



Databases

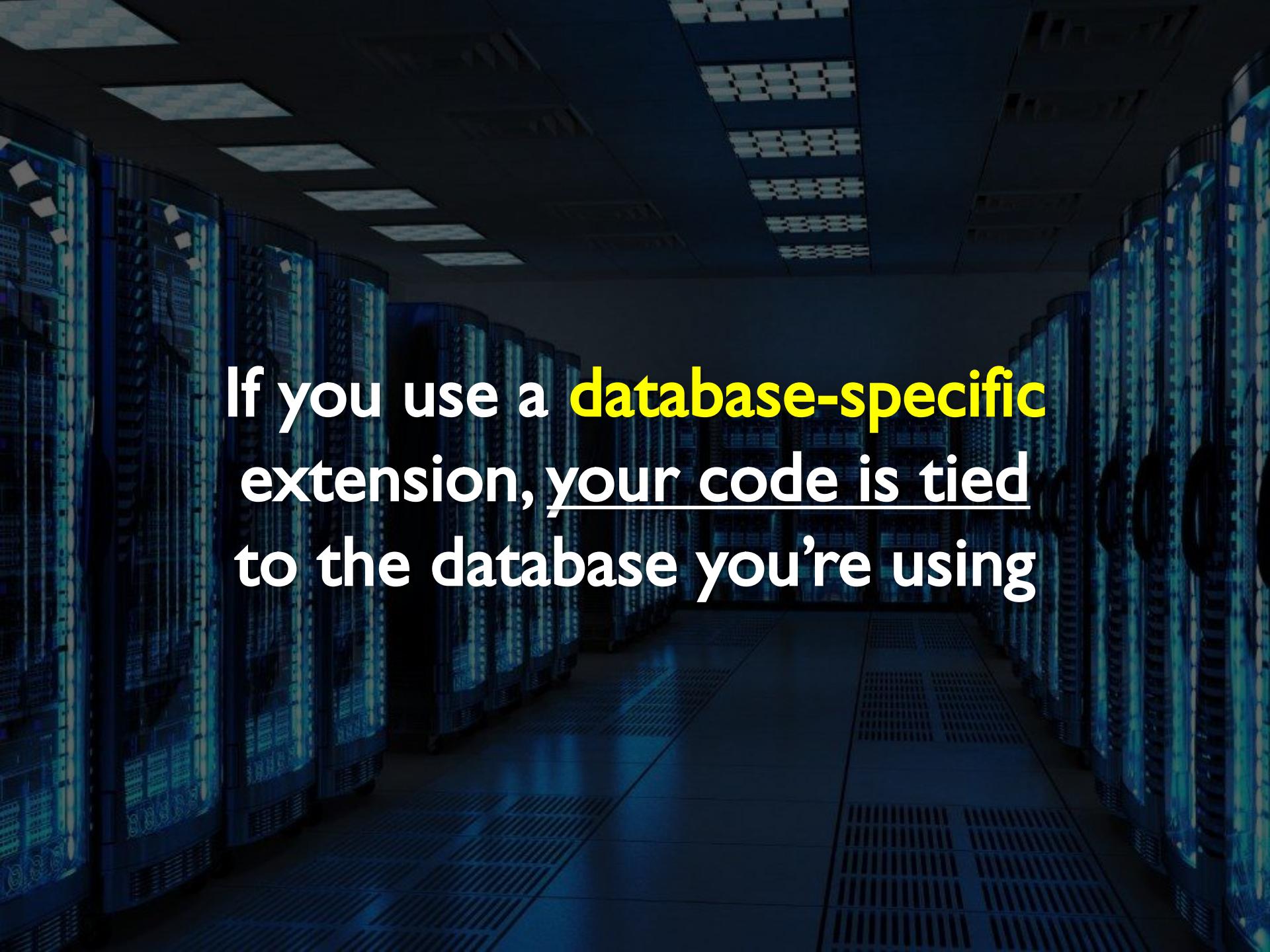
The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks. The racks are arranged in two main rows, receding into the distance. The floor has a metal grate pattern. The overall atmosphere is high-tech and industrial.

There are two ways to
access databases from PHP:

1. Using a **database-specific** extension
2. Using the **database-independent**
PDO (PHP Data Objects) library

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

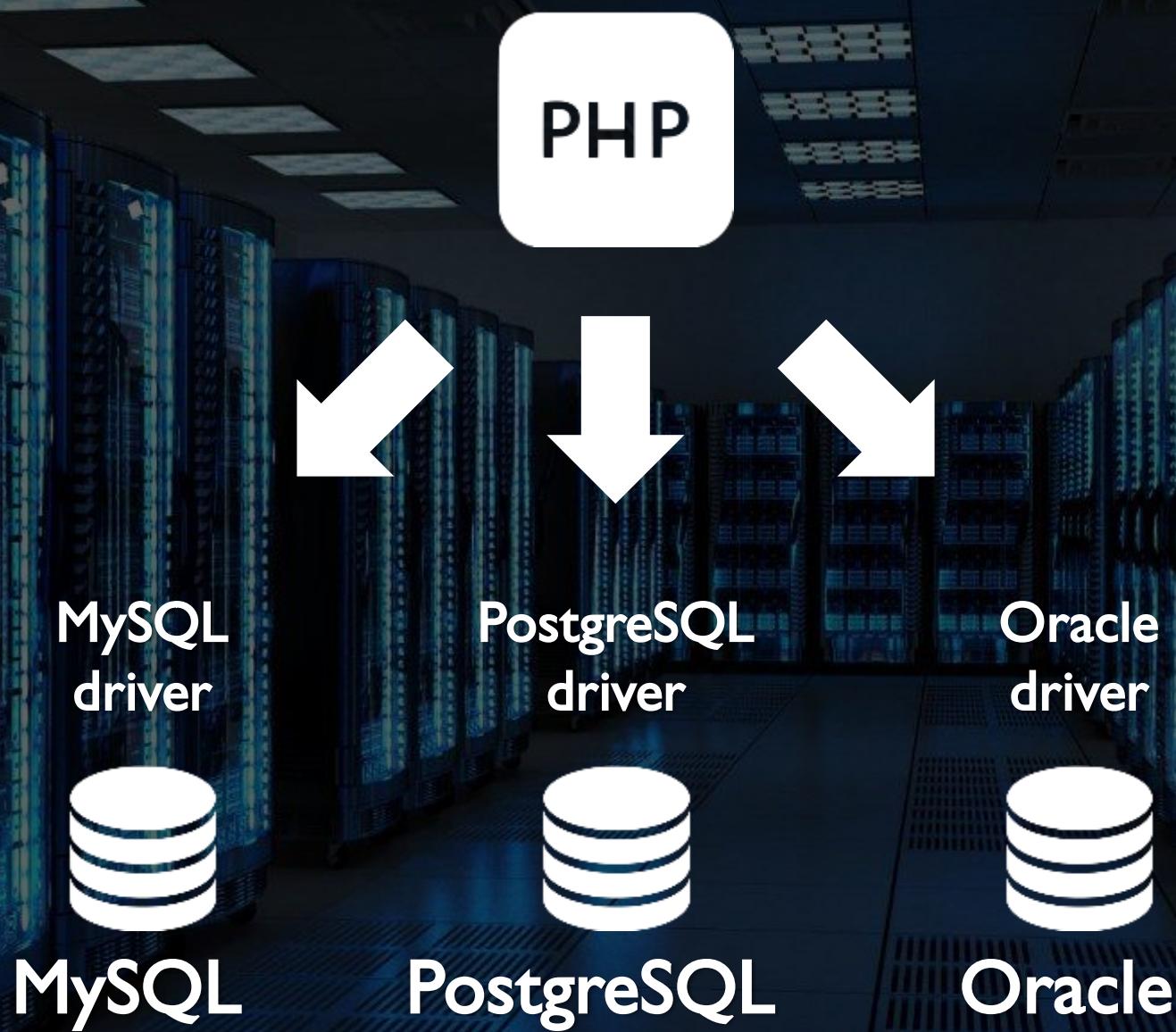
There are **advantages**
and **disadvantages**
to each approach

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

If you use a **database-specific extension**, your code is tied to the database you're using

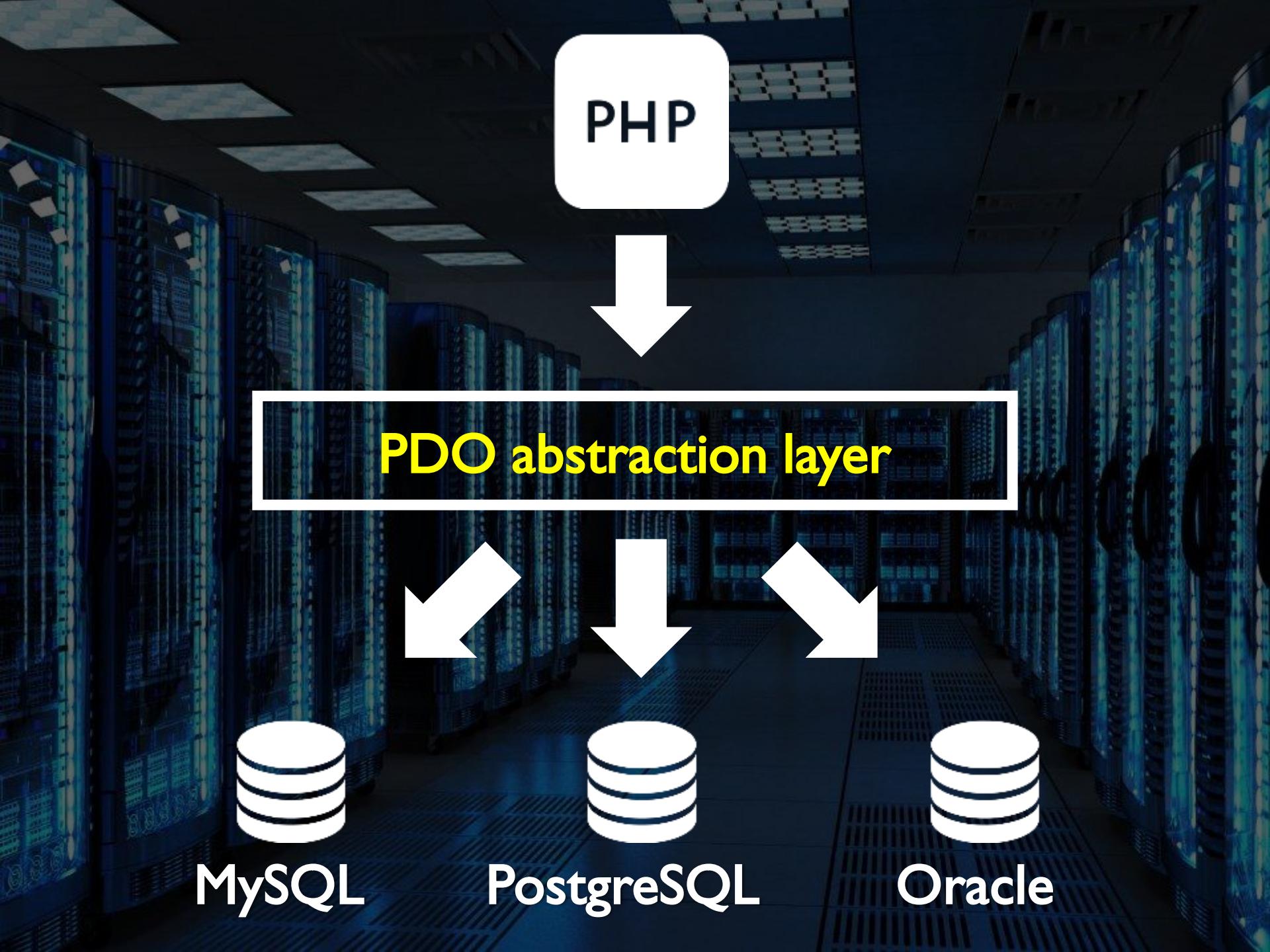
The background of the slide shows a dimly lit server room. Rows of server racks are visible, their front panels glowing with a bright blue light. The ceiling features a grid of rectangular light fixtures. The overall atmosphere is one of a high-tech, industrial data center.

If you want to move your
database from **MySQL** to
PostgreSQL, it will involve
significant changes to your code



A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

PDO, on the other hand,
hides the database-specific
functions from us with an
abstraction layer



