## Tietojen haku tietokannasta: Statement ja ResultSet

### Ennen tietojen hakua

Ennen kuin Java-ohjelmassa voidaan hakea tietoja tietokannasta, on ohjelmalla oltava käytössä yhteys tietokantaan. Käytännössä JDBC-teknologiaa hyödyntävä ohjelma tarvitsee `java.sql.Connection`-luokan ilmentymän. Tämän ilmentymän luominen kuvattiin edellä kohdassa Tietokantayhteyden muodostaminen.

### Tietokantakyselyn suorittaminen: Statement

Tietokantakyselyissä on kaksi perusosaa: `Statement`, jolla kysely suoritetaan ja `ResultSet`, jonne kyselyn tulokset tallennetaan.

`Statement` luodaan yksinkertaisesti hyödyntämällä `Connection`-luokkaa. `Connection`-luokasta löytyy metodi `createStatement()`, jolla uusi `Statement` saadaan käyttöön.

Itse `Statement`-luokasta löytyy useita eri metodeita, joilla voidaan suorittaa erilaisia tietokantaoperaatioita. Kyselyiden suorittamiseen perusoperaatio on `executeQuery(String sql)`, jolla voidaan suorittaa jokin SQL-lause. `ExceuteQuery()`-metodi palauttaa `java.sql.ResultSet`-tyyppisen muuttujan, joka sisältää kyselyn tulokset.

**Esimerkki**

Haetaan auton nimi ja rekisterinumero auto-nimisestä tietokantataulusta.

```
// Luodaan uusi Statement avattua tietokantayhteyttä käyttämällä
Statement s = conn.createStatement();

// Suoritetaan kysely
ResultSet rs = s.excetuteQuery(”SELECT nimi, rekisterinumero FROM
auto”);
```

### Kyselyn tulosten käsittely: ResultSet

Tietokantakyselyn tulokset tallennetaan `java.sql.ResultSet`-tyyppiseen muuttujaan. `ResultSet` on kursori, joka viittaa kulloinkin käsittelyssä olevaan tulosjoukon riviin. Kursorin saa siirtymään seuraavaan riviin `next()`-metodilla. `Next()`-metodi palauttaa `true`, jos tuloksissa on vielä lisää käsittelemättömiä rivejä, ja `false`, jos siirrytään viimeiselle riville.

`Next()`-metodin lisäksi ResultSet-luokasta löytyy joukko metodeita, joilla voidaan palauttaa tulosjoukon kenttien sisältö. Tällaisia metodeita ovat esimerkiksi `getString()`, `getDate()`, `getInt()` jne. Näille metodeille annetaan parametrina joko sen kentän järjestysnumero, jonka tieto halutaan palauttaa, tai vaihtoehtoisesti kentän nimi.

**Esimerkki**

Edellisessä esimerkissä haettiin `executeQuery()`-metodilla joukko auton nimiä ja rekisterinumeroita tietokannasta. Oletetaan, että taulussa, josta tiedot haettiin, oli kolme riviä. Kun kysely on suoritettu, tuloksena on seuraava `ResultSet`:

| Kursori | Nimi | Rekisterinumero |
|---|---|---|
| ⇨ | | |
| | Ford Mondeo | ABC-123 |
| | Volkswagen Golf | DDE-456 |
| | Mitshubishi Galant | ERT-987 |

Oletetaan, että edellä mainittu `ResultSet` on tallennettu muuttujaan nimeltä `rs`. Kun tuloksia halutaan käydä läpi, on kursori ensin siirrettävä tulosjoukon ensimmäiselle riville. Tämä onnistuu käskyllä `rs.next();` Tämän jälkeen tulosjoukko näyttäisi seuraavalta:

| Kursori | Nimi | Rekisterinumero |
|---|---|---|
| ⇨ | Ford Mondeo | ABC-123 |
| | Volkswagen Golf | DDE-456 |
| | Mitshubishi Galant | ERT-987 |

