Competitive Exams: Database: Life Cycle & Modelling

The Database Life Cycle

Any large software project may be divided up into several project phases. The five phases of the database life cycle are: Requirements analysis, data modeling, implementation, testing and maintenance. These are examined in detail below. However database development is an iterative process. Often the data modeling phase will highlight ambiguities or inconsistencies in the requirements analysis, while the implementation and test phases may indicate that errors were made in constructing the data model. A common scenario is for the user to place additional demands on the system following the implementation and testing phases. If the original design and implementation is poorly structured this iterative process can easily lead to a system which is of poor quality and which is difficult to maintain.

Requirements Analysis

The objective of the requirements analysis phase is to describe precisely the information content of the proposed database system and to determine the transaction demands that will be placed on the system. This analysis should lead to the production of a requirements specification, which is a detailed and precise description of the database system, its functions and its user interfaces. This requirements specification must be accepted by both users and database designer as complete and consistent and must therefore be clearly understood by everyone involved. For this reason, natural languages tend to be widely used for drafting requirements specifications. However, the imprecision and ambiguity of such languages tend to make it very difficult to test specifications for completeness and consistency. The discipline of software engineering has attempted to provide some methodical aids for constructing specifications, ranging from elementary aids such as flowcharts and decision tables to more elaborate techniques such as SADT (Structured Analysis and Design Technique) as described by Schoman and Ross (1977) . However most of these special techniques are rather inflexible and not well-suited to database design. Thus requirements specification remains, like data modeling and programming, a creative task for which a precise and comprehensive methodology has not yet been developed.

The first step in the requirements analysis phase is to identify the data objects present in the real-world situation to be modeled, the properties of those objects, and the associations among them.

Data Modelling

The quality of a database system is significantly influenced by the quality of the data model, and thus data modeling occupies an important position in the database life cycle. For that reason, a large portion of this book is devoted to the subject. The purpose of data modeling is to develop a global design for the database with the ultimate objective of achieving an efficient implementation which satisfies the requirements. As with any design process, data modeling requires a certain amount of ingenuity and creativity, but meaningful and effective guidelines can still be laid down:

Implementation

By implementation we mean the transformation of our design into a database system which operates on a particular machine, usually under the control of a database management system. The design of the system should ideally be independent of any particular DBMS, but not all such systems are equally well suited to the task of transforming the design into an efficient and easily-maintainable implementation.

Testing

The quality of a database system is measured first by its correctness and reliability, and second by the degree to which it satisfies the demands of the initial requirement analysis. The purpose of the testing phase therefore is twofold:

- To discover any errors that may have arisen during the modeling and implementation phases.
- To ascertain, in conjunction with the user community, whether the system satisfies the information demands of user and the requirements of application programs.

By error we mean any deviation from the behavior stipulated by the requirements analysis. Such errors may be caused by a faulty or inadequate requirements analysis, incorrect data modeling, or programming errors made during the implementation phase. Testing therefore is an activity which relates to the entire project and rather than be treated as an isolated phase, it should be distributed over all phases of the database life cycle, including the maintenance phase. For a complex database with a large number of different user groups, it may be necessary to develop prototype database systems in a number of iterations. At each stage of the iteration these prototypes can be discussed with the users to ensure that the needs of all applications are satisfied.

At the current level of sophistication in software engineering there is no universal recipe for testing software systems. The scope and number of tests will vary considerably from system to system. The effort put into this phase will depend on the complexity of the enterprise and the level of expertise available for the data modeling and implementation phases.

Note that it is also important to test the robustness of the system against incorrect user interaction and against errors in application programs.

Maintenance

As described above it is almost impossible to devise an exhaustive test phase for a large database system and in practice many errors and inadequacies in the system first come to light during actual use. These must be monitored and eliminated within the framework of the maintenance phase of the database life cycle. An exact, requirements analysis is also very difficult in many cases and so new user demands are often added during the operation of a database system which requires system modifications.

The maintenance phase involves:

- The correction of errors which arise during operation.
- The implementation of system modifications which arise due to new user requests or changes in the original user requirements.
- The implementation of performance enhancements and improvements to user interfaces.

As described the responsibility for maintaining a multiuser database system must reside centrally with the database administrator through whom all user requests and complaints are channeled. Database usage and performance must be constantly monitored with the assistance of DBMS utilities (such as a data dictionary) which provide statistics on important factors such as access times, storage utilization and transaction throughout. Corrections, modifications and enhancements to the system, whether at the logical or physical levels, must be carried out with the minimum of disruption to the user community.

Maintenance can be a very expensive part of any database project and it is therefore prudent to pay particular attention to the maintainability of the system. Designing and implementing a database system which is easily maintained means ensuring a good modular structure with good supporting documentation.

Developed by: Mindsprite Solutions