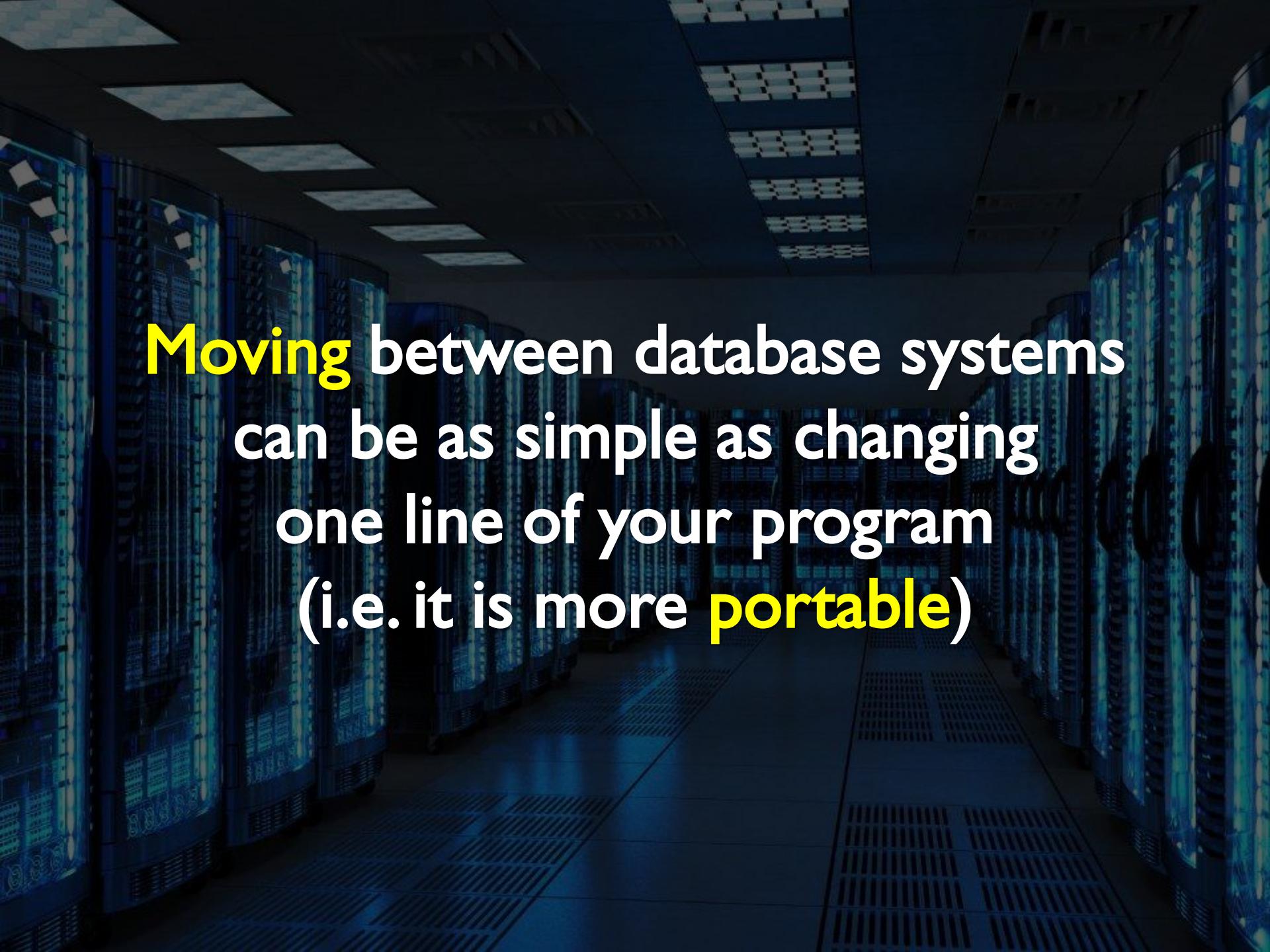
PHP

PDO abstraction layer

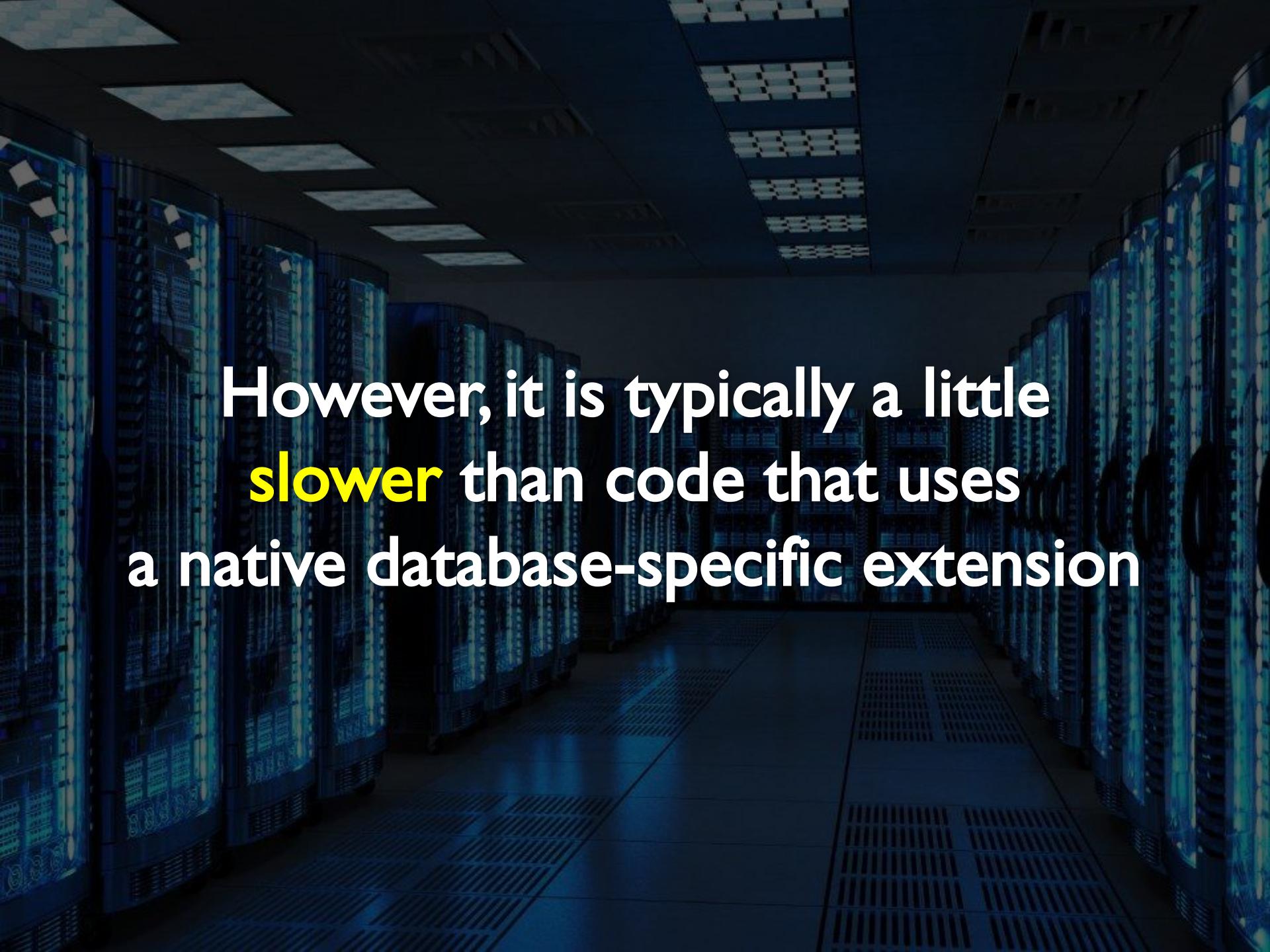
MySQL

PostgreSQL

Oracle

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

**Moving between database systems
can be as simple as changing
one line of your program
(i.e. it is more portable)**

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor is made of metal grates. The ceiling has several rectangular light fixtures. The perspective is looking down a aisle between the racks.

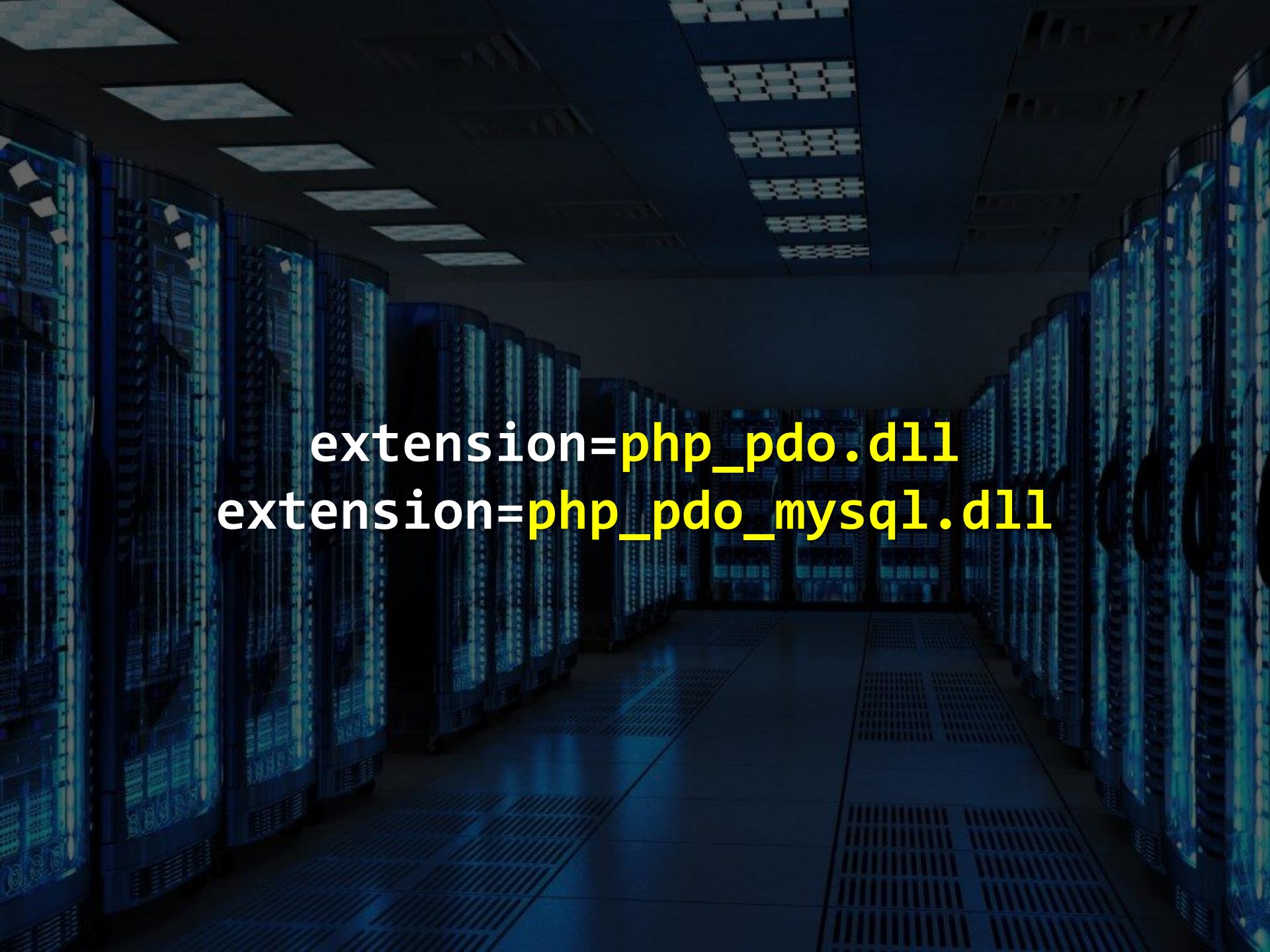
However, it is typically a little
slower than code that uses
a native database-specific extension

PDO

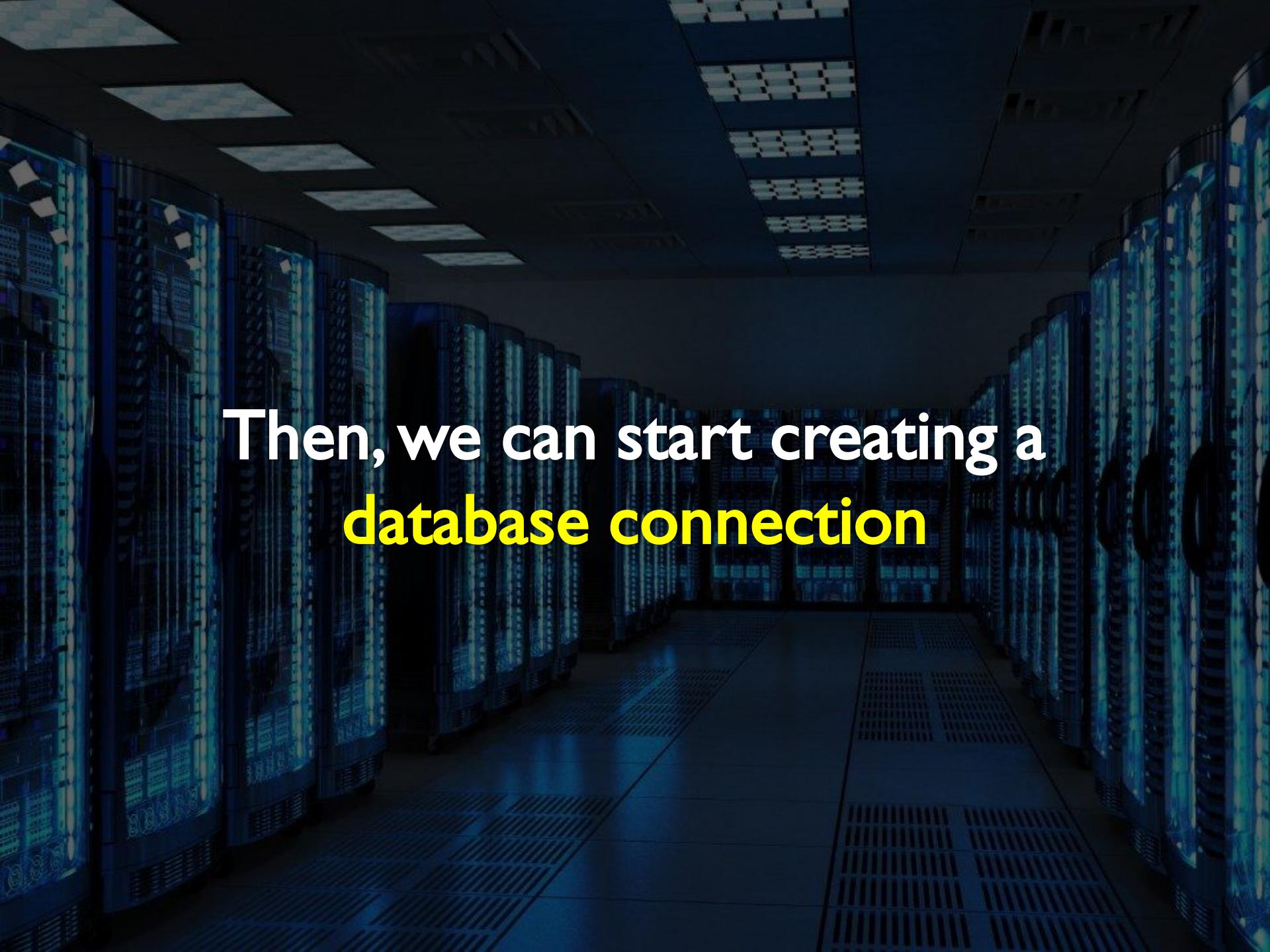


The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks, each with multiple glowing blue LED lights. The floor is made of dark tiles, and the ceiling has several rectangular light fixtures. The overall atmosphere is tech-oriented and professional.

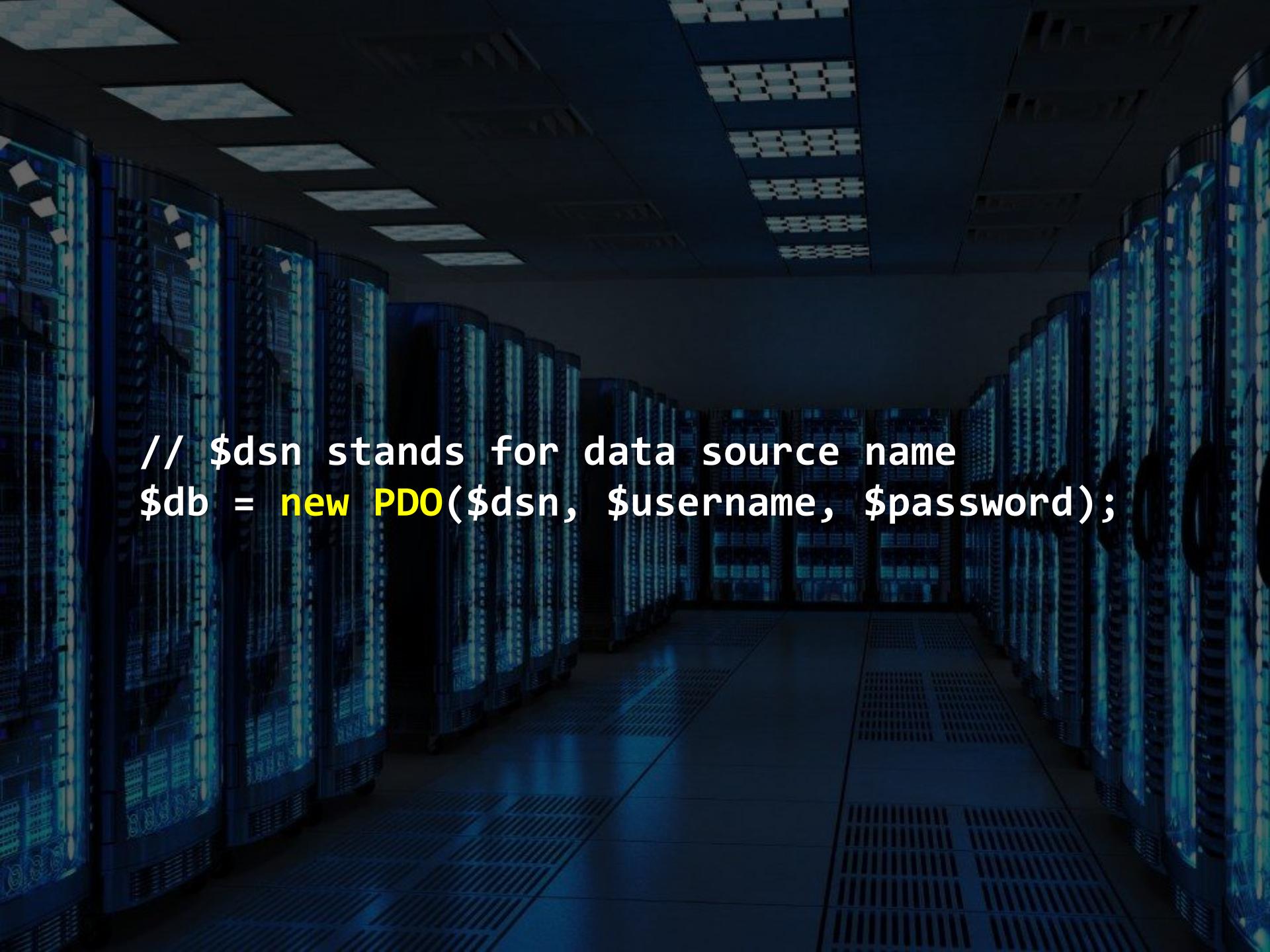
First, to use **PDO**, we need to
enable **PDO** extension in our
php.ini file

