples from R that match combination of every tuple in S.
- E.g.
  - (π clientNo, propertyNo (Viewing)) ÷ (π propertyNo (σ rooms = 3 (PropertyForRent))) → Identify all clients who have viewed all properties with three rooms

$\Pi_{clientNo,propertyNo}(Viewing)$

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

propertyNo
PG4
PG36

RESULT

clientNo
CR56

43

Aggregate Operations

$\mathfrak{I}_{AL}(R)$

- Applies aggregate function list, AL, to R to define a relation over the aggregate list
- AL contains one or more (<aggregate_function> <attribute>) pairs
- Common aggregate functions:
  - COUNT
  - SUM
  - AVG
  - MIN
  - MAX
- E.g.
  - \aggregate COUNT staffNo, SUM salary (Staff) → Find number of staff and the sum of their salaries
  - ρR(myCount) \aggregate COUNT propertyNo (σ rent > 350 (PropertyForRent)) → Number of properties cost more than

	350 per month to rent $\rho_R(myCount)\ \Im_{COUNT\ propertyNo}\ (\sigma_{rent > 350}\ (PropertyForRent))$ PropertyForRent propertyNo | street | city | postcode | type | rooms | rent | ownerNo | staffNo | branchNo PA14 | 16 Holhead | Aberdeen | AB7 5SU | House | 6 | 650 | CO46 | SA9 | B007 PL94 | 6 Argyll St | London | NW2 | Flat | 4 | 400 | CO87 | SL41 | B005 PG4 | 6 Lawrence St | Glasgow | G11 9QX | Flat | 3 | 350 | CO40 | | B003 PG36 | 2 Manor Rd | Glasgow | G32 4QX | Flat | 3 | 375 | CO93 | SG37 | B003 PG21 | 18 Dale Rd | Glasgow | G12 | House | 5 | 600 | CO87 | SG37 | B003 PG16 | 5 Novar Dr | Glasgow | G12 9AX | Flat | 4 | 450 | CO93 | SG14 | B003 myCount 5
Grouping Operation $_{GA}\Im_{AL}(R)$	• Groups tuples of R by grouping attributes, GA, and then applies aggregate function list, AL to define a new relation • AL contains one or more (<aggregate_function> <attribute>) pairs • Result: relation contains the grouping attributes, GA, along with results of each of the aggregate functions • E.g. ○ ρR (branchNo, myCount, mySum) branchNo \aggregate COUNT staffNo, SUM salary (Staff) Staff staffNo | fName | lName | position | sex | DOB | salary | branchNo SL21 | John | White | Manager | M | 1-Oct-45 | 30000 | B005 SG37 | Ann | Beech | Assistant | F | 10-Nov-60 | 12000 | B003 SG14 | David | Ford | Supervisor | M | 24-Mar-58 | 18000 | B003 SA9 | Mary | Howe | Assistant | F | 19-Feb-70 | 9000 | B007 SG5 | Susan | Brand | Manager | F | 3-Jun-40 | 24000 | B003 SL41 | Julie | Lee | Assistant | F | 13-Jun-65 | 9000 | B005 branchNo | myCount | mySum B003 | 3 | 54000 B005 | 2 | 39000 B007 | 1 | 9000

Sample Questions

Book (ISBN, title, edition, year)
BookCopy (copyNo, ISBN*, available)
Borrower (borrowerNo, borrowerName, borrowerAddr)
BookLoan (copyNo*, dateOut, dateDue, borrowerNo*)

1. List all book titles.
2. List all borrower details.
3. List all book titles published in the year 2019.
4. List all copies of book titles that are available for Borrowing.
5. List all copies of the book title “Lord of the Rings” that are available for borrowing.
6. List the names of borrowers who currently have the book title “Lord of the Rings” on loan.

7. List the names of borrowers with overdue books.
8. Produce a report of book titles that have been borrowed by “Peter Bloomfield”.
9. Produce a report with the details of borrowers who currently have books overdue

1. π title (Book)

2. Borrower

   Or

   π borrowerNo, borrowerName, borrowerAddr (Borrower)

3. π title (σ year = 2019 (Book))

4. π title (Book) $\bowtie$ (Book.ISBN = BookCopy. ISBN (σ available = 'Y' (BookCopy)))

5. π CopyNo (σ available = 'Y' (BookCopy)) $\bowtie$ (BookCopy. ISBN = Book. ISBN (σ title = "Lord of the Rings" (Book)))

6. π borrowerName (Borrower) $\bowtie$ (Borrower. borrowerNo = BookLoan. borrowerNo (BookLoan)) $\bowtie$ (BookLoan. CopyNo = BookCopy. CopyNo (σ available = 'N') (BookCopy))) $\bowtie$ (BookCopy. ISBN = Book. ISBN (σ title = "Lord of the Rings") (Book))

7. π borrowerName, (Borrower) $\bowtie$ (Borrower. borrowerNo = BookLoan. borrowerNo (σ dateDue $\leq$ "03/06/2025") (BookLoan))) $\bowtie$ (BookLoan. CopyNo = BookCopy. CopyNo (σ available = 'N') (BookCopy))

8. π title (Book) $\bowtie$ (BookISBN = BookCopy. ISBN (BookCopy)) $\bowtie$ (BookCopy. CopyNo = BookLoan. CopyNo (BookLoan)) $\bowtie$ (BookLoan. borrowerNo = Borrower. borrowerNo (σ borrowerName = "Peter Bloomfield" (Borrower)))

9. Borrower $\bowtie$ (Borrower. borrowerNo = BookLoan. borrowerNo (σ dateDue < "03/06/2025") (BookLoan))) $\bowtie$ (BookLoan. CopyNo = BookCopy (σ available = 'N' (BookCopy))

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return those professors who have taught 'csc6710' but never 'csc7710'.

