CSCE 155 – Intro to Database Connectivity

Lecture Notes

1. Introduction
    - Programs are short-lived; need a way to *persist* data across multiple runs, sessions, etc.
    - File I/O is not sufficient
        - Data records may be incomplete
        - Repetition of data
        - No way to enforce data integrity, formatting
        - No way for multiple programs to access/update data concurrently
    - Solution: Relational Database Management System
2. Database Overview
    - RDMS stores data in tables
    - Each table has rows of records
    - Each table has columns (fields) of data
    - Examples
        - Video Game
        - Platform
        - Publisher
    - Records may be unique if they have a primary key (may be automatically assigned by the database)
    - Records in different tables may be related through foreign keys: one table has a column that references a (primary) key in another table
3. Relational Structure: Video Game Example
    - Video Game has: ID, name, a publisher
    - Publisher has: ID, name
    - Platform has: ID, name
    - Relations
        - Many-to-One (or One-to-Many): A single video game has one publisher, but a publisher may have published many games
        - Achieved through the use of a foreign key (Video Game references a Publisher)
        - Entity-Relation Diagram
        - Many-to-Many: A single game may have been released on multiple platforms; one platform will certainly have many games
        - Solution: a JOIN table (2 many-to-one relations)
    - Importance of data integrity
        - Relations cannot be violated (in a well-designed database):

        - A game cannot exist without a valid publisher
        - A game and a platform must exist before we can associate them in the JOIN table
        - A publisher could not be deleted while some games still reference it
        - Etc.
4. Advantages
    - RDMS provide the ACID principles:
        - Atomicity – data modifications are an all-or-nothing process
        - Consistency – database schema (definition) will remain consistent (no constraints will be violated)
        - Isolation – no transaction interferes with another
        - Durability – Once committed, a transaction remains so
    - RDMS provide CRUD operators through SQL
        - C = Create
        - R = Retrieve
        - U = Update
        - D = Destroy
        - SQL = Structured Query Language
    - RDMSs usually provide a general API (Application Programmer Interface) for connecting to them
5. ODBC: Open Database Connectivity
    - Standard C library for connecting to almost any type of database
    - Many different databases: vendors, features, cost, etc.
        - MySQL
        - PostgreSQL
        - MS SQL
        - DB2 (IBM)
        - Oracle
    - Good application design will allow for growth, migration; not good to program toward one specific database (vendor lock)
    - Alternative: use an abstract API layer “above” the database system
    - Vendors provide specific drivers that actually connect to the database in a vendor-specific manner
    - Changing a database is now only a matter of changing a driver (no code changes)
    - ODBC offers:
        - Generic database resources (SQLHandles)
        - Generic functions to connect and query data (CRUD)
        - `SQLDriverConnect`
        - `SQLExecDirect`
        - `SQLPrepare`
        - `SQLExecute`

        - `SQLRowCount`
        - `SQLBindCol`
        - `SQLCloseCursor`
        - `SQLFreeStmt`
6. JDBC: Java Database Connectivity API
    - Standard Java API (Application Programmer Interface) for any type of database
    - API provides a generic interface for connecting and querying databases
    - Different vendors publish *drivers* that provide specific functionality for a particular database
    - JDBC offers:
        - `DriverManager`
        - `Connection`
        - `PreparedStatement`
        - `ResultSet`