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

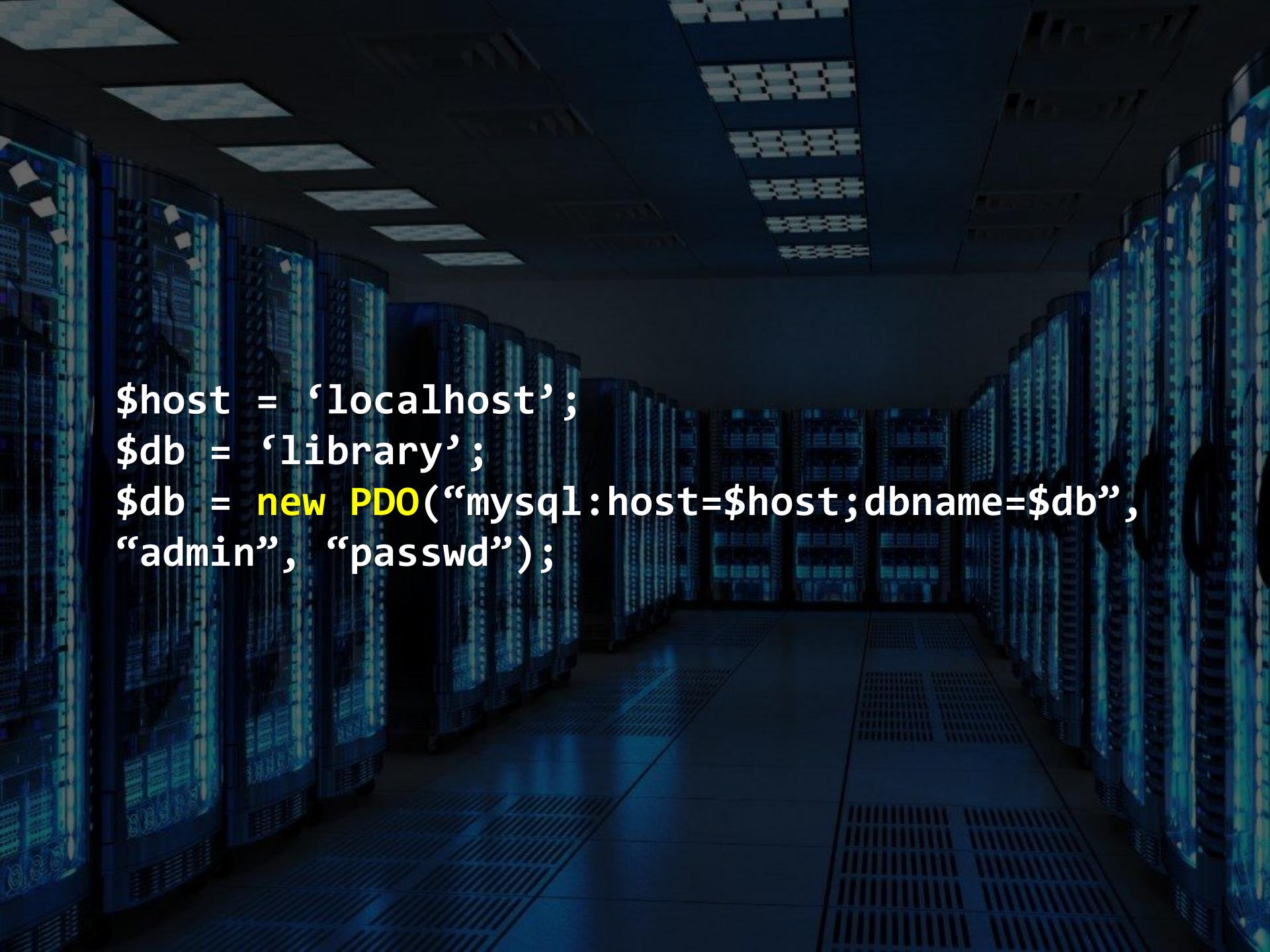
extension=php_pdo.dll
extension=php_pdo_mysql.dll

The background of the slide shows a dark server room. On both sides, there are long rows of server racks. The front panels of the servers are illuminated with a bright blue light, showing various ports and status indicators. The floor is made of grey tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is tech-oriented and professional.

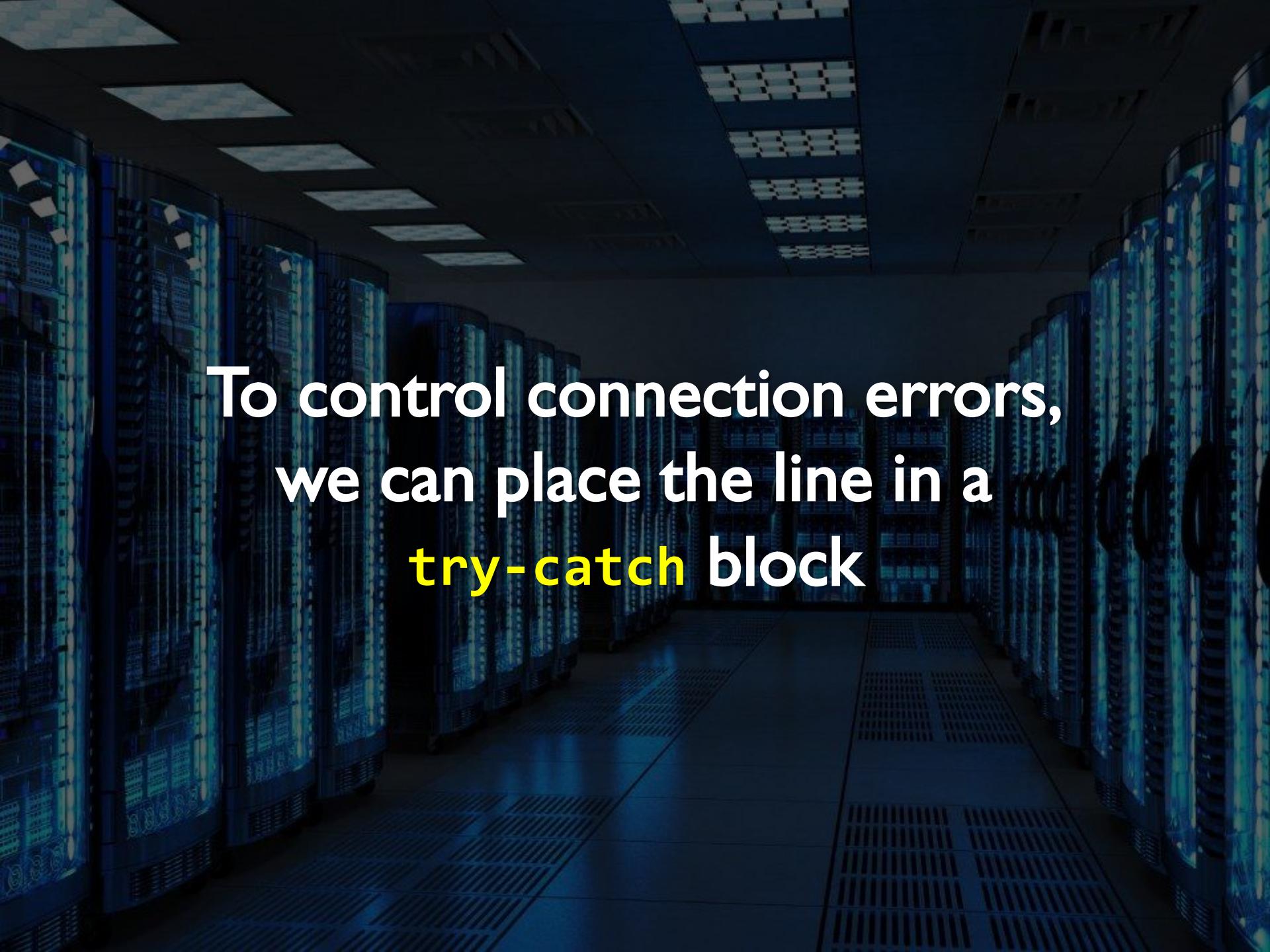
Then, we can start creating a
database connection

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing panels on the front of the server racks. These panels display various status indicators and data. The racks are arranged in two main rows that recede into the distance, creating a perspective effect. The floor is made of polished tiles, reflecting some of the ambient light. The overall atmosphere is one of a high-tech, industrial data center.

```
// $dsn stands for data source name  
$db = new PDO($dsn, $username, $password);
```

A dark server room with rows of server racks. The racks are illuminated from within, showing glowing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

```
$host = 'localhost';
$db = 'library';
$db = new PDO("mysql:host=$host;dbname=$db",
"admin", "passwd");
```

The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks. The front panels of the servers are illuminated with a bright blue light, showing various status indicators and data. The floor is made of polished tiles, reflecting the ambient light. The ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial data center.

To control connection errors,
we can place the line in a
try-catch block

```
$host = 'localhost';
$db = 'library';

try {
    $db = new PDO("mysql:host=$host;dbname=$db",
                  "admin", "passwd");
} catch (PDOException $e) {
    die('Error: ' . $e->getMessage() . '<br>');
}
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing panels on the server racks. These racks are arranged in two long rows that recede into the distance, creating a perspective effect. The floor is made of polished tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

It is **important to catch the exception to prevent PHP to display database connection details including username and password**

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the numerous server racks themselves, which are illuminated from within, creating a blue glow. The racks are arranged in two main rows that recede into the distance, creating a sense of depth. The floor is made of polished concrete with a grid pattern. The ceiling is dark with several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

Once we have the database connection, we can use the connection to send **SQL** commands with the **query** method

```
$sql = "SELECT * FROM student";
```

```
// Returns a PDOStatement object  
$query = $db->query($sql);
```

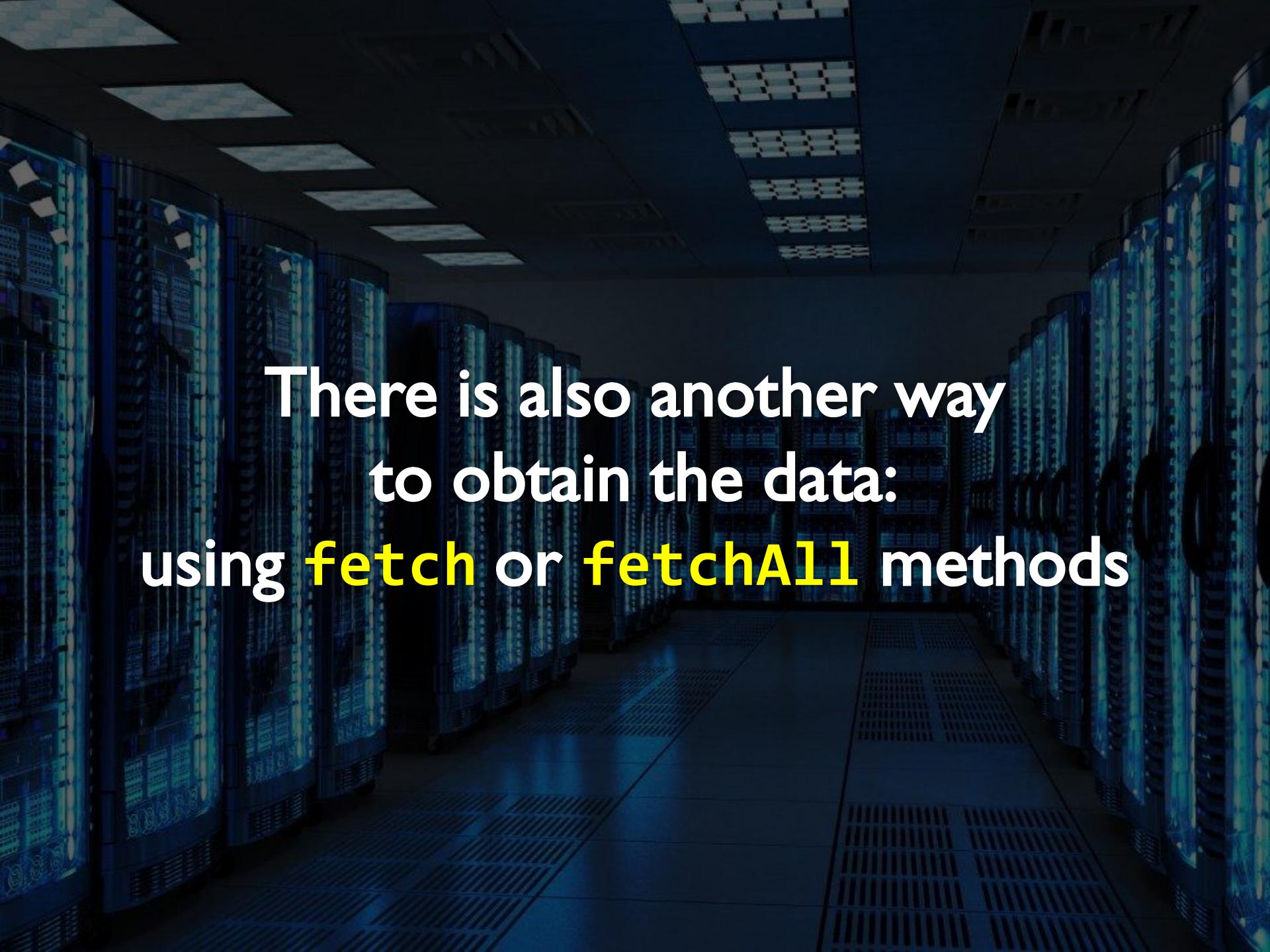
The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks. The front panels of the servers are illuminated with a bright blue light, showing various status indicators and data. The floor is made of polished concrete with a grid pattern. The ceiling has several rectangular recessed lights. The overall atmosphere is high-tech and industrial.

We can then **loop** through the
PDOStatement object **directly**
to obtain the data

```
$sql = "SELECT * FROM student";  
  
