A very basic web architecture
A very basic web architecture
A very basic web architecture
A very basic web architecture
A very basic web architecture

Client

Browser

Server

Web server

Database
A very basic web architecture

Client

Browser

(Private) Data

Server

Web server

Database
A very basic web architecture

Client

Browser

(Private) Data

Server

Web server

Database

DB is a separate entity, logically (and often physically)
SQL security
Databases

• Provide data **storage & data manipulation**

• Database designer lays out the data into tables

• Programmers query the database

• **Database Management Systems (DBMSes)** provide
  • semantics for how to organize data
  • transactions for manipulating data sanely
  • a **language** for creating & querying data
    - and APIs to interoperate with other languages
  • management via users & permissions
## Databases: basics

### Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
# Databases: basics

## Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
# Databases: basics

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
## Databases: basics

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
## Databases: basics

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
# Databases: basics

## Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
## Databases: basics

### Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
## Databases: basics

### Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>i3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
# Databases: basics

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
Database transactions

Transactions are the unit of work on a database
Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is listed as taking CMSC414 in the Classes table”

“Deduct $100 from Alice; Add $100 to Bob”
Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is listed as taking CMSC414 in the Classes table” 2 reads

“Deduct $100 from Alice; Add $100 to Bob” 2 writes
Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is listed as taking CMSC414 in the Classes table” 2 reads 1 transaction

“Deduct $100 from Alice; Add $100 to Bob” 2 writes
Database transactions

Transactions are the unit of work on a database

“Give me everyone in the User table who is listed as taking CMSC414 in the Classes table”  2 reads

“Deduct $100 from Alice; Add $100 to Bob”  2 writes

Typically want **ACID** transactions

- **Atomicity**: Transactions complete entirely or not at all
- **Consistency**: The database is always in a *valid* state (but not necessarily *correct*)
- **Isolation**: Results from a transaction aren’t visible until it is complete
- **Durability**: Once a transaction is committed, it remains, despite, e.g., power failures
## SQL (Standard Query Language)

### Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>
### SQL (Standard Query Language)

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>

**SQL Query:**

```
SELECT Age FROM Users WHERE Name='Dee';
```
## SQL (Standard Query Language)

**Users**

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>

```
SELECT Age FROM Users WHERE Name='Dee'; 28
```
### SQL (Standard Query Language)

#### Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:aneifjask@pp.com">aneifjask@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>

**SELECT** Age FROM Users WHERE Name='Dee';  
**28**

**UPDATE** Users SET email='readgood@pp.com'  
WHERE Age=32; -- this is a comment
# SQL (Standard Query Language)

## Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:readgood@pp.com">readgood@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>

```sql
SELECT Age FROM Users WHERE Name='Dee';
```

```sql
UPDATE Users SET email='readgood@pp.com'
WHERE Age=32;
-- this is a comment
```
SQL (Standard Query Language)

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:readgood@pp.com">readgood@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
</tbody>
</table>

**SELECT**  Age FROM Users WHERE Name='Dee';  

**UPDATE**  Users SET email='readgood@pp.com'  
WHERE Age=32;  -- this is a comment  

**INSERT**  INTO Users Values('Frank', 'M', 57, ...);
# SQL (Standard Query Language)

## Users

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:readgood@pp.com">readgood@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
<tr>
<td>Frank</td>
<td>M</td>
<td>57</td>
<td><a href="mailto:armed@pp.com">armed@pp.com</a></td>
<td>ziog9gga</td>
</tr>
</tbody>
</table>

**SELECT** Age FROM Users WHERE Name='Dee';  
**UPDATE** Users SET email='readgood@pp.com' WHERE Age=32;  
**INSERT** INTO Users Values('Frank', 'M', 57, ...);
SQL (Standard Query Language)

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Email</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dee</td>
<td>F</td>
<td>28</td>
<td><a href="mailto:dee@pp.com">dee@pp.com</a></td>
<td>j3i8g8ha</td>
</tr>
<tr>
<td>Mac</td>
<td>M</td>
<td>7</td>
<td><a href="mailto:bouncer@pp.com">bouncer@pp.com</a></td>
<td>a0u23bt</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>32</td>
<td><a href="mailto:readgood@pp.com">readgood@pp.com</a></td>
<td>0aergja</td>
</tr>
<tr>
<td>Dennis</td>
<td>M</td>
<td>28</td>
<td><a href="mailto:imagod@pp.com">imagod@pp.com</a></td>
<td>1bjb9a93</td>
</tr>
<tr>
<td>Frank</td>
<td>M</td>
<td>57</td>
<td><a href="mailto:armed@pp.com">armed@pp.com</a></td>
<td>zio9g9ga</td>
</tr>
</tbody>
</table>

