SQL Course

Introduction

An **I**ntroduction **T**o **D**atabases
Objectives

- This module defines **what a database is** and describes the **central concepts of a database**.
- It also explains the benefits of using database technology compared to traditional file processing techniques.

"**Database** is a logically coherent collection of related data". It should provide:

- **A single source of information** for user groups and multiple applications.
- **Independence** from the way information is actually stored in files on disk.
- **Concurrency Control** so that concurrent applications or users operations will appear to happen one after another (i.e. no interference between users or applications operating on data at the same time).
- **Data integrity** control. This controls the type, domain values and relationships between values of the information stored within a database.
- **Data security**. This guards the privacy of information stored in the database.

- A database uses a set of concepts that describe the structure of the information contained within it. These set of concepts differ between different types of database. A **Data Model** is the set of concepts characterising the way in which data is organised within a database.
- A **Database Management System (DBMS)** is a collection of programs that enable users to create and maintain a database.
- A DBMS is a general purpose software system that enables users to define, construct and manipulate databases for various applications.
- A DBMS usually supports one particular data model. Therefore most database management systems are described by the data model they support.

Hierarchical

This type of database represents data as hierarchical tree structures. Each hierarchy represents a number of related records. Although most hierarchical databases have a query language, these query languages differ from implementation to implementation and are not standardised.

Network

This type of database represents data as record types and also represents a limited kind of relationship between records. The network data model (also knows as CODASYL DBTG Model), has an associative record-at-a-time language that must be embedded in an application programming language e.g. COBOL.

Relational

This type of database represents information as a collection of Tables. Most relational DBMSs support a high level query language called SQL (Structured Query Language).

Object Oriented

This type of database represents information as objects which contain both data and operations which can process that data. A standard for an Object Oriented data model has not yet been agreed on but proposals of standards have already been made and agreement is expected in the next few years.

Databases were first designed to allow applications and users to share information in a controlled manner. This concurrent control of database retrievals and operations is still the major advantage of using databases. However several other important benefits can be summarised from using database management systems:

- **Isolation of application programs and data:** By not embedding the storage and retrieval operations in an application, many users and applications may access the same information.
- **Denial of Unauthorised Access** to information held within a database.
- **Representation of complex data-types and relationships** among data.
- **Enforcement of integrity constraints** on data (i.e. constraints on the type information or values that a piece of data can have).
- Provide **automatic backup and recovery** of information stored in the database.

SQL Course

The Relational Model

The **R**elational **M**odel **D**atabases
Objectives

- To introduce **the basic notion** of the Relational Data Model.
- Describe the **key concepts and theory** which supports this data model.

- The notion of Relational Modelling for Databases originally arose from work done by E.F. Codd at IBM research laboratories.
- The idea is to organise all data items into a number of relations. Informally, each relation resembles a table. The data can thus be inserted, modified, deleted and queried based on these data tables.
- Thus if a database organises its data in accordance with the relational data model, it is termed a **Relational Database**.

Relational Database Table

aircraft name	*model*	*club seats*	*econ seats*	*call sign*
Eagle Flyer	ATR42	22	40	N410C
(NULL)	Boeing 707-320C	50	102	9J-AEB
(NULL)	Boeing 727-200	34	100	N7255U
Finians Dream	Boeing 737-200	22	96	DQ-FDM
(NULL)	Boeing 737-200	8	120	N301SW

Value

The smallest update-able portion of a table. A value is specified by the intersection of a column and a row.

Columns

A column is a set of values which may vary over time. All values of a column are of the same data-type.
A column specification includes its type and integrity constraints on the values it may hold.
Column names are also called **attribute** names.

Rows

A row is a non empty set of values. Rows are also called **tuples**.