// Returns a PDOStatement object  
$query = $db->query($sql);  
  
foreach ($query as $row) {  
    // Column name as array index  
    echo $row['nim'], ', ', $row['name'], '<br>;  
}
```

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

**1234 Tim
1235 Berners
1236 Lee**

A dark server room with rows of server racks. The racks have glowing blue screens displaying binary code. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

There is also another way
to obtain the data:
using **fetch** or **fetchAll** methods

The background of the slide shows a dark server room with rows of server racks. The racks are illuminated from within, showing blue and white patterns that resemble circuit boards or data streams. The floor is made of metal grates, and the ceiling has several rectangular light fixtures. The overall atmosphere is tech-oriented and futuristic.

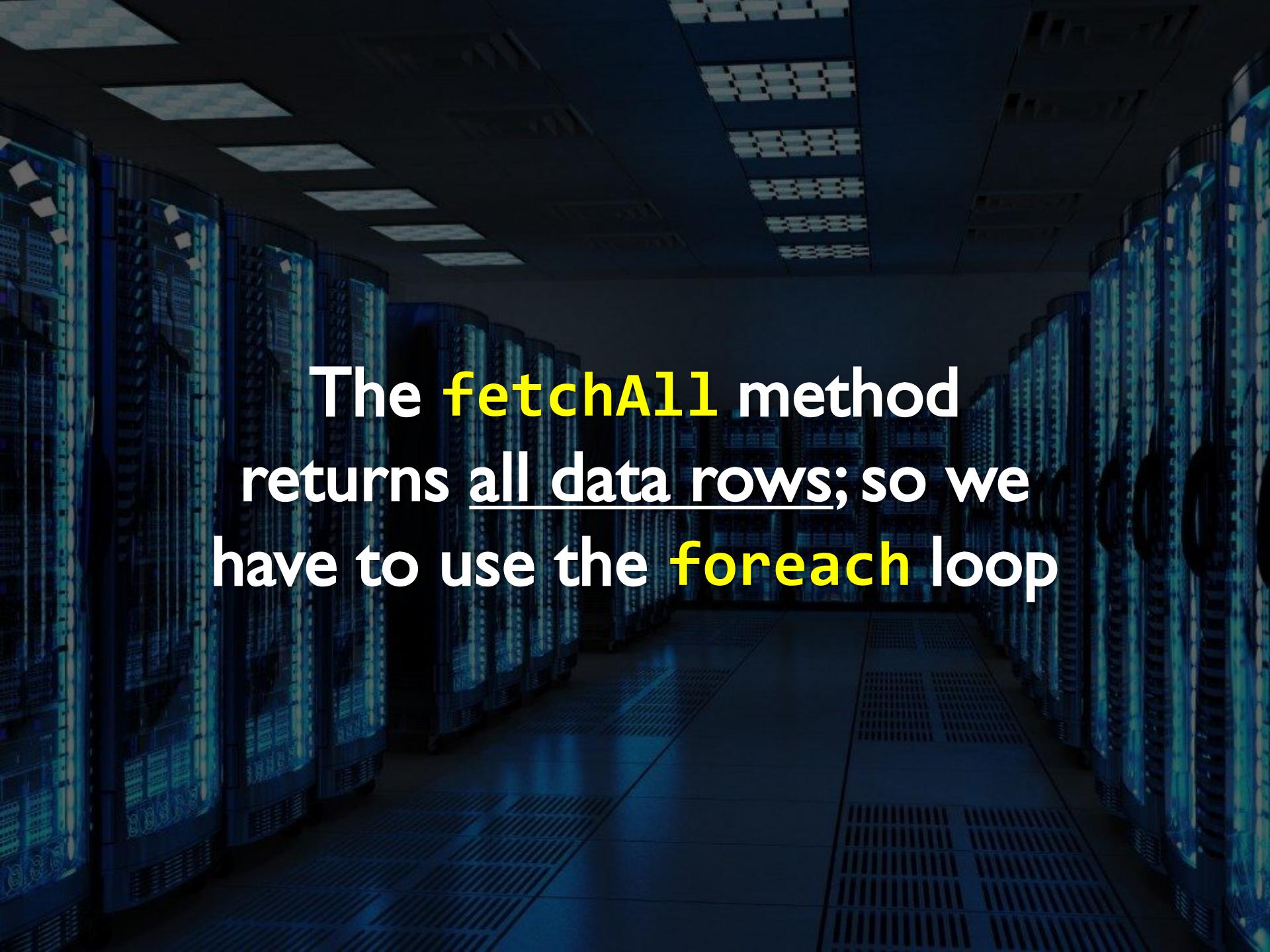
The **fetch** method returns
one data row at a time; so we
have to use the **while** loop

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);

while ($row = $query->fetch()) {
    // Using indexed array
    echo $row[0], $row[1], $row[2];

    // Using associative array
    echo $row['id'], $row['nim'], $row['name'];
}

}
```

The background of the slide shows a dark server room with multiple rows of server racks. The racks have blue glowing screens displaying data. The ceiling has several rectangular light fixtures. The overall atmosphere is dim and tech-oriented.

The **fetchAll** method
returns all data rows; so we
have to use the **foreach** loop

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);
$rows = $query->fetchAll();

foreach ($rows as $row) {
    // Using indexed array
    echo $row[0], $row[1], $row[2];

    // Using associative array
    echo $row['id'], $row['nim'], $row['name'];
}
```

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and various components. The floor has a metal grate pattern. The ceiling has several rectangular light fixtures.

We can specify a **fetch mode** to
the **fetch** and **fetchAll** method

Fetch modes we can use:

- `FETCH_ASSOC`
- `FETCH_NUM`
- `FETCH_BOTH` (**default**)
- `FETCH_OBJ`
- `FETCH_BOUND`
- `FETCH_CLASS`
- `FETCH_FUNC`

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);
$rows = $query->fetchAll(PDO::FETCH_NUM);

foreach ($rows as $row) {
    // Can use indexed array only
    echo $row[0], $row[1], $row[2];

    // Cannot use associative array
    // echo $row['id'], $row['nim'], $row['name'];
}
```

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);
$rows = $query->fetchAll(PDO::FETCH_ASSOC);

foreach ($rows as $row) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Can use associative array only
    echo $row['id'], $row['nim'], $row['name'];
}
```

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);
$rows = $query->fetchAll(PDO::FETCH_BOTH);

foreach ($rows as $row) {
    // Can use indexed array
    echo $row[0], $row[1], $row[2];

    // Or associative array
    echo $row['id'], $row['nim'], $row['name'];
}
```

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);
$rows = $query->fetchAll(PDO::FETCH_OBJ);

foreach ($rows as $row) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Nor associative array
    // echo $row['id'], $row['nim'], $row['name'];

    // Must use object-style
    echo $row->id, $row->nim, $row->name;
}
```

A dark server room with rows of server racks. The racks have glowing blue screens displaying data. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

We can **bind** the data
to our variables
by **column index**

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);

$query->bindColumn(1, $id);
$query->bindColumn(2, $nim);
$query->bindColumn(3, $name);

while ($row = $query->fetch(PDO::FETCH_BOUND)) {
    echo $id, '<br>';
    echo $nim, '<br>';
    echo $name, '<br>';
}
```

A dark server room with rows of server racks. The racks have glowing blue screens displaying data. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

We can also **bind** the data
to our variables
by **column name**

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);

$query->bindColumn('id', $id);
$query->bindColumn('nim', $nim);
$query->bindColumn('name', $name);

while ($row = $query->fetch(PDO::FETCH_BOUND)) {
    echo $id, '<br>';
    echo $nim, '<br>';
    echo $name, '<br>';
}
```

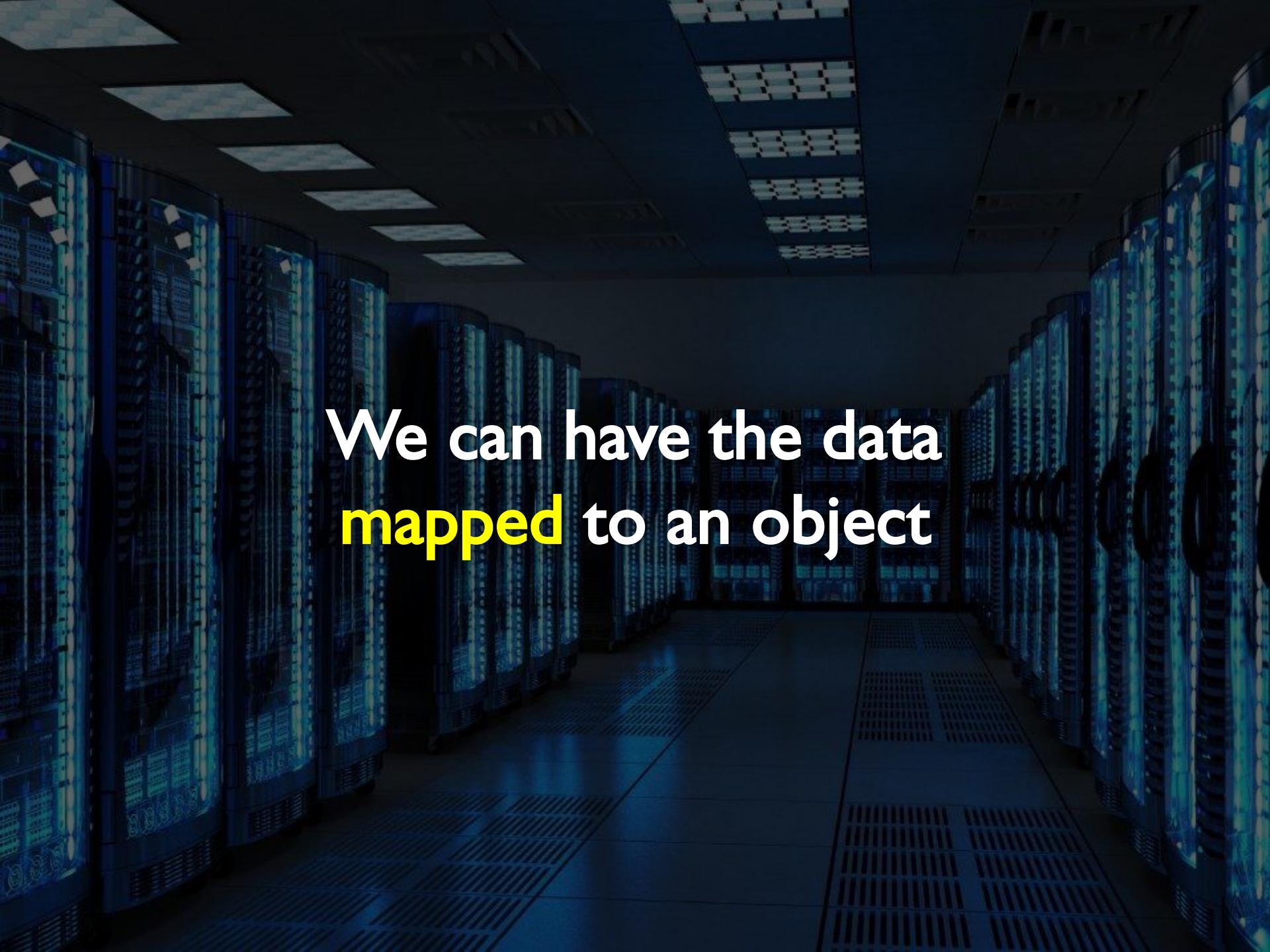
The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks, each with multiple glowing blue lights indicating active hardware. The floor is made of polished tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is high-tech and industrial.

We can even **bind** the data
to our variables by both
column name and **column index**

```
$sql = "SELECT * FROM student";
$query = $db->query($sql);

$query->bindColumn('id', $id);
$query->bindColumn(2, $nim);
$query->bindColumn('name', $name);

while ($row = $query->fetch(PDO::FETCH_BOUND)) {
    echo $id, '<br>';
    echo $nim, '<br>';
    echo $name, '<br>';
}
```

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit板 patterns. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

We can have the data
mapped to an object

```
class Student {  
    // must be public  
    public $id;  
    public $nim;  
    public $name;  
}  
  
$sql = "SELECT * FROM student";  
$query = $db->query($sql);  
$rows = $query->fetchAll(PDO::FETCH_CLASS);  
  