**SELECT** Age FROM Users WHERE Name='Dee'; **28**

**UPDATE** Users SET email='readgood@pp.com'
WHERE Age=32; -- this is a comment

**INSERT** INTO Users Values('Frank', 'M', 57, ...);

**DROP** TABLE Users;
SQL (Standard Query Language)

```
SELECT Age FROM Users WHERE Name='Dee';
UPDATE Users SET email='readgood@pp.com'
    WHERE Age=32;  -- this is a comment
INSERT INTO Users Values('Frank', 'M', 57, ...);
DROP TABLE Users;
```
Server-side code

```
$result = mysql_query("select * from Users
    where(name='{$user}' and password='{$pass}');");
```

Suppose you successfully log in as $user
if this query returns any rows whatsoever
Server-side code

Website

“Login code” (php)

```php
$result = mysql_query("select * from Users
  where(name='$user' and password='$pass');");
```

Suppose you successfully log in as $user if this query returns any rows whatsoever

How could you exploit this?
SQL injection

$result = mysql_query("select * from Users
    where(name='$user' and password='$pass');");
SQL injection

```
$result = mysql_query("select * from Users
  where(name='"$user' and password='"$pass'); --
```

SQL injection

$username = "frank" OR 1=1); --

$result = mysql_query("select * from Users where (name='".$user." and password='".$pass."');");

$result = mysql_query("select * from Users where (name='"frank." OR 1=1); -- ' and password='"x.");");
SQL injection

$result = mysql_query("select * from Users
    where(name='".$user.' and password='".$pass.');");

Can chain together statements with semicolon:
STATEMENT 1 ; STATEMENT 2
Can chain together statements with semicolon:

STATEMENT 1 ; STATEMENT 2
SQL injection
Buffer "errors"
XSS
CSRF
hi, this is your son's school. we're having some computer trouble.

oh, dear - did he break something? in a way -

did you really name your son 'robert'); drop table students;-- ?

oh, yes. little bobby tables, we call him.

well, we've lost this year's student records. i hope you're happy.

and i hope you've learned to sanitize your database inputs.
SQL injection countermeasures

- **Blacklisting**: Delete the characters you don’t want
  - ‘
  - __
  - ;

- **Downside**: “Peter O’Connor”
  - You want these characters sometimes!
  - How do you know if/when the characters are bad?
SQL injection countermeasures

1. Whitelisting

- Check that the user-provided input is in some set of values known to be safe
  - Integer within the right range

- Given an invalid input, better to reject than to fix
  - “Fixes” may introduce vulnerabilities
    - Principle of fail-safe defaults

- Downside:
  - Um.. Names come from a well-known dictionary?
SQL injection countermeasures

2. Escape characters

• Escape characters that could alter control
  • ’ ⇒ \
  • ; ⇒ \\;
  • - ⇒ \\-
  • \ ⇒ \\\n
• Hard by hand, but there are many libs & methods
  • magic_quotes_gpc = On
  • mysql_real_escape_string()

• Downside: Sometimes you want these in your SQL!
The underlying issue

```
$result = mysql_query("select * from Users
  where(name='$user' and password='$pass');");
```

- This one string combines the code and the data
- Similar to buffer overflows:

When the boundary between code and data blurs, we open ourselves up to vulnerabilities
The underlying issue

```
$result = mysql_query("select * from Users
where(name='\$user' and password='\$pass');");
```
The underlying issue

```
$result = mysql_query("select * from Users
    where(name='"$user"' and password='"$pass")");
```

```
select / from / where

*  Users  and

=  =

name  $user  password  $pass
```
SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```
$result = mysql_query("select * from Users
                   where(name='$user' and password='$pass');");
```
SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```php
$db = new mysql("localhost", "user", "pass", "DB");

$statement = $db->prepare("select * from Users
    where(name='\$user' and password='\$pass');");

$result = mysql_query("select * from Users
    where(name='\$user' and password='\$pass');");
```

```php
$db = new mysql("localhost", "user", "pass", "DB");

$statement = $db->prepare("select * from Users
    where(name=? and password=?);");

$statement->bind_param("ss", $user, $pass);

$statement->execute();
```
SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: *Decouple* the code and the data

```php
$db = new mysql("localhost", "user", "pass", "DB");

$statement = $db->prepare("select * from Users where(name=? and password=?);");

$result = mysql_query("select * from Users where(name='\$user' and password='\$pass');");

$statement->bind_param("ss", $user, $pass);
$statement->execute();
```

```php
$db = new mysql("localhost", "user", "pass", "DB");

$statement = $db->prepare("select * from Users where(name=? and password=?);");

$statement->bind_param("ss", $user, $pass);
$statement->execute();
```

