ARCHITECTURE OF DBMS

Example of a simplified database catalog

RELATIONS

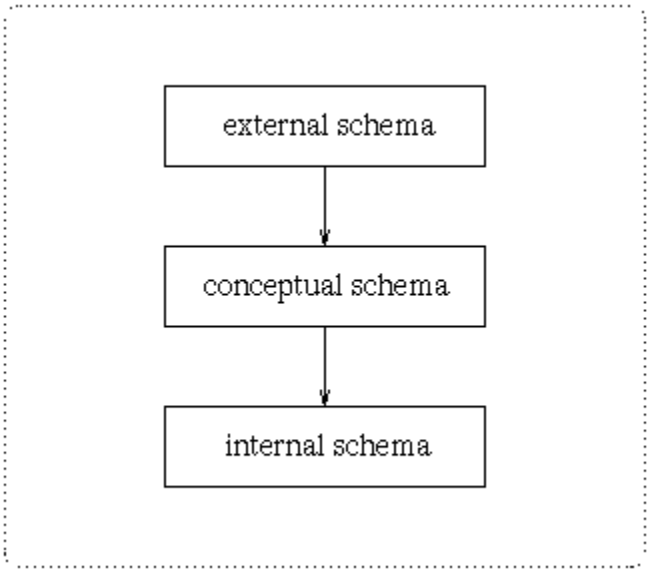
Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.



A commonly used views of data approach is the three-level architecture suggested by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee). ANSI/SPARC produced an interim report in 1972 followed by a final report in 1977. The reports proposed an architectural framework for databases. Under this approach, a database is considered as containing data about an *enterprise*. The three levels of the architecture are three different views of the data:

External - individual user view

Conceptual - community user view

Internal - physical or storage view

The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout. A database system that is able to separate the three different views of data is likely to be flexible and adaptable. This flexibility and adaptability is data independence that we have discussed earlier.

We now briefly discuss the three different views.

The external level is the view that the individual user of the database has. This view is often a restricted view of the database and the same database may provide a number of different views for different classes of users. In general, the end users and even the application programmers are only interested in a subset of the database. For example, a department head may only be interested in the departmental finances and student enrolments but not the library information. The librarian would not be expected to have any interest in the information about academic staff. The payroll office would have no interest in student enrolments.

The conceptual view is the information model of the enterprise and contains the view of the whole enterprise without any concern for the physical implementation. This view is normally more stable than the other two views. In a database, it may be desirable to change the internal view to improve performance while there has been no change in the

conceptual view of the database. The conceptual view is the overall community view of the database and it includes all the information that is going to be represented in the database. The conceptual view is defined by the conceptual schema which includes definitions of each of the various types of data.

The internal view is the view about the actual physical storage of data. It tells us what data is stored in the database and how. At least the following aspects are considered at this level:

Storage allocation e.g. B-trees, hashing etc.

Access paths e.g. specification of primary and secondary keys, indexes and pointers and sequencing.

Miscellaneous e.g. data compression and encryption techniques, optimization of the internal structures.

Efficiency considerations are the most important at this level and the data structures are chosen to provide an efficient database. The internal view does not deal with the physical devices directly. Instead it views a physical device as a collection of physical pages and allocates space in terms of logical pages.

The separation of the conceptual view from the internal view enables us to provide a logical description of the database without the need to specify physical structures. This is often called *physical data independence*. Separating the external views from the conceptual view enables us to change the conceptual view without affecting the external views. This separation is sometimes called *logical data independence*.

Assuming the three level view of the database, a number of mappings are needed to enable the users working with one of the external views. For example, the payroll office may have an external view of the database that consists of the following information only:

Staff number, name and address.

Staff tax information e.g. number of dependents.

Staff bank information where salary is deposited.

Staff employment status, salary level, leave information etc.

The conceptual view of the database may contain academic staff, general staff, casual staff etc. A mapping will need to be created where all the staff in the different categories are combined into one category for the payroll office. The conceptual view would include information about each staff's position, the date employment started, full-time or part-time etc. This will need to be mapped to the salary level for the salary office. Also, if there is some change in the conceptual view, the external view can stay the same if the mapping is changed.