foreach ($rows as $student) {  
    echo $student->id, $student->nim, $student->name;  
}
```

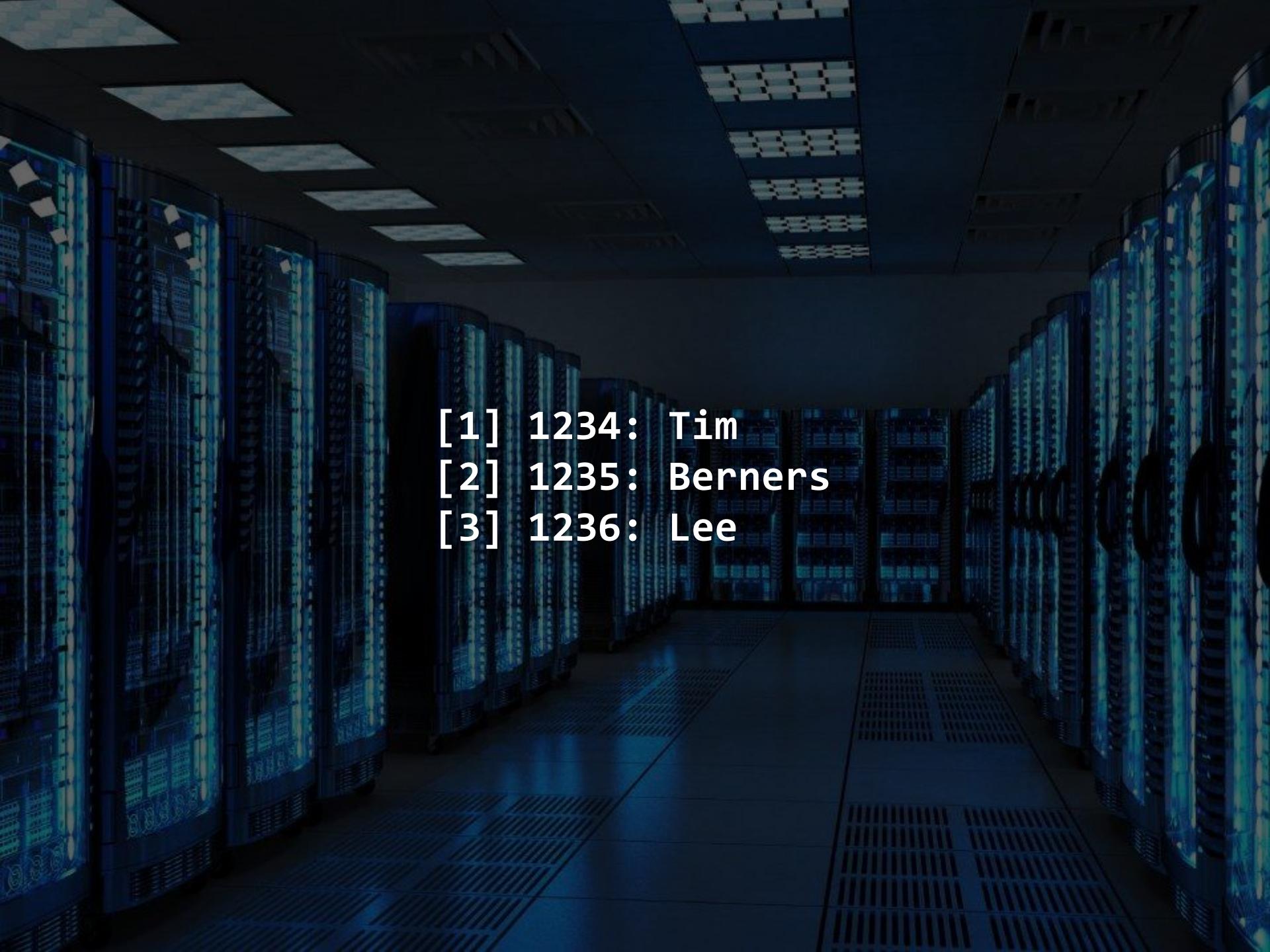
The background of the slide shows a dark server room with multiple rows of server racks. The racks have blue glowing screens displaying data. The ceiling has several rectangular light fixtures. The overall atmosphere is dim and tech-oriented.

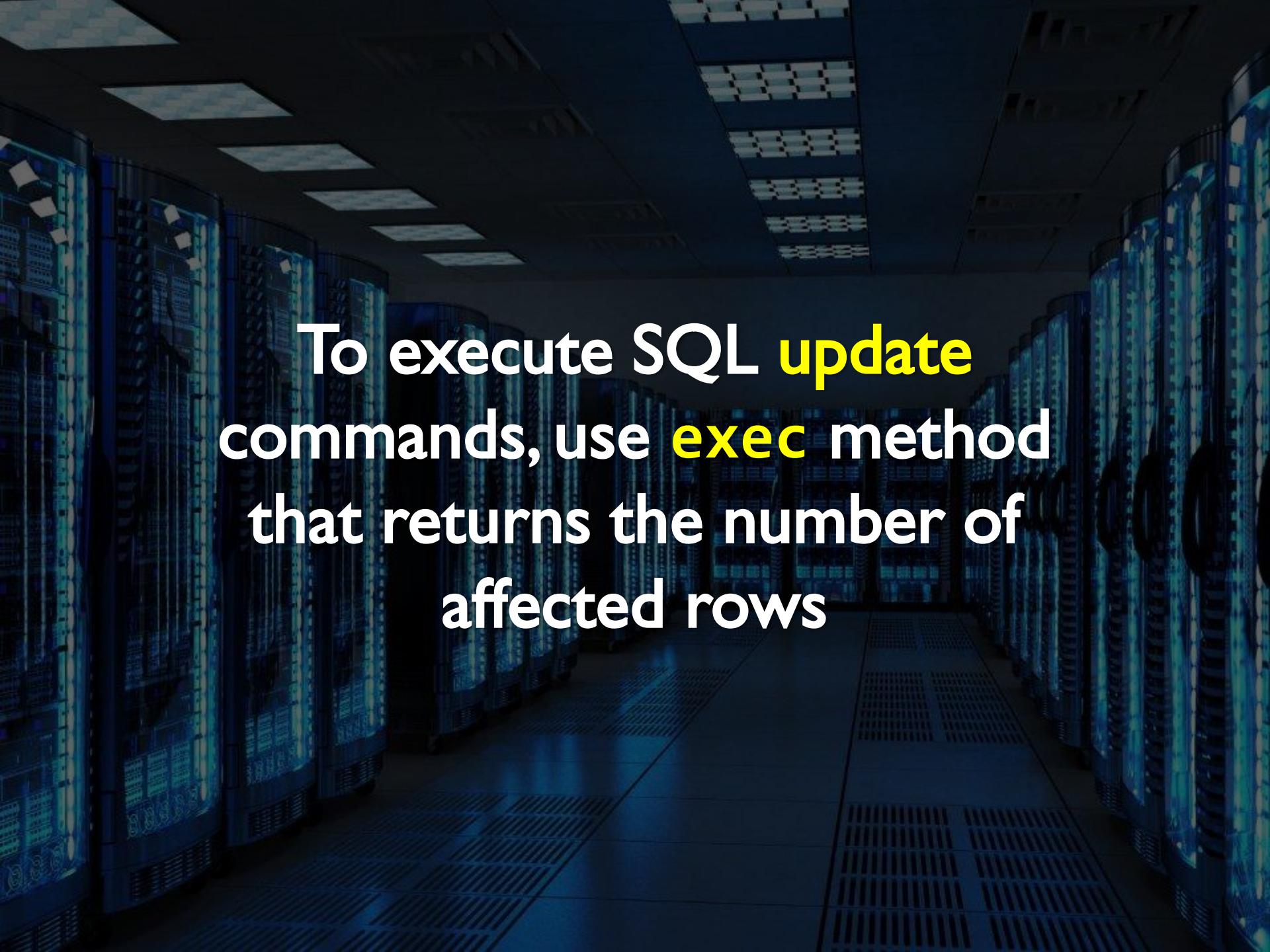
The **FETCH_FUNC** mode can be
used if want to “preprocess”
our data by passing it to a function

```
function preprocess($id, $nim, $name) {
    return “[{$id}] {$nim}: {$name}”;
}

$sql = “SELECT * FROM student”;
$query = $db->query($sql);
$rows = $query->fetchAll(PDO::FETCH_FUNC, ‘preprocess’);

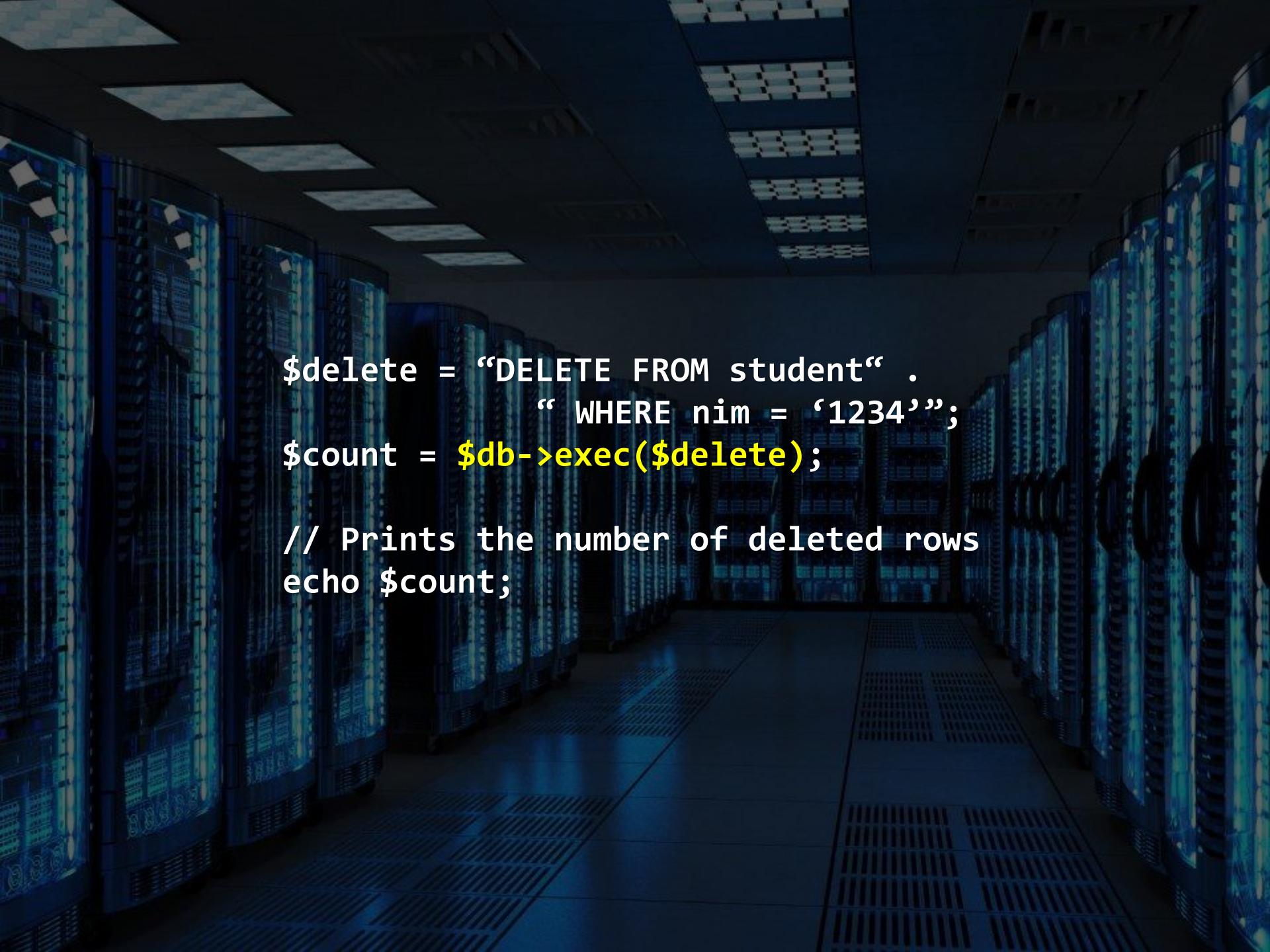
foreach ($rows as $row) {
    echo $row, ‘  
’;
}
```

- 
- A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor is made of metal grates. The ceiling has several rectangular light fixtures.
- [1] 1234: Tim
 - [2] 1235: Berners
 - [3] 1236: Lee

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks themselves. These racks are arranged in two long rows that recede into the distance, creating a perspective effect. The floor is made of polished tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial data center.

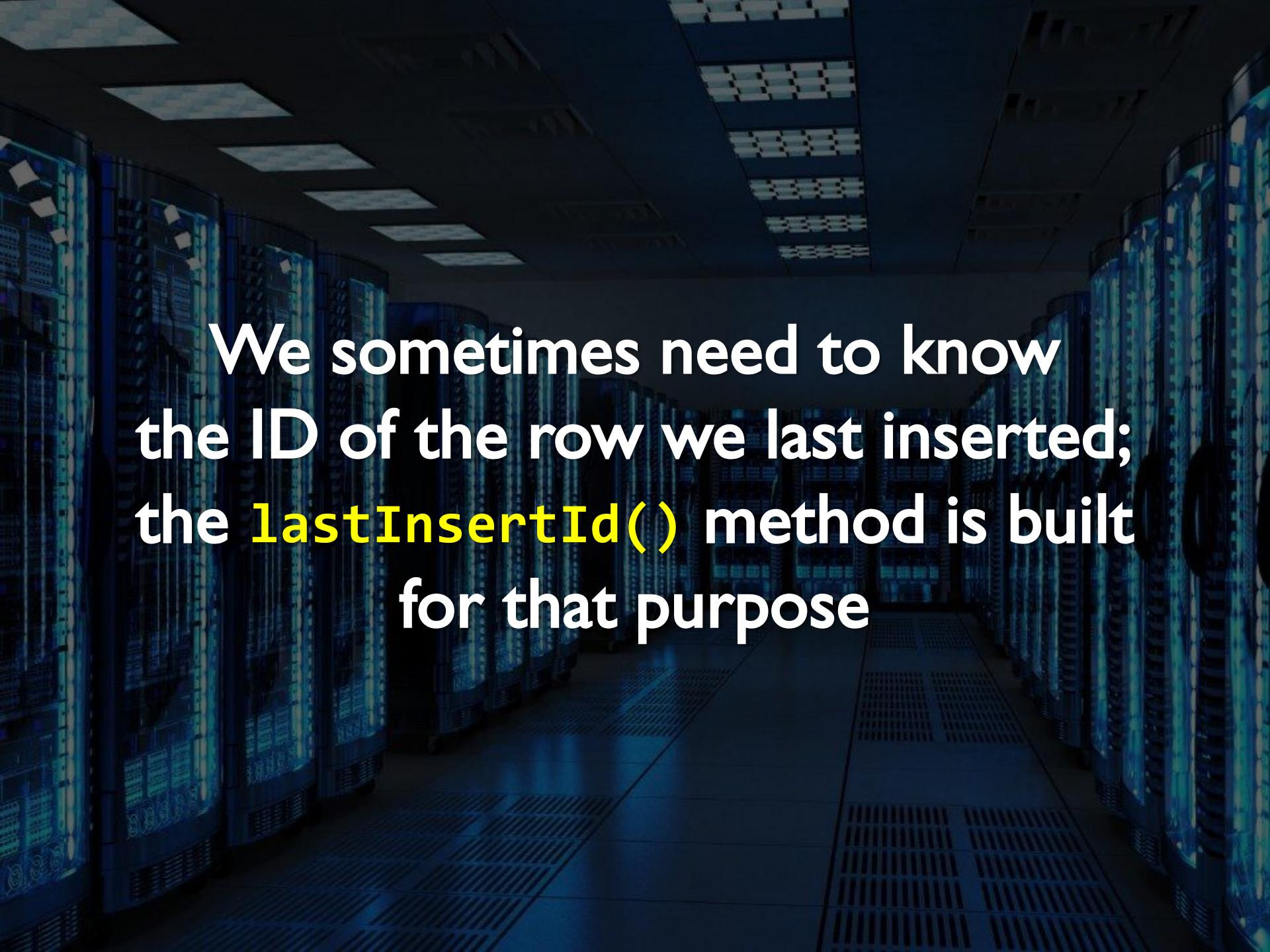
To execute SQL **update**
commands, use **exec** method
that returns the number of
affected rows

```
$update = "UPDATE student SET name = 'Lee'".  
        " WHERE nim = '1234'";  
$count = $db->exec($update);  
  
