IT IM01

advanced database system

NEUST

RA.G. SANTOS
E.T. BANTUG
J.M. PADRE
B.C. BONDOC
E. C. NAVARRO

1ST SEMESTER
AY 2020-2021

PREFACE

This module, ***Advanced Database System***, is primarily designed for the 3rd year BS Information Technology students of NEUST who are taking up the *Advanced Database Management* subject. This is a product of the collaborative efforts of the CICT faculty from Sumacab Campus and other satellite campuses.

It is composed of four major topics aligned with the course outline designed to provide the students with knowledge and skills in terms of database design and management. The different lessons are presented in a simple manner in order to facilitate faster learning and better understanding.

Each lesson includes comprehensive discussions and sample illustrations to enable students to understand every topic more clearly. Practical examples of SQL statements are also provided to aid the students in their application and practice of the database language. Assessment can be found at the end of each lesson in order to evaluate the students' learning and skills.

The Authors

TABLE OF CONTENTS

UNIT I. Database Development Process

Learning Objectives:

At the end of the unit, I will be able to:

1. differentiate the stages in the database life cycle; and
2. demonstrate understanding of the difference between centralized and decentralized database design.

(DBLC) Database Life Cycle

The ***database life cycle (DBLC)*** is described as the stages includes for implementing a database, which begins with the analysis of the requirements and ends with the checking and modification. Likewise, the DBLC never ends because these activities are continuing long after the implementation of the Database. In other words, the DBLC encompasses the database lifetime.

The database life cycles are:

1. **Requirements analysis**

   Requirements Analysis is the first activity in DBLC. It requires extensive work for the database designer. It involves evaluating the details needed in an organization so that the database can be designed to produce the required information.

2. **Logical design**

   A conceptual model is usually an entity-relationship (ER) diagram consisting of tables, field, and the primary that shows how tables are connected. The tables are then normalized to resolves redundancy and any problems that may be encountered with the design.

3. **Physical Design**

   The ***Physical design*** stage has one aim: *to optimize the productivity of the database.* That means finding ways to accelerate RDMS performance. Extracting of data from database and writing data into it which are the slowest operations in RDMS can speed up by manipulating some elements of database design.

Levels of the database

- Notional;
- Logical; and
- Physical

Notional and Logical Structure

The design and modification of the Database should always be at the standard level. *Two possible scenarios during the design phase are:*

1. Some use machine styles file manager, such as dBase, Paradox, or Foxpro. In a network environment, these have historical significance and can be accessed via file-servers.
   .
2. Some build more advanced systems to create a network ambiance for multi-users. We use client-server architecture; we may also be able to evolve it further to create multilayer applications. These systems include Oracle, MSSQL, PostgreSQL, and MySQL, for example.

Physical Structure

An acceptable database is when physical manifestation contests realities. There are, however, some problems which may arise:

1. ***Assertion.*** The data to be entered must be acceptable or valid. If we add a date, for example, its value and structure must be correct.

2. ***Data presentation.*** It is giving the specific type and size of data such as *textual, imaginary, logical, numerical, etc.*

3. ***Data management and storage method.*** The more advanced the management and storage method are, the less time it is needed to monitor the database behavior.

The Names of the Physical Data Structure

1. ***Data table.*** The file in table form represented by the entities or data sources.
2. ***Field***. The table columns which are represented by the attributes.
3. ***Record.*** The table rows or also known as tuple.
4. ***Elementary item.*** Values per table cell.

Step in Designing the Database

1. ***Requirement handling.*** It is to define the objectives of the database that will be the basis of the database design.

2. ***Identification of entities and their attributes.*** Once the entities are identified, table structures can be specified.

3. ***Declaration of identifiers such primary and foreign keys***. It is to establish the relationships.

4. ***Establishment of relationships***. It is to show the connection between relations. *Common types of relationships are:*
   - One-to-one relation
   - One-to-many relation
   - Many-to-many relation

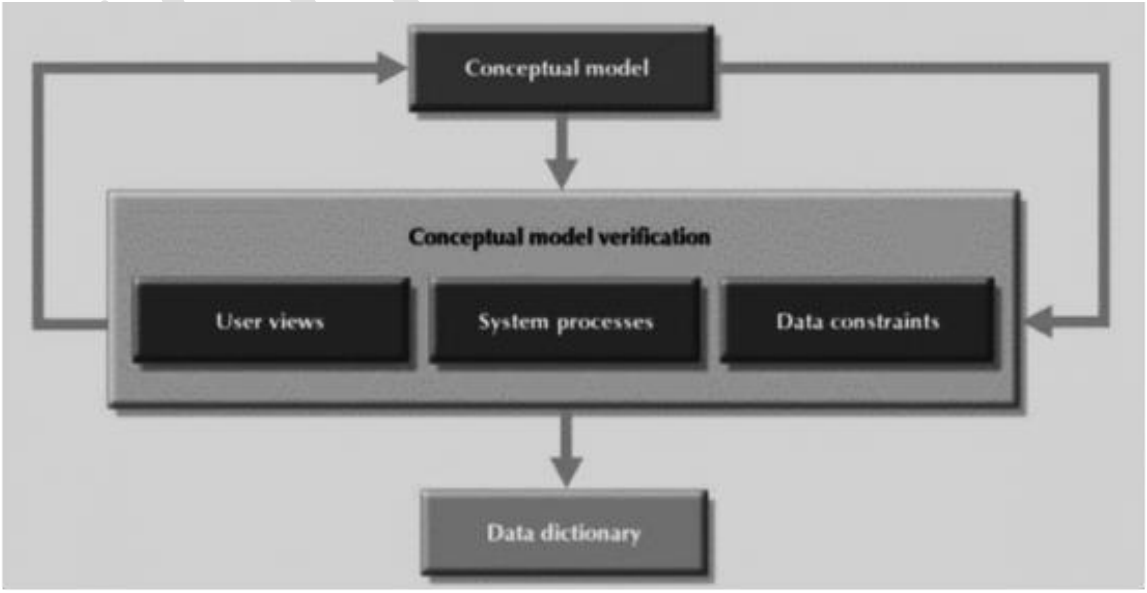
5. ***Testing.*** Once the tables are created and keys are in place, testing is required to ensure that the design does not cause any inconsistencies.

6. ***Data input.*** Uploading of data to the database.

Database Centralized Design

It is the database design that is usually employed when we are dealing with small –scale databases with small number of objects and procedures.

Centralized Design is shown on this figure, as you can see from the conceptual model down to conceptual model verification and data dictionary.

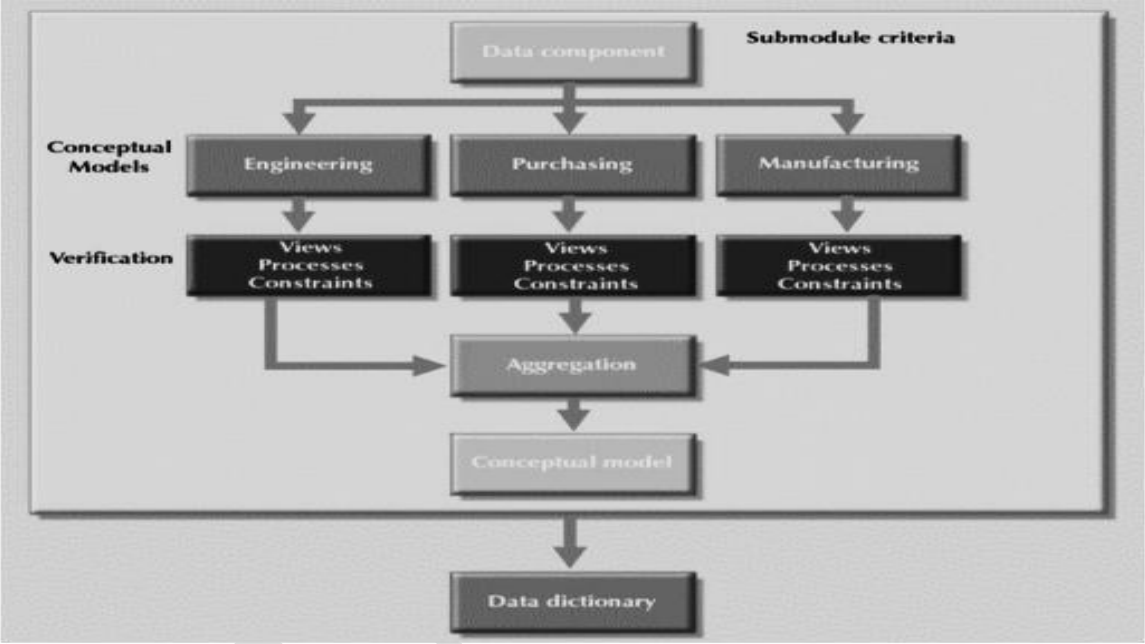


Database Systems: Design, Implementation, & Management, 6th Edition, Rob & Coronel

Database Decentralized Design

This is the database design that is ideal for bigger number of entities and complex relations.

Decentralized Design is shown on this figure, as you can see we have major stages, namely: *the submodule criteria for data component, a conceptual model for technical and conceptual verification goes down to aggregation and data dictionary.*



Database Systems: Design, Implementation, & Management, 6th Edition, Rob & Coronel

Assessing Learning

Name: _____ **Date: _____**
Section: _____

Part I. *Directions*: Encircle the best answer of the following statement.

1. What is the database life cycle?
    a. A process to define the budget of a project
    b. A methodology that defines the steps of a database development project
    c. Resource allocations for each project
    d. The process of defining requirements
2. What are the stages of the DBLC?
    a. Requirements gathering, design, coding, testing, deployment, and maintenance
    b. Design, resource allocation, and coding
    c. Testing, maintenance, coding, deployment, and budgeting
    d. Requirements analysis, logical design, physical design, implementation, and maintenance
3. What is requirements gathering?
    a. The process of gathering specifications from the client to determine requirements for the project
    b. The process of designing the code
    c. The process of implementing the code
    d. The process of designing the software
4. What DBLC stands for?
    a. Database life cycle.
    b. Database development life cycle.
    c. Data base lifes cycle
    d. Database develop life cycle.
5. An entity-relationship (ER) in which phase of the DBLC?
    a. In the analysis phase.
    b. In the maintenance phase.
    c. In the logical design phase.
    d. In the implementation phase.

