Constraints: Consistency and Integrity

Mukul K Sinha

NCSDCT, TIFR, Bombay 400 005

The set of assertions associated with a database is called in literature, either consistency constraints [3] or integrity constraints [1]. Correspondingly, the database is defined to be in a correct state if the set of data items satisfies the consistency/integrity constraints.

We classify the database assertions into two types of constraints: consistency constraints and integrity constraints. They are different and should not be used interchangeably. An integrity constraint specifies the characteristics of a data item independent of any other data item of the database; whereas a consistency constraint specifies a relationship among two or more data items of the database. Inconsistency between two related entities is also viewed as lack of semantic integrity of the database but here we talking about the types of constraints and not about the semantic integrity of the database.

Consider, for example, a multi-user distributed database, which is associated with only the following set of assertions:

(i) The value of the object with the name x is a natural number greater than 21.
(ii) A + B = 100.
(iii) y >= 30.
(iv) P is equal to Q.

The assertions (i) and (iii) are integrity constraints whereas assertions (ii) and (iv) are consistency constraints.

Again, the terms atomic actions and transaction are also used interchangebly in the literature but there is an important difference between them. In [5], Reed has discussed the issues differentiating an atomic action from critical section and recovery block, but he does not

distinguish an atomic action from a transaction. An atomic action is a set of primitive actions executed at different sites at different times and whose internal state should not be visible outside the computation of the atomic action. An atomic action remains atomic in face of failure. Thus, an atomic action has to solve concurrency control problem as well as the problem of recovery. On the other hand, a transaction is a set of actions on individual data items of the database such that consistency of relationships among data items is maintained. A transaction is a unit of consistency. Thus, a transaction must be an atomic action as well. Can we, conversely, assert that an atomic action is also a transaction? Our answer is in negation.

Consider three atomic actions T1, T2 and T3 for the database discussed above.

T1: (x = x + 20)
T2: (y = y + 15, x = x + 20)
T3: (A = A - 10, B = B + 10)

Each update action in the above atomic actions comprises of two accesses to the data item. First, to read the data item and then to write the new value of the data item. We presume that the intermediate value of a data item acquired during the processing of the atomic action is not shipped to the disk and the atomic action writes only the final value of a data item on disk. All three atomic actions must follow a concurrency control algorithm (T1 and T2, to take care of the lost updates problem [2]; and T3, to maintain the database consistency).

Now let us consider the case of a failure. Suppose the crash occurs while T3 is getting processed. If it has updated the data item A but not B then the database will be in an inconsistent state. The recovery of T3 is a must before any other user can access either A or B.

On the other hand, if a crash occurs any time during the processing of T1 or T2, the database does not need any recovery. Processing of, neither T1 nor T2 can ever destroy the database consistency and the database always remains in a consistent state. The participating data items never loose their integrity either. The data item is not dirty for the database and hence we can not talk in terms of degrees of consistency [4].

It is quite obvious that the atomic action T1 never creates the need for any recovery. In case of T2, even if the crash occurred after y has been updated but before the modification of x, the database does not need any recovery. If we define a transaction strictly as a set of actions which is a unit of consistency, the atomic action T3 is a transaction but then neither T1 nor T2 should be called transactions. The semantics of the application program may like to recover from a crash of T2, but this recovery is qualitatively different from a crash of T3. The recovery of T3 has an objective basis (i.e., consistency of the database) whereas the recovery of T2 has a subjective basis (i.e., user has specified it as an atomic action). Thus, while the former will be done by the system, the latter is the responsibility of the user and should be done at the application level.

The difference between T2 and T3 stems from the types of assertions of the database. Atomic actions T1 and T2 operate on data items which appear only in those assertions which are classified as integrity constraints of the database. So, any crash during the processing of T1 or T2 will never result in an inconsistent state of the database. Thus, the classification of assertions helps to distinguish transactions from non-transaction atomic actions. It also helps to divide the recovery responsibilities between the database system and the application program. We also feel that a non-transaction atomic action need not follow so strict a protocol as two-phase commit protocol where a data item has to go through an indefinite wait phase even when the database is not inconsistent.

References

1. Bayer, R., Heller, H., and Reiser, A., "Parallelism and Recovery in Database Systems," ACM Trans. on Database Systems, Vol 5, No 2, June 1980.

2. Bernstein, P. A., and Goodman, Nathan., "Concurrency Control in Distributed Database Systems, " ACM Computing Surveys, Vol 13, No 2, June 1981.

3. Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L., "The Notions of Consistency and Predicate Locks in a Database System,"

Communication of ACM 19, November 1976.

4. Gray, J. N., Lorie, R. A., Putzolu, G. R., and Traiger, I. L., "Granularity of Locks and Degrees of Consistency in a Shared Database," IBM Res. Lab., RJ 1606, 1975.

5. Reed, D. P., "Implementing Atomic Actions on Decentralized Data," The 7th SOSP, Asilomar, California, December 1979.