π ssn (σ crscode = 'csc6710' $\wedge$ crscode $\neq$ 'csc7710' (Taught))

OR

π ssn, profname (Professor) $\bowtie$ (Professor. ssn = Taught. ssn (σ crscode = 'csc7710' (Taught)))

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return those professors who have never taught 'csc7710'.

$$\Pi_{ssn}(\sigma_{crscode \neq 'csc7710'}(Taught))$$

OR

$$\Pi_{ssn,\ profname}(Professor) \bowtie (Professor.ssn = Taught.ssn\ (\sigma_{crscode \neq 'csc7710'}(Taught)))$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return those professors who taught 'csc6710' or 'csc7710' but not both.

$$\Pi_{ssn}(\sigma_{crscode = 'csc6710' \vee crscode = 'csc7710'}(Taught)) - \Pi_{ssn}(\sigma_{crscode = 'csc6710' \wedge crscode = 'csc7710'}(Taught))$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return those course that have never been taught.

$$\Pi_{crscode}(Course) - \Pi_{crscode}(Taught)$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return the names of professors who ever taught 'csc6710'.

$$\Pi_{\text{profname}}(\text{Professor}) \bowtie_{\text{Professor.ssn} = \text{Taught.ssn}} (\sigma_{\text{crscode} = \text{'csc6710'}}(\text{Taught}))$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return the names of full time professors who ever taught 'csc6710'.

$$\Pi_{\text{profname}}(\sigma_{\text{status} = \text{'full'}}(\text{Professor})) \bowtie_{\text{Professor.ssn} = \text{Taught.ssn}} (\sigma_{\text{crscode} = \text{'csc6710'}}(\text{Taught}))$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

List the names of those courses that professor Smith have never taught.

$$\Pi_{\text{crsname}}(\text{Course}) - (\Pi_{\text{crsname}}(\text{Course}) \bowtie_{(\text{Course.crscode} = \text{Taught.crscode}} (\text{Taught}) \bowtie_{(\text{Taught.ssn} = \text{Professor.ssn}} (\sigma_{\text{profname} = \text{'Smith'}}(\text{Professor}))))$$

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return the highest salary.

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return the highest salary of each professor group (Full, associate, partial).

Professor (ssn, profname, status, salary)
Course (crscode, crsname, credits)
Taught (crscode, semester, ssn)

* 3 semester in a year

Return those courses that have been taught in all semesters.

Hotel (hotelNo, hotelName, city)
Room (roomNo, hotelNo, type, price)
Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)
Guest (guestNo, guestName, guestAddress)

Generate the relational algebra expressions for the following queries:

1. List all hotels.
2. List all single rooms with a price below $20 per night.
3. List the names and cities of all guests staying at Setapak.
4. List the price and type of all rooms at the Hilton Hotel.
5. List the guest details (guestNo, guestName and guestAddress) of all guests currently staying at the Hilton Hotel.
6. List the details of all rooms at the Hilton Hotel, including the name of the guest staying in the room, if the room is occupied. All rooms that are not occupied with such a date will still be included in the join.

a) Hotel

b) σ type = 'S' $\wedge$ price < 20 (Room)

c) π guestName, guestAddress (σ guestAddress LIKE '%Setapak%' (Guest))

d) π price, type (Room) $\bowtie$ (Room.hotelNo = Hotel.hotelNo (σ Hotel.hotelName = 'Hilton Hotel' (Hotel)))

e) π guestNo, guestName, guestAddress (Guest) $\bowtie$ (Guest.guestNo = Booking.guestNo (σ dateFrom $\leq$ 'XXX' $\wedge$ dateTo $\geq$ 'XXX' (Booking)) $\bowtie$ (Booking.hotelNo = Hotel.hotelNo (σ hotelName = 'Hilton Hotel' (Hotel)))))

(substitute 'XXX' for today's date)

f) Room $\bowtie$ (Room.hotelNo = Hotel.hotelNo (σ hotelName = 'Hilton Hotel' (Hotel))) $\bowtie$ π guestNo, guestName, guestAddress (Guest) $\bowtie$ (Guest.guestNo = Booking.guestNo (σ dateFrom $\leq$ 'XXX' $\wedge$ dateTo $\geq$ 'XXX' (Booking)) $\bowtie$ (Booking.HotelNo = Hotel.hotelNo (σ hotelName = 'Hilton Hotel' (Hotel)))))

(substitute 'XXX' for today's date)

C5: Relational Database Design

Relational Database Design

- Purposes:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of integrity constraints
- Consequences of bad design:
  - Repetition of information
    - Leads to anomalies
  - Loss of information (lossy decomposition)

Problems of Data Redundancy

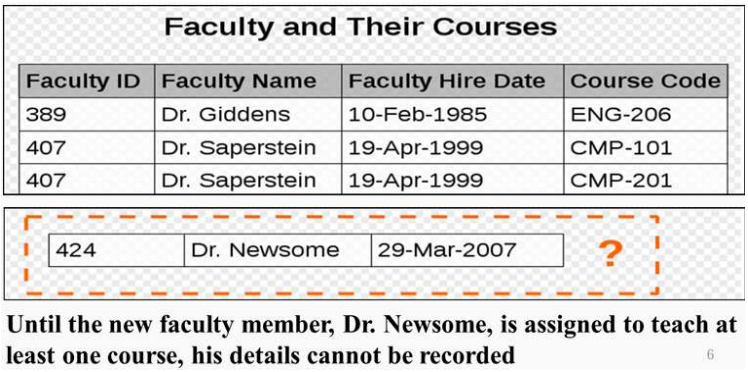
<table>
<tr>
<td>Modification Anomaly</td>
<td>

The same information can be expressed on multiple rows; therefore modification to the table may result in logical inconsistencies
Since the same information can be stored in multiple rows, modifying data in one row requires updating all corresponding rows. Failure to do so may result in logical inconsistencies, where some rows reflect outdated or incorrect data.