Part II. *Directions*. Explain briefly of the following statement.

1. What are the main functions of the Database Management Systems (DBMS)?

_____
_____

2. Discuss the difference between Centralized and Decentralized Design?

_____
_____

UNIT II. Data Modeling Advanced Concepts

Learning Objectives:

At the end of the unit, I will be able to:

1. demonstrate understanding of the different data model for a specific system/application; and
2. learn how to create an ERD and normalize the database.

Data Models

The hierarchical data model is a tree-like data model that stores data in a hierarchic structure. Each record-type is represented by a node in the tree.

Network data model is the advancement of a hierarchical data model. Datum in a network data model may have one or many ' ancestors. Having complicated relations and a need for a set of computers having large storage capacity are two of the main disadvantages of a Network data model.

Relational data model is a model composed of data-sets that may have different types but also contain fields that may be related to other tables. These fields serve as a link to another table to form a relation.

Object-oriented data model follows the concepts of the use of object-oriented technology in database creation.

The Three Factors of the Data Model

The data model is composed of three elements: ***entity, property, and relation.*** All of these are an equal part of a data model. That is, none of each element is greater than the other.

- Each entity has its properties which are also known as internal structure.
- Entities may have relations to other entity which is called the external structure of the entities.

Enhanced ER Model (EER)

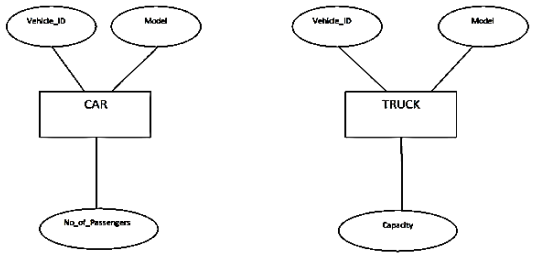
- It can also be called as an ***extended ER model*** as it is an outcome of an ER model being extended and incorporated with the new modeling constructs.
- The supertype/subtype relationship is the most significant modeling construct that is fused in the EER.

EER Features

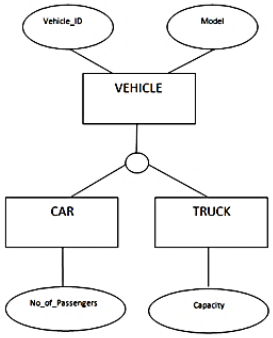
1. Generalization

It is the process of determining entities with common attributes to form a more general entity type and is also known as a ***supertype entity.***

EXAMPLE (without generalization)



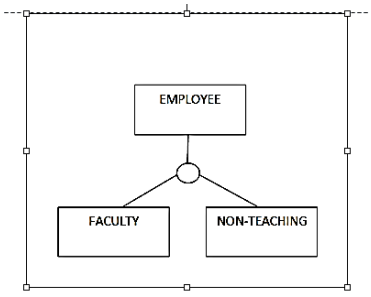
EXAMPLE (Generalization to a supertype)



2. Specialization

It is the process of determining one or more subtype from the supertype to create a supertype or a subtype relationship.

Example: In a school setting, for example, instead of just generalizing the employee into one entity, we can narrow it down to subtypes which may contain other attributes exclusive to that subtype only.



Supertype / Subtype Relationship

Allows us the identification of the general entities (subtype) and their subdivision into specialized parts (subtype).

- **Supertype**
  It is a general entity type that may be related to a single or more subtype.

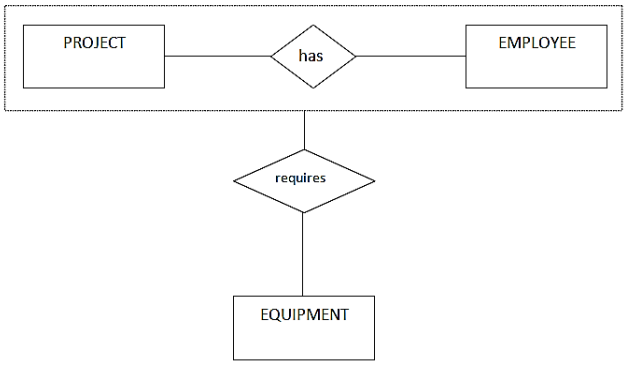
  Examples:
  STUDENT
  VEHICLE
- **Subtype**
  It is a more specific subgrouping that shares attributes with the supertype.

  Examples:
  1. A general entity: STUDENT
     Subtype: GRADUATE and UNDERGRADUATE
  2. A general entity: VEHICLE
     Subtype: CAR, MOTORCYCLE, Truck, Bus

3. Aggregation

It is when two entities having relation is treated as a sole entity, and also sometimes called the ***"Has a" relationship***.

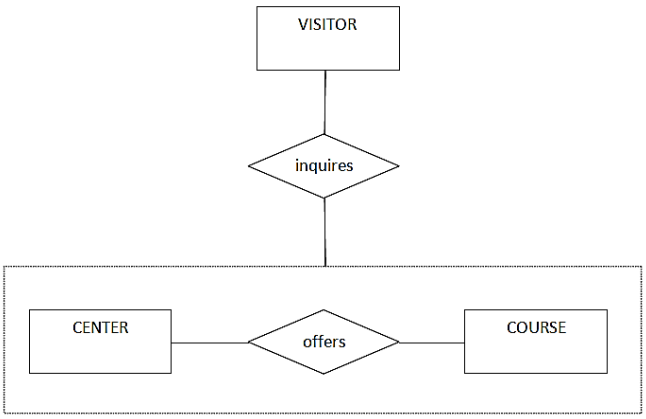
Example No.1:



Explanation:

This case implies that if a project requires an equipment, there is an employee working on a project that will use the equipment. Hence, project and employee is considered as an aggregation.

Example No.2



Explanation:

Normally, when a visitor goes to a training center to inquire, it may not only be about the center but also about its courses.

4. Composition

It is also known as a ***special or restricted aggregation*** where in the existence of one entity is dependent on the existence of another.

Example:

- A class contains students
- Library contains books

Aggregation vs. Composition

Imagine an everyday scenario in a school library.

Library and Student = **Aggregation**

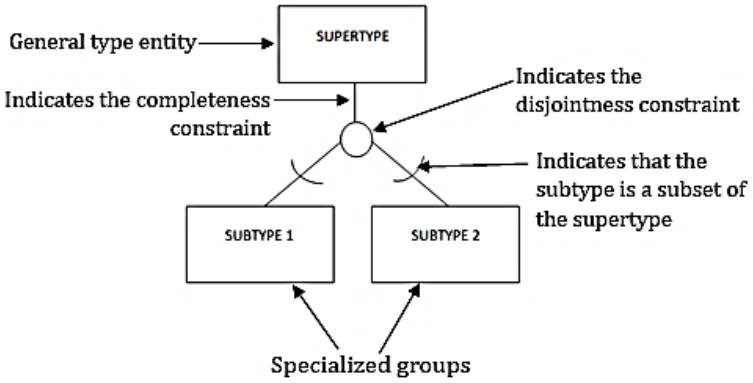
Because without Student, a Library can still exist

Library and Books = **Composition**

But library without books cannot exist

Basic Concepts & Notation

Representation in EER



EER constraints

1. Completeness constraints

It is a type of constraint used to determine or specify if an occurrence of an entity supertype can also be included to least one subtype.

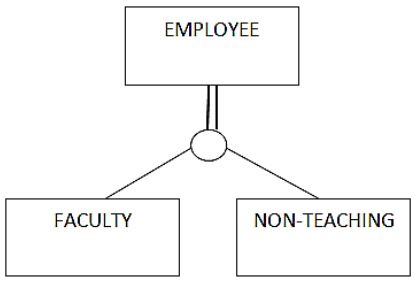
Two rules:

- ***Total Specialization Rule***

  It demands that for each instance of an entity in supertype should be a part of a subtype in the relation. It can be represented using a double line that interconnects supertype to its subtypes.

  Example:

  This one denotes that the EMPLOYEE type can only be either of the two.

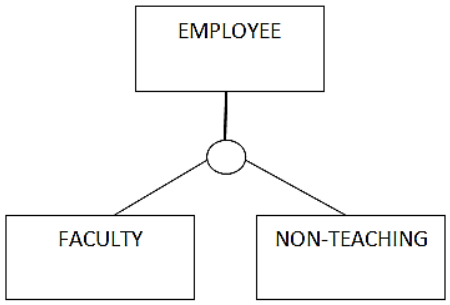


- ***Partial Specialization Rule***

Unlike *Specialization Rule*, ***Partial Specialization*** states that an occurrence of an entity in supertype is permissible not to belong to any subtype. It is represented using a single line to connect the supertype to its subtype.

Example:

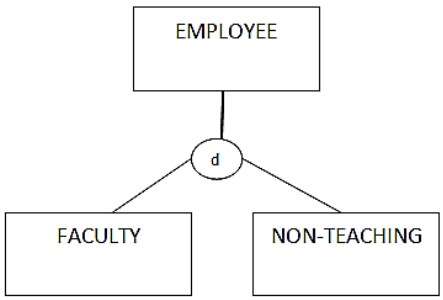
This denotes that the EMPLOYEE type can belong to either of the two subtypes or can also be not.



2. Disjointness Constraint

It denotes an "either or" scenario wherein the supertype can belong to one of the subtypes but cannot belong to others. A letter **"d"** inside a circle denotes disjointness.

Example:



- ***Overlap Rule***

State that a single entity CAN be a part of two or more subtypes. A letter "o" inside a circle denotes an overlap rule.

Database Normalization

What is Normalization?

Normalization is a process of structuring attributes on a relational database to eliminate data repetition/redundancy. On the other hand, ***denormalization*** produces a lower normal form that may increase performance but has greater redundancy.

Objectives of Normalization

1. To avoid anomalies in insertion, modification, and deletion.
2. To reduce restructuring of tables.
3. To make the models more informative.
4. To make a neutral query of the collection of relations despite the fluidity of data.

