NET, IAS, State-SET (KSET, WBSET, MPSET, etc.), GATE, CUET, Olympiads etc.: Software Engineering and Database

The term 'software engineering' implies the application of scientific knowledge and discipline to the construction of computer software systems. In practice, this involves the production of a specification, a problem solution, and a program design prior to the inception of actual coding. Ideally therefore, software development should be like mathematics or any other branch of science or engineering. Where one adopts a methodical approach to solving a problem and most importantly one is required to show that the solution is correct, i.e. … that it solves the given problem. Die following sections outline some of the most important issues in software engineering and show how they relate to die subject of database technology.

Data Abstraction

One of the most important techniques for dealing with complexity in software systems is the principle of abstraction. Abstraction is used to construct solutions to problems without, having to take account of the intricate details of the various component sub-problems. The program design phase of any software project involves abstraction as a means for developing representations of the complex data objects and concepts inherent in the application. Abstraction allows the programmer to apply the principles of stepwise refinement by which at each stage of program design unnecessary details associated with representation or implementation may be hidden from the surrounding program. This ability to decompose a large problem into a number of smaller ones, and to specify unambiguously the interactions among the components, is a vital tool in the construction of large software systems.

Database systems are typically large software systems, which are distinguished by the fact that they involve the processing of data with three important characteristics:

- There is a large amount of data, which must be held in external storage.
- The data objects have complex interrelationships.
- The data must be shared among many different users.

Very few of the popular programming languages provide adequate constructs for the efficient processing of large volumes of interrelated data. The simple sequential and random file structures found in most languages offer only minimal facilities in this area. However many modern languages do offer very powerful algorithmic constructs for the representation of complex data objects. The ability to define such abstract objects, together with appropriate operations, is invaluable for database applications.

However, any language which supports abstraction allows the programmer to build more sophisticated file management structures which may provide a range of high-level operations for the manipulation of external data, while at the same time absolving the user of the mundane responsibility for reading from and writing to actual Physical storage devices. In addition, abstraction facilities allow one to create complex data objects within a program which model closely those used in the problem solution. Thus the process of transforming the problem solution into a program is greatly eased.

Concurrency

When several application programs or terminal users are interacting with a shared database simultaneously, their operations on the database must be carefully controlled. Specifically, the effect of a database procedure (often called a transaction) must be that which would be obtained if no father transaction were executing concurrently. The effect of executing several transactions concurrently, therefore, must be the same as if they had been executed serially in some order. By ‘effect’ here, we mean the final state of the database together with any results produced by the transactions.

To prevent any transaction from reading or updating data that are being updated by another transaction we could provide a locking mechanism which guarantees exclusive access to an item of data while a lock is in force. Although most operating systems provide for file locking, few support locks on pages or records. Such smaller locks are essential in many database environments and must therefore be provided by the database management system, A programming language which offers high-level facilities for concurrency control gives the database programmer great flexibility in scheduling transactions while preserving the integrity of the database.

Program Correctness

Program correctness has been a strong motivation for much of the work in software engineering and programming language design. Ideally correctness should be determined by a formal mathematical proof that a program conforms to its specification. However the current widespread practice of using natural language for specifications and machine-oriented languages for implementation is not conducive to presenting mathematical proofs of programs and this obligation is normally discharged by testing, i.e. … by showing that the solution solves some instances of the problem.

The construction of formal proofs of correctness of programs is beyond the scope of this work and is adequately covered by other authors (Brady, 1977; Mc Getterick, 1982) . However, we shall in this work be intimately concerned with those aspects of programming language design and programming style which influence the ease of program verification. It has long been recognized that both the characteristics of a given programming language and the practices used to write programs in the language have a significant influence on program correctness. Support in a language for concepts such as data abstraction and strong type-checking greatly ease program testing and verification. Also highly desirable is the ability to break a large program down into modules which may be separately compiled and tested in isolation.