// Prints the number of updated rows  
echo $count;
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks. There are several rows of server racks extending from the foreground into the distance. The floor has a dark, polished surface with some reflective highlights. The overall atmosphere is one of a high-tech, industrial data center.

```
$delete = "DELETE FROM student" .  
        " WHERE nim = '1234'";  
$count = $db->exec($delete);  
  
// Prints the number of deleted rows  
echo $count;
```

```
$insert = "INSERT INTO student (nim, name)" .  
        " VALUES ('1236', 'Dennis')";  
$count = $db->exec($insert);  
  
// Prints the number of inserted rows  
echo $count;
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing panels on the server racks. These racks are arranged in two long rows that recede into the distance, creating a perspective effect. The floor is made of polished tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

We sometimes need to know
the ID of the row we last inserted;
the `lastInsertId()` method is built
for that purpose

7

1234

Tim

8

1235

Berners

9

1236

Lee

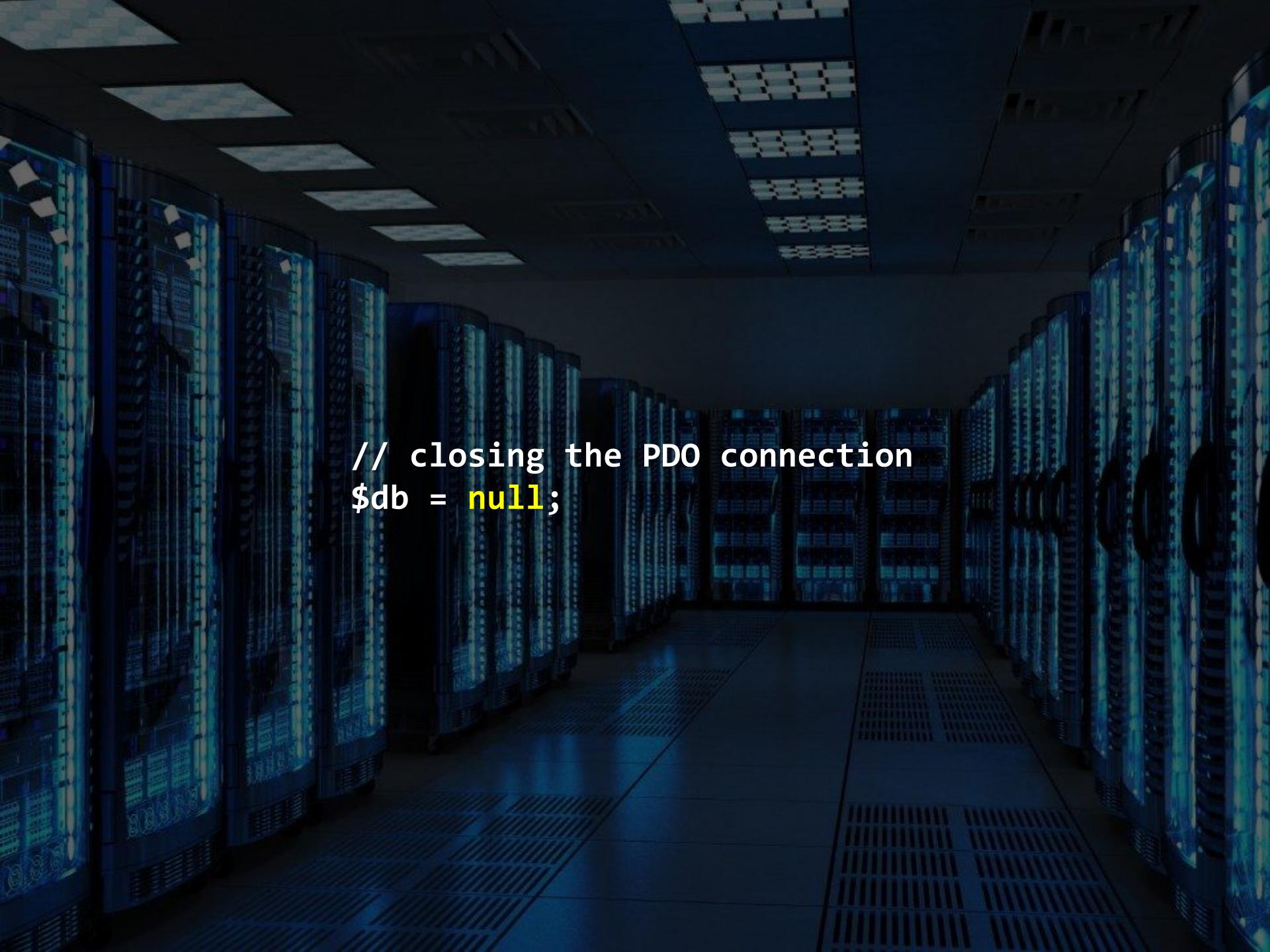
```
$insert = "INSERT INTO student (nim, name)" .  
          " VALUES ('1236', 'Dennis')";  
$count = $db->exec($insert);  
  
// Prints the number of inserted rows  
echo $count;  
  
// Prints the id of the new row  
echo $db->lastInsertId();
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing panels on the server racks. The racks are arranged in two main rows, receding into the distance. The floor has a metal grate pattern. The overall atmosphere is high-tech and industrial.

If we are to execute **multiple**
SQL statements, the **exec** method
is not recommended, i.e. it is slow;
use **prepared statement** instead

The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks. The front panels of the servers are illuminated with a bright blue light, showing various status indicators and data. The floor is made of polished tiles, reflecting the ambient light. The ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

We can **close** the PDO connection
by assigning **null** to the
connection variable

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

```
// closing the PDO connection  
$db = null;
```

Prepared Statements



A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

**It is often better to prepare
our SQL commands
before executing them**

Connection

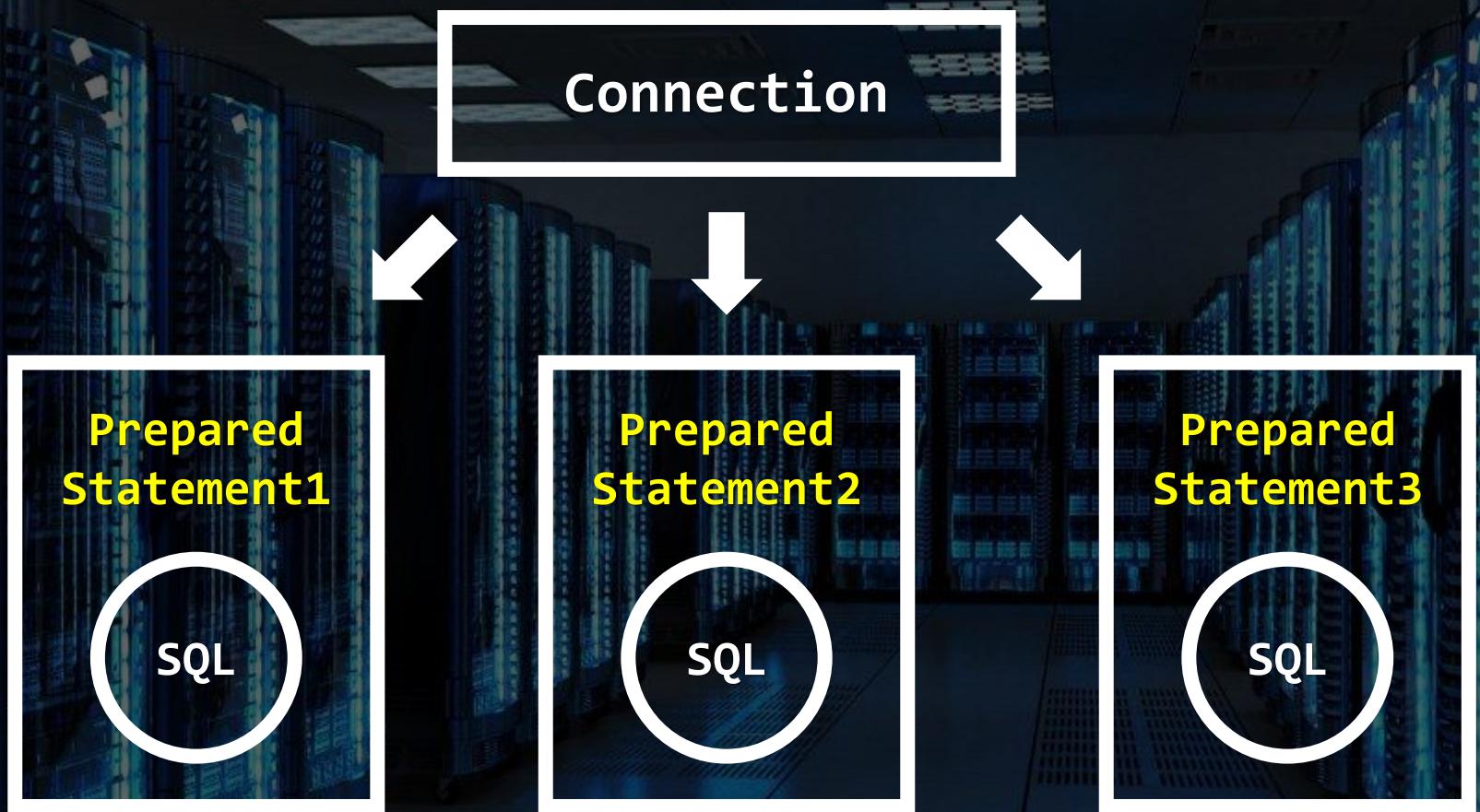
The diagram illustrates a flow of data or operations from a central point down to three distinct components. At the top, a white rectangular box contains the word "Connection". A thick white arrow points downwards from this box to another white rectangular box below it, which contains the word "Statement". From the "Statement" box, three separate white arrows point downwards to three circular icons at the bottom. These icons are labeled "SQL1", "SQL2", and "SQL3" respectively. The background of the diagram is a dark image of a server room with multiple server racks.

Statement

SQL1

SQL2

SQL3



The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks, their front panels illuminated with a bright blue light that casts a glow on the floor and the ceiling. The ceiling features several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial data center.

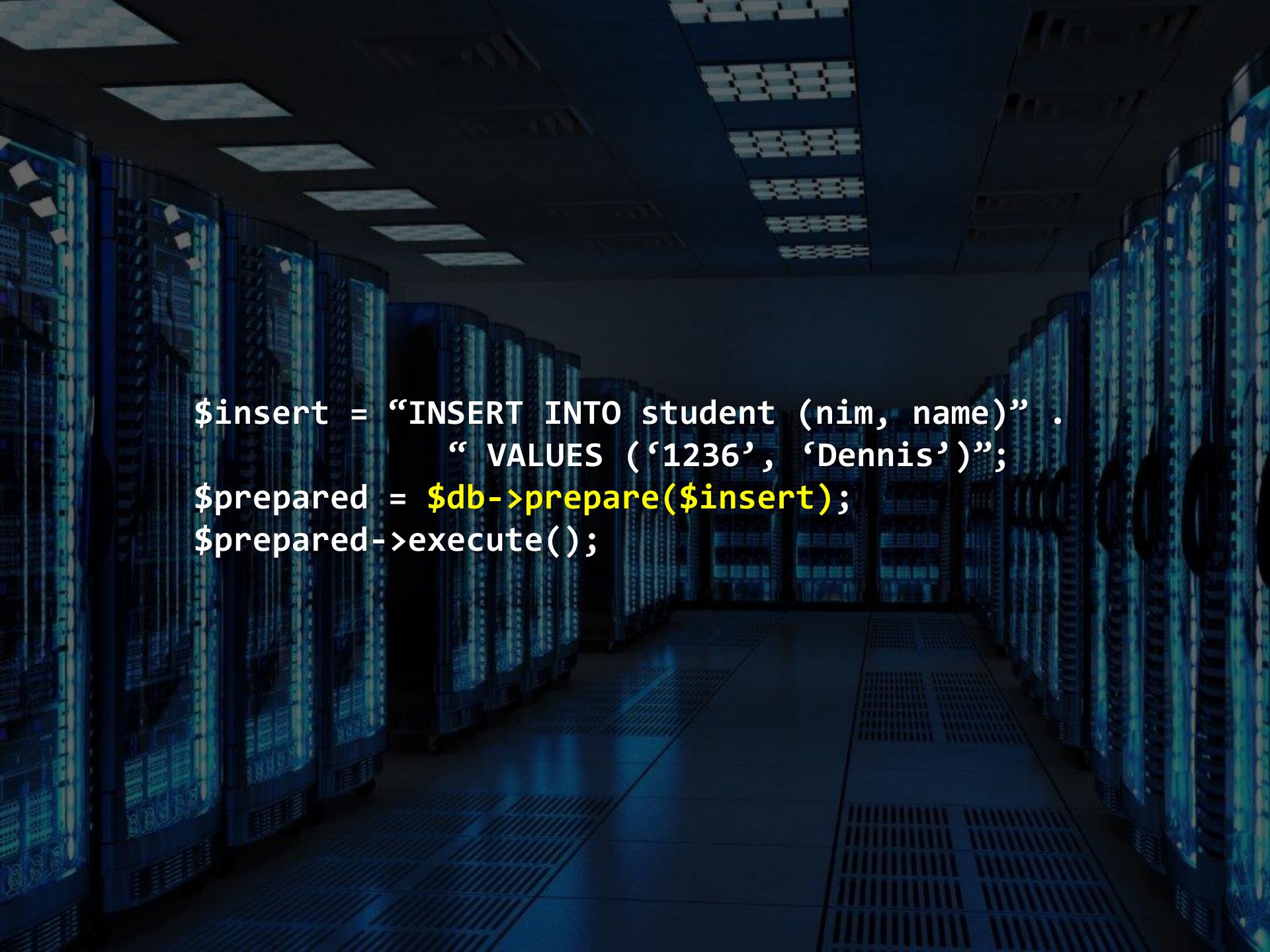
It also offers better performance,
especially when sending multiple
SQL statements



**Prepared statements can be used
with various SQL commands:
SELECT, INSERT, UPDATE, DELETE**

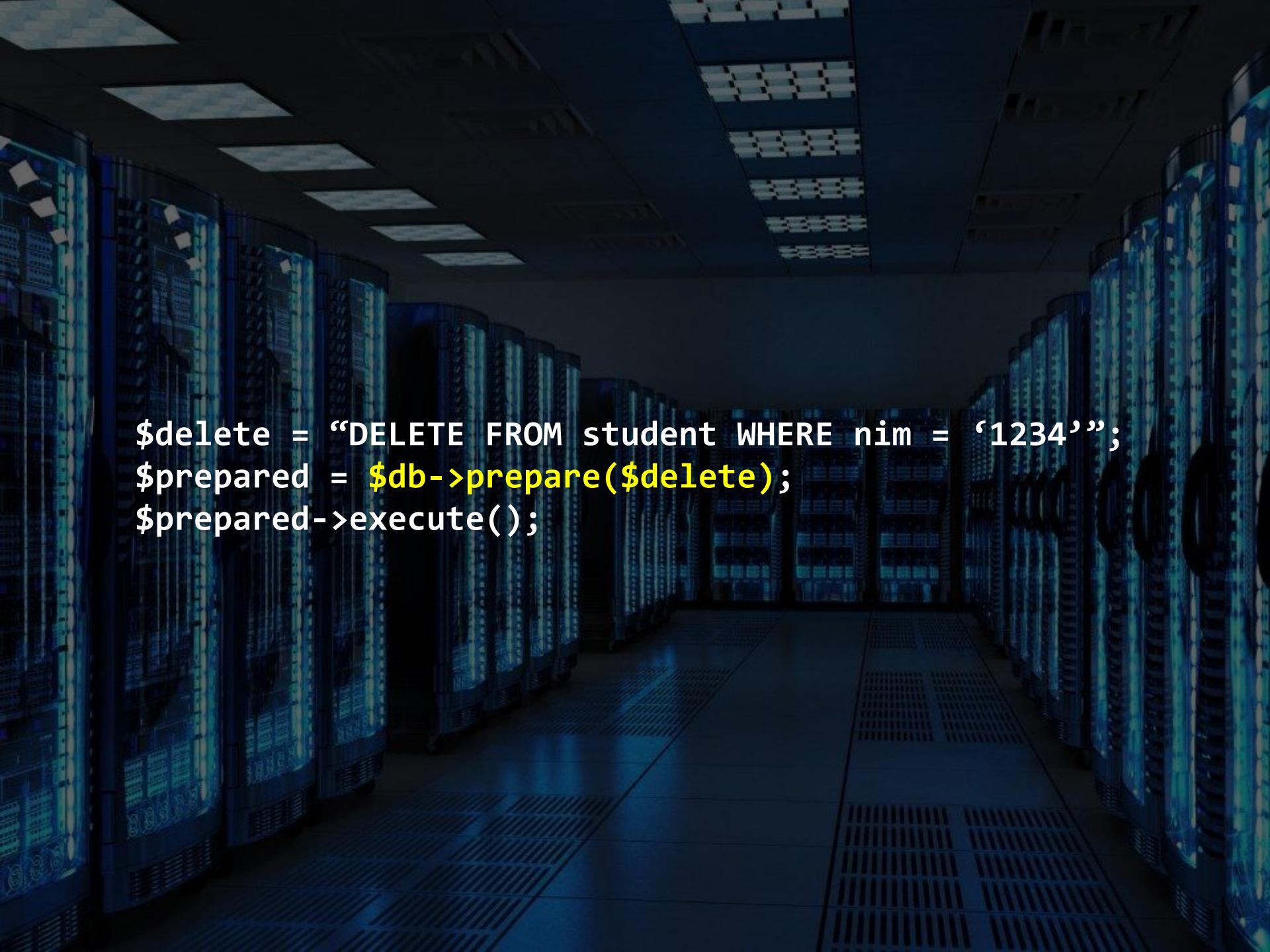
```
$select = "SELECT * FROM student";
$prepared = $db->prepare($select);
$prepared->execute();
$rows = $prepared->fetchAll(PDO::FETCH_OBJ);

foreach ($rows as $row) {
    echo $row->id, '<br>';
    echo $row->nim, '<br>';
    echo $row->name, '<br>';
}
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks. There are several rows of server racks extending into the distance, creating a perspective effect. The floor has a dark, polished surface with some reflective highlights.