Employees' Skills
<table>
<tr><th>Employee ID</th><th>Employee Address</th><th>Skill</th></tr>
<tr><td>426</td><td>87 Sycamore Grove</td><td>Typing</td></tr>
<tr><td>426</td><td>87 Sycamore Grove</td><td>Shorthand</td></tr>
<tr><td>519</td><td>94 Chestnut Street</td><td>Public Speaking</td></tr>
<tr><td>519</td><td>96 Walnut Avenue</td><td>Carpentry</td></tr>
</table>
</td>
</tr>
<tr>
<td>Insertion Anomaly</td>
<td>

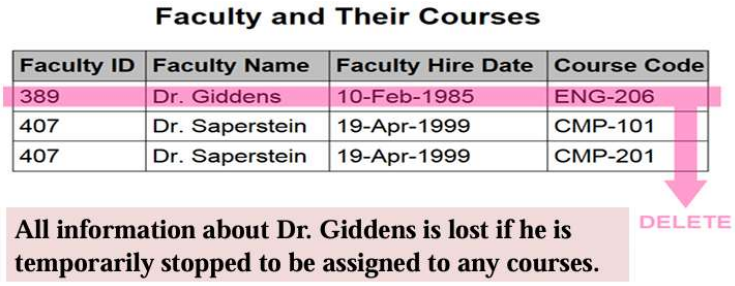
There are circumstances in which certain facts cannot be recorded at all
In some cases, it may be impossible to insert new data into a table without existing related data. This occurs when certain attributes depend on others that are not yet available, making data insertion infeasible.

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>


</td>
</tr>
<tr>
<td>Deletion Anomaly</td>
<td>

Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts.
Deleting a row can inadvertently remove essential data if that row contains the only record of a specific fact. This may lead to unexpected loss of important information unrelated to the primary deletion purpose.

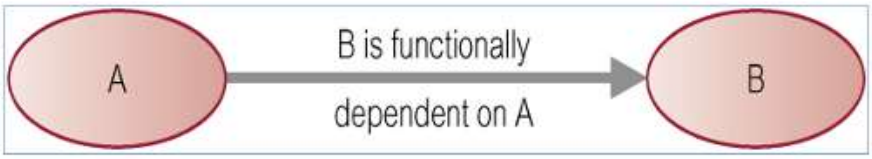


</td>
</tr>
</table>

Functional Dependency

- Describes relationship between attributes

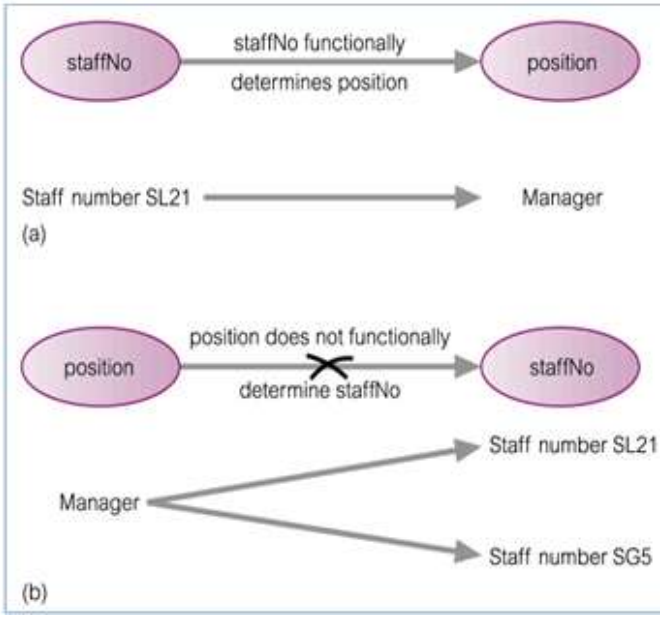
If A and B are attributes of relation R,
B is functionally dependent on A
(denoted A → B),
if each value of A in R is associated with exactly one value of B in R.



- The determinant of functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow
- A → B (B is functionally dependent on A) if each value of A in R is associated with exactly one value of B in R.
- e.g.:

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005



Full Functional Dependency	• A functional dependency A → B is fully functional if **B depends on the entire A**, but not on any subset of A.
	• This usually involves **primary keys** and attributes that can **only** be determined when the **entire primary key** is present
	• E.g.:
	  ○ Staff relation: staffNo, sName → branchNo
	  ○ Each unique combination of staffNo and sName determines a single branchNo (Full functional dependency)

Partial Dependency	• Indicates that **B is dependent on only a part of A**
	• This typically involves **foreign keys** and attributes that can be determined using **only part** of a composite key instead of the whole key
	• It often appears in **2nd Normal Form (2NF) violations** in relational databases
	• E.g.:
	  ○ Staff relation: staffNo, sName → branchNo
	  ○ The branchNo is actually dependent only on staffNo, not on (staffNo, sName).
	  ○ Since branchNo can be determined by staffNo alone, this means that using (staffNo, sName) as a

	determinant introduces a **partial dependency**.
Transitive Dependency	• Can potentially cause **update anomalies** • This usually involves **non-primary keys** and attributes that depend on other **non-key attributes** • Transitive dependencies can cause **update anomalies** and are typically removed in **3rd Normal Form (3NF)** • Describes a condition where A, B and C are attributes of a relation such that ○ If A → B and B → C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C)

Normalization

- Formal methodology for refining and creating good relational designs using common schema refinement techniques, decomposition

