Database Management Systems, ACID Properties, and their Implications

Overview

- **Database**: organized collection of data

- **Database Management System (DBMS)**: a software system that controls the organization, storage, retrieval, security and integrity of data in a database and across multiple databases.

- **Database Management System Pipeline**:
  1. Accept query from user
  2. Process query
  3. Output result

  Step 2 - **core** of DBMS.

- **Query Processing**:

<table>
<thead>
<tr>
<th>High Level</th>
<th>Low Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Evaluation</td>
<td>Data storage</td>
</tr>
<tr>
<td>Use of index structures</td>
<td>Index organization</td>
</tr>
<tr>
<td>Query rewrite rules</td>
<td>File system organization</td>
</tr>
<tr>
<td>Query evaluation plans</td>
<td><strong>Transaction Management</strong></td>
</tr>
<tr>
<td>Cost models</td>
<td></td>
</tr>
</tbody>
</table>

Transaction (externally) Execution of any user program by a DBMS.

Transaction (internally) A series of reads and writes.

*Informally, a transaction* is a sequence (set) operations on the database that serves a single purpose and yields a single outcome.
Concurrency in DBMS: Modern DBMS must be able to process multiple transactions at the same time.

To do this successfully, DBMS must have certain properties.

ACID Properties

(A)tomictiy: The DBMS executes either all actions of the transaction or none. Incomplete transactions should not result in permanent changes in the database.

(C)onsistency: Each transaction run by itself must preserve the consistency of the database.

(I)solation: Each transaction should be protected from effects of other transactions being executed alongside.

For each pair of transactions $T_i$ and $T_j$, it appears to $T_i$ that either $T_j$ has finished execution before $T_i$ started, or that $T_j$ has not commenced yet.

(D)urability: Once the transaction has successfully completed, its effects must become permanent in the database and shall be able to survive system crashes.

Why Atomicity? Transactions are defined as sequences of actions in pursuit of a single goal. As a result of executing the transaction, either the goal is reached and all effects of all actions from the transaction must take hold, or the goal has been abandoned and then no results of any actions should survive.

Why Consistency? This question lies at the core of our course. The short argument for consistency is this. Writing software that works with consistent-state databases (as defined by the requirements of the specific application) is easier than writing software that is not allowed to make this assumption. Inconsistent database states that exist for short periods of time might be benign, but any actions performed on the data while the database is in an inconsistent state are dangerous.

Why Isolation? Nothing will screw with the data in the database more than simultaneous changes to the same data from multiple not yet completed transactions. Further actions in such transactions may wind up with incorrect assumptions about the stored data, and may be unable to produce correct actions.

Why Durability? We need to make sure that changes to the database state persist once the process of making these changes succeeds.

Effect of ACID Properties on DBMS Design.

What do we need to do to ensure that a DBMS has ACID properties?
For Atomicity. To ensure atomicity DBMS must:

- know all actions in each transaction;
- have a roll-back mechanism to undo partial effects of aborted transactions.

For Consistency. To ensure consistency DBMS must:

- ensure that all transactions access the correct data at all times;

Ensuring consistency is essentially the DB application’s job.

For Isolation. To ensure isolation DBMS must:

- have a mechanism for correctly interleaving actions from multiple transactions (concurrency control mechanism);
- ensure that all transactions access the correct data at all times;

For Durability. To ensure durability DBMS must:

- have access to persistent storage, i.e., storage that does not get emptied when the DBMS is powered down;
- carefully and appropriately manage exchange of data between persistent storage and volatile storage (RAM);
- be able to recover from systemwide crashes.

Satisfying ACID Properties: Relational DBMS Organization

- Read database access (see Fig. 1)
  - Data is stored in records on pages on physical storage devices (e.g., hard disks).
  - Pages are retrieved from physical storage and loaded into main memory buffer.
  - Records are retrieved from pages in main memory buffer and passed to the transaction program where they are stored as variables.

- Write database access (see Fig. 1)
  - Data stored in the variables of the transaction program is written into a page in main memory buffer
  - Pages of main memory buffer are written to physical storage.
Figure 1: Data Flow in a Database Management System.