1.264 Lecture 10

SQL, part 3
Connecting to database servers

Data definition language (DDL)

- We've explored the data manipulation language (DML) so far.
- SQL also has a data definition language (DDL):
  - CREATE DATABASE
  - CREATE TABLE
  - CREATE INDEX (and other CREATE statements)
  - ALTER TABLE
  - ALTER VIEW (and other ALTER statements)
  - DROP DATABASE
  - DROP TABLE
  - DROP VIEW (and other DROP statements)

Indexes

- Index is a separate data object in the database that lists the table rows in order to allow rapid lookup.
  - Each index for each table is a separate object
  - Primary keys and foreign keys are automatically indexed
- Rapid access to indexed columns
  - Each index may be updated when a row is updated, so indexes slow updates, insertions and deletes
  - Practical maximum of 3 or 4 indexes per table. If others are needed on occasion, add and drop them as needed
  - If a database is mostly read, use many indexes to speed performance
  - If database is mostly updates, use as few indexes as possible
- Clustered indexes
  - Physically rearrange rows by a single index to maximize disk access speed

Example of indexes

- Customer database
  - Customer ID is primary key
  - We also want to search by:
    - Customer name (last, first)
    - City, state
    - Postal (zip) code
    - Address
  - Index the name, city/state, zip and address
    - Four indexes: slow insert, update, delete, but fast lookup
    - If customer database is fairly stable, this is fine
  - Similar logic for parts catalog, bill of materials, etc.
- Internet search engines use 'text retrieval engines'
  - Index every word in the entire database; count occurrences and rank matches. Recent advances (frequency of links, usage…) enhance this.
- Syntax:
  - CREATE INDEX IX_Orders ON Orders (Cust, OrderNbr)
  - Try this in MSE. First select MIT1264 in toolbar or enter:

```
Use MIT1264
Go
```

Security

- Security options
  - Use operating system logon/password (weak) to identify user
    - User gets access to all databases, all tables ("Windows authentication")
  - Use database logon/password (stronger)
    - Restrict access to databases, tables, but can still use all applications
    - "SQL Server authentication": we'll use this for the Web->db connection
  - Application level security (stronger still, but tough to administer)
    - Each application must look in its database to see if user authorized
    - Can set authorization based on data model (which entities)
  - Network level security (strongest)
    - Use directory and public key infrastructure (PKI) (encryption)
    - Single sign-on (Kerberos, Microsoft Active Directory)
- Classes of users: super-user (dba or sa), data owner, data user
- Assignment of database privileges (permissions)
  - GRANT and REVOKE: E.g.,
    - GRANT ALL ON TableName TO PUBLIC WITH GRANT OPTION
    - REVOKE ALL ON TableName FROM PUBLIC CASCADE
    - Order matters for GRANTs and REVOKEs. Last one governs.
    - Try these two statements; look at the table properties in Enterprise Mgr

Transactions

- Group of operations often must be treated as atomic unit
  - Start transaction
    - Insert OrderHeader
    - While more OrderDetail (line items) exist:
      - Select Part
      - Update Part inventory
      - Insert OrderDetail row
  - Commit transaction if everything succeeds
  - Roll back transaction if any error occurs:
    - In Order Header
    - In OrderDetail
    - Server crashes
    - Disk crashes
    - Network dies
    - Etc.

Transaction properties (ACID)

- Atomicity. Either all of transactions are executed or all are rolled back
  - Account transfer debit and credit both succeed or fail
- Consistency. Only legal states can exist
  - If order detail cannot be written, order header is rolled back
- Isolation. Results not seen by other transactions until the transaction is complete
  - Account transfer debit and credit either both seen or neither is seen
- Durability. Data is persistent even if hardware or software crashes: What is written on the disk is correct
  - Account balance is maintained

