DBS211

Week 1 - Introduction to Database Systems

Table of Contents

Suggested Reading Materials

- Text - Chapter 1 (Before the Advent of Database Systems)
- Text - Chapter 2 (Fundamental Concepts)
- Text - Chapter 3 (Characteristics and Benefits of a Database)

Welcome to Week 1

Welcome to week 1 of DBS211, An Introduction to Database Systems. In this course students will be introduced to data management concepts, database systems and database application programming. Students learn to represent information with a relational database model and manipulate data using structured query language (SQL). Students will also learn the basic concepts of database modelling and the design and development of a new database to be used for business purposes.

Week 1 acts as an introduction to the course in addition to some of the key terms and concepts needed throughout the remainder of the course. After this week you should be able to:

- describe the difference between data and information
- be able to name a variety of file systems and database management systems
- explain why databases are important in software development
- communicate the difference between a database server and a graphical user interface (Oracle vs. SQL Developer)
- list the different types of data anomalies and be able to avoid them in the future
- effectively describe the database software development process.

Data vs. Information

To understand the use of databases, the first thing one needs to understand is the different between data and information and how they lead to knowledge and wisdom.

Data: is the raw unprocessed strings, numbers, dates, etc. that is simply stored. Data is raw, unorganized facts that needs to be processed. Data can be something simple and seemingly random and incoherent until organized. Data is the building blocks of information. Data can also be referred to as nothing more than symbols.

Examples: Date of Birth, Student Grades, course, employee SIN.

Information: reveals the meaning of data and is produced through the processing of the data into meaningful output, summations and categorizations. Can often be associated with the answers to: "Who", "What", "Where", and "When". Information is the basis of Knowledge which is essential in daily decision making.

Examples: Persons Age, Average student Age, Average student mark, Students GPA, Total Amount Owing

File-based Systems

File based systems in the context of this module refers to the way data is managed or stored before databases are used. Many current businesses still use non0database driven file-based systems.

One way to keep information stored on a computer is by saving various files on your computer. You create a series of folders and files to store the information in a way that makes sense to you. Maybe you create a folder on your conputer for each term of school and then sub-folders for each course, and then sub-folders for assignments, downloads, labs, etc. When you need to locate those files, you are familiar with the structure of the folders and can navigate to the required file where hopefully the name of the file clearly indicates what is contains.

Now let's expand this simple example to hundres or thousands of students taking hundreds or thousands of courses. Every student naming files with their own convention and style and creating their own folder structures. As a 3rd party coming in to find a specific file, it may be very difficult and time consuiming to find the exact file wanted. This gets even more complex in businesses that have many departments and a workflow for each particular file.

A Simple business example:

- A customer **orders** products from a Retailer
- Sales Department would take the **order** information from the **customer** and send a **request** to the Shipping Department to deliver a **product** to a **customer**
- The Shipping Department would request the **products** from the **warehouse** for packaging
- The warehouse locates the **products**, gathers them **together** and **packages** them
- The warehouse sends the **package** to shipping

- Shipping enters in the information about the **customer order** and create a **shipping document**
- The Shipping Department would print the **shipping document** and send the **products** to the **Customer**

Possible files to store:

- Sales has files for: customer, product, and sales
- Shipping has files for: customer, products, packages, shipping document
- Warehouse has files for: products, shipping document, packages

Can you see a problem with this?

- Where do you go to look u p a customer phone number? (both sales and shipping have customer data)
- Where do you get the product code for a specific product? (Sales, Shipping and Warehousing all have product data)

Data Redundancy

Data that is stored in more than one location has inherant problems. If a customer needed to change their phone number, they would have to contact multiple places (Shipping and Sales) in order to completely change the number. If any one data source was missed, now there is conflicts and this means errors.

Example:

Courses Tables

Course Code	Course Name	Student ID	**Student Name**
dbs211	intro to dbase systems	900111111	**John Smith**
dbs211	intro to dbase systems	900222222	**Allyson Jones**

Students Table

Student ID	**Student Name**	Date of Birth	Phone
900111111	**John Smith**	May 16, 2002	9055551111
900222222	**Allyson Jones**	Sept 12, 2001	4165552222

web222	web programming fundamentals	900111111	**John Smith**

900333333	**Raj Patel**	Feb 28, 1999	2865553333

In the above example, the student names are redundantly stored in multiple locations. If for some reason, a student needs to change their name (a marriage), then the update would have to occur in multiple locations. Note, the fact that John Smith is in the courses table twice is not redundent, it is repetitive.

Data Repetition

Data that is stored in one location, but repeated many times also has enherent problems. For instance, the same customer may place many orders and therefore, sales will have many order files for a specific customer. If the customer needed to update their phone number, they might have to do it for all past orders palced in order to have the most up to date information associated with those orders in case of warranties, returns etc.