Nyt rivillä olevien tulosjoukon kenttien arvot voidaan hakea `rs.getString()` –metodia kutsumalla. Jos halutaan palauttaa vaikkapa `Nimi`-kentän arvo, onnistuisi tämä joko antamalla kentän järjestysnumero (1) tai kentän nimi (”Nimi”) parametriksi:

```
// nimi1-muuttuja saa arvon ”Ford Mondeo”
String nimi1 = rs.getString(1);

// nimi2-muuttuja saa arvon ”Ford Mondeo”
String nimi2 = rs.getString(”Nimi”);
```

Jos halutaan siirtyä tarkastelemaan seuraavaa riviä, voidaan antaa jälleen käsky `rs.next()`, jolloin tulosjoukko näyttäisi seuraavalta:

| Kursori | Nimi | Rekisterinumero |
|---|---|---|
| | Ford Mondeo | ABC-123 |
| ⇨ | Volkswagen Golf | DDE-456 |
| | Mitshubishi Galant | ERT-987 |

Jos nyt haluttaisiin tietää seuraavan auton nimi ja rekisterinumero, onnistuisi se seuraavilla tavoilla:

```
// nimi1-muuttuja saa arvon ”Volkswagen Golf”
String nimi1 = rs.getString(1);

// rekisteri1-muuttuja saa arvon ”DDE-456”
String rekisteri1 = rs.getString(2);
```

vastaavasti asian voisi myös tehdä seuraavilla käskyillä:

```
// nimi2-muuttuja saa arvon ”Volkswagen Golf”
String nimi2 = rs.getString(”Nimi”);

// rekisteri2-muuttuja saa arvon ”DDE-456”
String rekisteri2 = rs.getString(”Rekisterinumero”);
```

Usein tulosjoukko käydään läpi silmukassa. Jos esimerkiksi halutaan tulostaa kaikkien autojen tiedot ruudulle, onnistuisi se seuraavalla tavalla:

```
ResultSet rs = s.excetuteQuery(”SELECT nimi, rekisterinumero FROM auto”);

while (rs.next()) {
  System.out.println(rs.getString(1) + ” ” + rs.getString(2));
}
```

Ruudulle tulostuisi tässä tapauksessa:

```
Ford Mondeo ABC-123

Volkswagen Golf DDE-456

Mitshubishi Galant ERT-987
```

### Yhteyden sulkeminen

Kun tietokantakysely on suoritettu, eikä `ResultSet`- tai `Statement`-muuttujia eikä tietokantayhteyttä enää tarvita, on yhteydet syytä sulkea. Tämä onnistuu `ResultSet`-, `Statement`- ja `Connection`-luokista löytyvillä `close()`-metodeilla.

**!** Mikäli yhteyksiä ei suljeta sen jälkeen, kun niitä ei enää tarvita, on mahdollista, että tietokanta ylikuormittuu, eikä uusia yhteyksiä ole enää mahdollista avata. Käytännössä tämä näkyy usein niin, että ohjelma lakkaa toimimasta kun sitä on käytetty jonkin aikaa. Poikkeuskäsittelyyn kannattaa yleensä aina laitta finally-lohko, jossa yhteydet suljetaan myös siinä tapauksessa, että ohjelmassa tapahtuu virhe..

**Esimerkki**

Tietojen haku Auto-taulusta Derby-tietokantaa käyttämällä. Kaikki vaiheet alusta loppuun:

```
//
// # 1. Avataan yhteys tietokantaan
//

// Vaihe 1.1: Ladataan JDBC-ajuri
String driver = ”org.apache.derby.jdbc.EmbeddedDriver”;
Class.forName(driver).newInstance();

// Vaihe 1.2: Määritellään yhteyden parametrit
Properties props = new Properties();
props.put(”user”, “user1”);
props.put(“password”, “user1”);

// Vaihe 1.3: Avataan yhteys
String url = ” jdbc:derby:derbyDB;create=true”;
conn = DriverManager.getConnection(url, props);

//
// # 2. Suoritetaan kysely
//
// Vaihe 2.1: Luodaan uusi statement avattua tietokantayhteyttä käyttämällä
Statement s = conn.createStatement();

// Vaihe 2.2: Suoritetaan kysely
ResultSet rs = s.excetuteQuery(”SELECT nimi, rekisterinumero FROM auto”);

//
// # 3. Käydään läpi kyselyn tulokset ja tulostetaan haetut tiedot ruudulle
//
```