```
$insert = "INSERT INTO student (nim, name)" .  
        " VALUES ('1236', 'Dennis')";  
$prepared = $db->prepare($insert);  
$prepared->execute();
```

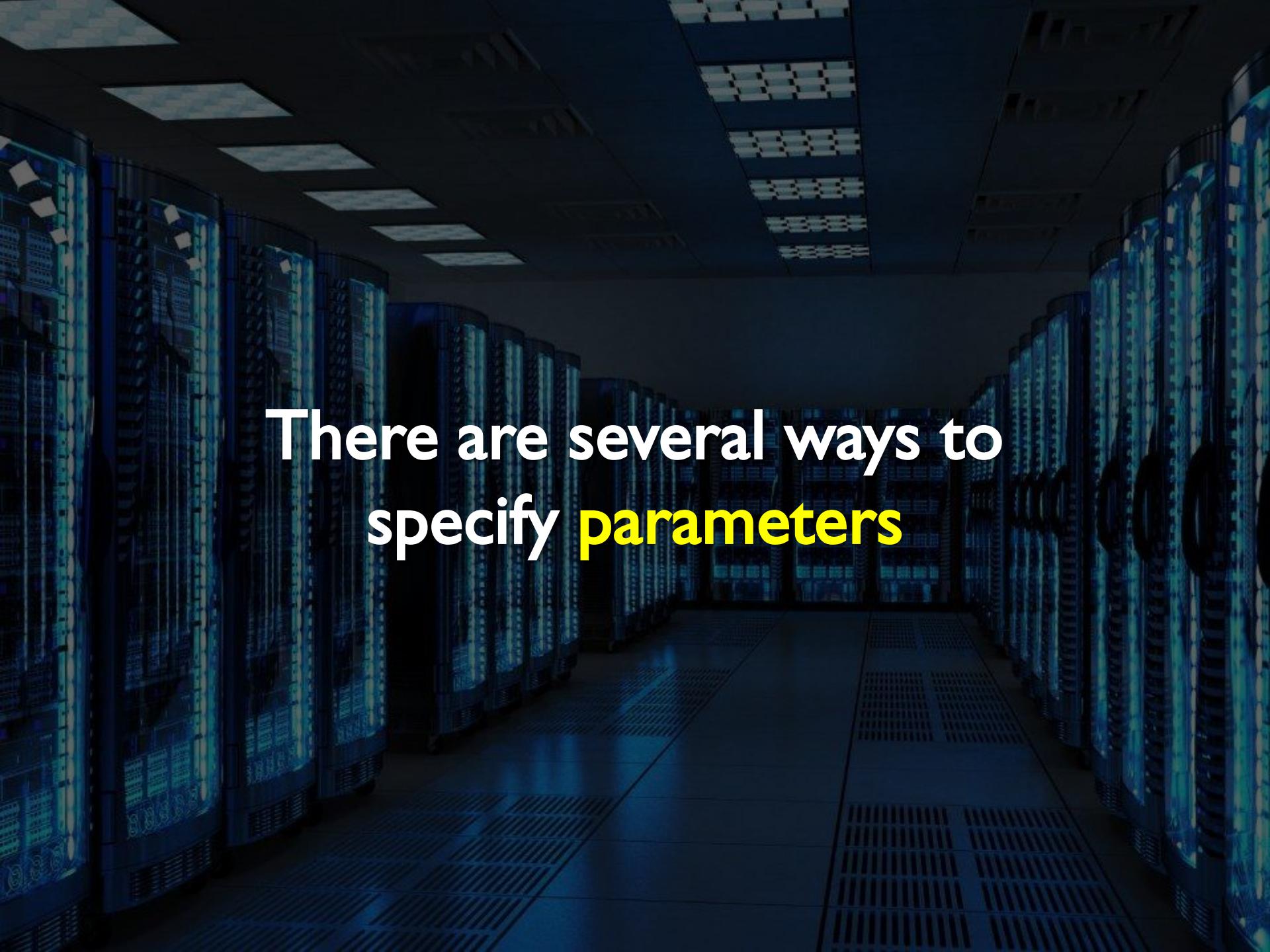
```
$update = “UPDATE student SET name = ‘Ritchie’” .  
        “ WHERE nim = ‘1234’”;  
$prepared = $db->prepare($update);  
$prepared->execute();
```

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks. There are several rows of server racks extending into the distance, creating a perspective effect. The floor has a metal grate pattern. The overall atmosphere is futuristic and tech-oriented.

```
$delete = "DELETE FROM student WHERE nim = '1234'";
$prepared = $db->prepare($delete);
$prepared->execute();
```

A dark server room with rows of server racks. The racks have glowing blue screens displaying binary code. The floor has a metal grate. The ceiling has several rectangular light fixtures.

One advantage of prepared
statements is that we can
use **parameters**

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit板 patterns. The floor has a metal grating. The ceiling has several rectangular light fixtures.

There are several ways to
specify **parameters**

```
// Using question-mark parameter
$insert = "INSERT INTO student (nim, name) VALUES (?, ?)";
$prepared = $db->prepare($insert);

// With bindValue() method
$prepared->bindValue(1, '1234');
$prepared->bindValue(2, 'Smith');
$prepared->execute();
```

```
// Using question-mark parameter
$insert = "INSERT INTO student (nim, name) VALUES (?, ?)";
$prepared = $db->prepare($insert);

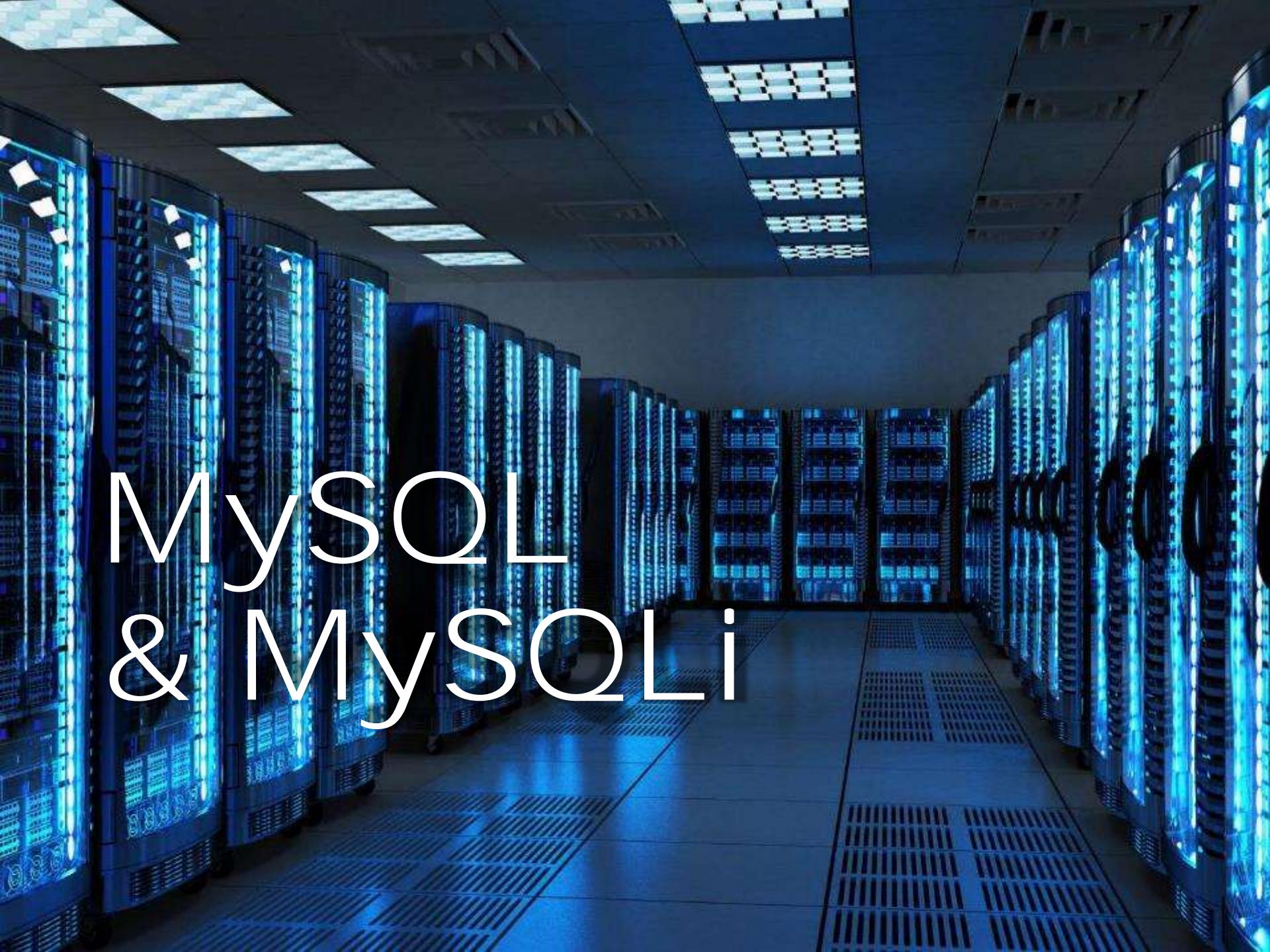
// Directly to the execute() method
$prepared->execute(array('1234', 'Smith'));
```

```
// Using named parameter
$insert = "INSERT INTO student (nim, name)" .
    " VALUES (:nim, :name)";
$prepared = $db->prepare($insert);

// With bindValue() method
$prepared->bindValue(1, '1234');
$prepared->bindValue(2, 'Smith');
$prepared->execute();
```

```
// Using named parameter
$insert = "INSERT INTO student (nim, name) .
           " VALUES (:nim, :name)";
$prepared = $db->prepare($insert);

// Directly to the execute() method
$prepared->execute(
    array(
        'nim' => '1234',
        'name' => 'Smith'
    )
);
```



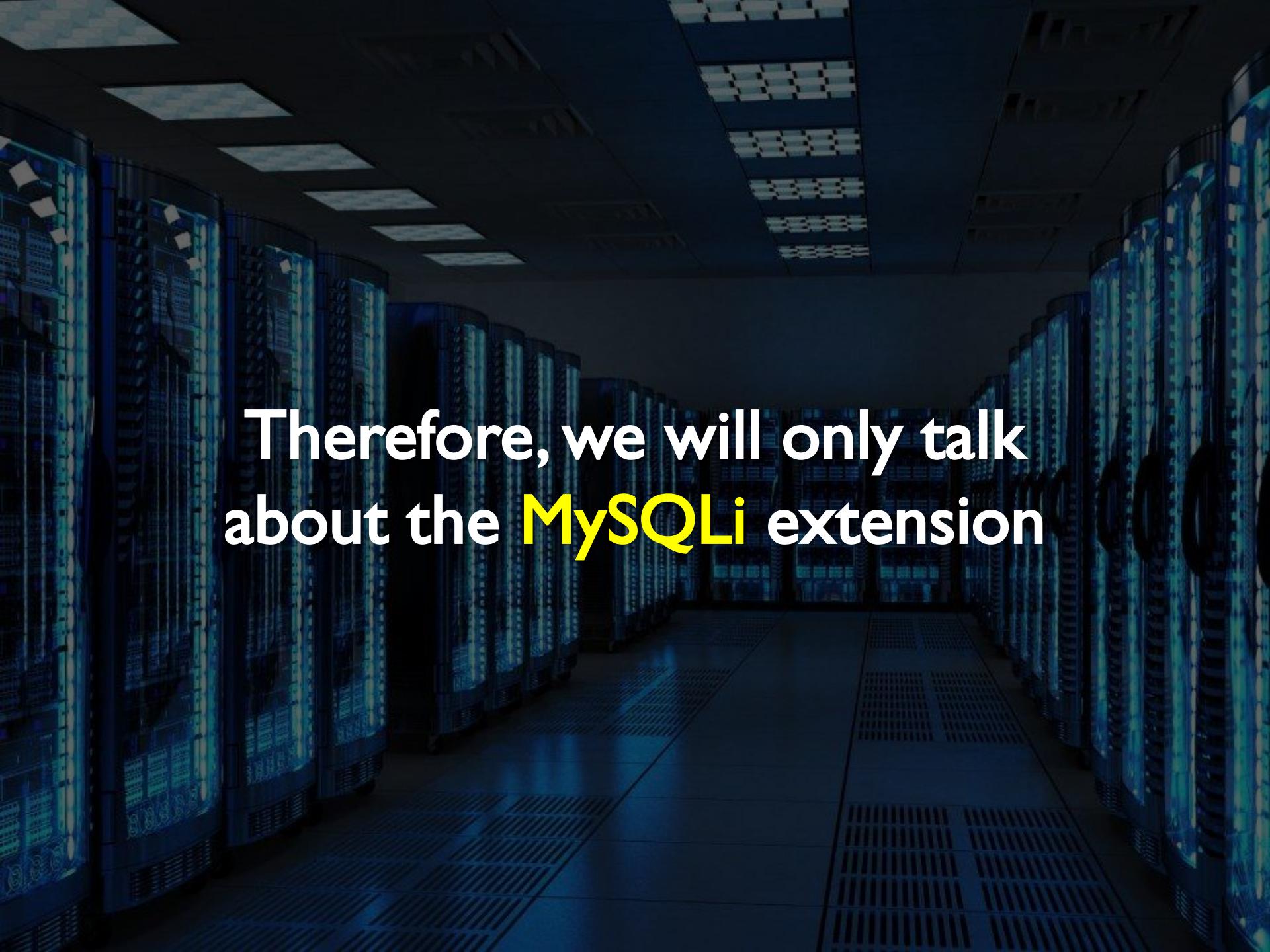
MySQL
& MySQLi



**MySQL and MySQLi are
MySQL-specific database
extensions**

The background of the slide shows a dimly lit server room. Rows of server racks are visible on both sides, their front panels glowing with a bright blue light. The floor is made of dark tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is technical and modern.

However, MySQL extension is
deprecated; we should use
MySQLi or **PDO_MySQL** instead

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks themselves. These racks are arranged in two long rows that recede into the distance, creating a sense of depth. The floor is made of grey tiles, and the ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

Therefore, we will only talk
about the **MySQLi** extension

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing panels on the server racks. These racks are arranged in two long rows that recede into the distance, creating a perspective effect. The floor is made of polished tiles, reflecting some of the ambient light. The overall atmosphere is one of a high-tech, industrial environment.

**MySQLi extension supports
procedural and object-oriented
programming style**

