

## Chapter 7 Database Management

### Introduction

The information age has brought about heavy demand on storing, organizing and retrieving data as well as producing usable information out of the stored data in the form of reports or answers to queries. Due the enormous nature of the data stored, they are placed in many files in an organized way and relationships between the records in these files are established taking special care to minimize duplication of the data. A collection of programs that handle all these data requirements is called a Database Management System (DBMS). The collection of files that contain related data is called a database.

Traditional file processing systems concentrated on a particular task such as Accounting or a particular department such as Human Resources. These files were sequential, random or indexed access depending on the program. These programs were isolated and did not share the files with another program, and would only generate a pre-defined set of reports. If a new report is needed, a separate program had to be written by those familiar with the file structure of that program. It was difficult to obtain file structures for proprietary programs. The main disadvantages of these traditional file processing systems were the data redundancy and resulting inconsistency. The database management system attempts to answer these shortcomings.

### Some Terminology

A *relation* in a database is organized as a *table* of *rows* and *columns*, each row containing a *tuple* (record) and each column containing an *attribute* (field) of the record. Each attribute must have a *type declaration* associated with it. For example, a student record may contain name, age, social security number, address, city, state, zip, telephone number and other relevant attributes. Each attribute will have an allowable range of values called a *domain*. The way that data is organized and encoded on the hard disk drive is the *physical aspect*, and selection of the data to be stored and their relationship make up the *logical aspect*. The design of the database at the physical aspect is called the *physical schema*, and the design of the logical aspect is called the *logical schema*. Only a portion of the database will be visible to a given user and it is called the *view level* of a database. The collection of all data in the tables of a database at any given time is called an *instance* of the database. When a relation has minimized duplication of data and other rules of relation have been satisfied a relation is said to be *normalized*.

There are three database models, *relational*, *network* and *hierarchical*. In the relational model data and their relations are incorporated into tables with one or more data items as *key fields*. One or more fields identified as key fields will uniquely distinguish one record from another. This key will be used to retrieve data and describe relationship with another table. When a key field from one table appears in another, it is called a *foreign key*. The relational database is modeled after *relational algebra* and *relational calculus*. E. F. Codd is credited for his work in the relational database model. In the network model relationship between records are depicted as a *graph* with *nodes* and *edges*. Graphs can be mathematically described using *sets*. In the hierarchical model

the database is constructed like a *tree*, each node having only one *parent*. In this chapter, only the relational database will be dealt with.

### Relational Algebra and Relational Calculus

Relational Algebra and Relational Calculus are two formal languages for relational database queries. Examples of unary relational algebraic operators are select and project. Selection returns certain tuples of a relation based on some criteria. `SELECT * FROM Student WHERE zip='78501'` will list all students with the zip code of 78501. Algebraic expression for this selection is:

$\sigma_{\text{zip}='78501'}(\text{Student})$

The Project Operation extracts specified columns from a relation. For example,

$\Pi_{\text{StLName, zip}}(\text{Student})$

--I will add more later --

### Relational Databases

The earliest version of widely accepted relational database for personal computers was the dBase II that was released in 1979. The dBase I was written for the CP/M operating system, while the dBase II ran under DOS and Apple OS. After release IV it was discontinued in 1993 as FoxPro and Microsoft Access gained the market share. Microsoft Access is the predominant database management system for personal computers today.

In addition to Microsoft Access and FoxPro mentioned earlier there many database management systems available in the market today for all sorts of computers from PCs to multiprocessor systems running various operating systems. Frequently encountered DBMS are Microsoft SQL, Oracle, SAP, Informix, Sybase, Pervasive (Btrieve), NCR Teradata, MySQL, PostgreSQL, and Inprise InterBase.

### DBMS Languages

The Structured Query Language (SQL) provides for creating, manipulating and retrieving data from databases. It incorporates both the Data Definition Language (DDL) and the Data Manipulation Language (DML). The DDL is the part of the SQL that is used to define the database schema that provides for creating, altering, dropping, creating index, dropping index, and granting privileges. The DDL creates a data dictionary containing metadata (data about data) when a table is created. The DDL contains a subset of language called the data storage and definition language that specifies the storage structure and access methods for the database. The Data Manipulation Language (DML) handles manipulation of data through SQL commands like select, update, delete and insert into. Most commonly used SQL statements and their syntax are given below:

Creating a Table – CREATE TABLE studentInfo (ID char(11), name char(50), grade char(1)).

This creates a table named studentInfo with three columns, ID, name, and grade.

Inserting a new record – INSERT INTO studentInfo VALUES ('463-47-5455', 'David Egle', 'A'). This will place values into each of the respective columns.

Inserting into specified columns – INSERT INTO studentInfo (ID, name) VALUES ('463-47-5455', 'David Egle').

