**CSIS 2300:**

**Chapter 01: Getting Started**

**Why Use a Database?**

* Store data more complicated than simple lists.
* Keep track of data and relations.

**The Problem with Lists:**

* **Modification problems:** redundancy and multiple themes can create modification problems. The figure bellow illustrates a problem the operations update, delete and insert.

Diagram

Description automatically generated

* + Updating the advisor requires the update of all advisor’s related fields.
  + Deleting a row cause a student and advisor data lost.
  + Inserting a row ay imply on incomplete data.

**Relational Database:**

* **Relational model** is a methodology used as a solution for database design.
* A **relational database** contains a collection of separate tables.
* A **table** holds data about only one *theme*.
* Each **column**,also knownas fields, in a table *stores a characteristic* common to all rows in a table.
* A **row** in a table, also known as a *record*, has data about an *occurrence*.
* The leading technique for data definition and manipulation is **Structured Query** **Languages** (SQL).
* **SQL** is an international standard for creating, processing, and querying databases and their tables. With SQL you can:
  + Reconstruct lists from their underlying tables.
  + Query for specific data conditions.
  + Perform calculations on data in tables.
  + Insert, Update, and Delete data.

**What is a Database System?**

A **Database system** has four components consisting of:

Diagram

Description automatically generated

1. Users:
   1. Employ a database application to keep track of things.
   2. Use forms to read, enter, and query data.
2. Database application.
   1. Create and process forms.
   2. Process user queries.
   3. Create and process reports.
   4. Execute application logic.
   5. Control application.
3. Database management system (DBMS).
   1. The purpose of a DBMS is to create, process, and administer databases and are licensed from a vendor. The DBMS enforce the **referential integrity constraints** (rules to ensure values of a column in one table are valid when compared to values in another table).
   2. The function of a DBMS is:
      1. Create databases.
      2. Create tables.
      3. Create supporting structures (e.g., indexes).
      4. Read database data.
      5. Modify (insert, update, or delete) database data.
      6. Maintain database structures.
      7. Enforce rules.
      8. Control concurrency.
      9. Provide security.
      10. Perform backup and recovery.
4. Database:
   1. Self-describing collection of related tables, where **self-describing** means a description of the structure of the database is contained with the database itself.
   2. **Metadata** is a data about the structure of the database.

Diagram

Description automatically generated

**Personal Database Systems:**

Diagram

Description automatically generated

**Enterprise-Class Database Systems:**

Diagram

Description automatically generated

**Web Database Applications:**

* A **web database application** is an application with a web user interface that is dependent on a database to store the data needed by the application.
* An **Application Programming Interface** (API) is a programming language such as PHP or JavaScript to connect to a DBMS allowing the sending of SQL commands to the DBMS and then to receive them back

**Data Warehouse and Business Intelligence (BI) Systems?**

* **Transactions** are purchases bought online that are recorded in a company’s database, also referred to as an **Online Transaction Processing** (OLTP) database.
* Data analysis is done on an organization’s **Online Analytical Processing** (OLAP) database and is used for research.
* A **Business Intelligence System** consists of tools used to analyze and report on company data.

**Chapter 02: The Relational Model**

**Relational Terms:**

* An **entity** is something of importance to the user that needs to be represented in a database.
  + In an entity-relational model, entities are restricted to things that can be represented by a single table.
* A **relation** is a two-dimensional table consisting of rows and columns that has the characteristics shown above:
  1. Rows contain data about an entity.
  2. Columns contain data about attributes of the entity.
  3. Cells of the table hold a single value.
  4. All entries in a column are of the same kind.
  5. Each column has a unique name.
  6. The order of the columns is unimportant.
  7. The order of the rows is unimportant.
  8. No two rows may hold identical sets of data values.

**Relational Structures:**

* When writing out relation structures use the following format:
  + Relation names are written first in all caps (if two words, then use an underscore between them), and they are always singular.
  + A column name is written with the first letter capitalized (if two words, then run them together and capitalize the first letter of each word).
* A **database schema** is the design on which a database and its associated applications are built.

Table

Description automatically generated

**Types of Keys:**

* A **key** is one or more columns of a relation that is used to identify a row.
  + A key can be unique (primary key) or nonunique (foreign key).
* A **composite key** contains two or more attributes.
* **Candidate keys** are the keys that uniquely identify each row in a relation.
* A **primary key** is a candidate key that is chosen as the key that the DBMS will use to uniquely identify each row in a relation.
  + The primary key in a relation will be underlined.

