

PR FIFA - HC 5

SQL Injection
Prepared statements

Recap:

Tbl_users		
Id	First name	Last name
1	Kees	Smit
2	Kees	Pieters

```
SELECT * FROM tbl_users WHERE id = 1  
INSERT INTO tbl_users (id, name) VALUES(2, "Jantje")  
UPDATE tbl_users SET name = 'Pietje' WHERE id = 1  
DELETE FROM tbl_users WHERE id = 1
```

Gebruik in WHERE altijd id, dat is het enige unieke!

Anders loop je risico meerdere mensen met dezelfde naam te verwijderen

Voorbeeld:

Vind een gebruiker

Voornaam:

Zoeken

```
SELECT * FROM tbl_users WHERE firstname = $firstname
```

- Kees Smit (x) <input type="hidden" name="id" value="1">
- Kees Pieters (x) <input type="hidden" name="id" value="2">

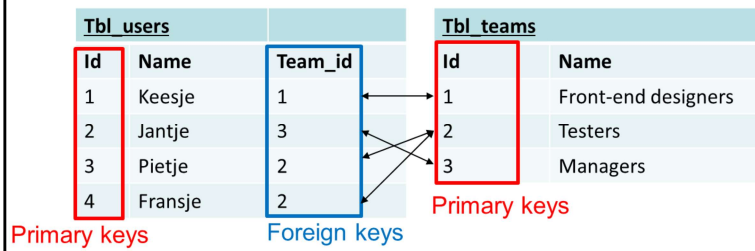
```
DELETE FROM tbl_users WHERE id = $id
```

© Copyright van DCC - Niet te verspreiden

Gebruik in WHERE altijd id, dat is het enige unieke!

Anders loop je risico meerdere mensen met dezelfde naam te verwijderen

"Keys"



Je gebruikt dus nummers om tabellen te koppelen, we noemen dat "primary keys" of "foreign keys".

Iedere tabel heeft een eigen primary key.

Een foreign key verwijst naar een andere tabel

INNER JOIN

Tbl_users		
Id	Name	Team_id
1	Keesje	1
2	Jantje	3
3	Pietje	2
4	Fransje	2
5	Liesje	5

Tbl_teams	
Id	Name
1	Front-end designers
2	Testers
3	Managers

```
SELECT *
FROM tbl_users
INNER JOIN tbl_teams
ON tbl_users.team_id =
tbl_teams.id
```

Resultaat		
Id	Name	Team
1	Keesje	Front-end designers
2	Jantje	Managers
3	Pietje	Testers
4	Fransje	Testers

“Inner join”: krijgt alleen resultaat als in beide tabellen een overeenkomst is.
Team 5 bestaat niet, dus je krijgt “Liesje” niet terug.

SQL Injection Prepared statements

Voorbeeld van SQL-injection

Prepared statements

1. Verzend éérs de query naar de database (met placeholders ipv gebruikersdata)
2. Query wordt gecompileerd
3. Verzend dan de parameters
4. SQL in de parameters wordt niet gecompileerd en dus niet uitgevoerd

© Universiteit van Dordrecht

Hiermee voorkom je dus SQL-injection!

Prepared statements - PHP

Gebruik PDO!

```
$id = $_GET['id'];  
$db = new PDO(...);  
$sql = "SELECT * FROM tbl_users WHERE id = :id"  
$stmt = $db->prepare($sql);  
$stmt->execute(array("id" => $id));  
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

`$stmt` betekent dus "statement"!

Ga dit niet onthouden, maar zoek het terug op mysite of elders op internet ("php pdo prepared statements")

Prepared statements – C#

```
String sql = "INSERT INTO [tblUsers] ([Username], [Password],  
[IsAdmin]) VALUES (@Username, @Password, @IsAdmin)";  
using (SqlCommand cmd = new SqlCommand(sql))  
{  
    cmd.Parameters.AddWithValue("Username", txtUsername.Text);  
    cmd.Parameters.AddWithValue("Password", txtPassword.Text);  
    cmd.Parameters.AddWithValue("IsAdmin", 0);  
    cmd.Connection = dbh.GetCon();  
    cmd.ExecuteNonQuery();  
}
```

Wat is dat using-ding?

Zorgt ervoor dat het object na het blok netjes wordt weggegooid.

Je hebt het cmd-object immers niet meer nodig verderop in je code.

Prepared statements – C#

```
String sql = "INSERT INTO [tblUsers] ([Username], [Password],  
[IsAdmin]) VALUES (@Username, @Password, @IsAdmin)";  
  
using (SqlCommand cmd = new SqlCommand(sql))  
{  
    cmd.Parameters.AddWithValue("Username", txtUsername.Text);  
    cmd.Parameters.AddWithValue("Password", txtPassword.Text);  
    cmd.Parameters.AddWithValue("IsAdmin", 0);  
    cmd.Connection = dbh.GetCon();  
    cmd.ExecuteNonQuery();  
}
```

Wat is dat?

Gebruik je intellisense!