Advantages of Normalization

1. It will guarantee faster maintenance because of the fewer indexes.
2. Normalization removes the redundancy of data which results in a smaller size of the database.
3. As mentioned above, normalization results in a smaller database size which then has an outcome of faster response time.
4. Normalized tables can allow more data storage because redundant columns are removed to narrow the table.
5. Also, by narrowing of tables, we can easily specify which tables are to be joined.

The Normalization Process

- ***Transitive dependency***

  It exists when there is an indirect relationship between the two columns. ***For example***: if column Q is functionally dependent on column W (Q→W), and column W is functionally dependent on column E (W→E), then, we can say that Qcisa transitive dependency of E (Q→E).

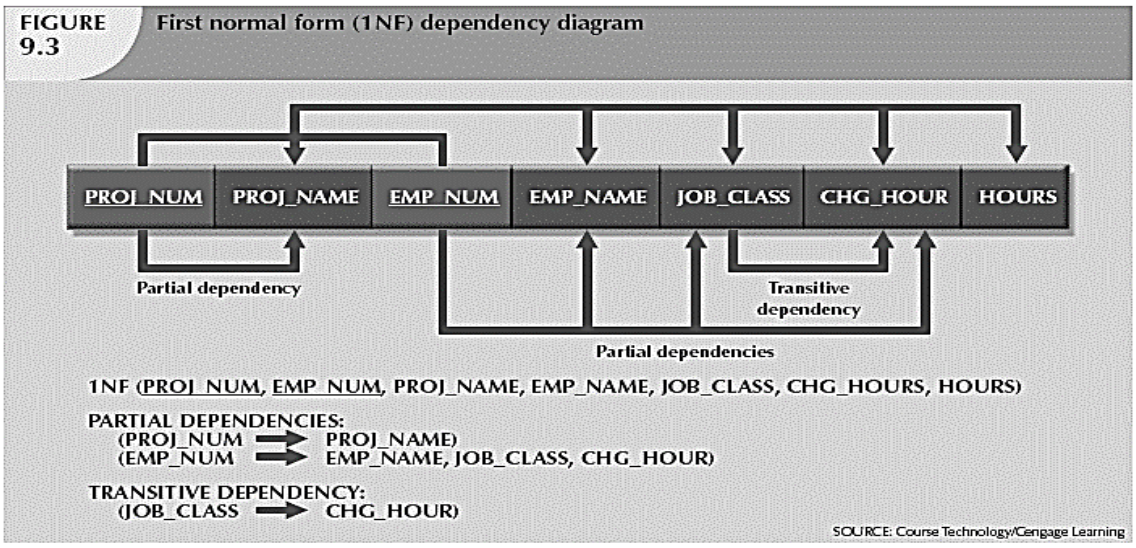
- ***Partial dependency***

  When the determinant of a functional dependency is only an attribute or a part of the primary key.

Dependency Diagram

Dependency diagram:

- Dependencies that are determined in a table structure are illustrated using a Dependency Diagram
- It also provides an overview of relationships that are existing on a table.
- Moreover, it would reduce the risk of having an important dependency being overlooked.



Steps in Normalization

First Normal Form (1NF)

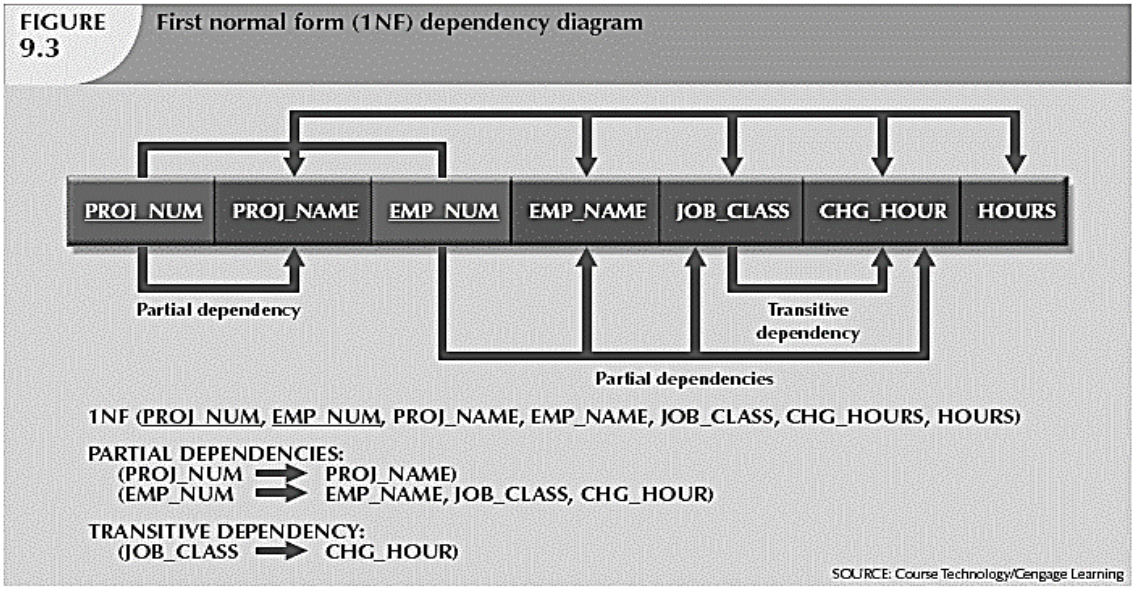
- Repeating groups happens when a single key attribute contains data of multiple entries with the same type.
- Removing repeating groups will reduce redundancies on a database

Steps to First Normal Form (1NF)

- **Step 1.** All repeating groups must be removed
  - No single key attribute containing multiple entries of data with the same type.

- **Step 2**. A Primary key must be determined
  - Identify a unique key or attribute on a given table
  - We can create a unique key for a table

- **Step 3.** All dependencies in a table must be identified
  - Determine all dependencies that exist in the table



Given below is an example of how the renting of tools on a particular shop is being recorded.

1NF

Date	Name	Address	Tool	Category	Price
05.02.1997	Jhonny Aguilar Martin Luna	Eper street 5. Nap street 3.	polisher welding-machine	small medium	500 1000
06.02.1997	Jhonny Aguilar Susan Moreno Martin Luna	Eper street 5. Fö street 1. Nap street 3.	paint-sprayer lawnmower chainsaw	medium big big	1000 2000 2000

From the example above, some columns contain multiple values. For this reason, we cannot call it a relation.

If we fill out dates for every tool rented, it will be like it is in 1NF. However, it will cause too much redundancy since similar dates will appear multiple times. It will irregularities or what we called ***anomalies:***

- ***Deletion anomaly.*** We might want to delete a single tool on a specified date, but some dates appeared more than once which may lead to unintentional deletion of certain data.

- ***Modification anomaly.*** As a result of data redundancy, changing the value on a specific row might cause unwanted change on values on different rows or columns.

- ***Insertion anomaly.*** User cannot fill out all the fields when adding new data

Characteristics of First Normal Form (1NF)

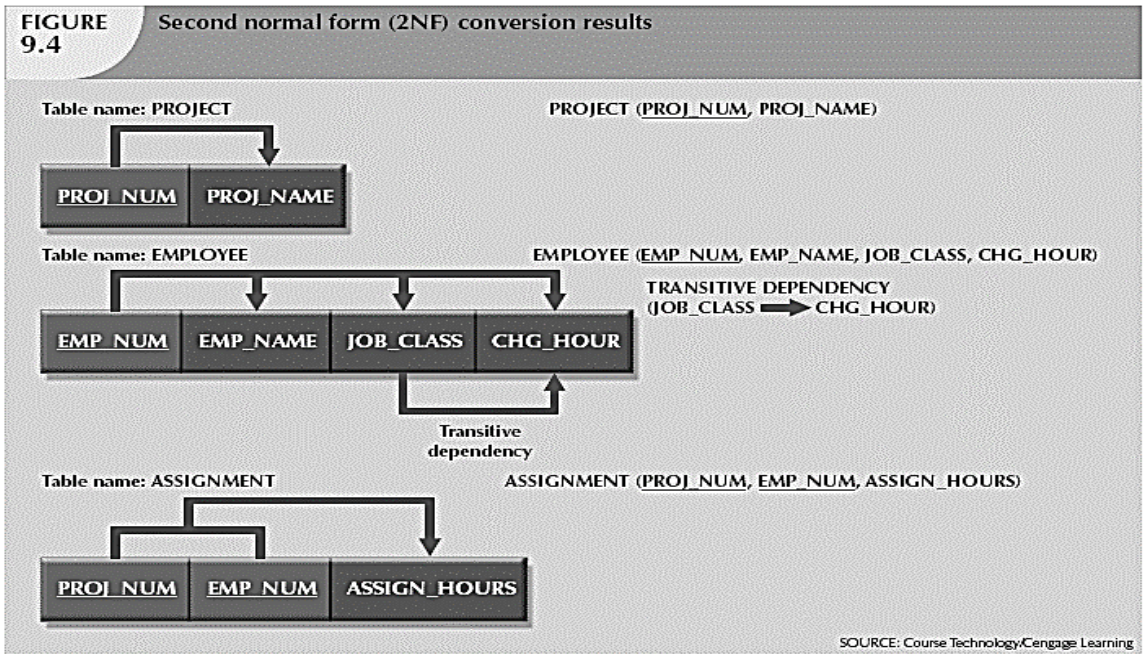
- Key attributes are determined
- Repeating groups are removed
- A primary key where all other attributes are dependent is specified
- Partial dependencies may exist in some tables
    - *Partial dependencies* exist when a dependency is based on a part of a primary key.
    - You must be careful in using partial dependencies.

Second Normal Form (2NF)

Steps to Second Normal Form (2NF)

- **Step 1**. Removing of partial dependencies by creating new tables
    - Specify key components and place them on a separate column.
    - Designate each key component as the primary key to new tables.

- **Step 2.** Specify attributes that are dependent on other attributes.
    - Most anomalies are removed in the 2NF



Below is a sample of a database in 2NF wherein it shows three individual tables with attributes with a partial dependency of the primary key.