- In Relational databases a table is termed a **relation**, a row in a table is termed a **tuple** and a column in a table is termed an **attribute**.
- The fact that the information is presented in tables does not constrain the actual underlying storage mechanism of a relational database system. The data may be stored in files and records on disk but the storage formats are hidden from the user.
- Since all data in a relational database is organised into data tables, all operations in the

database are in fact operations on the data tables themselves e.g. database queries are merely retrieval operations performed across one or more of these tables
- The relational data model is based on the mathematics of set theory Relational algebra.

The notion of a Relational data model is based on the mathematical foundations of set theory. A relation is a special type of set, where:

- A set is a collection of values of the same type.
- A set has no implied ordering.
- A set has no duplicate elements.

Therefore the operations supported by a relational database are set oriented and NOT record oriented. Therefore unlike third generation programming languages like C and PASCAL which store and operate on information in 'a record at a time' fashion, a single relational query can operate over a set of values.
For example consider a table containing information describing different types of aircraft e.g. aircraft name, manufacturer, number of seats in economy class, number of seats in business class. The relational query to retrieve 'all information about aircraft where there are more than 100 economy seats' would be a single relational query.

However, if a user tried to retrieve the same information using PASCAL or C, it would involve writing a small program which looped through the entire table checking the economy value in each row and returning all row values where the economy attribute value is greater than 100.

Further Reading

For further reading into the theoretical basis and mathematics of Relational database the reader is referred to the following references.

- "'Database Systems: Models, Languages, Design and Application Programming', R. Elmasri, S. Navathe, 6th edition, published by Pearson Education, 2010 (ISBN-10: 0132144980 | ISBN-13: 9780132144988).
- "'An Introduction to Database Systems' Vol. 1, C.J. Date, published by Addison Wesley,1990.

SQL Course

Relational DBMS Architecture

Objectives

- To introduce the basic concepts of what a Query language is.
- To focus particurarly on Structrued Query Language.
- To give a brief history of SQL.
- To deal with some of the Standards that exist within Query Languages.

- **Simple data structures:** Since the user need only deal with tabular information, the data structures are both simple and intuitive.

- **Simple operators:** All operations are based on tables. Therefore there is much greater uniformity in the operations and their syntax.

- **View mechanism:** The Database administrator can easily allow parts of the database to be presented to different users in different ways without changing the underlying table definitions. For example the finance department of a company may need to see employee financial information where as the personnel department may wish to see similar information but organised in a different way reflecting the use they have for such information. Relational databases allow users to have have different presentations of the database information which suit their different purposes for that information.

- **Standard language support.** The Relational Database Community have defined a standard language for database query and control. This language is therefore common across ALL implementations of relational databases.

- **Data independence:** Information is presented to users in tables. However the underlying storage organisation and database file formats are completely hidden from the user. This allows **physical data independence** where the storage format may be changed without affecting the users view of the information. Likewise, changing the tables (by adding records or attributes) need not affect the users view of the database **(logical data independence)**

- **Dynamic data definition:** Relational query language allows table structures to be modified at run-time which means that the database need not be recompiled or re-booted.