To retrieve data from a table – SELECT \* FROM studentInfo. The \* means all. This statement will retrieve all columns (in this case all 3 columns) from studentInfo table. To choose desired columns use this command. SELECT ID, name FROM studentInfo.

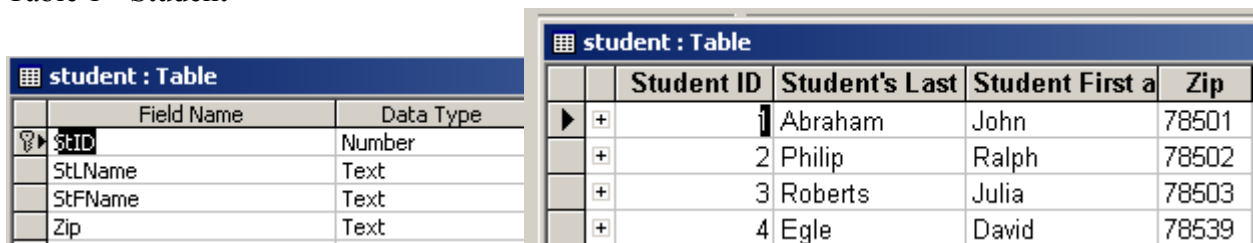
To select data that meets certain conditions – SELECT \* FROM studentInfo WHERE grade = 'A'. The WHERE clause will select records with grade 'A'. Operators that can be used with the clause are: =, <, >, <=, >=, <>, BETWEEN, LIKE. The BETWEEN is an inclusive range and the LIKE searches for a pattern.

Update data in the table – UPDATE studentInfo SET grade = 'B' WHERE name = 'David Egle'. This statement will replace the grade for David Egle.

Delete a record from a table – DELETE FROM studentID WHERE name = 'David Egle'. This statement will delete David Egle's record from the table. If the WHERE clause is left out all records will be deleted from the table.

In order to demonstrate some statements it is necessary to create two or more tables. The following tables were created using Microsoft Access. The first table, Student has three fields, student identification (StID), student last name (StLName), and student first name (StFName). The student identification is the unique key for this table.

Table 1 - Student



The screenshot shows the 'student : Table' in Microsoft Access. On the left, a table structure view shows the following fields and data types:

| Field Name | Data Type |
|------------|-----------|
| StID       | Number    |
| StLName    | Text      |
| StFName    | Text      |
| Zip        | Text      |

On the right, a data view shows the following records:

| Student ID | Student's Last | Student First a | Zip   |
|------------|----------------|-----------------|-------|
| 1          | Abraham        | John            | 78501 |
| 2          | Philip         | Ralph           | 78502 |
| 3          | Roberts        | Julia           | 78503 |
| 4          | Egle           | David           | 78539 |

The second table, Courses has three fields as well, course identification (courseID), course description (courseDescription), and course hours (courseHours). The course identification will be the course number or a unique number assigned to a course offered at the university, example: CS 1380. The course description is the name of the course such as CS1 C++, and the course hours is the number of semester credit assigned that course. The courseID is the key for this table.

Table 2 - Courses

|  | Field Name        | Data Type |
|--|-------------------|-----------|
|  | courseID          | Text      |
|  | courseDescription | Text      |
|  | courseHours       | Number    |
|  |                   |           |