1. Table

Serial number	Date	Name	Tool

2. Table

Tool	Category	Price

3. Table

Name	Address

Although it is in 2NF, there are still some anomalies that may exist. For the given example above, we might have a deletion anomaly specifically in table 2. Wherein deletion of a single tool may cause the deletion of category and price.

Characteristics of Second Normal Form (2NF)

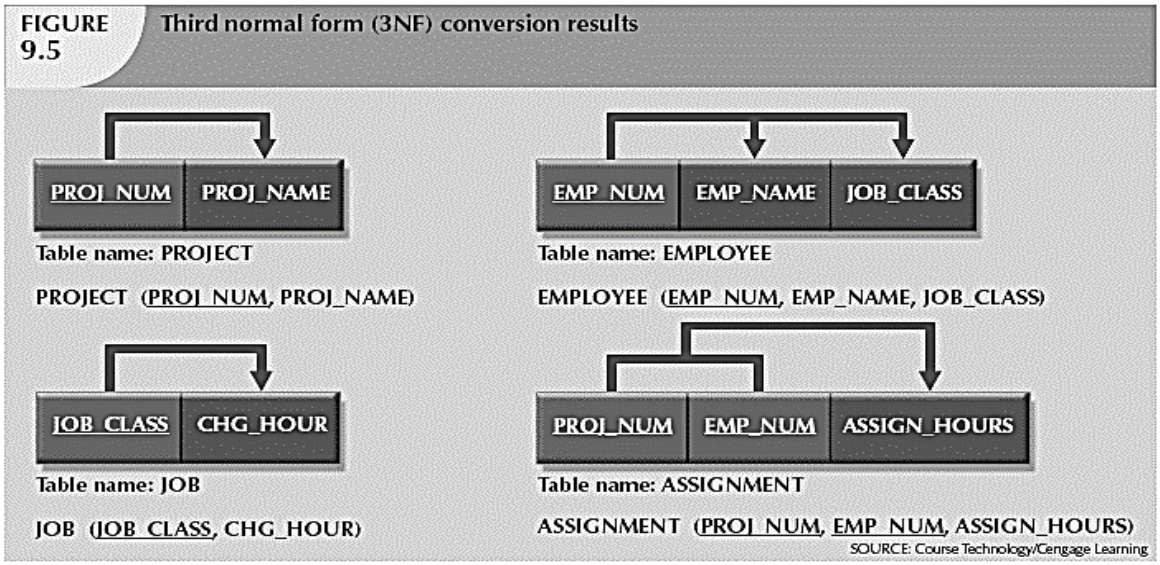
- It MUST be in 1NF
- All attributes are dependent on the primary key.
- Partial dependencies may exist in some tables
    - Partial dependencies exist when a dependency is based on a part of a primary key.
    - You must be careful in using partial dependencies.

Third Normal Form (3NF)

Steps to Third Normal Form (3NF)

- **Step 1.** Remove all transitive dependencies
    - For every transitive dependency, we can write PK for its determinant
    - A *determinant* is an attribute that may be used to determine the value of another row in a table.

- **Step 2**. Reassign dependency of attributes
    - The goal is to determine dependencies in a table



Below is an example of 3NF. As mentioned in the previous example on 2NF, table 2 may have a deletion anomaly, which is then addressed by managing the table to create two separate tables.

2. a. Table

Tool	Category

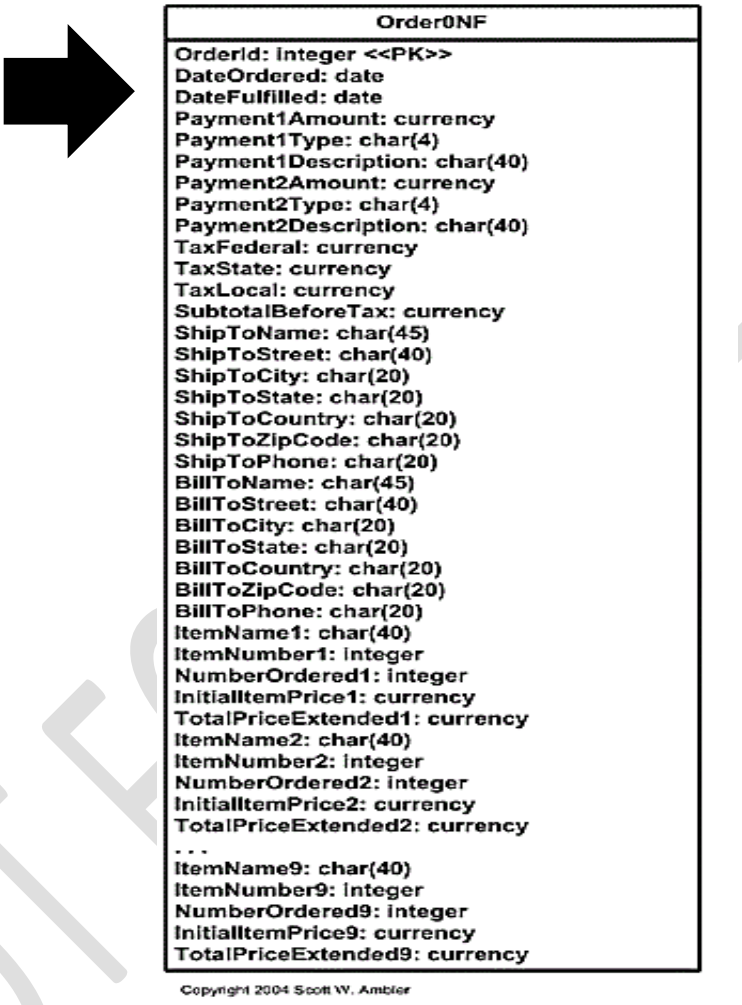
2. b. Table

Category	Price

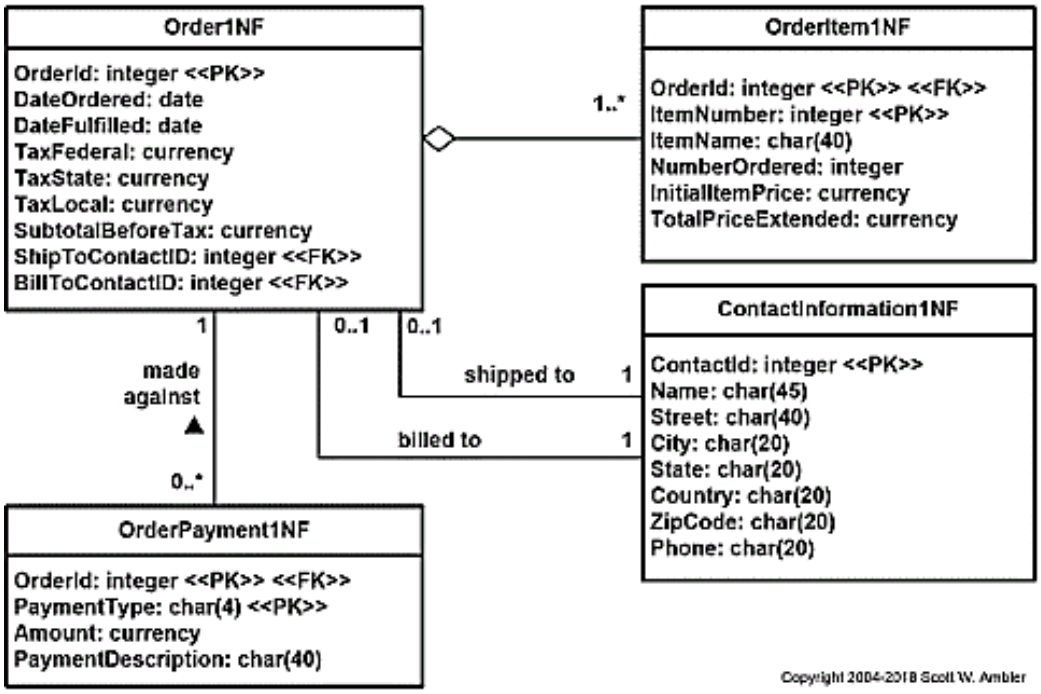
On the previous examples given, we may assign “Serial Number” as a super key. For the reason that it can be used to determine other rows that are part of the relation. All four rows are part of a relation; therefore, they must be used together. To make it possible there must be a field that would interconnect each table. That field is called the relationship field or what we know as the foreign key.

Here is the example of Normalization Form in UML (Unified Modeling Language)