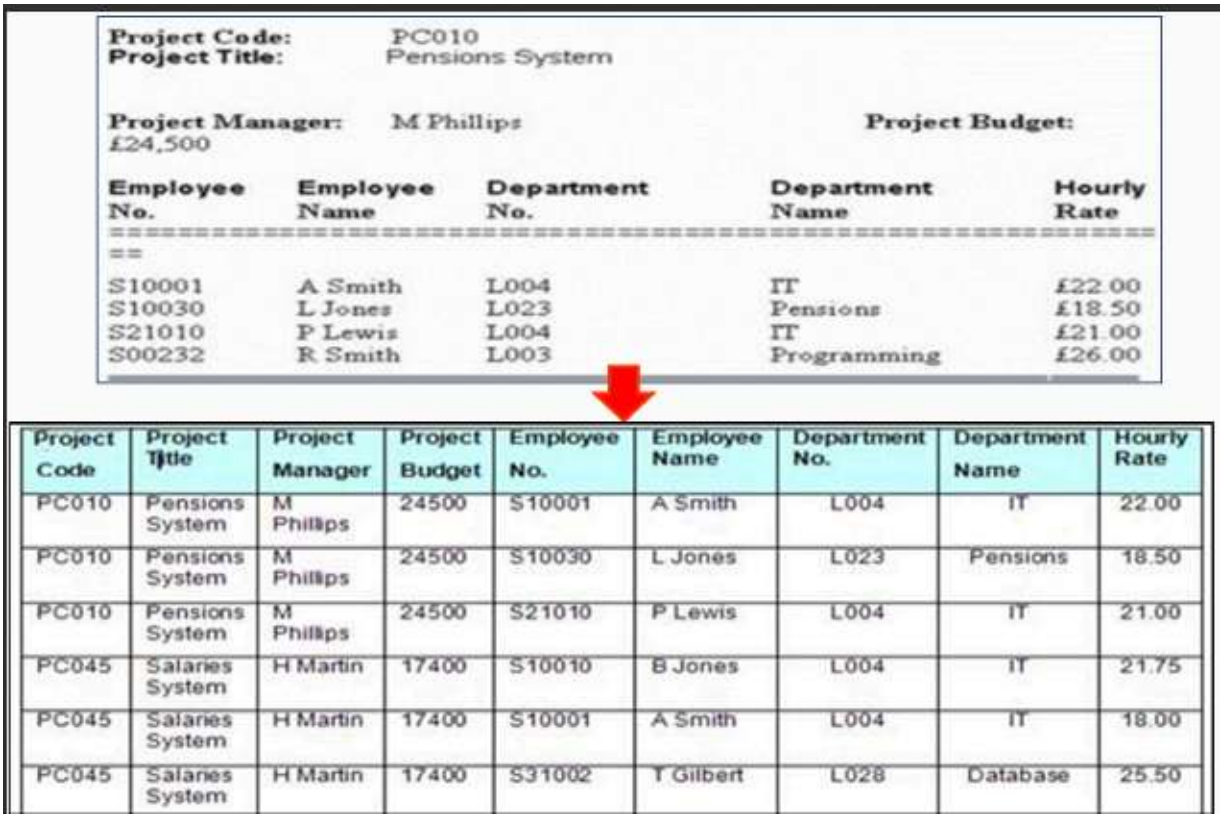
Properties of Decomposition

Lossless-join property	The final relations must contain exactly the information contained in the original relation, without losing any nor adding any other ones
Dependency preservation property	The final relations must be characterized by the same FDs as the original relation

Unnormalized Form (UNF)

- A table that contains one or more **repeating groups**
- How to create unnormalized table:
  - Transform the data from the information source (e.g. form) into **table format with columns and rows**

- e.g.:



Project Code: PC010
Project Title: Pensions System

Project Manager: M Phillips **Project Budget:** £24,500

Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
S10001	A Smith	L004	IT	£22.00
S10030	L Jones	L023	Pensions	£18.50
S21010	P Lewis	L004	IT	£21.00
S00232	R Smith	L003	Programming	£26.00

Project Code	Project Title	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Pensions System	M Phillips	24500	S10001	A Smith	L004	IT	22.00
PC010	Pensions System	M Phillips	24500	S10030	L Jones	L023	Pensions	18.50
PC010	Pensions System	M Phillips	24500	S21010	P Lewis	L004	IT	21.00
PC045	Salaries System	H Martin	17400	S10010	B Jones	L004	IT	21.75
PC045	Salaries System	H Martin	17400	S10001	A Smith	L004	IT	18.00
PC045	Salaries System	H Martin	17400	S31002	T Gilbert	L028	Database	25.50

First Normal Form (1NF)

- A relation in which the **intersection of each row and column contains one and only one value**
- How to convert UNF to 1NF:
    - Nominate an **attribute or group of attributes to act as the key** for the unnormalized table
    - Identify the **repeating group(s)** in the unnormalized table which repeats for the key attribute(s)
    - **Remove the repeating group** by **entering appropriate data into the empty columns of rows** containing the repeating data ('flattening' the table)
    - e.g.:

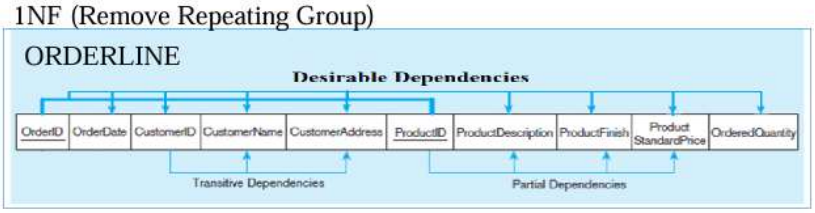
UNF

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

1NF

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

1NF (Dependency Diagram)



ORDERLINE (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress, <u>ProductID</u>, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)