The solution to both Data Redundancy and Data Repetition is centralizing the files in one location and allowing each department or request to access the data from a single central source. But now what if shipping needs the updated address, but a salesman has the one file and it is not currently available. The shipping of the products would then be delayed resulting in an unhappy customer.

What is you wanted to change the name of the file where the data is stored. All the departments would need to know the new name so they could find the file when needed. This becomes more troublesome with highly heirarchal systems with many levels of depth.

Example:

Courses Tables

Course Code	**Course Name**	**Student ID**	Student Name
dbs211	**intro to dbase systems**	900111111	**John Smith**

Students Table

Student ID	Student Name	**Date of Birth**	**Phone**
900111111	John Smith	May 16, 2002	9055551111

Course Code	Course Name	Student ID	Student Name
dbs211	**intro to dbase systems**	900222222	**Allyson Jones**
web222	web programming fundamentals	900111111	**John Smith**

Student ID	Student Name	Date of Birth	Phone
900222222	Allyson Jones	Sept 12, 2001	4165552222
900333333	Raj Patel	Feb 28, 1999	2865553333

In this example, data stored in a single table multiple times is repetitive data. Everytime, the course code dbs211 is used, the same course name will be present. If the name of the source was to change, it would need to be updated in every row where it occurs.

Also, the student name is directly related to the student ID and therefore John Smith is repetitive.

It must be clear, that although the Student ID and Course Codes are also repeated, they are not repetitive from an inefficiency perspective, they are required to maintain the relationship between course and student.

In this example, the student name column is not needed in the Courses table at all as the name can be "looked up" using the student ID and the students table.

Additionally, a new Courses table could be created linking course code to course name and then the course name can be removed from this table leaving only the course code and the student ID, which gives us all the information we need to determine which student is taking which course.

Data Anomalies

In order to understand further why data repetition and data redundancy must be avoided, it is important to understand the consequences of these issues in more detail.

There are 3 main types of data anomalies that can occur in a database:

- Modification Anomalies
- Insertion Anomalies
- Deletion Anomalies

Modification Anomalies

A modification anomaly occurs when changing one records data leaves another records data in error.Skill

Employee's Skills		
Employee ID	**Employee Address**	**Skill**
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	**94 Chestnut Street**	Public Speaking
519	**96 Walnut Avenue**	Carpentry

If we need to change the address for employee 519: it has to be done for all rows with employee ID 519.

What if this employee address is in another table: it has to be done for all tables

Insertion Anomalies

Faculty Hires			
Faculty ID	**Faculty Name**	**Hire Date**	**Course Code**
389	Bob Giddens	10-Feb-1985	ENG206
407	Ruth Saperstein	19-Apr-1999	CMP101

Faculty ID	Faculty Name	Hire Date	Course Code
407	Ruth Saperstein	19-Apr-1999	CMP201

424	**Henry Newsome**	**29-Mar-2007**

If the the Faculty Hires table requires a Course Code, then we would not be able to enter new Faculty members hired before they are assigned a course.

Deletion Anomalies

In the Faculty Hires table above, if Bob Giddens in not teaching ENG206 this semester, we would remove him from this table. The removal would result in the loss of the hire date data stored in this table. Therefore, 2 independent fields may actually impact each other due to data. This is not a good database design.

Databases

A database is a structure that contains logically related data in a single repository. Through careful design and development a database can be a centralized storage facility that is accessed simulataneously by several departments and if one peice of data is updated, it is autmoatically updated everywhere as it is centralized and unique.

By using a database, data is much less likely to be subject accidental disorganization, is much more accessible, and integrated with all aspects of the business. The database is primarily a central data sotrage facility, but additionally allows data to be manipulated through sorting, matching, linking, aggregating, calculations and arrangements. The database can simultaneously be accessed by a web site, a mobile application, the warehouse computer, the sales department, accounting, and management all at the same time in real-time.

Benefits of Database Implementaion

- elimination of data redundancy

- elimination of data repetition
- storing great amounts of data with little to no physical space requirements
- tactical and strategic decision support
- supporting and feeding day to day business operations through production and transactions.
- ability to get more information from the same amount of data through relations
- sharing of data
- security of data to authorized users
- software independance (can restrucutre database without requiring updates to the software)

Database Management Systems

A database management system is a collection of programs that manages a database structure and controls access to the database (and ultimately the data). The database is just one of those programs in the collection. For the purposes of this course, we will be using the Oracle DBMS which comprised of Oracle12c (Data Server) and SQL Developer (User Interface and Access tool).

The database management system typically has many components that:

- manages the sharing of data amongst multiple applications and/or users
- ensures data consistency
- add the ability to do ad hoc querying
- are extensions to the manufactured components to add addition and customized access and usage (A custom developed API may be an example)

Some examples of popular DBMSs are:

Database Design

This course covers many aspects of database design and development. At this time we will briefly introduce the concept.