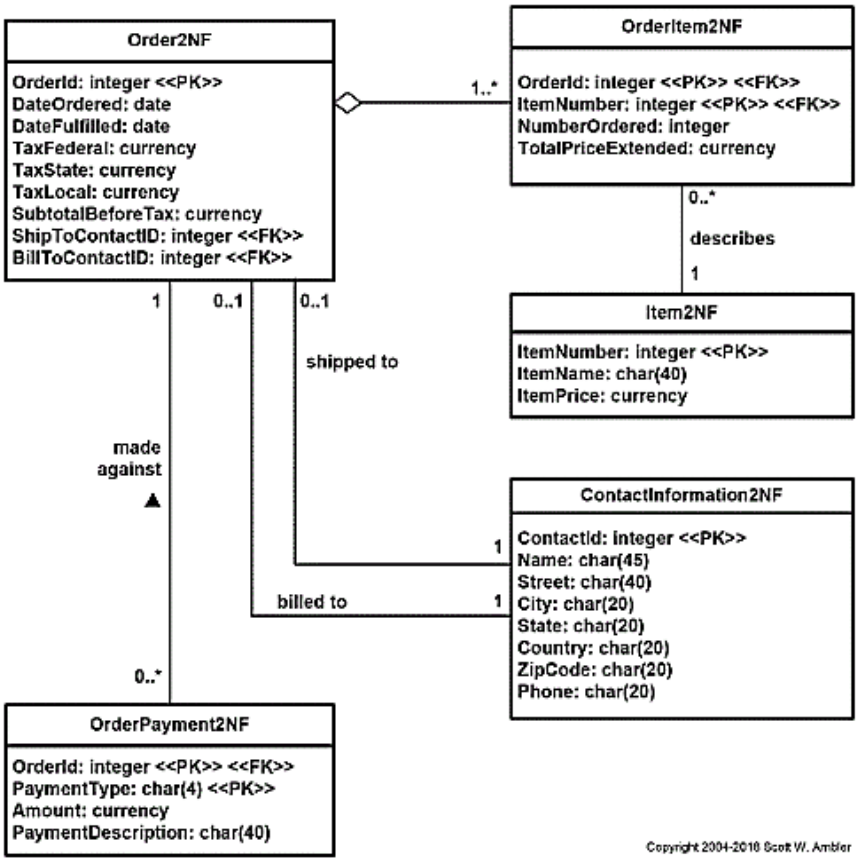
UnNormalized sample



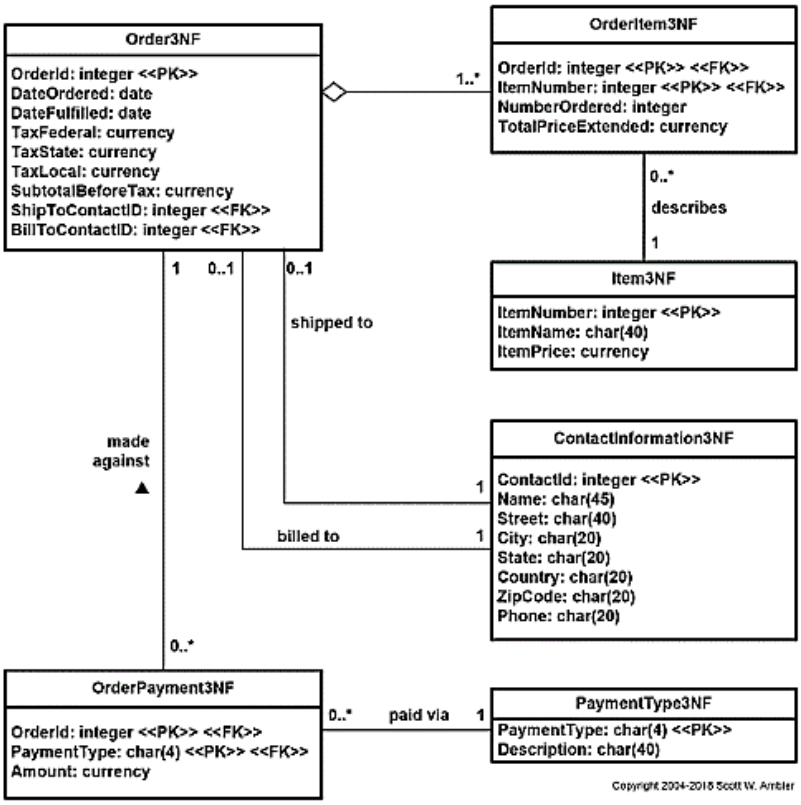
First Normal Form (1NF)



Second Normal Form (2NF)



Third Normal Form (3NF)



NoSQL and NewSQL databases

NoSQL databases features document-oriented databases and fast key-value stores. An example of a structured document-oriented database is an XML database that allows XML document attributes to be queried. NoSQL databases are the type of database that does not require a fixed table. Also, it stores denormalized data to avoid join operations. Some of the NoSQL systems in the market today are:

- MongoDB;
- Oracle NoSQL Database;
- CouchDB;
- Hazelcast;
- Apache Cassandra; and
- HBase.

****Note that all are open-source software products.

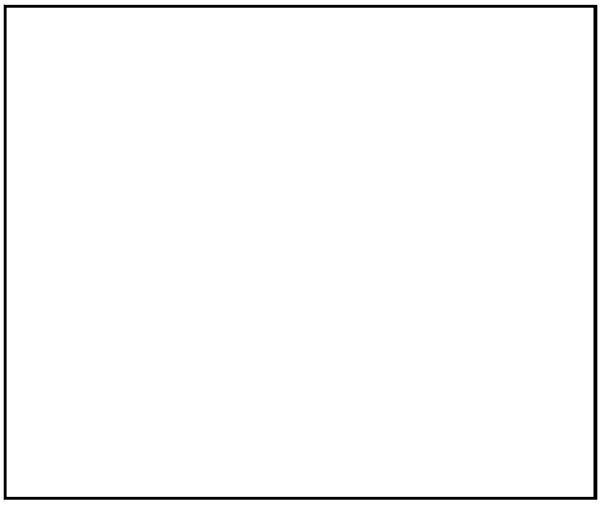
Assessing Learning

Name: _____ **Date: _____**
Section: _____

Part I. *Directions:* Draw a simple diagram – one supertype and two subtypes that show the use of the different EER rules and constraints. Add a brief explanation per answer (one to two sentences will do).

1. Partial specialization rule

Diagram:



Explanation:

_____

_____

_____

_____

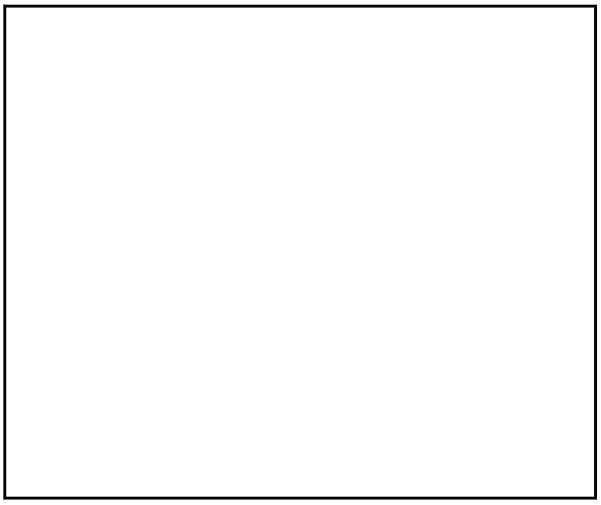
_____

_____

_____

2. Total specialization rule

Diagram:



Explanation:

_____

_____

_____

_____

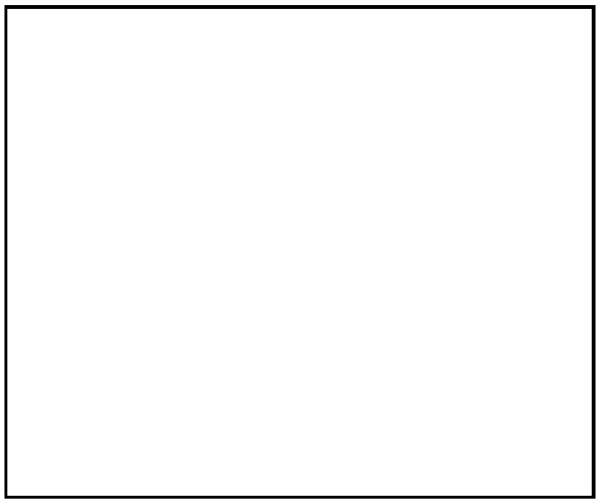
_____

_____

_____

3. Disjointness constraint / Disjoint rule

Diagram:



Explanation:

_____

_____

_____

_____

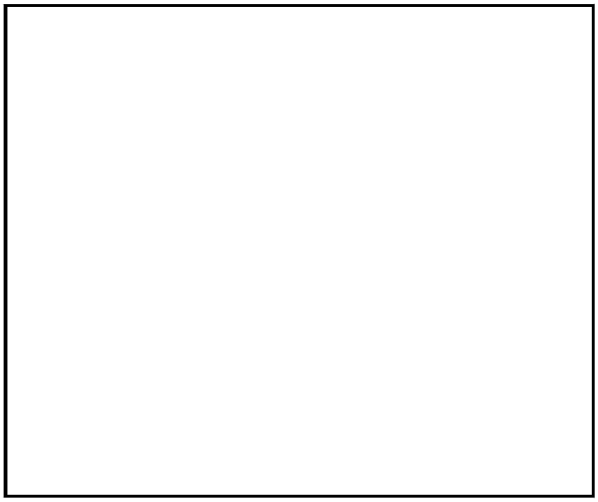
_____

_____

_____

4. Overlap constraint / Overlap rule

Diagram:



Explanation:

_____

_____

_____

_____

_____

_____

_____

Part II. Normalization Forms

Directions: Normalize the given table below from the first normal form until the third normal form.

Student NO	Student Name	Address	Course Name	Course Units	Department Name	Subject Teacher

UNIT III. Procedural Language SQL and Advanced SQLs

Learning Objectives:

At the end of the unit, I will be able to:

1. define and use the MySQL views;
2. determine how to make use of the SQL join operator;
3. determine how to install SQL server;
4. demonstrate understanding of the notion of client/server system;
5. determine how to create a SQL store procedure; and
6. determine the importance of database administration in the Database Management System.

Structured Query Language (SQL)

SQL or Structured Query Language is utilized for collecting, managing, and retrieving information stored in an electronic structured database.

SQL instructions standardized the language needed to create and execute the exclusive operations in a variety of applications and is a universally used database management system.

Why SQL?

SQL is extensively popular in managing facts due to the fact it provides the following advantages:

1. It allows access to statistics in relational database management systems.
2. It allows us to describe the data.
3. It allows us to outline the data manipulation in a Database.
4. It allows embedding within different languages using SQL modules, libraries & pre-compilers.
5. It allows us to create databases as well as tables and drop/delete it.
6. It allows us to create a view, stored procedure, features in a database.
7. It grants permissions on the collection of records or tables, as well as permissions in procedures, and views.

The figure below is a simple diagram of SQL architectures

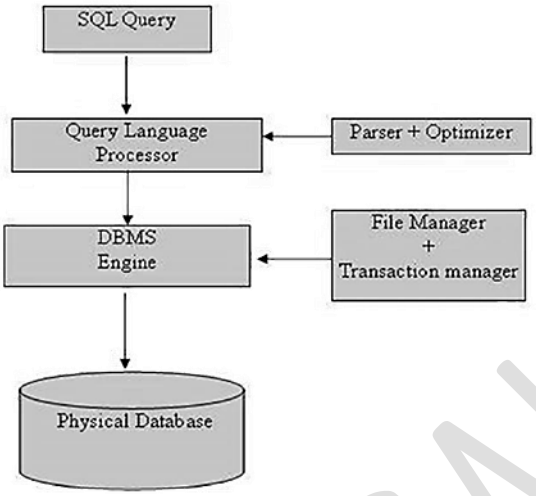


Image captured from https://www.tutorialspoint.com

The SQL Commands

There was a generic syntax for SQL commands that interact with relational databases. These are the CREATE statement, SELECT statement, the INSERT statement, UPDATE statement, DELETE statement, and DROP statement, which is classified into the following groups according to their needs.