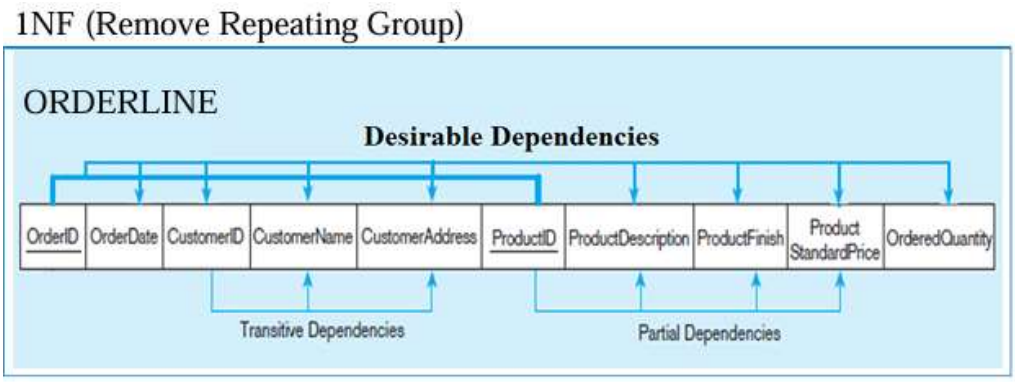
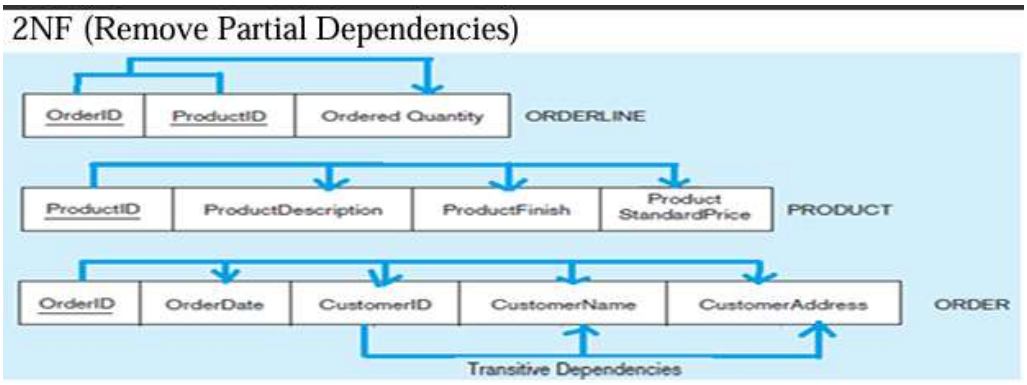
Convert into

ORDER (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress)

	ORDERLINE(OrderID*, ProductID, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)

Second Normal Form (2NF)

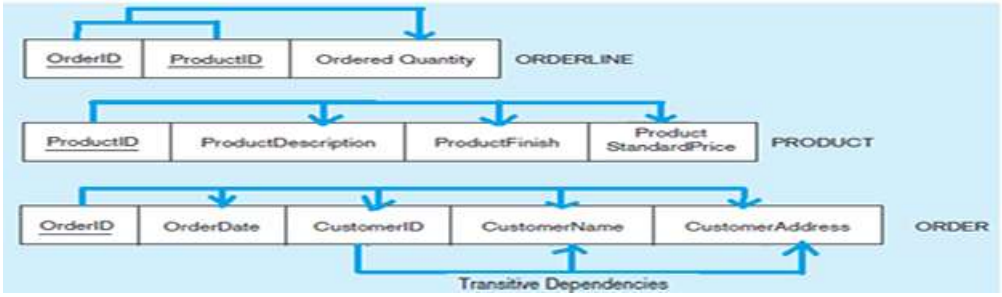
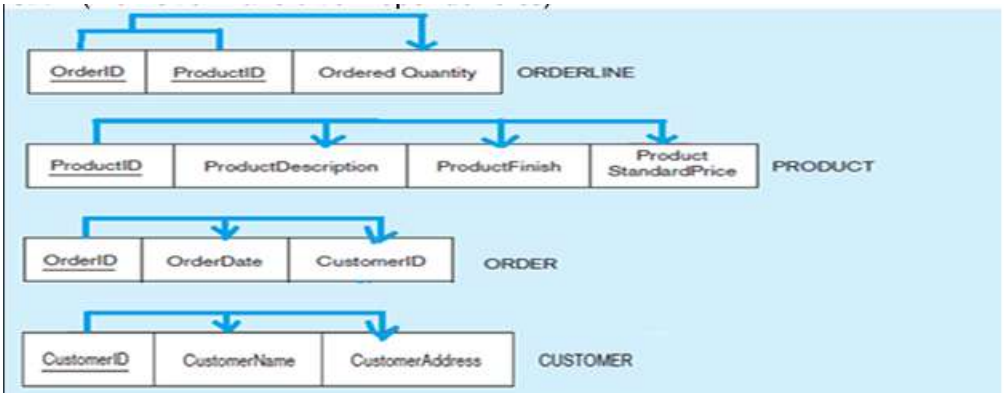
- Based on the concept of **full functional dependency**
- A relation that is
  - In **1NF**
  - **Every non primary-key attribute is fully functional dependent on the primary key**
  - E.g.
    - studID, studName → programmeCode
    - Partial dependency exists
- How to convert 1NF to 2NF:
  - Identify the **primary key** for 1NF relation
  - Identify the **functional dependencies** in the relation
  - If partial dependency exist on the primary key, **remove them by placing them in a new relation** along with a copy of their determinant
- E.g.

1NF	 ORDER (OrderID, OrderDate, CustomerID, CustomerName, CustomerAddress) ORDERLINE(OrderID*, ProductID, ProductDescription, ProductFinish, ProductStandardPrice, OrderQuantity)
2NF	 ProductID → ProductDescription, ProductFinish, ProductStandardPrice (partial

	dependency) ORDERLINE (<u>OrderID*</u>, <u>ProductID*</u>, OrderedQuantity) PRODUCT (<u>ProductID</u>, ProductDescription, ProductFinish, ProductStandardPrice) ORDER (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress)

Third Normal Form (3NF)

- Based on the concept of **transitive dependency**
- A relation that is:
    - In **1NF** and **2NF**
    - In which **no non primary-key attribute is transitively dependent on the primary key**
- How to convert 2NF to 3NF:
    - Identify the **primary key** in the 2NF relation
    - Identify **functional dependencies** in the relation
    - If **transitive dependencies** exist on the primary key, **remove them by placing them in a new relation** along with a copy of their determinant
- e.g.:

2NF	 ORDERLINE (<u>OrderID*</u>, <u>ProductID*</u>, OrderedQuantity) PRODUCT (<u>ProductID</u>, ProductDescription, ProductFinish, ProductStandardPrice) ORDER (<u>OrderID</u>, OrderDate, CustomerID, CustomerName, CustomerAddress)
3NF	 CustomerID → CustomerName, CustomerAddress (Transitive Dependency) ORDERLINE (<u>OrderID*</u>, <u>ProductID*</u>, OrderedQuantity)