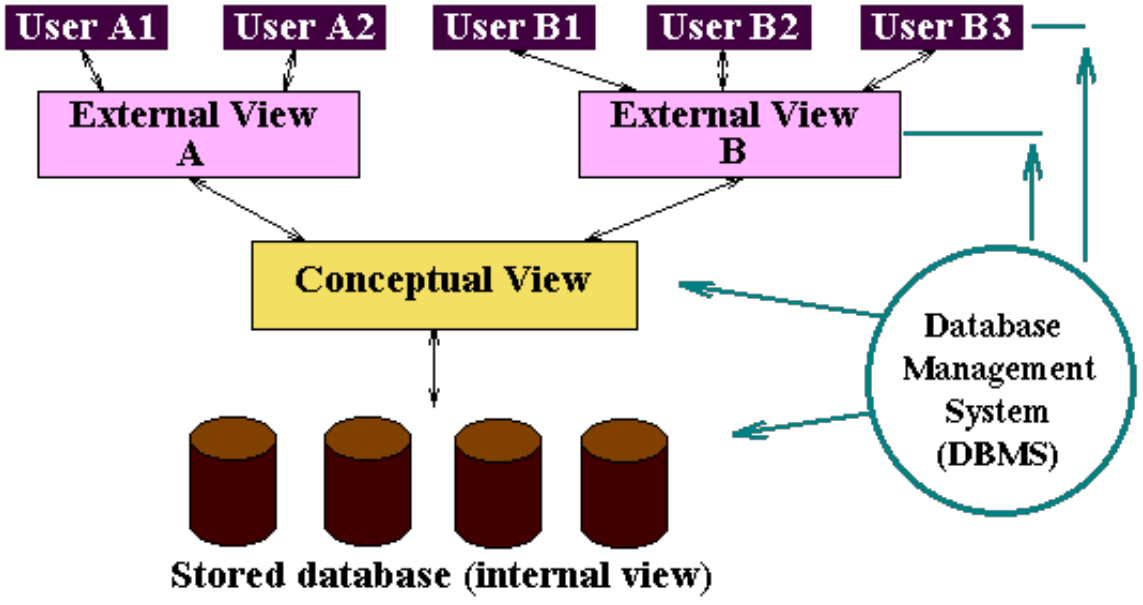
- **Integrated database dictionary:** Relational database automatically maintain information about all information contained in the database. For example it maintains dictionary (catalog) tables which identify what attributes are stored in what tables, what data-types make up each attribute, the sizeof all tables etc.

- **Ease of application development:** Since the user need only concern himself/herself with WHAT information to retrieve (rather than how to perform the retrieval), application development time is considerably reduced as no file and record programming need be written.

- **Distribution support:** Most relational databases now provide distributed access to the database (client server) where the database, although actually stored on a remote machine (server), appears to be locally available.
- **Concurrency Control and Security:** Relational databases provide mechanisms to enforce serialisation of queries and provide authorisation and authentication of the database.

Relational Database Management Systems conform to the 3 - schema architecture which is a layering of different levels of the information in a database:

1. The **External View** is the view presented to the actual user or application programmer. In this view, the information is presented in such a form as to suit the intended use of the information. Each view typically describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
2. The **Conceptual Level** contains a description the structure of the whole database for a community of users. This description is called the **Conceptual Schema** of a database and is a global description of the database that hides the details of physical storage structures and concentrates on describing entities, data types, relationships, and constraints.
3. The **Internal Level** contains the **internal schema.** This internal schema describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.



RDBMS Architecture

SQL Course

Structured Query Language

- Much of the user benefits of relational database technology are offered by the powerful relational query languages.
- These languages exploit the relational theory to provide ad hoc query and data management.
- The standard language for defining, constructing and manipulating information in a relational database is called **SQL**(Structured Query Language).
- The **SQL**(pronounced **sequel**) has been standardised over the last ten years and is now the most widely used database query language.

What is SQL?

- SQL is the **standard** relational query language and is based on relational algebra.
- It is between a third and fourth generation language. (3GL/4GL)

SQL supports the:

- Creation and maintenance of database information.
- Rapid prototyping and testing as it makes information retrieval and Storage easy.
- Porting of database application programs onto different RDBMS implementations (with only minor alterations required) since SQL is a standard language.
- Control over security, concurrency and locking (of tables, rows and values).

SQL is not just an information retrieval language, it has 3 sub languages:

- **Data Definition Language:** allows definition of relations (tables), their content type and integrity rules which govern attribute usage and values.
- **Data Manipulation Language:** allows user to populate, manipulate, retrieve and delete information in the database.
- **Data Control Language:** allows users and database administrators to specify security checks and controls on the database.

