Bölüm 10: Veri Tabanı

JAVA ile Nesne Yönelimli Programlama

Veritabanı Neden Önemlidir?

- Kalıcı Depolama:
  - Verileri uzun süre depolama.
- Dosya Sistemleri vs. Veritabanları:
  - Dosya sistemleri bazı uygulamalar için yeterli olabilir.
  - Veritabanları daha karmaşık ihtiyaçlar için gereklidir.
  - Verilerin verimli bir şekilde işlenmesi için önemli özellikler sunar.

Veritabanlarının Temel Özellikleri

- Verilere hızlı erişim sağlama
- Alan seviyesinde güvenlik
- Platform bağımsız ve standartlaştırılmış API'ler
- Belirli bir dosya sistemi veya yapı ile bağımlı değil

Veritabanı Türleri

- İlişkisel Veritabanları (%77):
  - Oracle, SQL Server, MySQL, Postgres
- NoSQL/Arama Veritabanları (%23):
  - Belirli senaryolarda kullanılır (örneğin, MongoDB, Elasticsearch)
- SQLite:
  - Gömülü bir ilişkisel veritabanı,
  - Veri paylaşımının minimum olduğu durumlar için yaygın.

SQLite Nedir?

- Gömülü bir ilişkisel veritabanıdır
- Sunucuya ihtiyaç duymaz
- Kendine yeten bir yapıya sahiptir
- Çapraz platform desteği sunar
- Kolayca kurulabilir ve kullanılabilir
- Yaygın kullanım alanları:
  - Mobil uygulamalar, Gömülü sistemler, Küçük ölçekli uygulamalar

SQLite

- Dünyada en yaygın kullanılan gömülü (serverless) ilişkisel veritabanıdır.
- Açık kaynak ve ücretsiz bir yazılımdır.
- Oracle ve SQL Server gibi sunucu tabanlı veritabanlarının aksine, tamamen uygulama içinde çalışır ve tüm verileri tek bir disk dosyasında saklar.

Eclipse'te SQLite Bağlantısı

- Java, veritabanlarına bağlanmak için standart bir API JDBC'yi destekler.
- SQLite JDBC Sürücüsü Kurma:
  - SQLite JDBC sürücüsü indirilip projeye eklenir
- Bağlantı Kurma:
  - jdbc:sqlite:<veritabanı_dosyanız> formatında bağlantı kurulur
- SQL Komutları ile Çalışma:
  - SQL komutları çalıştırılarak veritabanı ile etkileşimde bulunulur
  - (örneğin, CREATE TABLE, INSERT, SELECT)

DB Browser for SQLite

- SQLite için popüler bir SQL istemcisi DB Browser for SQLite'tır.
- https://sqlitebrowser.org/dl/
- https://github.com/xerial/sqlite-jdbc

Yaygın SQLite Komutları

- CREATE TABLE:
  - Yeni tablo oluşturur
- INSERT INTO:
  - Yeni kayıt ekler
- SELECT:
  - Tabloyu sorgular
- UPDATE:
  - Var olan kayıtları günceller
- DELETE:
  - Kayıtları siler

SQLite'a Bağlanma

```
// SQLite veritabanı bağlantısı oluşturuluyor.
String url = "jdbc:sqlite:sample.db"; // Veritabanı dosyasının yolu
try (Connection conn = DriverManager.getConnection(url)) {
    // Bağlantı başarılı ise aşağıdaki mesaj yazdırılır.
    if (conn != null) {
        System.out.println("Bağlantı başarılı!");
    }
} catch (SQLException e) {
    // Bağlantı hatası oluşursa, hata mesajı yazdırılır.
    System.out.println(e.getMessage());
}
```

Tablo Oluşturma

```
String sql = "CREATE TABLE IF NOT EXISTS users
                                (id INTEGER PRIMARY KEY, name TEXT, email TEXT);";
try (Connection conn = DriverManager.getConnection(url);
     Statement stmt = conn.createStatement()) {
    // Tablo oluşturuluyor.
    stmt.execute(sql);
    System.out.println("Tablo oluşturuldu!");
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Veri Ekleme

```
String sql = "INSERT INTO users(name, email)
                              VALUES('Ahmet Yılmaz', 'ahmet@example.com');";
try (Connection conn = DriverManager.getConnection(url);
    Statement stmt = conn.createStatement()) {
    // Veritabanına veri ekleniyor.
    stmt.executeUpdate(sql);
    System.out.println("Veri eklendi!");
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Veri Okuma

```
String sql = "SELECT * FROM users;";
try (Connection conn = DriverManager.getConnection(url);
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sql)) {
    // Veri okunuyor ve ekrana yazdırılıyor.
    while (rs.next()) {
        System.out.println("ID: " + rs.getInt("id") + ", Name: " +
                rs.getString("name") + ", Email: " + rs.getString("email"));
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Veri Güncelleme

```
String sql = "UPDATE users SET name = 'Mehmet Akdeniz' WHERE id = 1;";
try (Connection conn = DriverManager.getConnection(url);
    Statement stmt = conn.createStatement()) {
    // Veritabanındaki veri güncelleniyor.
    stmt.executeUpdate(sql);
    System.out.println("Veri güncellendi!");
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

Veri Silme

```
String sql = "DELETE FROM users WHERE id = 1;";
try (Connection conn = DriverManager.getConnection(url);
    Statement stmt = conn.createStatement()) {
    // Veritabanındaki veri siliniyor.
    stmt.executeUpdate(sql);
    System.out.println("Veri silindi!");
} catch (SQLException e) {
    System.out.println(e.getMessage());
}
```

SON