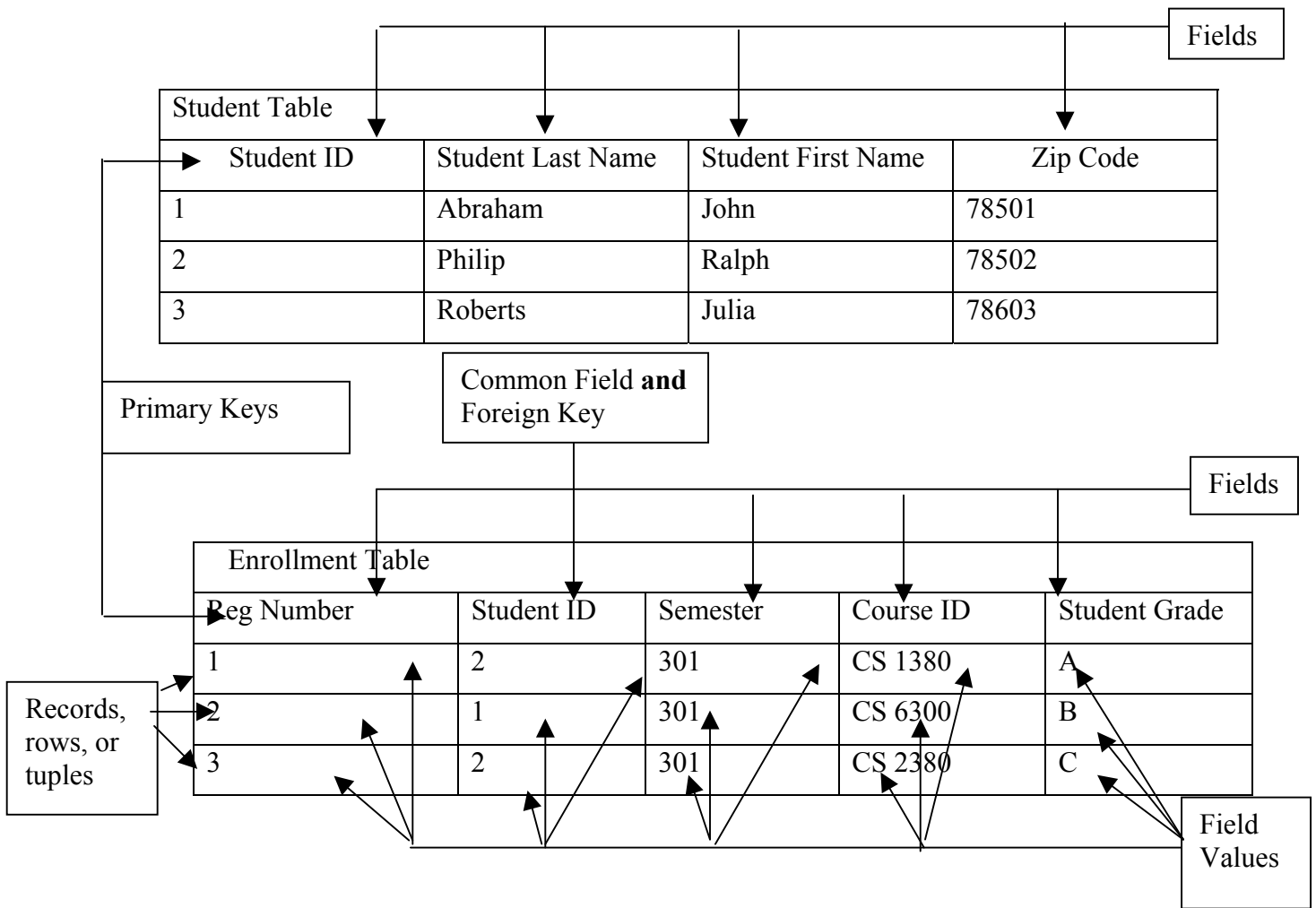
| courses : Table |          |                   |             |
|-----------------|----------|-------------------|-------------|
|                 | courseID | courseDescription | courseHours |
|                 | CS 1380  | CS1 C++           | 3           |
|                 | CS 2380  | CS2 DS            | 3           |
|                 | CS 4345  | Comp Network      | 3           |
|                 | CS 6300  | Comp Systems      | 3           |
|                 | CS 6303  | Inf Technology    | 3           |

The third table has four fields. An auto number for registration, the semester enrolled (for example for Fall 2003 enter 301 or any such code that make sense), course identification (see table 2), student identification (see table 1), and student grade, stGrade.

Table 3 - Enrollment

|  | Field Name | Data Type  |
|--|------------|------------|
|  | RegNo      | AutoNumber |
|  | Semester   | Number     |
|  | courseID   | Text       |
|  | stID       | Number     |
|  | stGrade    | Text       |
|  |            |            |

| Enrollment : Table |       |          |          |      |         |
|--------------------|-------|----------|----------|------|---------|
|                    | RegNo | Semester | courseID | stID | stGrade |
|                    | 1     | 301      | CS 1380  | 2    | A       |
|                    | 2     | 301      | CS 6300  | 1    | B       |
|                    | 3     | 301      | CS 2380  | 2    | C       |
|                    | 4     | 301      | CS 2380  | 1    | A       |
|                    | 5     | 301      | CS 6302  | 3    | B       |
|                    | 6     | 301      | CS 6300  | 3    | A       |



Now, running a query as follows will show all courses enrolled in by specified student ID. The WHERE clause is a JOIN condition: tables Student, and Enrollment are joined to give the results that satisfy the condition.

```
SELECT distinct StFName, StLName, courseDescription
FROM student, courses, enrollment
WHERE enrollment.StID = student.StID AND enrollment.courseID=courses.courseID;
```

| Student First a | Student's Last | courseDescripl |
|-----------------|----------------|----------------|
| John            | Abraham        | Comp Systems   |
| John            | Abraham        | CS2 DS         |
| Julia           | Roberts        | Comp Systems   |
| Ralph           | Philip         | CS1 C++        |
| Ralph           | Philip         | CS2 DS         |