The story of SQL includes the following milestones:

- **1970** Codd publishes relational paper.
- **1973/75** University Ingres and System R prototypes.
- **1980** Oracle product released.
- **1983** IBM release DB2 & SQL/DS.
- **1986** ANSI standard SQL.
- **1989** Updated ANSI & ISO standard for SQL.
- **1992** Latest SQL standard (ISO & ANSI).*[Known as **SQL2**]*

The benefits of the adoption of the SQL standard are:

- Reduced training costs as the language is common across all RDBMS implementations.
- Increases application longevity as the applications use the standard which evolves slowly over time and maintains backward compatibility.

- Increases application portability as all RDBMS vendors support the SQL standard.
- Provides a basis for inter system communication.
- Simplifies customer choice as comparison of different vendor offerings is easier.

Caveat: Although all RDBMS vendors implement the SQL standard, several dialects have evolved where vendors have added in extra features for their customer benefits. These dialects differ very slightly between each vendor RDBMS.

- The **SQL** syntax is designed to be simple and consistent in its construction
- The language supports "English like" statements: i.e. say what you want, from where you want it, and under what conditions
- Statements are always terminated with a semi-colon **;**

Example

Consider a table called 'aircraft' which contains different aircraft names, models, number of club class seats, number of economy class seats and the call sign of each aircraft as shown in the diagram below.

aircraft name	*model*	*club seats*	*econ seats*	*call sign*
Eagle Flyer	ATR42	22	40	N410C
(NULL)	Boeing 707-320C	50	102	9J-AEB
(NULL)	Boeing 727-200	34	100	N7255U
Finians Dream	Boeing 737-200	22	96	DQ-FDM
(NULL)	Boeing 737-200	8	120	N301SW

To find out the call signs of the aircraft and the number of club class seats on each aircraft of model 'Boeing 737-200', the SQL query would be:

SELECT *call_sign, no_club_seats*
FROM *aircraft*
WHERE *model* = 'Boeing 737-200';

Although the central core of a database management system is it's information storage and query abilities, many other commonly used facilities are usually provided by a Relational Database System. Here are just a few of the most widely used database tools:

- **Report Writers:** These allow the automatic production of printed formatted reports based on current database values.
- **Screen Generators Forms:**These allow the rapid development screens or forms to facilitate data entry or data querying.
- **Network Connectivity Tools:** These allow applications which use the relational database to reside on a different computer (perhaps in a different location) to use the relation database without the user being aware of the distribution over the network.
- **Support for Embedded 3rd Generation Languages:** This allows programs written in 3rd generation programming languages like PASCAL, C, etc. to make runtime queries to a relational database. They usually take the form of either a Library of functions which the programmer can invoke with his/her program or a means of actually specifying relational queries in the programming language which a special pre-compiler processes and translates into appropriate runtime calls to the database.

- **CASE dictionaries.**

Many relational Database vendors distinguish their products by the number, sophistication, and performance of such database tools.

SQL Course

Data Types

- Illustrate and describe the various data types and constants available in the SQL language.
- Describe how SQL handles missing or unknown values using the **NULL** keyword.

SQL2 Data Types

- A database is a named allocation of storage space that will contain tables, views, indexes and other database objects.
- Every table is made up of a collection columns in which data will be stored.
- All data that is stored in a given column must be of the same data type.
- The data type is therefore an attribute of the column.

The data type of a column specifies what kind of information the column will hold (characters, numbers, dates and so on), so that the system will know how the data is to be physically stored, and how it can be manipulated. The data that will be stored in these columns will vary greatly from numeric numbers to names and places to dates and times. As a result of this the database must support certain data types. The ANSI/ISO standards specify various types of data that can be stored in an SQL2 compliant database and hence be manipulated by the SQL2 language. The SQL2 data types are listed below:

SQL2 Data Types

Data Type	Description
CHAR(len) **CHARACTER**(len)	Fixed length character strings
VARCHAR(len) **CHAR VARYING**(len) **CHARACTER VARYING**(len)	Variable length character strings*
NCHAR(len) **NATIONAL CHAR**(len) **NATIONAL CHARACTER**(len)	Fixed length national character strings*
NCHAR VARYING(len) **NATIONAL CHAR VARYING**(len) **NATIONAL CHARACTER VARYING**(len)	Variable length national character strings*
INTEGER **INT**	Integer numbers
SMALLINT	Small integer numbers

BIT(len)	Fixed length bit string*
BIT VARYING(len)	Variable length bit string*
NUMERIC(precision, scale) **DECIMAL**(precision, scale) **DEC**(precision, scale)	Decimal numbers
FLOAT(precision)	Floating point numbers
REAL	Low precision floating point numbers
DOUBLE PRECISION	High precision floating point numbers
DATE	Calendar Date*
TIME(precision)	Clock time*
TIMESTAMP(precision)	Date and time*
INTERVAL	Time interval*

String

Character strings are either fixed or variable length. Columns of variable length are allowed to store character strings that vary in length from row to row, up to some maximum length. Different character sets may be used when assigning a data type to a column. The different character sets that are supported will depend on the specific product that will be used. The character set is assigned by specifying **'CHARACTER SET Kanji'** for example after the character string data type (Kanji is a Japanese character set).**NATIONAL CHARACTER** is provided as a shorthand to conform to previous Japanese SQL standards in an upwardly compatible manner.

Bit

The bit data type is used to define bit strings. A bit string is a sequence of binary digits(bits), each having the value 0 or 1. The bit data type is intended to allow for data whose structure is unknown or is not supported by SQL to be stored in SQL tables.

Date-Time

The SQL2 standard supports an elaborate specification for **DATE**, **TIME**, **TIMESTAMP** and **INTERVAL** data types, including support for time zones and time precision. Every date-time data type consists of a contiguous subset of the fields: **YEAR**, **MONTH**, **DAY**, **HOUR**, **MINUTE**, **SECOND**, **TIMEZONE_HOUR**, **TIMEZONE_MINUTE** . Each of the fields are listed in significant order with the most significant being listed first.

The ANSI/ISO SQL2 standard specifies the format of numeric and string constants, or literals, which represent specific data values. Constants are categorised as follows:

Numeric

Integer or decimal constants or exact numeric literals as they are also known are written as ordinary decimal numbers in SQL statements, with an optional leading plus or minus

sign. Floating point constants or approximate numeric literals as they are also known are specified using scientific notation.

String

The ANSI/SQL standard specifies that SQL constants for character data be enclosed in single quotes. Some implementations allow the use of double quotes but this causes problems in portability to other implementations.

Date and Time

Constant data values for dates, times and time intervals are specified as string constants.

Symbolic

In addition to user supplied constants, the SQL language includes special symbolic constants that return data values maintained by the DBMS itself. The SQL1 standard specified only one symbolic constant which was the **USER** constant. The SQL2 standard supports the following symbolic constants: **CURRENT_DATE**, **CURRENT_TIME**, **CURRENT_TIMESTAMP**, **USER**, **SESSION_USER** and **SYSTEM_USER**.

In practical real world scenarios where databases are used frequently there often arise situations where data is either not applicable, not known or doesn't apply in a particular situation. In our example database we have a table called **customers**. In this table the customers telephone number is stored. It may be the case that some customers may not have a telephone. What do we do in this situation? It would be incorrect to place a zero in lieu of a missing telephone number as the persons telephone number is not zero, it is just not known. SQL supports missing, unknown or inapplicable data explicitly, through the concept of a **NULL** value. A null value is an indicator that tells SQL and the user, that the data is missing or not applicable. This is a special value used to signify the absence of a value. It is distinct from a numeric zero or a blank character. It is a signal that the data value is missing or unknown. A **NULL** value takes up no storage space in a row. Due to its unusual nature the **NULL** value requires special consideration when it is encountered by other statements in the SQL language.