	PRODUCT (ProductID, ProductDescription, ProductFinish, ProductStandardPrice) ORDER (OrderID, OrderDate, CustomerID*) CUSTOMER (CustomerID, CustomerName, CustomerAddress)

Extra Questions

Given the CovidExamination table as follows:

PatientID	PatientName	StateID	StateName	ExamDate	VariantID	VariantName	HospitalID	HospitalName
P5010	Alice	S001	Selangor	08/01/2022	V0001	Alpha	H223	Hospital KL
P5010	Alice	S001	Selangor	08/01/2022	V0004	Delta	H223	Hospital KL
P5010	Alice	S001	Selangor	11/11/2022	V0001	Alpha	H881	Hospital PJ
P7888	Thomas	S008	Melaka	11/11/2022	V0004	Delta	H666	Hospital SB
P3002	Penny	B009	Perak	15/03/2022	V0004	Delta	H666	Hospital SB
P3002	Penny	B009	Perak	12/12/2022	V0004	Delta	H223	Hospital KL
P3355	Julie	S008	Melaka	12/12/2022	V0002	Beta	H881	Hospital PJ

Table 1: Details of CovidExamination Table

a) Normalize Table 1 to a set of Third Normal Form (3NF) relations. Your answer should show all the three stages of normalization (1NF, 2NF and 3NF) by using the Database Design Language format (underline all primary keys, composite keys and use an * to indicate the foreign keys). State the functional dependency/dependencies that is/are removed from second and third Normal Form. Besides that, 1NF must be divided into repeating and non-repeating group relations from its original 1NF table. (16 marks)

b) Based on the sample data shown in the **CovidExamination** table above, provide a specific example for insertion, modification and deletion anomalies. (9 marks)

[Total: 25 marks]

Question a)
UNF
CovidExamination (PatientID, PatientName, StateID, StateName, ExamDate, VariantID, VariantName, HospitalID, HospitalName)

1NF
CovidExamination (PatientID*, ExamDate, VariantID, VariantName, HospitalID, HospitalName)
Patient (PatientID, PatientName, StateID, StateName)

2NF
PatientID, ExamDate, VariantID → VariantName, HospitalID, HospitalName
VariantID → VariantName (Partial Dependency)

CovidExamination (PatientID*, ExamDate, VariantID*, HospitalID, HospitalName)
Patient (PatientID, PatientName, StateID, StateName)

Variant (VariantID, VariantName)

3NF
PatientID → PatientName, StateID, StateName
StateID → StateName (Transitive Dependency)

PatientID, ExamDate, VariantID → VariantName, HospitalID, HospitalName
HospitalID → HospitalName (Transitive Dependency)

CovidExamination (PatientID*, ExamDate, VariantID*, HospitalID*)
Patient (PatientID, PatienName, StateID*)
Variant (VariantID, VariantName)
Hospital (HospitalID, HospitalName)
State (StateID, StateName)

Question b)

- Insert anomaly: It is not possible to add a new patient unless that new patient is perform the covid examination.
- Modification anomaly: When we update the variant name of V0004 from 'Delta' to 'Delta Plus', we need also update the similar variant name in another row, if not it will cause data inconsistency.
- Deletion anomaly: When we delete the Julie (P3355) record we also will delete this covid variant record of Beta (V0002).

1. VideoStore Relation

TransID	RentDate	CustomerID	CustName	CustPhone	VideoID	VideoTitle	#Copy	ChargePerCopy
T001	01/12/02	C1223	Ai Li Lim	07-12345678	V001	Titanic	2	RM2.00
T001	01/12/02	C1223	Ai Li Lim	07-12345678	V006	The Tuxedo	1	RM1.50
T002	02/12/02	C1225	Aee Sy Go	07-87654321	V001	Titanic	1	RM2.00
T002	02/12/02	C1225	Aee Sy Go	07-87654321	V002	Office Ghosts	1	RM1.50
T002	02/12/02	C1225	Aee Sy Go	07-87654321	V008	The Signs	1	RM1.50

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

TransID, VideoID → #Copy (full dependency)
TransID → RentDate, CustomerID, CustName, CustPhone (Partial dependency)
VideoID → VideoTitle, ChargePerCopy (Partial Dependency)
CustomerID → CustName, CustPhone (Transitive dependency)

1NF
VideoStore (TransID*, VideoID, VideoTitle, #Copy, ChargePerCopy)
Transaction(TransID, RentDate, CustomerID, CustName, CustPhone)

2NF
VideoStore (TransID*, VideoID*, #Copy)
Transaction(TransID, RentDate, CustomerID, CustName, CustPhone)
Video (VideoID, VideoTitle, ChargePerCopy)

3NF
VideoStore (TransID*, VideoID*, #Copy)
Transaction(TransID, RentDate, CustomerID*)
Video (VideoID, VideoTitle, ChargePerCopy)
Customer (CustomerID, CustName, CustPhone)

2. A construction company manages several building projects. Each project has its project number, name and employees assigned to the project. Each employee has an employee number, name and job classification. The pay rate is dependent on the employee's position.

ProjectNumber	ProjectName	EmployeeNumber	EmployeeName	JobClassification	ChargePerHour	Hours
PRO3001	Marianne	E1001	Eric Heng	Computer Technology	80	18
		E1002	Gary Chee	Electronic Engineering	75	16
		E1004	Raymond Low	Computer Technology	80	19
PRO3002	Coast	E1001	Eric Heng	Computer Technology	80	13
		E1003	Melvin Sim	Biology Engineering	77	17
PRO3003	Satellite	E1004	Raymond Low	Computer Technology	80	15
		E1002	Gary Chee	Electronic Engineering	75	15

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

ProjectNumber, EmployeeNumber —> EmployeeName, JobClassification, ChangePerHour, Hours (Full dependency)
ProjectNumber —> ProjectName (Partial dependency)
EmployeeNumber —> EmployeeName (Partial dependency)
JobClassification —> ChargePerHour (Transitive dependency)

