Database Dictionary

Provided by GeekGirls.com

http://www.geekgirls.com/database_dictionary.htm

database: A collection of related information stored in a structured format. *Database* is sometimes used interchangeably with the term **table**. Technically, they're different: A table is a single store of related information; a database can consist of *one or more tables* of information that are related in some way. For instance, you could track all the information about the students in a school in a students table. If you then created separate tables containing details about teachers, classes and classrooms, you could combine all four tables into a timetabling database. Such a multi-table database is called a *relational database*.

data entry: The process of getting information into a database, usually done by people typing it in by way of data-entry forms designed to simplify the process.

dbms: Database management system. A program which lets you manage information in **databases**. Lotus Approach, *Microsoft Access* and FileMaker Pro, for example, are all DBMSs, although the term is often shortened to 'database'. So, the same term is used to apply to the program you use to organize your data and the actual data structure you create with that program.

field: Fields describe a single aspect of each member of a **table**. A student **record**, for instance, might contain a last name field, a first name field, a date of birth field and so on. All records have exactly the same structure, so they contain the same fields. The *values* in each field vary from record to record, of course. In some database systems, you'll find fields referred to as *attributes*.

flat file: A **database** that consists of a single **table**. Lightweight database programs such as the database component in Microsoft Works are sometimes called 'flat-file managers' (or list managers) because they can only handle single-table databases. More powerful programs, such as FileMaker Pro, Access, Approach and Paradox, can handle multi-table databases, and are called **relational database** managers, or **RDBMSs**.

foreign key: A key used in one **table** to represent the value of a **primary key** in a related table. While primary keys must contain unique values, foreign keys may have duplicates. For instance, if we use student ID as the primary key in a Students table (each student has a unique ID), we could use student ID as a foreign key in a StudentCourse table: as each student may do more than one course, the student ID field in the StudentCourse table (often shortened to StudentCourse.student ID) will hold duplicate values.

index: A summary table which lets you quickly look up the contents of any **record** in a **table**. Think of how you use an index to a book: as a quick jumping off point to finding full information about a subject. A database index works in a similar way. You can create an index on any **field** in a table. Say, for example, you have a customer table which contains customer numbers, names, addresses and other details. You can make indexes based on any information, such as the customers' customer number, last name + first name (a composite index based on more than one field), or postal code. Then, when you're searching for a particular customer or group of customers, you can use the index to speed

up the search. This increase in performance may not be noticeable in a table containing a hundred records; in a database of thousands of records it will be a blessing.

key field: You can sort and quickly retrieve information from a **database** by choosing one or more **fields** to act as *keys*. For instance, in a students table you could use a combination of the last name and first name fields (or perhaps last name, first name and birth dates to ensure you identify each student uniquely) as a key field. The database program will create an **index** containing just the key field contents. Using the index, you can quickly find any **record** by typing in the student's name. The database will locate the correct entry in the index and then display the full record.

Key fields are also used in **relational databases** to maintain the structural integrity of your tables, helping you to avoid problems such as duplicate records and conflicting values in fields (see **primary key** and **foreign key**).

normalization: The process of structuring data to minimize duplication and inconsistencies. The process usually involves breaking down a single **table** into two or more tables and defining relationships between those tables. Normalization is usually done in stages, with each stage applying more rigorous rules to the types of information which can be stored in a table. While full adherence to normalization principles increases the efficiency of a particular database, the process can become so esoteric that you need a professional to create and understand the table design. Most people, when creating a database, don't need to go beyond the third level of normalization, called *third normal form*. And there's no need to know the terminology: simply applying the principles is sufficient.

The first three levels in normalizing a database are:

First Normal Form (1NF): There should be no repeating groups in a table.

For example, say you have a students table with the following structure:

student ID
name
date of birth
advisor
advisor's telephone
student
course ID 1
course description 1
course instructor 1
course ID 2
course description 2
course instructor 2

The repeating course fields are in conflict with first normal form. To fix the problems created by such repeating fields, you should place the course information in a separate course table, and then provide a linking field (most likely student ID) between the students table and the course table.

Second Normal Form (2NF): No non-key fields may depend on a *portion* of the primary key.

For example, say we create a course table with the structure:

student ID
course ID
course description
course instructor

We can create a unique primary key by combining student ID + course ID (student ID is not unique in itself, as one student may take multiple courses; similarly, course ID is not unique in itself as many students may take the same course; however, each student will only be taking a particular course once at any one time, so the combination of student ID + course ID gives us a unique primary key).

Now, in 2NF, no non-key fields (course description, course instructor) may depend on a *portion* of the primary key. That, however, is exactly what we have here: the course instructor and course description are the same for any course, regardless of the student taking the course.

To fix this and put the database in second normal form, we create a third table, so our database structure now looks like this (with key fields in italics):

Student table

student ID
name
date of birth
advisor
advisor's telephone

Student courses table

student ID
course ID

Courses table

course ID
course description
course instructor

Third Normal From (3FN): No fields may depend on other non-key fields. In other words, each field in a record should contain information about the entity that is defined by the primary key.

In our students table, for example, each field should provide information about the particular student referred to by the key field, *student ID*. That certainly applies to the student's name and date of birth. But the advisor's name and telephone doesn't change

depending on the student. So, to put this database in third normal form, we need to place the advisor's information in a separate table:

Student table

student ID
name
date of birth
advisor ID

Student courses table

student ID
course ID

Courses table

course ID
course description
course instructor

Advisor table

advisor ID
advisor name
advisor telephone

primary key: A field that *uniquely* identifies a record in a table. In a students table, for instance, a key built from last name + first name might not give you a unique identifier (two or more Jane Does in the school, for example). To uniquely identify each student, you might add a special Student ID field to be used as the primary key.

query: A view of your data showing information from one or more tables. For instance, using the sample database we used when describing **normalization**, you could query the Students database asking "Show me the first and last names of the students who take both history and geography and have Alice Hernandez as their advisor" Such a query displays information from the Students table (firstname, lastname), Courses table (course description) and Advisor table (advisor name), using the keys (student ID, course ID, advisor ID) to find matching information.

rdbms: Relational database management system. A program which lets you manage structured information stored in **tables** and which can handle **databases** consisting of multiple tables.

record: A record contains all the information about a single 'member' of a **table**. In a students table, each student's details (name, date of birth, contact details, and so on) will be contained in its own record. Records are also known as *tuples* in technical **relational database** parlance.

relational database: A **database** consisting of more than one **table**. In a multi-table database, you not only need to define the structure of each table, you also need to define the relationships between each table in order to link those tables correctly.

report: A form designed to print information from a database (either on the screen, to a file or directly to the printer).

SQL: Structured Query Language (pronounced *sequel* in the US; *ess-queue-ell* elsewhere). A computer language designed to organize and simplify the process of getting information out of a database in a usable form, and also used to reorganize data within databases. SQL is most often used on larger databases on minicomputers, mainframes and corporate servers.

table: A single store of related information. A table consists of **records**, and each record is made up of a number of **fields**. Just to totally confuse things, tables are sometimes called *relations*. You can think of the phone book as a table: It contains a record for each telephone subscriber, and each subscriber's details are contained in three fields – name, address and telephone.