Review exercises

Friday, February 09, 2024 5:57 PM

QUESTION 1

Data Independence: Application programs should not, ideally, be ex□posed to details of data representation and storage. The DBMS provides an abstract view of the data that hides such details. Efficient Data Access: A DBMS utilizes a variety of sophisticated tech□niques to store and retrieve data efficiently. This feature is especially im□portant if the data is stored on external storage devices. Data Integrity and Security: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce access controls that govern what data is visible to different classes of users. Data Administration: When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient. Concurrent Access and Crash Recovery: A DBMS schedules concur□rent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures. Reduced Application Development Time: Clearly, the DBMS sup□ports important functions that are common to many applications accessing data in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick application development. DBMS applications are also likely to be more robust than similar stand-alone applications because many important tasks are handled by the DBMS (and do not have to be debugged and tested in the application)

QUESTION 2

We probably do not have 500 GB of main memory to hold all the data. We must therefore store data in a storage device such as a disk or tape and bring relevant parts into main memory for processing as needed. Even if we have 500 GB of main memory, on computer systems with 32-bit addressing, we cannot refer directly to more than about 4 GB of data. We have to program some method of identifying all data items. We have to write special programs to answer each question a user may want to ask about the data. These programs are likely to be complex because of the large volume of data to be searched. We must protect the data from inconsistent changes made by different users accessing the data concurrently. If applications must address the details of such concurrent access, this adds greatly to their complexity. We must ensure that data is restored to a consistent state if the system crashes while changes are being made. Operating systems provide only a password mechanism for security. This is not sufficiently flexible to enforce security policies in which different users have permission to access different subsets of the data

QUESTION 3

A data model is a collection of high-level data description constructs that hide many low-level storage details. A DBMS allows a user to define the data to be stored in terms of a data model. Most database management systems today are based on the relational data model, which we focus on in this book. While the data model of the DBMS hides many details, it is nonetheless closer to how the DBMS stores data than to how a user thinks about the underlying enterprise. A semantic data model is a more abstract, high-level data model that makes it easier for a user to come up with a good initial description of the data in an enterprise.

A very important advantage of using a DBMS is that it offers data indepen□dence. That is, application programs are insulated from changes in the way the data is structured and stored. Data independence is achieved through use of the three levels of data abstraction; in particular, the conceptual schema and the external schema provide distinct benefits in this area. Relations in the external schema (view relations) are in principle generated on demand from the relations corresponding to the conceptual schema.4 If the underlying data is reorganized, that is, the conceptual schema is changed, the definition of a view relation can be modified so that the same relation is computed as before

users can be shielded from changes in the logical structure of the data, or changes in the choice of relations to be stored. This property is called logical data independence. In turn, the conceptual schema insulates users from changes in physical storage details. This property is referred to as physical data independence. The conceptual schema hides details such as how the data is actually laid out on disk, the file structure, and the choice of indexes.

QUESTION 4

questions involving the data stored in a DBMS are called queries. A DBMS provides a specialized language, called the query language, in which queries can be posed. A very attractive feature of the relational model is that it supports powerful query languages. Relational calculus is a formal query language based on mathematical logic, and queries in this language have an intuitive, precise meaning. Relational algebra is another formal query language, based on a collection of operators for manipulating relations, which is equivalent in power to the calculus.

QUESTION 5

A transaction is any one execution of a user program in a DBMS. (Executing the same program several times will generate several transactions.) This is the basic unit of change as seen by the DBMS: Partial transactions are not allowed, and the effect of a group of transactions is equivalent to some serial execution of all transactions.

Every object that is read or written by a transaction is first locked in shared or exclusive mode, respectively. Placing a lock on an object restricts its availability to other transactions and thereby affects performance. 2. For efficient log maintenance, the DBMS must be able to selectively force a collection of pages in main memory to disk. Operating system support for this operation is not always satisfactory. 3. Periodic checkpointing can reduce the time needed to recover from a crash. Of course, this must be balanced against the fact that checkpointing too often slows down normal execution

Mechanism 1
A locking protocol is a set of rules to be followed by each transaction (and enforced by the DBMS) to ensure that, even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions in some serial order. A lock is a mechanism used to control access to database objects. Two kinds of locks are commonly supported by a DBMS: shared locks on an object can be held by two different transactions at the same time, but an exclusive lock on an object ensures that no other transactions hold any lock on this object. Suppose that the following locking protocol is followed: Every transaction begins by obtaining a shared lock on each data object that it needs to read and an exclusive lock on each data object that it needs to modify, then releases all its locks after completing all actions. Consider two transactions T1 and T2 such that T1 wants to modify a data object and T2 wants to read the same object. Intuitively, if T1's request for an exclusive lock on the object is granted first, T2 cannot proceed until T1 releases this lock, because T2's request for a shared lock will not be granted by the DBMS until then. Thus, all of T1's actions will be completed before any of T2's actions are initiated.