1NF
ProjectAssignment (ProjectNumber*, EmployeeNumber, EmployeeName, JobClassification, ChargePerHour, Hours)
Project (ProjectNumber, ProjectName)

2NF
ProjectAssignment (ProjectNumber*, EmployeeNumber*, Hours)
Project (ProjectNumber, ProjectName)
Employee (EmployeeNumber, EmployeeName, JobClassification, ChargePerHour)

3NF
ProjectAssignment (ProjectNumber*, EmployeeNumber*, Hours)
Project (ProjectNumber, ProjectName)
Employee (EmployeeNumber, EmployeeName, JobClassification*)
Job (JobClassification, ChargePerHour)

3. Examine the following relation describing the facial treatment transactions for a beauty therapy saloon.

CustID	CustName	CustSkinType	TreatmentDate	TreatmentCode	TreatmentDesc	BeauticianID	BeauticianName
C1003	Yee Si Go	Normal	08/01/2003	NOF	Normal Facial	B008	Susan
C1003	Yee Si Go	Normal	08/01/2003	ETR	Eye Treatment	B203	Linda
C1003	Yee Si Go	Normal	18/01/2003	NOF	Normal Facial	B108	Esther
C1004	Ai Li Lim	Dry	02/01/2003	NOF	Normal Facial	B108	Esther
C1004	Ai Li Lim	Dry	10/01/2003	NTR	Neck Treatment	B008	Susan
C1005	Ellen Tan	Oily	11/01/2003	NOF	Normal Facial	B203	Linda
C1005	Ellen Tan	Oily	11/01/2003	ETR	Eye Treatment	B203	Linda

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

CustID, TreatmentDate, TreatmentCode —> TreatmentDesc, BeauticianID, BeauticianName (Full dependency)
CustID —> CustName, CustSkinType (Partial dependency)
TreatmentCode —> TreatmentDesc (Partial dependency)
BeauticianID —> BeauticianName (Transitive dependency)

1NF
TreatmentTransaction (CustID*, TreatmentDate, TreatmentCode, TreatmentDesc, BeauticianID, BeauticianName)
Customer (CustID, CustName, CustSkinType)

2NF
TreatmentTransaction (CustID*, TreatmentDate, TreatmentCode*, BeauticianID, BeauticianName)
Customer (CustID, CustName, CustSkinType)
Treatment (TreamentCode, TreatmentDesc)

3NF
TreatmentTransaction (CustID*, TreatmentDate, TreatmentCode*, BeauticianID*)
Customer (CustID, CustName, CustSkinType)
Treatment (TreamentCode, TreatmentDesc)
Beautician (BeauticianID, BeauticianName)

4. The table shown below is the Account table called Account in a banking system.

CustomerName	CustomerAddress	AccountNumber	TransactionCode	TransactionDescription	TransactionDate	Amount
Gary Chee	Segamat	A1223	DEP	Deposit	08/01/2003	1000
Gary Chee	Segamat	A1223	WDR	Withdraw	08/01/2003	500
Gary Chee	Segamat	A1223	WDR	Withdraw	10/01/2003	300
Gary Chee	Segamat	A1223	DEP	Deposit	30/01/2003	1200
Ellen Tan	Bukit Siput	A1224	WDR	Withdraw	08/01/2003	500
Melvin Sim	Labis	A1225	DEP	Deposit	08/01/2003	800

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

AccountNumber, TransactionCode, TransactionDate —> Amount (Full dependency)
AccountNumber —> CustomerName, CustomerAddress (Partial dependency)
TransactionCode —> TransactionDescription (Partial dependency)

CustomerName —> CustomerAddress (Transitive dependency)

1NF
AccountTransaction (AccountNumber*, TransactionCode, TransactionDate, TransactionDescription, Amount)
Account (AccountNumber, CustomerName, CustomerAddress)

2NF
AccountTransaction (AccountNumber*, TransactionCode*, TransactionDate, Amount)
Account (AccountNumber, CustomerName, CustomerAddress)
Transaction (TransactionCode, TransactionDescription)

3NF
AccountTransaction (AccountNumber*, TransactionCode*, TransactionDate, Amount)
Account (AccountNumber, CustomerName*)
Transaction (TransactionCode, TransactionDescription)
Customer (CustomerName, CustomerAddress)

5. The table shown below lists dentist-patient appointment data. A patient is given an appointment at a specific time and date with a dentist located at a particular surgery. On each day of patient appointments, a dentist is allocated to a specific surgery for that day.

StaffNo	DentistName	PatientNo	PatientName	Appointment		SurgeryRoomNo	SurgeryRoomDesc
				Date	Time		
S1011	Gary Chee	P100	Eddie Koh	12-Sept-02	1.00p.m.	S15	Room C
		P100	Eddie Koh	12-Sept-02	4.30p.m.	S15	Room C
S1024	Jenny Tay	P108	Melvin Sim	12-Sept-02	12.30p.m.	S10	Room B
		P110	Alice Go	14-Sept-02	12.30p.m.	S10	Room B
S1032	Ellen Tan	P105	Raymand Low	14-Sept-02	10.00p.m.	S15	Room C
		P110	Alice Go	15-Sept-02	10.30p.m.	S13	Room A

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

StaffNo, PatientNo, AppointmentDate, AppointmentTime —> SurgeryRoomNo, SurgeryRoomDesc (Full dependency)
StaffNo —> DentistName (Partial dependency)
PatientNo —> PatientName (Partial dependency)
SurgeryRoomNo —> SurgeryRoomDesc (Transitive dependency)

1NF
Appointment (StaffNo*, PatientNo, AppointmentDate, AppointmentTime, PatientName, SurgeryRoomNo, SurgeryRoomDesc)
Staff (StaffNo, DentistName)

2NF
Appointment (StaffNo*, PatientNo*, AppointmentDate, AppointmentTime, SurgeryRoomNo, SurgeryRoomDesc)
Staff (StaffNo, DentistName)
Patient (PatientNo, PatientName)

3NF
Appointment (StaffNo*, PatientNo*, AppointmentDate, AppointmentTime, SurgeryRoomNo*)
Staff (StaffNo, DentistName)
Patient (PatientNo, PatientName)
SurgeryRoom (SurgeryRoomNo, SurgeryRoomDesc)

6. Pigeon Healthcare Center provides a free treatment daily for each customer. Each treatment time (in minutes) is recorded.