1. **DDL - Data Definition Language**

Command	Description
CREATE	Create a new table object in the database.
ALTER	Modifyan existing database object, such as a table.
DROP	Drop ordelete an existing tableobject in the database.

2. **DML - Data Manipulation Language**

Command	Description
SELECT	Retrievecertain records from one or more tables.
INSERT	Create a record.
UPDATE	Modify a record.
DELETE	Delete a record.

3. DCL - Data Control Language

Command	Description
GRANT	Give privilege to the user.
REVOKE	Takes back privileges granted by the user.

The Structured Query Language Syntax

In all the SQL statements the numerous keyword begins with the **SELECT** statement, the **INSERT** statement, **UPDATE** statement, **DELETE** statement, **ALTER** statement, **DROP** statement, **CREATE** statement, **USE** statement, and **SHOW** statement and this will end with a semicolon (;). SQL is case insensitive and has the equal SQL statements you want to provide table names as it appears in the current database.

To further discussion of the SQL commands you are going to visit the following websites:

Command	Website
DML	https://www.tutorialspoint.com/sql/sql-syntax.htm
	https://sitenxt.com/sql/sql-syntax/
Operations	https://www.tutorialspoint.com/sql/pdf/sql-quick-guide.pdf https://congnghejava.blogspot.com/search/label/22.SQL
Joins	https://mafiadoc.com/download-sql-tutorial-pdf-version-tutorials-
Union and Intersect Clause	https://mafiadoc.com/download-sql-tutorial-pdf-version-tutorials-
Functions	https://www.tutorialspoint.com/sql/sql-syntax.htm

Here are the various SQL syntax statements

SQL SELECT Statement

```
SELECT column1, column2.... columnN
FROM source_table_name;
```

SQL DISTINCT Clause

```
SELECT DISTINCT column1, column2....columnN
FROM source_table_name;
```

SQL WHERE Clause

```
SELECT column1, column2....columnN
FROM source_table_name
WHERE CONDITION;
```

SQL AND/OR Clause

```
SELECT column1, column2....columnN
FROM source_table_name
WHERE CONDITION-1 {AND|OR} CONDITION-2;
```

SQL IN Clause

```
SELECT column1, column2....columnN
FROM source_table_name
WHERE column_name IN (Val-1, Val-2,...Val-N);
```

SQL BETWEEN Clause

```
SELECT column1, column2....columnN
FROM source_table_name
WHERE column_name BETWEEN Val-1 AND Val-2;
```

SQL LIKE Clause

SELECT column1, column2...columnN FROM source_table_name WHERE column_name LIKE {PATTERN };

SQL ORDER BY Clause

SELECT column1, column2....columnN FROM source_table_name WHERE CONDITION ORDER BY column_name {ASC|DESC};

SQL GROUP BY Clause

SELECT SUM(column_name) FROM source_table_name WHERE CONDITION GROUP BY column_name;

SQL COUNT Clause

SELECT COUNT(column_name) FROM source_table_name WHERE CONDITION;

SQL HAVING Clause

SELECT SUM(column_name) FROM source_table_name WHERE CONDITION GROUP BY column_name HAVING (arithematic function condition);

SQL CREATE TABLE Statement

CREATE TABLE source_table_name (column1 datatype, column2 datatype, column3 datatype ...
ColumnN datatype, PRIMARY KEY (one or more columns));

SQL DROP TABLE Statement

DROP TABLE source_table_name;

SQL CREATE INDEX Statement

CREATE UNIQUE INDEX index_name ON source_table_name (column1, column2,... columnN);

SQL DESC Statement

DESC source_table_name;

SQL TRUNCATE TABLE Statement

TRUNCATE TABLE source_table_name;

SQL ALTER TABLE Statement

ALTER TABLE source_table_name {ADD|DROP|MODIFY} column_name {data_ype};

SQL ALTER TABLE Statement (Rename)

ALTER TABLE source_table_name RENAME TO new_source_table_name;

SQL INSERT INTO Statement

INSERT INTO source_table_name (column1, column2...columnN) VALUES (value1, value2...valueN);

SQL UPDATE Statement

UPDATE source_table_name SET column1 = value1, column2 = value2...columnN=valueN [WHERE CONDITION];

SQL DROP DATABASE Statement

DROP DATABASE database_name;

SQL USE Statement

USE database_name;

SQL COMMIT Statement

COMMIT;

SQL ROLLBACK Statement

ROLLBACK;

SQL EXPRESSIONS

The ***SQL EXPRESSIONs*** are similar to a formula written in the query language. This particular expression can use to query a set of data in the database.

There are SQL Expressions according to their types

SQL Boolean Expressions

```
SELECT column1, column2, columnN
FROM source_table_name
WHERE SINGLE VALUE MATCHING EXPRESSION;
```

Here is a simple example showing the usage of SQL Boolean Expressions:

```
SQL>SELECT * FROM CUSTOMERS WHERE SALARY= 10000;
```

SQL Numeric Expressions

```
SELECT numerical_expression as OPERATION_NAME
[FROM source_table_name
WHERE CONDITION];
```

Here is a simple example showing the usage of SQL Number Expressions:

```
SQL>SELECT (15 + 6) AS ADDITION
```

SQL Date Expressions

```
SQL> SELECT CURRENT_TIMESTAMP;
```

Here is a simple example showing the usage of SQL Date Expressions:

```
SQL> SELECT CURRENT_TIMESTAMP;
```

The SQL AND and OR operators

The ***SQL Operator AND and OR*** is a condition that mixes multiple SQL statements help to narrow down the fetching of information from the Database. These operators are oftentimes named as conjunctive operators. The AND and OR operators provide measures to get various comparisons with other operators within the same SQL statement.

The AND operator

```
SELECT column1, column2, columnN
FROM source_table_name
WHERE [condition1] AND [condition2]...AND [condition];
```

Here is a simple example showing the usage of AND operator:

```
SQL> SELECT ID, NAME, SALARY
     FROM CUSTOMERS
     WHERE SALARY> 2000 AND age < 25;
```

The OR operator

```
SELECT column1, column2, columnN
FROM source_table_name
WHERE [condition1] OR [condition2]...OR [condition]
```

Here is a simple example showing the usage of the OR operator:

```
SQL> SELECT ID, NAME, SALARY
     FROM CUSTOMERS
     WHERE SALARY > 2000 OR age < 25;
```

The SQL Joins

A ***JOIN*** means connecting attributes into two entities through the use of values common to each.

Here is the example showing the usage of joining two tables.

```
SQL>  SELECT id, name, age, amount
FROM customers, orders
WHERE customers.id = orders.customer_id;
```

Here is another example: Suppose we want to join the relations

Movies (title, year, length, genre, studio_name, producerC#)
Starsln (movie_title, movie_year, star_name)

Explanation

In with the situation that the only tuples to be joined are those that refer to the identical movie. That is, the titles and years from both relations are similar. We can ask this query by

```
SQL> Movies JOIN Starsln ON title = movie_title AND year = movie_year;
```

Here are the six types of Joins

1. Inner;
2. Left ;
3. Right;
4. Full;
5. Self; and
6. Cartesian.

INNER JOIN

The frequently used take part SQL server is that the ***INNER JOIN***. This join renders a brand new table by connecting two tables (table1 and table2) based totally

upon the join-predicate. The query works to check each row of table1 with every row of table2 to seek out all rows that fulfill the join-predicate. And when the join-predicate is satisfied, column values for each matched pair of A and B rows are mixed into a consequence row.

```
syntax:
    SELECT table1.column1, table2.column2...
    FROM table1
    INNER JOIN table2 ON table1.common_field = table2.common_field;

Here is a simple example showing the usage of INNER JOIN.
SQL> SELECT ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    INNER JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

LEFT JOIN

This means that a left join returns all the values from the left table, and also coordinated values from the proper table or Invalid just in case of no coordinating connect predicate.

```
Syntax:
   SELECT table1.column1, table2.column2...
   FROM table1
   LEFT JOIN table2 ON table1.common_field = table2.common_field;

Here is a simple example showing the usage of the LEFT JOIN.
SQL> SELECT ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    LEFT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

RIGHT JOIN

This means that a right join returns all the values from the right table, and in addition, coordinated values from the cleared out table or Invalid just in case of no coordinating connect predicate.

```
Syntax:
    SELECT table1.column1, table2.column2...
    FROM table1
    RIGHT JOIN table2 ON table1.common_field = table2.common_field;

Here is a simple example showing the usage of the RIGHT JOIN.
SQL> SELECT ID, NAME, AMOUNT, DATE
    FROM CUSTOMERS
    RIGHT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

FULL JOIN

The joined table will carry all records from both tables, and fill in NULLs for those fields that have no value.

```
Syntax:
    SELECT table1.column1, table2.column2...
    FROM table1
    FULL JOIN table2 ON table1.common_filed = table2.common_field;

Here is a simple example showing the usage of the FULL JOIN.
SQL> SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
FULL JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

SELF JOIN

The ***SELF JOIN*** is a special kind of joins in which it allows an entity to join itself. This join is beneficial in comparing rows within one table.

```
Syntax:
      SELECT a.column_name, b.column_name...
      FROM table1 a, table1 b
      WHERE a.common_field = b.common_field;

Here is a simple example showing the usage of SELF JOIN.
SQL> SELECT a.ID, b.NAME, a.SALARY
      FROM CUSTOMERS a, CUSTOMERS b
      WHERE a.SALARY < b.SALARY;
```

CARTESIAN JOIN

Returns all the rows in all the tables query. Each row of distinctive tables combined and assesses to true.

```
Syntax:
        SELECT
          table1.column1, table2.column2...
        FROM
          table1, table2 [, table3]

Here is a simple example showing the usage of CARTESIAN JOIN.
SQL> SELECT ID, NAME, AMOUNT, DATE
      FROM CUSTOMERS, ORDERS;

SQL> SELECT ID, NAME, AMOUNT, DATE
      FROM CUSTOMERS CROSS JOIN ORDERS;
```

SQL in a Server Environment

a. SQL terminology for client-server computing and connecting to a database.
b. SQL must be used to access a database as part of a typical application.
c. SQL relations and the variables of the surrounding, or "host," language.