**Surrogate Keys:**

* A **surrogate key** is a column with a unique, DBMS-assigned identifier that has been added to a table to be the primary key.
  + The ideal surrogate key is short, numeric, and never changes.
* Suppose we have the following table: ***PROPERTY (Street, City, State, ZIP, OwnerID)***
  + Notice that it takes the street, city, state, and zip to uniquely identify a row in a table.
* Now let’s use a surrogate key as a unique identifier! ***PROPERTY (PropertyID, Street, City, State, ZIP, OwnerID)***

**Foreign Keys:**

* A **foreign key** is a primary key of another relation that has been placed in the current relation to represent a relationship between two tables.
  + It is represented in a relation by italics as seen in the example bellow.

***EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, EmailAddress, Phone)***

***DEPARTMENT (DepartmentName, BudgetCode, OfficerNumber, DepartmentPhone)***

**Referential Integrity:**

* A **referential integrity constraint** states that every value of a foreign key must match a value of an existing primary key.
* In the relationship between EMPLOYEE and DEPARTMENT seen in the **Foreign Keys** section, the department attribute located in the EMPLOYEE table is the foreign key and whatever value is placed in that column, the same value MUST exist in the Department attribute in the DEPARTMENT table.

**The NULL value:**

* A **null value** is a missing value in a cell in a relation.
* The problem with null vales is that it is ambiguous:
  + Is it that no value is appropriate?
  + Is it known, but not entered?
  + Is it unknown, thus not entered?
* You can eliminate null values by requiring an attribute value.

**Functional Dependencies:**

* A functional dependency occurs when a candidate key determines all the other attributes in a relation.
  + In other words, all the attributes in a relation are functionally dependent on the candidate key.
  + A dependency is shown with the determinant on the left and then an arrow showing the attribute(s) that depend on it, as shown: ***CustomerNumber-> (CustomerLastName, CustomerFirstName, Phone)***

**Normalization:**

* **Normalization** is the process of (or a set of steps for) breaking a table or relation with more than one theme into a set of tables such that each has only one theme.
* Relational design principles for a well-formed relation:
  + Every determinate must be a candidate key.
  + Any relation that is not well formed should be broken into two or more relations that are well formed
* A relation is in first normal form (1NF) if it:
  + Has characteristics listed in Figure 2.1.
  + Has a defined primary key.
  + No repeating groups.

**Algorithm (Normalization Process):**

1. Identify the candidate keys of the relation.
2. Identify all the functional dependencies in the relation.
3. Examine the determinants of the functional dependencies. If any determinant is not a candidate key, the relation is not well formed. In this case:
   1. Place the columns of the functional dependency in a new relation of their own.
   2. Make the determinant of the functional dependency the primary key of the new relation.
   3. Leave a copy of the determinant as a foreign key in the original relation.
   4. Create a referential integrity constraint between the original and the new relation.
4. Repeat step 3 until every determinant of every relation is a candidate key.

**Example:**

Both the prescription and its customer details are in the same table.
The data from the table are,
1. 
Prescription Number,  P10001
Date,  10/17/2019
Drug,  Drug A
Dosage,  10mg
Customer Name,  Smith, Alvin
Customer Phone,  575-523-2233
Customer Email Address,  ASmith@somewhere.com

2.
Prescription Number,  P10003
Date,  10/17/2019
Drug,  Drug B
Dosage,  35mg
Customer Name,  Rhodes, Jeff
Customer Phone,  575-645-3455
Customer Email Address,  JRhodes@somewhere.com

3.
Prescription Number,  P10004
Date,  10/17/2019
Drug,  Drug A
Dosage,  20mg
Customer Name,  Smith, Sarah
Customer Phone,  575-523-2233
Customer Email Address,  SSmith@somewhere com

4.
Prescription Number,  P10007
Date,  10/18/2019
Drug,  Drug C
Dosage,  20mg
Customer Name,  Frye, Michael
Customer Phone,  575-645-4566
Customer Email Address,  MFrye@somewhere.com

5.
Prescription Number,  P10010
Date,  10/18/2019
Drug,  Drug B
Dosage,  30mg
Customer Name,  Rhodes, Jeff
Customer Phone,  575-645-3455
Customer Email Address,  JRhodes@somewhere.com