CustNo	CustName	CustContact	StateCode	StateDesc	TreatDate	TreatID	TreatDesc	TreatTime
C1003	Yee Si Go	012-22552255	S01	Kuala Lumpur	08/01/2017	T001	Electronic Chair	35
C1004	Ai Li Lim	012-33882288	S02	Selangor	08/01/2017	T001	Electronic Chair	30
C1005	Ellen Tan	012-55335533	S01	Kuala Lumpur	08/01/2017	T002	Foot Massage	75
C1003	Yee Si Go	012-22552255	S01	Kuala Lumpur	09/01/2017	T001	Electronic Chair	38
C1004	Ai Li Lim	012-33882288	S02	Selangor	09/01/2017	T002	Foot Massage	70
C1005	Ellen Tan	012-55335533	S01	Kuala Lumpur	09/01/2017	T003	Body Slimming	55

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

CustNo, TreatDate, TreatID —> TreatDesc, TreatTime

1NF
TreatmentHistory (CustNo*, TreatDate, TreatID*, TreatDesc, TreatTime)
Customer (CustNo, CustName, CustContact, StateCode, StateDesc)

2NF
TreatmentHIstory (CustNo*, TreatDate, TreatID*, TreatTime)
Customer (CustNo, CustName, CustContact, StateCode, StateDesc)
Treatment(TreatID, TreatDesc)

3NF
TreatmentHIstory (CustNo*, TreatDate, TreatID*, TreatTime)
Customer (CustNo, CustName, CustContact, StateCode*)
Treatment(TreatID, TreatDesc)
State(StateCode, StateDesc)

6. Pigeon Healthcare Center provides a free treatment daily for each customer. Each treatment time (in minutes) is recorded.

CustNo	CustName	CustContact	StateCode	StateDesc	TreatDate	TreatID	TreatDesc	TreatTime
C1003	Yee Si Go	012-22552255	S01	Kuala Lumpur	08/01/2017	T001	Electronic Chair	35
C1004	Ai Li Lim	012-33882288	S02	Selangor	08/01/2017	T001	Electronic Chair	30
C1005	Ellen Tan	012-55335533	S01	Kuala Lumpur	08/01/2017	T002	Foot Massage	75
C1003	Yee Si Go	012-22552255	S01	Kuala Lumpur	09/01/2017	T001	Electronic Chair	38
C1004	Ai Li Lim	012-33882288	S02	Selangor	09/01/2017	T002	Foot Massage	70
C1005	Ellen Tan	012-55335533	S01	Kuala Lumpur	09/01/2017	T003	Body Slimming	55

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

8. Pigeon Home Estate has three branches that manage a lot of properties within Selangor and Kuala Lumpur. Each property is allocate to one and only one branch and manages by one and only one agent.

BranchNo	BranchName	BranchManager	PropertyNo	PropertyAddress	Owner	Rent	AgentCode	AgentName
B001	Love Pigeon	Eric	P0001	Setapak Ria	Rose	800	A01	Coco
B001	Love Pigeon	Eric	P0002	Menara Alpha	May	1000	A02	Thomas
B002	Big Pigeon	Almond	P0003	TAR Villa	Rich	1200	A03	Jack
B003	Little Pigeon	Justine	P0004	Beta Condo	Mandy	1300	A04	Alice
B003	Little Pigeon	Justine	P0005	Setapak Ria	Rose	850	A04	Alice

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

1NF
Branch(BranchNo, BranchName, BranchManager)
PropertiesManage (BranchNo, PropertyNo, PropertyAddress, Owner, Rent, AgentCode, AgentName)

2NF
PropertiesManage (BranchNo*, PropertyNo*, Rent, AgentCode, AgentName)
Branch(BranchNo, BranchName, BranchManager)
Property (PropertyNo, PropertyAddress, Owner)

3NF
PropertiesManage (BranchNo*, PropertyNo*, Rent, AgentCode*)
Branch(BranchNo, BranchName, BranchManager)
Property (PropertyNo, PropertyAddress, Owner)
Agent(AgentCode, AgentName)

9. Consider the following first normal form relation for Travel_Agency:

TripID	**TripDescription**	**BusPlateNo**	**NoOfSeats**	**DepartureDate**	**CustID**	**CustName**	**CustTelNo**
T0001	Tioman Island	JBA1234	40	08/11/02	C001	Chong PF	7153918
T0001	Tioman Island	JBA1234	40	08/11/02	C002	Fong KA	6129015
T0002	Pangkor Island	WFU3333	35	08/11/02	C003	Liew HW	7696551
T0003	Langkawi Island	JBA1234	40	18/11/02	C001	Chong PF	7153918
T0001	Tioman Island	JBA5678	30	08/12/02	C003	Liew HW	7696551
T0002	Pangkor Island	WFU3333	35	18/12/02	C003	Liew HW	7696551

a) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and their functional dependencies.
b) The table is susceptible to anomalies. Provide examples of insertion, deletion and modification anomalies.
c) Normalize the above structure to 1NF, 2NF and 3NF by providing meaningful relation names and using dependency diagram.

1NF
Travel_Agency (TripID*, DepartureDate, CustID, CustName, CustTelNo)
Trip (TripID, TripDescription, BusPlateNo, NoOfSeats)

2NF
Travel_Agency (TripID*, DepartureDate, CustID*)

Trip (TripID, TripDescription, BusPlateNo, NoOfSeats)
Customer (CustID, CustName, CustTelNo)

3NF
Travel_Agency (TripID*, DepartureDate, CustID*)
Trip (TripID, TripDescription, BusPlateNo*)
Customer (CustID, CustName, CustTelNo)
Bus (BusPlateNo, NoOfSeats)