**Lesson 9**

**Objectives**

* Database Languages
  + Data Definition Languages
  + Data Manipulation Languages
    - Procedural
    - Non-Procedural

**Database Languages**

Answers the following questions

* How does an application interact with DBMS?
* How does a user look at a database systems?
* How can a user query a database system and view the result in his/her application?

A data sublanguage consists of two parts:

* Data Definition Language (DDL)
* Data Manipulation Language (DML).

The DDL is used to specify the database schema and the DML is used to both read and update the database. These languages are called data sublanguages because they do not include constructs for all computing needs such as conditional or iterative statements, which are provided by the high-level programming languages. Many DBMSs have a facility for embedding the sublanguage in a high-level programming language such as COBOL, Fortran, Pascal, Ada, ‘C’, C++, Java, or Visual Basic. In this case, the high-level language is sometimes referred to as the host language.

To compile the embedded file, the commands in the data sublanguage are first removed from the host-language program and replaced by function calls. The pre-processed file is then compiled, placed in an object module, linked with a DBMS-specific library containing the replaced functions, and executed when required. Most data sublanguages also provide non-embedded, or interactive, commands that can be input directly from a terminal.

**Data Definition Language (DDL)**

A language that allows the DBA or user to describe and name the entities, attributes, and relationships required for the application, together with any associated integrity and security constraints.

The database schema is specified by a set of definitions expressed by means of a special language called a Data Definition Language. The DDL is used to define a schema or to modify an existing one. It cannot be used to manipulate data.

The result of the compilation of the DDL statements is a set of tables stored in special files collectively called the system catalog. The system catalog integrates the metadata, that is data that describes objects in the database and makes it easier for those objects to be accessed or manipulated. The metadata contains definitions of records, data items, and other objects that are of interest to users or are required by the DBMS. The DBMS normally consults the system catalog before the actual data is accessed in the database. The terms data dictionary and data directory are also used to describe the system catalog, although the term ‘data dictionary’ usually refers to a more general software system than a catalog for a DBMS. At a theoretical level, we could identify different DDLs for each schema in the three level architecture, namely a DDL for the external schemas, a DDL for the conceptual schema, and a DDL for the internal schema. However, in practice, there is one comprehensive DDL that allows specification of at least the external and conceptual schemas.

**Data Manipulation Language (DML)**

A language that provides a set of operations to support the basic data manipulation operations on the data held in the database.

Data manipulation operations usually include the following:

* insertion of new data into the database;
* modification of data stored in the database;
* retrieval of data contained in the database;
* deletion of data from the database.

Therefore, one of the main functions of the DBMS is to support a data manipulation language in which the user can construct statements that will cause such data manipulation to occur. Data manipulation applies to the external, conceptual, and internal levels. However, at the internal level we must define rather complex low-level procedures that allow efficient data access. In contrast, at higher levels, emphasis is placed on ease of use and effort is directed at providing efficient user interaction with the system.

The part of a DML that involves data retrieval is called a query language. A query language can be defined as a high-level special-purpose language used to satisfy diverse requests for the retrieval of data held in the database. The term ‘query’ is therefore reserved to denote a retrieval statement expressed in a query language. The terms ‘query language’ and ‘DML’ are commonly used interchangeably, although this is technically incorrect.

DMLs are distinguished by their underlying retrieval constructs. We can distinguish between two types of DML: procedural and non-procedural. The prime difference between these two data manipulation languages is that procedural languages specify how the output of a DML statement is to be obtained, while non-procedural DMLs describe only what output is to be obtained. Typically, procedural languages treat records individually, whereas non-procedural languages operate on sets of records.

**Procedural DMLs**

Procedural A language that allows the user to tell the system what data is needed and exactly how to retrieve the data.

With a procedural DML, the user, or more normally the programmer, specifies what data is needed and how to obtain it. This means that the user must express all the data access operations that are to be used by calling appropriate procedures to obtain the information required. Typically, such a procedural DML retrieves a record, processes it and, based on the results obtained by this processing, retrieves another record that would be processed similarly, and so on. This process of retrievals continues until the data requested from the retrieval has been gathered. Typically, procedural DMLs are embedded in a high-level programming language that contains constructs to facilitate iteration and handle navigational logic.

**Non-procedural DMLs**

Non-procedural A language that allows the user to state what data is needed rather than how it is to be retrieved.

Non-procedural DMLs allow the required data to be specified in a single retrieval or update statement. With non-procedural DMLs, the user specifies what data is required without specifying how it is to be obtained. The DBMS translates a DML statement into one or more procedures that manipulate the required sets of records. This frees the user from having to know how data structures are internally implemented and what algorithms are required to retrieve and possibly transform the data, thus providing users with a considerable degree of data independence. Non-procedural languages are also called declarative languages.

Relational DBMSs usually include some form of non-procedural language for data manipulation, typically SQL (Structured Query Language) or QBE (Query-By-Example). Non-procedural DMLs are normally easier to learn and use than procedural DMLs, as less work is done by the user and more by the DBMS

**Fourth-Generation Languages (4GLs)**

There is no consensus about what constitutes a fourth-generation language; it is in essence a shorthand programming language. An operation that requires hundreds of lines in a third-generation language (3GL), such as COBOL, generally requires significantly fewer lines in a 4GL.

Compared with a 3GL, which is procedural, a 4GL is non-procedural: the user defines what is to be done, not how. A 4GL is expected to rely largely on much higher-level components known as fourth-generation tools. The user does not define the steps that a program needs to perform a task, but instead defines parameters for the tools that use them to generate an application program. It is claimed that 4GLs can improve productivity by a factor of ten, at the cost of limiting the types of problem that can be tackled. Fourth generation languages encompass:

* presentation languages, such as query languages and report generators;
* specialty languages, such as spreadsheets and database languages;
* application generators that define, insert, update, and retrieve data from the database to
* build applications;
* very high-level languages that are used to generate application code.

SQL and QBE, mentioned above, are examples of 4GLs. We now briefly discuss some of the other types of 4GL.

**Forms generators**

A forms generator is an interactive facility for rapidly creating data input and display layouts for screen forms. The forms generator allows the user to define what the screen is to look like, what information is to be displayed, and where on the screen it is to be displayed. It may also allow the definition of colors for screen elements and other characteristics, such as bold, underline, blinking, reverse video, and so on. The better forms generators allow the creation of derived attributes, perhaps using arithmetic operators or aggregates, and the specification of validation checks for data input.

**Report generators**

A report generator is a facility for creating reports from data stored in the database. It is similar to a query language in that it allows the user to ask questions of the database and retrieve information from it for a report. However, in the case of a report generator, we have much greater control over what the output looks like. We can let the report generator automatically determine how the output should look or we can create our own customized output reports using special report-generator command instructions.

There are two main types of report generator:

* language-oriented
* visually oriented.

In the first case, we enter a command in a sublanguage to define what data is to be included in the report and how the report is to be laid out. In the second case, we use a facility similar to a forms generator to define the same information.

**Graphics generators**

A graphics generator is a facility to retrieve data from the database and display the data as a graph showing trends and relationships in the data. Typically, it allows the user to create bar charts, pie charts, line charts, scatter charts, and so on.

**Application generators**

An application generator is a facility for producing a program that interfaces with the database.

The use of an application generator can reduce the time it takes to design an entire software application. Application generators typically consist of pre-written modules that comprise fundamental functions that most programs use. These modules, usually written in a high-level language, constitute a ‘library’ of functions to choose from. The user specifies what the program is supposed to do; the application generator determines how to perform the tasks.