Database transactions

- A **database transaction** is any (possibly multi-step) action that reads from and/or writes to a database
  - It may consist of a single SQL statement or a collection of related SQL statements
  - ex: Adding a new lunch to the class database
    - requires **two** related INSERT statements

Transactions (cont.)

- A **successful transaction** is one in which all of the SQL statements are completed successfully
  - A **consistent** database state is one in which all data integrity constraints are satisfied
  - A successful transaction changes the database from one consistent state to another
Transaction management

• Improper or incomplete transactions can have a devastating effect on database integrity
  Ex: INSERT only items into the Lunch_item table

• If a DBMS supports transaction management, it will *roll back* an inconsistent database (i.e., the result of an unsuccessful transaction) to a previous consistent state.

Properties of a transaction

• Atomicity
  • The "all or nothing" property
  • All transaction operations must be completed i.e. a transaction is treated as a single, indivisible, logical unit of work

• Consistency
  • When a transaction is completed, the database must be in a consistent state

Properties of a transaction

• Atomicity
  • Consistency
  • Isolation
  • Durability

• Every transaction MUST exhibit these four properties
Properties of a transaction

- **Isolation**
  - Data used during the execution of a transaction cannot be used by a second transaction until the first one is completed

- **Durability**
  - Once transaction changes are committed (successfully applied to the database), they cannot be undone or lost due to a subsequent failure

Transaction examples

- **T1: Examine the current account balance**
  
  ```sql
  SELECT Cust_Number, Cust_Balance
  FROM Customer WHERE Cust_Number = 238
  
  - Transaction: read from the Customer table
  No changes are made to the database => The database remains in a consistent state after the transaction
  ```

- **T2: Sell 100 units of X to customer 238 for $500**
  
  ```sql
  UPDATE Product
  SET Prod_QOH = Prod_QOH - 100
  WHERE Prod_Code = 'X';

  UPDATE Customer
  SET Cust_Balance = Cust_Balance + 500
  WHERE Cust_Number = 'Y';
  
  - The database is in a consistent state only if both database requests are successfully completed.
Transaction management

- ANSI SQL standards for managing transactions:
  - COMMIT;
    - Permanently records all changes in the database
    - Automatically ends the transaction
  - ROLLBACK;
    - Aborts all uncommitted changes
    - Database is rolled back to its previous state
    - A rolled back transaction can typically be restarted later

BEGIN TRANSACTION also used by some DBMSs

Transaction log

- A DBMS maintains a transaction log to support recovery to a consistent state
- The transaction log stores:
  - A record for the beginning of the transaction
  - A record for each transaction component
    - Type of operation
    - Tables / attributes affected
    - "before" and "after" values for attributes affected
  - A record for the end of the transaction (COMMIT)

Transaction log (cont.)

- Example:

<table>
<thead>
<tr>
<th>TXN_ID</th>
<th>ROW_ID</th>
<th>COL_ID</th>
<th>OPERATION</th>
<th>TABLE</th>
<th>BEFORE VALUE</th>
<th>AFTER VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>108</td>
<td>322</td>
<td>36.3</td>
<td>UPDATE</td>
<td>PRODUCT</td>
<td>555.44</td>
<td>155.44</td>
</tr>
<tr>
<td>108</td>
<td>360</td>
<td>36.3</td>
<td>UPDATE</td>
<td>CUSTOMER</td>
<td>1013.75</td>
<td>1112.75</td>
</tr>
<tr>
<td>108</td>
<td>361</td>
<td>36.3</td>
<td>NULL</td>
<td>COMMIT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TID, TXN_ID = Transaction log record ID
TXN_ID = Transaction identifier
TXN, ROW_ID = Transaction record
Note: The transaction number is automatically assigned by the DBMS.
Transaction log (cont.)

- The transaction log is typically implemented as one or more files that are stored separately from the database itself
- It is automatically updated when the DBMS executes transactions that modify the database
- Tradeoffs:
  - Increases DBMS processing overhead
  - But...provides ability to restore a corrupted database

Transactions in MS Access

- MS Access only supports multi-step transactions when using code to access the DBMS
  - The SQL window within Access only allows one SQL statement to be run at a time
  - Only one user => no problems with conflicts
  - You can implement multi-step transactions when:
    - connecting from other office applications via VBA
    - connecting from a .NET application
    - connecting from a web-based application, etc....