Transactions

- Multi-user databases have other transaction issues
- Two database actions conflict if one or both are write operations. Examples of problems:
  - Lost updates:
    - 7 parts in inventory
    - Transactions 1 and 2 simultaneously read 7 as the current quantity
    - Transaction 1 finishes first, adds 3 parts, writes 10 as quantity
    - Transaction 2 finishes second, subtracts 5 parts, writes 2 as quantity!
  - Uncommitted changes:
    - Transaction 1 adds 3 parts, writes 10 as quantity
    - Transaction 2 reads 10 as quantity
    - Transaction 1 aborts (rolls back), leaving transaction 2 with wrong data

Transactions

- Databases use locks for concurrency. One simple scheme is pessimistic locking:
  - Writes obtain an exclusive lock on a record, preventing reads or writes
  - Reads obtain nonexclusive locks, allowing other reads but preventing a writer from obtaining an exclusive lock
- Or you can use optimistic locking (logs)
  - No locks are used. Check if row exists, is same after operation
  - If not, issue error and program must retry. Better performance.
- Databases use logs for recovery.
  - Log file of all changes is written in addition to making the changes in the database. (This is a key bottleneck in architecture.)
  - Change cannot be committed until the log is written to stable storage.
    - Changes usually committed before tables actually updated on disk
  - If a change is rolled back, the log is read to reverse the transactions.
  - If a system or disk crashes, the log is rerun from the last checkpoint to restore the database.
  - Turn off logs when loading batch data or recovering

Transaction example

```
INSERT Customers VALUES (212, 'Smith Co', 89, 20000)  -- Independent INSERTs
INSERT Orders VALUES (212, 'Lathe', 3, 20000, 0.1)
INSERT Orders VALUES (212, 'Latte', 10, 2, 0.0)

-- INSERTs as a transaction
BEGIN TRAN
INSERT Customers VALUES (213, 'Wang Co', 53, 100000)
IF @@ERROR    0
 BEGIN
  INSERT Orders VALUES (213, 'Mill', 1, 50000, 0.2)
  IF @@ERROR    0
   BEGIN
    INSERT Orders VALUES (213, 'Malt', 1, 2, 0.0)
    IF @@ERROR    0
     COMMIT TRAN
    ELSE
     ROLLBACK TRAN
   END
  ELSE
   ROLLBACK TRAN
 END
ELSE
 ROLLBACK TRAN
```

Exercise: Modify the transaction:
It's in Lecture10.sql on the Web site
INSERT Customer 214
INSERT first order for 214 correctly
INSERT 2nd order incorrectly: leave out the last two fields
Then open Customers and Orders:
Are any of the INSERTs present?

Performance

- Benchmarks (TPC-App through H…)
  - www.tpc.org
- Caches: disks are slow!
  - Place pages into memory and beyond for fast access
  - Disk configuration
  - More on this when we do hardware
- Query optimizers
  - Many ways to do joins; depends on table size, characteristics of keys (length, ‘uniqueness’), etc.
- Set and tune indexes, use cluster indexes
- Denormalize only for read-only access

Application-to-Database Connectivity

- Initial stage: Embedded SQL (ESQL) in each database server (early 1990s)
  - Compiled into server application, cannot be changed by end user
  - Could not work reasonably across multiple databases
- Second stage: As client-server applications appeared (late 1990s)
  - Each database vendor provided an application programming interface(API) to allow client programs to query the database
  - Each vendor's API was different, of course
- Current stage: ODBC, JDBC, ADO.NET, … (late 1990s-now)
  - ODBC was first common Windows API capable of accessing most major databases
    - Oracle, SQL Server, Sybase, DB2, Informix
  - JDBC is very similar for Java environment
  - .NET, J2EE extend database capabilities much further still
  - Web services, WSDL, UDDI go even farther—later in term

ODBC API (and the rest too)

- Library of procedures (methods) to connect from an application (Web, Windows, other) to DBMS, execute SQL statements and retrieve results
- SQL syntax based on SQL-92 standard
- Standard set of error codes
- Standard way to connect and log on to DBMS
- Standard representation of data types
- Standard methods for data type conversions
- ODBC has core, layer 1 and layer 2 functionality to deal with simple and sophisticated interfaces. Others are similar.

These features overcome many nonstandard SQL issues noted in the last lecture.

ODBC Architecture (and others)