Mechanism 2
the DBMS maintains a log of all writes to the database. A crucial property of the log is that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself—otherwise, if the system crashes just after making the change in the database but before the change is recorded in the log, the DBMS would be unable to detect and undo this change. This property is called Write-Ahead Log, or WAL.
The log is also used to ensure that the changes made by a successfully completed transaction are not lost due to a system crash, as explained in Chapter 18. Bringing the database to a consistent state after a system crash can be a slow process, since the DBMS must ensure that the effects of all transactions that completed prior to the crash are restored, and that the effects of incomplete transactions are undone. The time required to recover from a crash can be reduced by periodically forcing some information to disk; this periodic operation is called a checkpoint.

QUESTION 6
A locking protocol is a set of rules to be followed by each transaction (and enforced by the DBMS) to ensure that, even though actions of several transactions might be interleaved, the net effect is identical to executing all transactions in some serial order. A lock is a mechanism used to control access to database objects. Two kinds of locks are commonly supported by a DBMS: shared locks on an object can be held by two different transactions at the same time, but an exclusive lock on an object ensures that no other transactions hold any lock on this object. Suppose that the following locking protocol is followed: Every transaction begins by obtaining a shared lock on each data object that it needs to read and an exclusive lock on each data object that it needs to modify, then releases all its locks after completing all actions. Consider two transactions T1 and T2 such that T1 wants to modify a data object and T2 wants to read the same object. Intuitively, if T1's request for an exclusive lock on the object is granted first, T2 cannot proceed until T1 releases this lock, because T2's request for a shared lock will not be granted by the DBMS until then. Thus, all of T1's actions will be completed before any of T2's actions are initiated.
the DBMS maintains a log of all writes to the database. A crucial property of the log is that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself—otherwise, if the system crashes just after making the change in the database but before the change is recorded in the log, the DBMS would be unable to detect and undo this change. This property is called Write-Ahead Log, or WAL.

QUESTION 7
The DBMS accepts SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers. (This is a simplification: SQL commands can be embedded in host-language application programs, e.g., Java or COBOL programs.
When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. Execution plan is a blueprint for evaluating a query, usually represented as a tree of relational operators (with annotations that contain additional detailed information about which access methods to use, etc.).The code that implements relational operators sits on top of the file and access methods layer. This layer supports the concept of a file, which, in a DBMS, is a collection of pages or a collection of records. Heap files, or files of unordered pages, as well as indexes are supported. In addition to keeping track of the pages in a file, this layer organizes the information within a page.The files and access methods layer code sits on top of the buffer manager, which brings pages in from disk to main memory as needed in response to read requests. The lowest layer of the DBMS software deals with management of space on disk, where the data is stored. Higher layers allocate, deallocate, read, and write pages through (routines provided by) this layer, called the disk space manager.
The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database. DBMS components associated with concurrency control and recovery include the transaction manager, which ensures that transactions request and release locks according to a suitable locking protocol and schedules the execution transactions; the lock manager, which keeps track of requests for locks and grants locks on database objects when they become available; and the recovery manager, which is responsible for maintaining a log and restoring the system to a consistent state after a crash. The disk space manager, buffer manager, and file and access method layers must interact with these components.

QUESTION 8

Quite a variety of people are associated with the creation and use of databases. Obviously, there are database implementors, who build DBMS software, and end users who wish to store and use data in a DBMS. Database implementors work for vendors such as IBM or Oracle. End users come from a diverse and increasing number of fields.

Database application programmers develop packages that facilitate data access for end users, who are usually not computer professionals, using the host or data languages and software tools that DBMS vendors provide. (Such tools include report writers, spreadsheets, statistical packages, and the like.)

Application programs should ideally access data through the external schema. It is possible to write applications that access data at a lower level, but such applications would compromise data independence.

A personal database is typically maintained by the individual who owns it and uses it. However, corporate or enterprise-wide databases are typically important enough and complex enough that the task of designing and maintaining the database is entrusted to a professional, called the database administrator (DBA).