The architecture called "three-tier "or "three-layer" because it distinguishes three different, interacting functions:

1. **Web Servers.** These can refer to *hardware* and s*oftware*. These are processes that connect clients to the database system, usually over the Internet or possibly a local connection.

2. **Application Servers.** These provide a wide variety of access to the data on the network. These processes operate the "business logic," whatever the system intends to do.

3. ***Database Servers***. These access a large number of databases in one server. These processes run the DBMS and perform queries and modifications at the request of the application servers.

The Web-Server Tier

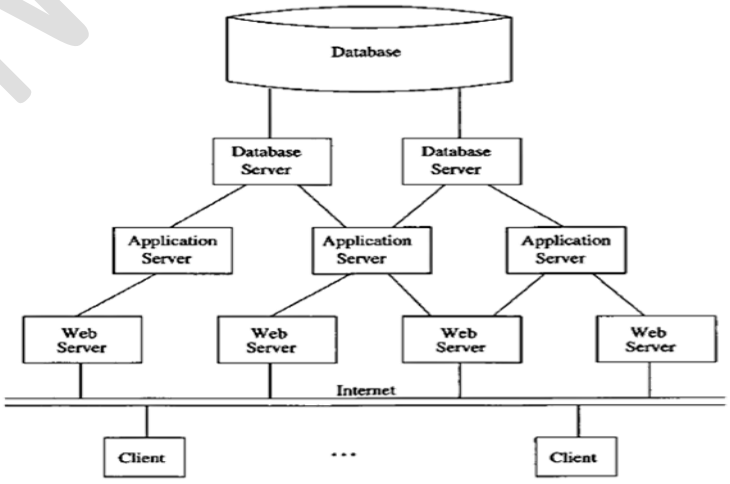
The webserver processes manage the interactions with the user. When a user makes contact, perhaps by opening a URL, a web server, typically running Apache, responds to the request. The user then becomes a client of this webserver process.

Usually, the web server performed the client's actions, such as managing the filling of forms, which later posted to the webserver.

Clients and Servers in the SQL Environment

A SQL environment is more than a collection of catalogs and schemas. It contains elements whose purpose is to support operations on the database.

"SQL server" plays the role of what we called a "database server there. A "SQL client" is like the application servers from that Figure below. The SQL standard does not define processes analogous to "Web servers" or "clients."



The Three-Tier Architecture
Image captured from Database Systems The Complete Book Second Edition

Stored procedure

A ***stored procedure*** is a subroutine accessible to functions that get the right of entry to a relational database system. The database data dictionary saves the stored procedure (sometimes known as a *proc, sproc, StoPro, StoredProc, sp, or SP*).

The important functions of stored procedure

1. The traditional use of stored procedures consists of data validation.
2. Access manipulate mechanisms.
3. Consolidate and centralize logic that initially implemented.
4. Extensive or complicated processing requires the execution of quite a few SQL statements moved into saved procedures, and all functions call the methods.
5. Nested stored procedures through executing one stored procedure from with each other.

Advantages of Stored Procedure to dynamic SQL

There are *four fundamental benefits* of store procedure with the aid of using queries or enhancing a file in the database:

1. **Overhead**

   Stored procedure declaration is saved straight to the database, and they will additionally suspend all or phases of the compilation overhead typically required in conditions the place software functions send inline (dynamic) SQL queries to a database.

2. **Avoidance of network traffic**

   They can run immediately within the database engine. Typically, it is potential that the techniques completely run on a specialized database server during a manufacturing system, which has direct get right of entry to access data.

3. **Encapsulation of business logic**

   Stored processes permit programmers to embed business logic as an Application Programming Interface (API) within the database, simplifying records management, and lowering the need to transform into coded form logic someplace else in customer programs.

4. **Delegation of access-rights**

   Stored procedures will not protect against SQL injection attacks.

Document-oriented database or so-called NoSQL

A ***document-oriented database*** could be a computer software designed to store, retrieve, and manage document-oriented information, and also considered semi-structured data. Document-oriented databases are one amongst the basic classes of so-called NoSQL databases, and also the recognition of the fundamental measure "document-oriented database" (or "document store") has grown with the employment of the fundamental measure NoSQL itself. In distinction to prevalent relational databases and their notions of "Relations" (or "Tables"), these structures are designed around a summary concept of a "Document."

Documents in a document-oriented database are similar, in some ways, to records or rows in relational databases, however, they're much less rigid. They're not required to stick to a preferred schema, nor will they need all the identical sections, slots, parts, or keys. For instance, the subsequent may be a document:

```
{
        FirstName: "Bob",
        Address: "5 Oak St.",
        Hobby: "sailing"
}
```

A second document might be:

```
{
        FirstName: "Jonathan",
        Address: "15 Wanamassa Point Road",
        Children: [
                {Name: "Michael", Age: 10},
                {Name: "Jennifer", Age: 8},
                {Name: "Samantha", Age: 5},
                {Name: "Elena", Age: 2} ]
}
```

Explanation:

The above documents distribute the same structural elements, however, each one represents a unique element. Not like with the relational database where each record consists of the identical fields, leaving unused fields empty. In the example given above, there are no empty 'fields' in both documents (a record). This strategy approves new facts to be brought to some data, barring, and requires every other file in the Database to share the same structure.

Assessing Learning

Name: _____ Date: _____
Section: _____

Directions: Able to apply SQL commands to create database and table structure, and that will be based on the given problem and data dictionary below:

1. Create a database NEUSTCICTdB.
2. Create 6 tables from the data dictionary.
3. Add one column StudTelNo in the STUDENT table and the data type is INT (9).
4. Alter data type of CourseName column for COURSE table to VARCHAR(30).
5. Drop table Stud_Sub from the database.

Problem: A database named NEUSTCICTdB will be developed by means of an application software for House Company. You will provide a relationship scheme for the 6 tables is as below:

1. Student(StudID, StudName,StudAddress,StudBirthDate,CourseID);
2. Course(CourseID,CourseName,LectID);
3. Lecturer(LectID, LectName,LectTelNo,DepartID);
4. Subject(SubID, SubName);
5. Department(DepartID, DepartName);
6. Stud_Sub (StudSubID,StudID,SubID, Mark,Grade)
7. (Primary Key showed by underlined and bold while in italic and dotted lined is Foreign Key)

Data Dictionary

Tables Name	Attributes Name	Data Types	PK/FK	FK refer to
STUDENT	StudID	VARCHAR(5)	PK	
	StudName	VARCHAR(20)		
	StudAddress	VARCHAR(30)		
	StudBirthDate	DATE		
	CourseID	VARCHAR(5)	FK	COURSE
COURSE	CourseID	VARCHAR(5)	PK	
	CourseName	VARCHAR(10)		
	LectID	VARCHAR(5)	FK	LECTURER
LECTURER	LectID	VARCHAR(5)	PK	
	LectName	VARCHAR(20)		
	LectTelNo	INT(10)		
	DepartID	VARCHAR(5)	FK	DEPARTMENT
DEPARTMENT	DepartID	VARCHAR(5)	PK	
	DepartName	VARCHAR(30)		
SUBJECT	SubID	VARCHAR(5)	PK	
	SubName	VARCHAR(20)		
STUD_SUB	StudSubID	VARCHAR(5)	PK	
	StudID	VARCHAR(5)	FK	STUDENT
	SubID	VARCHAR(5)	FK	SUBJECT
	Mark	INT(3)		
	Grade	CHAR(2)		

UNIT IV. Data Mining and Warehousing

Learning Objectives:

At the end of the unit, I will be able to:

1. demonstrate understanding of the different data mining and warehousing techniques that are beneficial to in supporting decision making for organizations; and
2. determine the specific algorithm as be applied to data warehousing and mining.

Data Mining

From the term "mining" on its name, ***data mining*** entails searching for valuable information and discovering patterns in a given dataset that can be interpreted and used to support the decision-making process. It uses various methods from different disciplines, such as database systems, statistics, artificial intelligence, and machine learning.

Major data mining capabilities include: automated pattern recognition, prediction of outcomes, creation of actionable insight; and ability to manage any size of data.

Scope of Data Mining

Various sectors such as business, education, healthcare, government, and agriculture benefit from the data mining process. Its main scope is to sift through a given dataset to find new patterns and possible associations. Through its knowledge-discovery capabilities, data mining can do the following:

a. **Prediction of trends and behaviors**. Data mining can find and calculate predictions based on given data. Some examples of such are the prediction of infection cases based on the healthcare reports, prediction of responses on given situations based on population segments, and the likelihood of sales based on buyer responses.

b. **Discovery of previously unknown patterns**. Data mining can identify previously undiscovered trends by establishing possible links between attributes. Some examples of pattern discovery are detecting behaviors that can result in fraud, which can be useful for banks and lending companies and identify unrelated items often bought together, which can be helpful in marketing and sales.

Data Mining Tasks

The data mining process involves ***six common task categories:***

1. **Anomaly detection**. Identifies unusual objects and events or outliers that are suspiciously different from the majority. This task is typically useful in fraud detection, network intrusion, and identification of outliers.

2. **Association rule learning**. Looks for links and relationships between entities and attributes.

3. **Clustering.** Groups together abstract objects that have similarities then assigning labels to each cluster formed. Clustering is commonly used in pattern recognition and image processing.

4. **Classification.** Involves the task of observing existing observable structures then applies them to new data to be able to classify them on predefined classes. One best example of the practical use of the classification technique is for prediction.

5. **Regression.** Finds the best fit model with the least error and is commonly used to predict numerical values.

6. **Summarization.** Provides data visualization, graphs, and report generation for a more detailed, yet summarized result presentation.