A database design should:

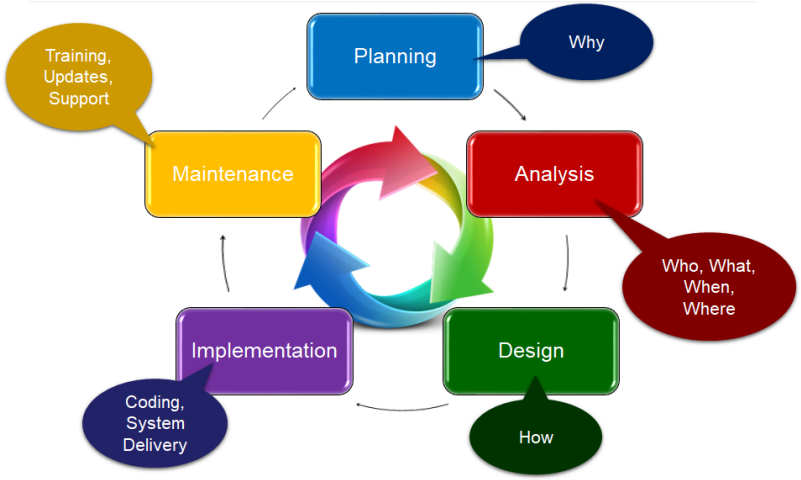
- define the expected use of the database
- avoids redundant and repetitive data
- completed in association with the Software Development LifeCyle (SLDC)
- Completed using the Database Design Life Cycle (DBLC)

To further expand upon the design of databases, there are 5 basic rules of good database design. There are lots of design concepts, but these 5 will guide the computer programmer towards a well thought out relational design. The 5 Basic rules are:

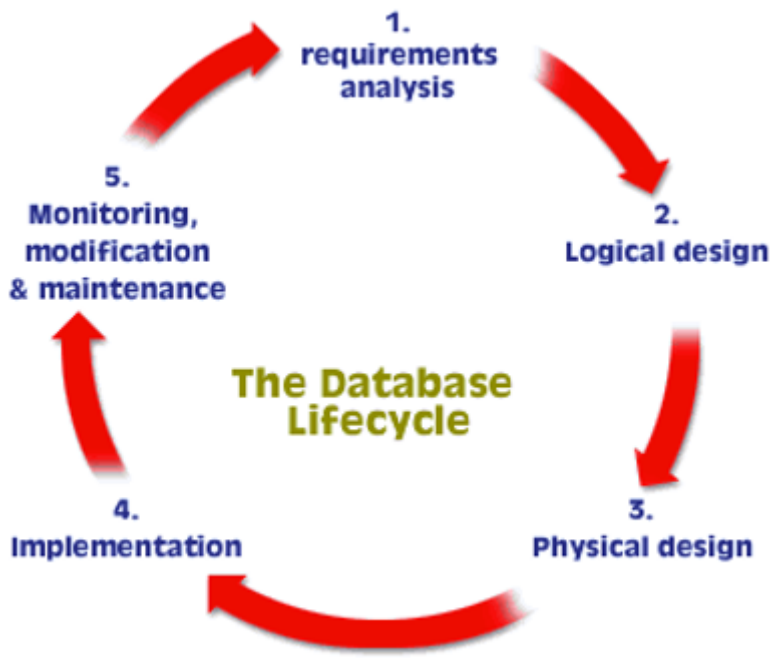
1. Every Table or Entity requires a **unique identifier** to determine exact row of interest
2. Avoid **redundent** data storage
3. Avoid **repetitive** data sotrage
4. Do not include any **calculated fields**, or values that are directly derived from other fields.
5. Every cell in the database must have **atomic values** (a single value).

Database Development Life Cycle

SDLC - Software Development Life Cycle



Planning	We have an idea! Why do we need this software, how will it help the organization? Who will champion this project? What are the goals or objectives of the software?
Analysis	Needs analysis: What does the software need to be successful? Should we build it? Do we have the right team to build it? Can we financially afford to built it? Do we have the right technologies available?
Design	Specifications, architecture, choice of languages, frameworks, diagrams, pseudo-code, scoping.
Implementation	Coding the software, creating the graphics, testing the software, actually creating the product, installation and delivery.
Maintenance and Support	Maintenance plan, on-going monitoring and review, what are the next steps, new versions? Training and on-going customer support



Requirements Analysis	What does the database need? What information do we need later? How is the database going to be used? What other applications will the database support?
Logical Design	Design Specifications, Data Modeling, Entity Relationship Diagrams, Data Types, Normalization,
Physical Design	Actual process of creating tables and database structure (SQL, DDL)
Implementation	Inserting or importing data, installation on live server, backup systems, and more
Monitoring, modification, and Maintenance	On-going backups, data archival, monitoring, additional queries as required

End of Week 1

TOP | HOME