Preparing Statements

> Preparing Statements on menetelmä, joka mm. estää SQL-injektion mahdollisuuden. Menetelmää on suositeltavaa käyttää aina, kun SQL-lauseen osaksi luetaan dataa ulkoisesta lähteestä esim. käyttäjän syöttämänä.

Edellisessä luvussa esitetty SQL-injektio-esimerkki `sql-inject.php` voidaan kirjoittaa Preparing Statements -menetelmää käyttäen alla esitetyllä tavalla. Tässä materiaalissa esitellään ainoastaan nimettyjen place holdereiden käyttöä.

Esimerkki 2201: mysql05.php

- Rivillä 8 käytetään **nimettyä** place holderia `:id`, jota ei ympyröidä edes merkkijonojen tapauksessa heittomerkeillä
- Rivillä 10 metodilla `$db->prepare()` SQL-kysely lähetetään ensin palvelimelle esikäännettäväksi
- Rivillä 11 esikäännöksen jälkeen place holderiin `:id` liitetään ukoisesta lähteestä tullut data muuttujasta `$_GET['id']`. Tässä vaiheessa suoritettavan SQL-lauseen **merkitys** ei voi enää muuttua, jolloin SQL-injektion mahdollisuus estyy.
- Rivin 11 arvon liittäminen voi tapahtua merkkijonona esim. seuraavasti `$stmt->bindValue(':name', $_GET['name'], PDO::PARAM_STR);`
- Rivillä 12 suoritetaan lopullinen SQL-kysely

```
1  <?php
2  // mysql05.php
3  require_once("/home/N1234/db-config/db-init.php");
4
5  // Ota kommentti pois, jos et halua syöttää id:tä URLin avull
6  //$_GET['id'] = 1;
7
8  $sql     = "SELECT * FROM customer WHERE id = :id";
9
10 $stmt = $db->prepare($sql);
11 $stmt->bindValue(':id', $_GET['id'], PDO::PARAM_INT);
12 $stmt->execute();
```

```
13
14  echo "<table border='1'>";
15  while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
16    echo "<tr><td>{$row['id']}</td><td>{$row['name']}</td></tr>
17  }
18  echo "</table>";
19
20  ?>
```

INSERT-, UPDATE- ja DELETE-lauseet ja Preparing Statements

Tällöin aiemmin esitetty *esimerkki 2012: mysql03.php* muuntuu seuraavaan muotoon.

Esimerkki 2202: mysql06.php

Huomio esimerkissä, että

- kyselyihin tuodaan "ulkoista dataa" esimerkin vuoksi riveillä 7-10, 24-25 ja 37-38
- ulkoinen data voidaan liittää place holdereihin myös hajautustaulukkona riveillä 15-17.
- kyselyjen suorittaminen tapahtuu `execute()`-metodilla riveillä 17, 30 ja 43

```
1   <?php
2   // mysql06.php
3   require_once("/home/N1234/db-config/db-init.php");
4
5   //-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
6   // Lisätään yksi tietue
7   // "Ulkoinen data":
8   $id = 6;
9   $nimi = 'Mieli Kaino';
10  $pvm = '2011-01-01';
11
12
13  $sql ="INSERT INTO customer VALUES(:id, :nimi, :pvm, (select
14  $stmt = $db->prepare($sql);
15  $bind_array = array(':id' => $id, ':nimi' => $nimi, ':pvm' =>
```

```php

$affected_rows = $stmt->execute($bind_array);
echo "<br>" . $affected_rows . " riviä lisättiin:<br>";

print_customers($db);

//-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Päivitetään yksi tietue
// "Ulkoinen data":
$pvm = '2019-01-09';

$sql ="UPDATE customer SET birth_date = :pvm WHERE name = 'Mi
$stmt = $db->prepare($sql);
$stmt->bindValue(':pvm', $pvm, PDO::PARAM_STR);
$affected_rows = $stmt->execute();
echo "<br>" . $affected_rows . " riviä muutettiin:<br>";

print_customers($db);

//-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Poistetaan yksi tietue
// "Ulkoinen data":
$nimi = 'Mieli Kaino';

$sql ="DELETE FROM customer WHERE name = :nimi";
$stmt = $db->prepare($sql);
$stmt->bindValue(':nimi', $nimi, PDO::PARAM_STR);
$affected_rows = $stmt->execute();

echo "<br>" . $affected_rows . " riviä poistettiin:<br>";

print_customers($db);

//-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Tulostetaan customer-taulu HTML-taulukkona
function print_customers($db) {
  $result = $db->query('SELECT * FROM customer');
  $row_count = $result->rowCount();
  echo "Näytetään " . $row_count. " riviä:<br>";

  echo "<table border='1'>";
  while($row = $result->fetch(PDO::FETCH_ASSOC)) {
    echo "<tr><td>{$row['id']}</td><td>{$row['name']}</td><td
```

```
59      }
60      echo "</table>";
61  }
62  ?>
```

Prepare Statements ja SQL-funktiot

SQL-funktiot (esim. `now()`) on liitettävä seuraavalla tavalla

```
1  $status = 'online';
2  $stmt = $db->prepare("INSERT INTO table(`time`, `status`) VAL
3  $stmt->bindValue(':status', $status, PDO::PARAM_STR);
4  $stmt->execute();
```

Funktiolle voi antaa myös parametrit place holdereilla seuraavasti

```
1  $uid = 'ara';
2  $pwd = 'sala';
3  $stmt = $db->prepare("INSERT INTO usertable(`uid`, `pwd`) VAL
4  $stmt->bindValue(':uid', $uid, PDO::PARAM_STR);
5  $stmt->bindValue(':pwd', $pwd, PDO::PARAM_STR);
6  $stmt->execute();
```

Prepare Statements ja toistorakenteet

Kertaalleen esivalmistellun kyselyn voi suorittaa toistuvasti myös toistorakenteessa. Huomaa, että $name-muuttuja pitää määritellä etukäteen (tyhjänä), jotta se voidan sitoa bindParam()-metodissa:

```
1  $values = array('elephant', 'tiger', 'bird');
2  $name = '';
3
4  $stmt = $db->prepare("INSERT INTO animals(`name`) VALUES(:nam
5  $stmt->bindParam(':name', $name, PDO::PARAM_STR);
6  foreach($values as $name) {
7     $stmt->execute();
8  }
```

```
```

```
© #AriRantala
```