Data Mining System Architecture

A classic data mining system is comprised of the following major components: ***user interface, pattern evaluation, data mining engine, and knowledge base.***

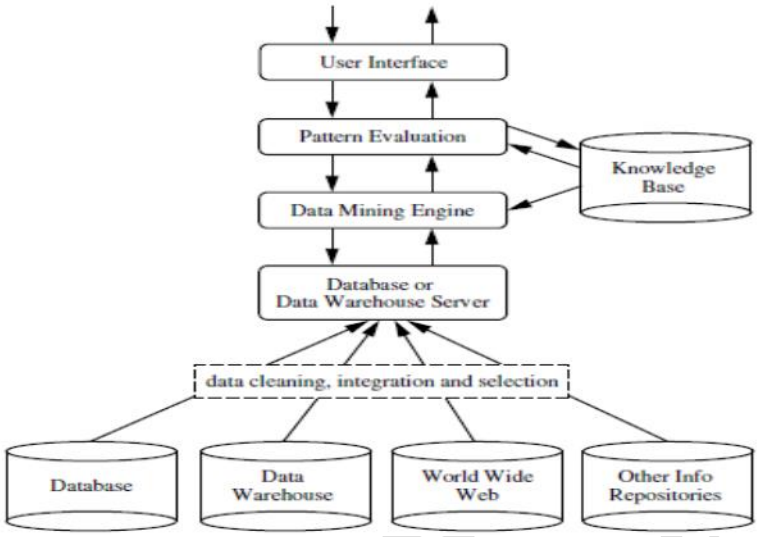
1. **Knowledge Base**
   This is a knowledge repository that can be used as guides for finding and evaluating query results and patterns.

2. **Data Mining Engine**
   This involves the essential functional modules that are responsible for characterization, association, analysis, and prediction.

3. **Pattern Evaluation**
   This component uses measures based on the interest of the users to discover new patterns and filter known ones.

4. **User interface**

The user interface promotes interaction and communication between the users and the system. It also helps to facilitate searching and browsing of data and viewing the extracted knowledge and results.



Captured image from https://www.zentut.com/data-mining/advantages-and-disadvantages-of-data-mining/

Data Mining Process

Data mining is a method of finding various models, summaries, and values extracted from a given data set. It follows the general experimental approach that consists of the following steps:

1. **Statement of the problem and hypothesis**
   Identifying the problem and hypothesis formulation can help set the domain and scope of the data mining process.

2. **Data collection**
   With the established domain, this step focuses on what data needs to be collected and how.

3. **Data pre-processing**
   Data pre-processing is concerned with "cleaning" the collected data. It is usually done by means though various activities such as removing outliers, scaling, feature selection, formatting, conversion, and encoding.

4. **Model building and implementation**
   Once the data is cleansed and formatted, the model can be built using the selected features and run to get the desired output.
5. **Result formulation and interpretation**

Outputs from the data mining activity are presented in an interpretable manner where conclusions can be drawn to validate the hypothesis and aid in the decision making process.

Challenges

The array of knowledge requirement. Different users may request different information. Therefore, to cater to each user's needs, the data mining process must be able to cover and handle a wide range of knowledge.

1. **The interactivity of the process.** More straightforward navigation and interaction can allow users to be more focused on pattern searching, knowledge discovery, and result interpretation.

2. **The requirement of background knowledge**. Background knowledge of data mining techniques and algorithms is essential to better design and implement the models to produce more satisfactory results.

3. **The manner of output presentation and visualization**. The delivery of results must be comprehensive and easy to understand.

4. **The handling of data irregularities.** Data must be "cleaned" or pre-processed to increase the accuracy and integrity of the results.

5. **The validity of pattern evaluation**. The patterns discovered must be something new and worth exploring. Otherwise, it defies he purpose of the exploration.

6. **The efficiency, flexibility, and capacity of the algorithms.** The algorithm's scalability and effectiveness in dealing with various amounts of data are essential because of their impact on the overall performance and results. Factors such as database size, data distribution, and method complexity are also worth considering because the process adjusts based on the scale of the need.

Advantages of Data Mining

Since the key benefit of data mining is the availability of knowledge that can be utilized in making decisions, below are practical application of data mining for certain industries:

Data mining can aid in marketing research by providing insights on market trends, consumer behavior, sales, and other factors that can help establish strategies and targets.

1. Data mining can search through historical data to detect fraudulent transactions and credit standing that can be beneficial for financial institutions.

2. Data mining can help detect product defects and predict parameters that can be applied on product quality improvement.

3. Data mining can provide comparative information to detect criminal activities, fraud, and corruption.

Disadvantages of Data Mining

1. **Privacy Issues.** Data collection may be challenging as consideration for both personal privacy and data privacy laws should be considered.

2. **Security issues.** Databases and data warehouses should be highly secured as the information they contain may be compromised by hackers and unauthorized access.

3. **Prone to information misuse and inaccuracy**. Data should be safeguarded and should be used with utmost ethical responsibility. Moreover, because the data mining results are used for strategic decision making, data consistency and accuracy should be observed.

Data Warehouse

A ***data warehouse*** is a consolidation of non-volatile and subject-oriented data from extracted various databases. The ***data warehouse architecture's idea*** is that data from various sources are extracted and combined in one global repository. For the users, a data warehouse can look like an ordinary database wherein they can perform queries on how they usually do on a typical database.

Data Warehouse Models

1. **Enterprise warehouse**. An enterprise warehouse has a cross functional scope stores organization-wide information. It involves integration of data from one or more functional systems and information sources. It is an extensive data warehouse model that requires time to build and implement on mainframes, super serves, and parallel architecture platforms.

2. **Data mart.** A data mart consists of data that are beneficial to a specific user group. The scope is limited to selected users and the data contained are usually summarized.

3. **Virtual warehouse.** A virtual warehouse is a collection of views made for faster processing. It is easy to build but offers limited summaries of data.

Advantages of a Data Warehouse

1. **Fast data retrieval.** Data in a data warehouse are non-volatile, making it available anytime it needs to be accessed.

2. **Efficient error detection and correction.** Data warehouses can detect inconsistencies during data loading, thus enabling the correction to avoid inaccuracy on data entries.

3. **Easy Integration.** Despite its complex nature, the data warehouse is capable of providing simplified information to make it easier for users to understand.

Disadvantages of a Data Warehouse

1. **Time Consuming.** It requires time to input and loads raw data into the warehouse. The more extensive data an organization deals with, the more time it requires.

2. **Compatibility Issues.** Compatibility with operating systems, software packages, and applications should be considered when planning to design and use one to ensure the system is working as intended and to make the most out of the data warehouse technology.

3. **Maintenance Expenses.** Constant updates are needed to optimize the performance of the data warehouse. However, the more complex and intricate the features entail a more expensive upgrade.

4. **Limited Use Due to Confidential Information.** Access levels are needed when dealing with sensitive data, as only selected individuals are allowed to view them. But, as access limitation increases, the overall value of the data warehouse decreases.

Assessing Learning

Name: _____ **Date: _____**
Section: _____

Directions: Explain briefly and concisely of the following statements:

1. What can data mining contribute?

_____
_____

2. What are the six common tasks of data mining?

_____
_____
_____
_____

3. How is a data warehouse beneficial to an organization?

_____
_____
_____

4. What are the different data warehouse models?

_____
_____
_____
_____

5. Cite one advantage and one disadvantage of data warehouse. Explain briefly.

_____
_____
_____
_____
_____
_____
_____

References

Database administration. (2020). Retrieved 30 July 2020, from https://en.wikipedia.org/wiki/Database_administration_and_automation

Database Management Systems eBooks For All Edition (Retrieved from www.ebooks-for-all.com)

Dept of Cse & It Vssut, Burla, Lecture Notes On Data Mining& Data Warehousing Course Code:Bcs-403

Documentation, E. (2020). Timesheet :: Chapter 6. Database Design. Retrieved 30 July 2020, from https://macrotoneconsulting.co.uk/Extension-Documentation/timesheet/ch06/

Document-oriented database - Infogalactic: the planetary knowledge core. (2020). Retrieved 30 July 2020, from https://infogalactic.com/info/Document-oriented_database

Dr. Radványi Tibor, 2011, Advanced DBMS, Kiadó © Dr. Radványi Tibor, 2011

Hoffer, Jeffrey et al. (2002). Modern Database Management Sixth Edition. Prentice Hall.

HSQLDB - Joins - Tutorialspoint. (2020). Retrieved 30 July 2020, from https://www.tutorialspoint.com/hsqldb/hsqldb_joins.htm

Molina, Hector Garcia- et.al 2009., Database Systems The Complete Book Second Edition. Pearson Education Inc., Pearson Prentice Hall

SQL - Quick Guide - Tutorialspoint. (2020). Retrieved 29 July 2020, from https://www.tutorialspoint.com/sql/sql-quick-guide.htm

SQL - INNER JOINS - Tutorialspoint. (2020). Retrieved 30 July 2020, from https://www.tutorialspoint.com/sql/sql-inner-joins.htm

SQL, L. (2020). Learn Learn SQL and much more on Memrise. Retrieved 30 July 2020, from https://www.memrise.com/course/700039/learn-sql/34/

SQL Interview Questions and Answers with examples | Skillslelo.Com. (2020). Retrieved 30 July 2020, from https://skillslelo.com/sql-interview-questions-and-answers-with-examples/

Stored procedure. (2020). Retrieved 30 July 2020, from https://store.tutorialspoint.com

Wang, John, Data warehousing and mining: concepts, methodologies, tools and applications, Copyright © 2008 by IGI Global

https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/

https://www.relationaldbdesign.com/database-design/module3/dblc-design-stages.php

http:// tutorialpoint.com/Simply Easy Learning

https://www.google.com.ph/amp/s/www.geeksforgeeks.org/generalization-specialization-and-aggregation-in-er-model/amp/

https://javapapers.com/oops/association-aggregation-composition-abstraction-generalization-realization-dependency/

https://www.javatpoint.com/dbms-aggregation

http://agiledata.org/

https://whatisdbms.com/

https://www.zentut.com/data-mining/advantages-and-disadvantages-of-data-mining/

https://businessimpactinc.com/blog/the-pros-cons-of-data-warehouses/