Bind variables
SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: Decouple the code and the data

```
$result = mysql_query("select * from Users
    where(name='\$user' and password='\$pass');");
```

$db = new mysql("localhost", "user", "pass", "DB");

```php
$statement = $db->prepare("select * from Users
    where(name=? and password=?);");  

$statement->bind_param("ss", $user, $pass);

$statement->execute();
```

Bind variables are typed
SQL injection countermeasures

3. Prepared statements & bind variables

Key idea: Decouple the code and the data

```
$result = mysql_query("select * from Users
    where(name='\$user' and password='\$pass');");
```

$db = new mysql("localhost", "user", "pass", "DB");

$statement = $db->prepare("select * from Users
    where(name=? and password=?);”);

Bind variables

Decoupling lets us compile now, before binding the data

$statement->bind_param("ss", $user, $pass);

Bind variables are typed
The underlying issue

```php
$statement = $db->prepare("select * from Users
    where(name=? and password=?);");
```
The underlying issue

```php
$statement = $db->prepare("select * from Users where(name=? and password=?);");
```
The underlying issue

```php
$statement = $db->prepare("select * from Users
where(name=? and password=?);");
```

Prepare is only applied to the leaves, so the structure of the tree is fixed.
Mitigating the impact

- **Limit privileges**
  - Can limit commands and/or tables a user can access
    - Allow SELECT queries on Orders_Table but not on Creditcards_Table
  - Follow the principle of least privilege
  - Incomplete fix, but helpful

- **Encrypt sensitive data** stored in the database
  - May not need to encrypt Orders_Table
  - But certainly encrypt Creditcards_Table.cc_numbers
FOLLOWUP READING

Steve Friedl’s Unixwiz.net Tech Tips

SQL Injection Attacks by Example

A customer asked that we check out his intranet site, which was used by the company’s employees and customers. This was part of a larger security review, and though we'd not actually used SQL injection to penetrate a network before, we were pretty familiar with the general concepts. We were completely successful in this engagement, and wanted to recount the steps taken as an illustration.

"SQL Injection" is subset of an unverified/unsanitized user input vulnerability ("buffer overflows" are a different subset), and the idea is to convince the application to run SQL code that was not intended. If the application is creating SQL strings naively on the fly and then running them, it's straightforward to create some real surprises.

We'll note that this was a somewhat winding road with more than one wrong turn, and others with more experience will certainly have different -- and better -- approaches. But the fact that we were successful does suggest that we were not entirely misguided.

There have been other papers on SQL injection, including some that are much more detailed, but this one shows the rationale of discovery as much as the process of exploitation.

The Target Intranet

This appeared to be an entirely custom application, and we had no prior knowledge of the application nor access to the source code; this was a "blind" attack. A bit of poking showed that this server ran Microsoft's IIS 6 along with ASP.NET, and this suggested that the database was Microsoft's SQL server; we believe that these techniques can apply to nearly any web application backed by any SQL server.

The login page had a traditional username-and-password form, but also an email-me-my-password link; the latter proved to be the downfall of the whole system.

When entering an email address, the system presumably looked in the user database for that email address, and mailed something to that address. Since my email address is not found, it wasn't going to send me anything.

So the first test in any SQL-ish form is to enter a single quote as part of the data; the intention is to see if they construct an SQL string literally without sanitizing. When submitting the form with a quote in the email address, we got a 500 error (server failure), and this suggests that the "broken" input is actually being parsed literally. Bingo.

We speculate that the underlying SQL code looks something like this:

```
SELECT Fieldlist
FROM Table
WHERE Field = '$EMAIL';
```

Here, $EMAIL is the address submitted on the form by the user, and the larger query provides the quotation marks that set it off as a literal string. We don't know the specific names of the fields or table involved, but we do know their nature, and we'll make some good guesses later.