```
// Procedural style
$mysqli = mysqli_connect($host, $user, $pass, $db);

if (mysqli_connect_errno($mysqli)) {
    echo "Error: ", mysqli_connect_error();
}
```

```
// Object-oriented style
$mysqli = new mysqli($host, $user, $pass, $db);

if ($mysqli->connect_errno) {
    echo "Error: ", $mysqli->connect_error;
}
```

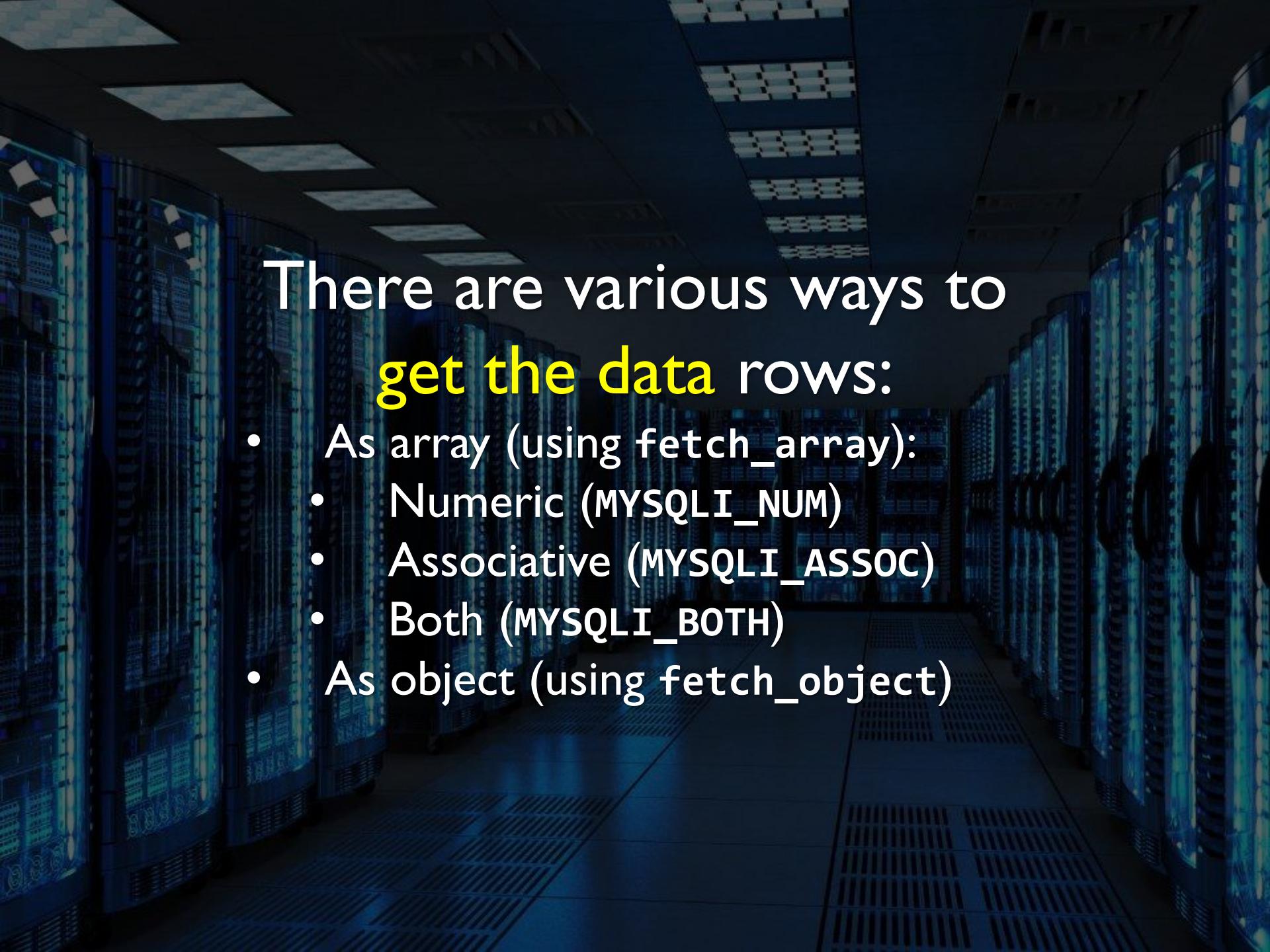
The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks. The front panels of the servers are illuminated with a bright blue light, showing various status indicators and data. The floor is made of polished tiles, reflecting the ambient light. The ceiling has several rectangular recessed lights. In the distance, a person wearing a white lab coat and safety glasses is visible, looking towards the camera.

It is possible to **switch between styles** at any time although it is
not recommended

```
$query = $mysqli->query("SELECT * FROM student");

// Prints the number of rows
// Object-oriented style
echo $query->num_rows;

// Procedural style
echo mysqli_num_rows($query);
```



There are various ways to
get the data rows:

- As array (using `fetch_array`):
 - Numeric (`MYSQLI_NUM`)
 - Associative (`MYSQLI_ASSOC`)
 - Both (`MYSQLI_BOTH`)
- As object (using `fetch_object`)

```
// Object-oriented style
$query = $mysqli->query("SELECT * FROM student");

while ($row = $result->fetch_array(MYSQLI_NUM)) {
    // Can use indexed array only
    echo $row[0], $row[1], $row[2];

    // Cannot use associative array
    // echo $row['id'], $row['nim'], $row['name'];
}
```

```
// Object-oriented style
$query = $mysqli->query("SELECT * FROM student");

// Shorter: $result->fetch_assoc()
while ($row = $result->fetch_array(MYSQLI_ASSOC)) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Can use associative array only
    echo $row['id'], $row['nim'], $row['name'];
}
```

```
// Object-oriented style
$query = $mysqli->query("SELECT * FROM student");

while ($row = $result->fetch_array(MYSQLI_BOTH)) {
    // Can use indexed array
    echo $row[0], $row[1], $row[2];

    // Or associative array
    echo $row['id'], $row['nim'], $row['name'];
}

}
```

```
// Object-oriented style
$query = $mysqli->query("SELECT * FROM student");

while ($row = $result->fetch_object()) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Nor associative array
    // echo $row['id'], $row['nim'], $row['name'];

    // Must use object style
    echo $row->id, $row->nim, $row->name;
}
```

```
// Procedural style
$query = mysqli_query($mysqli, "SELECT * FROM student");

while ($row = mysqli_fetch_array($result), MYSQLI_NUM) {
    // Can use indexed array only
    echo $row[0], $row[1], $row[2];

    // Cannot use associative array
    // echo $row['id'], $row['nim'], $row['name'];
}

}
```

```
// Procedural style
$query = mysqli_query($mysqli, "SELECT * FROM student");

// Shorter: mysqli_fetch_assoc($result)
while ($row = mysqli_fetch_array($result), MYSQLI_ASSOC) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Can use associative array only
    echo $row['id'], $row['nim'], $row['name'];
}
```

```
// Procedural style
$query = mysqli_query($mysqli, "SELECT * FROM student");

while ($row = mysqli_fetch_array($result), MYSQLI_BOTH) {
    // Can use indexed array
    echo $row[0], $row[1], $row[2];

    // Or associative array
    echo $row['id'], $row['nim'], $row['name'];
}

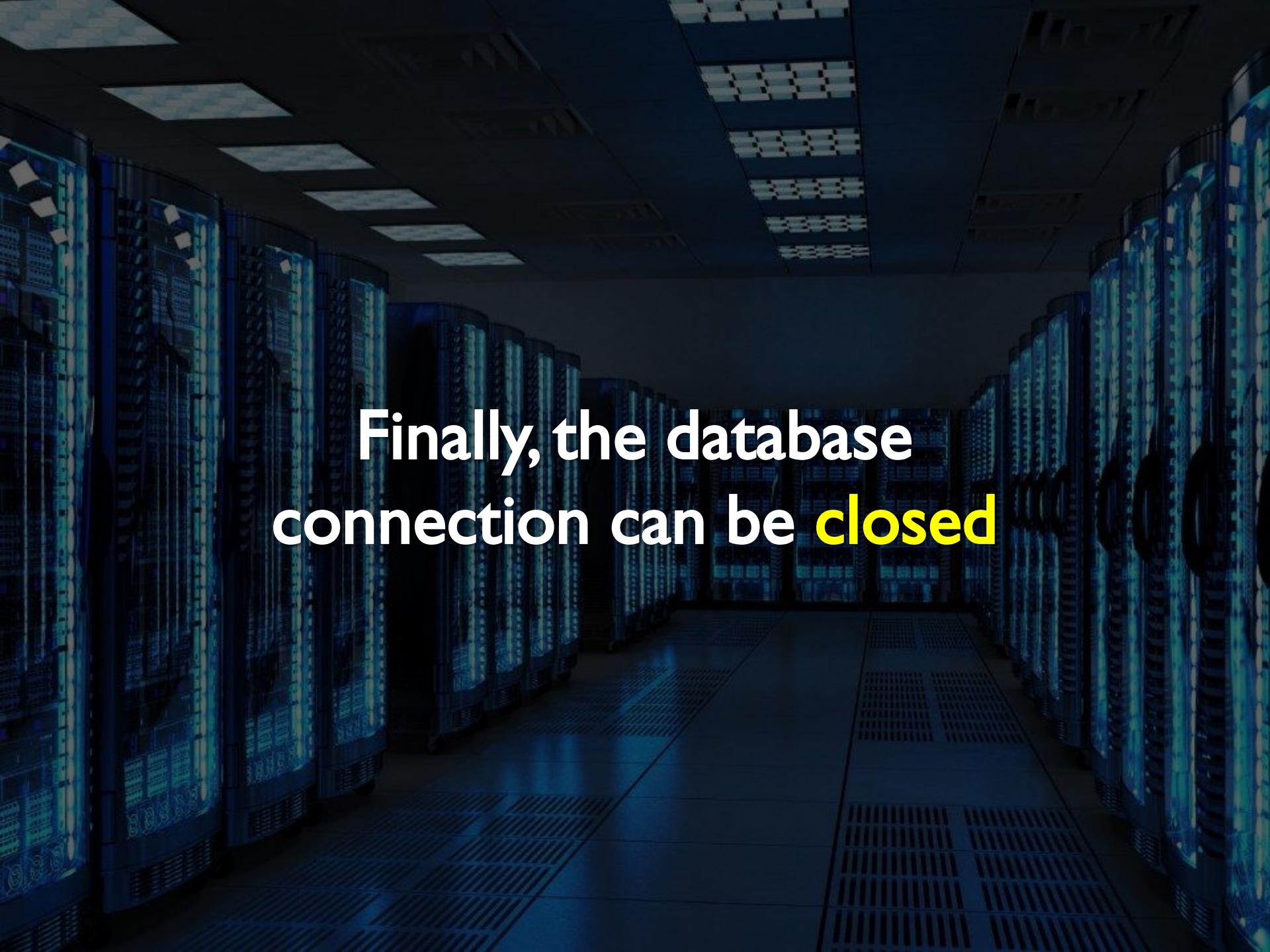

```

```
// Procedural style
$query = mysqli_query($mysqli, "SELECT * FROM student");

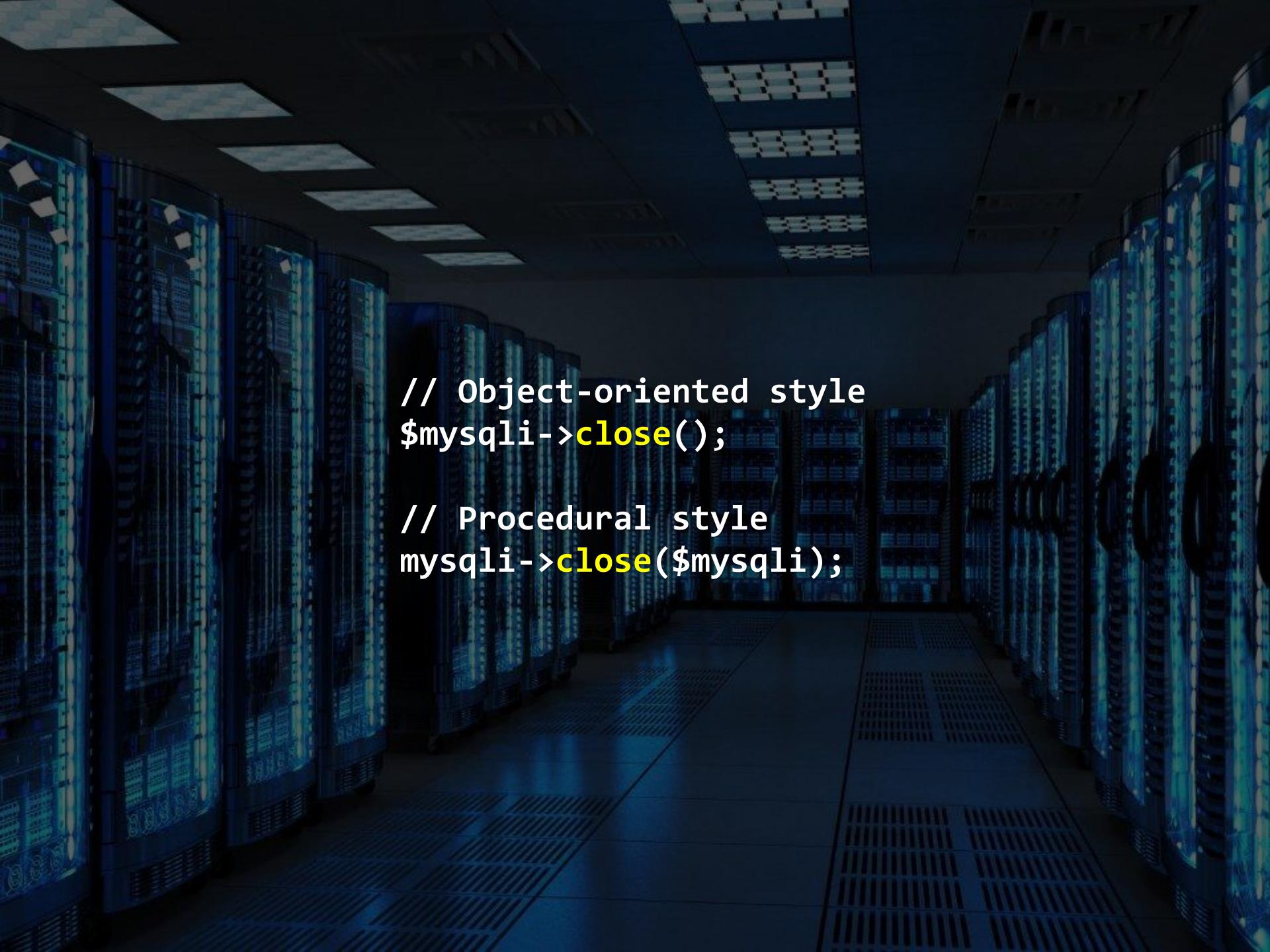
while ($row = mysqli_fetch_object($result)) {
    // Cannot use indexed array
    // echo $row[0], $row[1], $row[2];

    // Nor associative array
    // echo $row['id'], $row['nim'], $row['name'];

    // Must use object style
    echo $row->id, $row->nim, $row->name;
}
```

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit板 patterns. The floor has a metal grate. The ceiling has several rectangular light fixtures.

Finally, the database
connection can be **closed**



```
// Object-oriented style  
$mysqli->close();
```

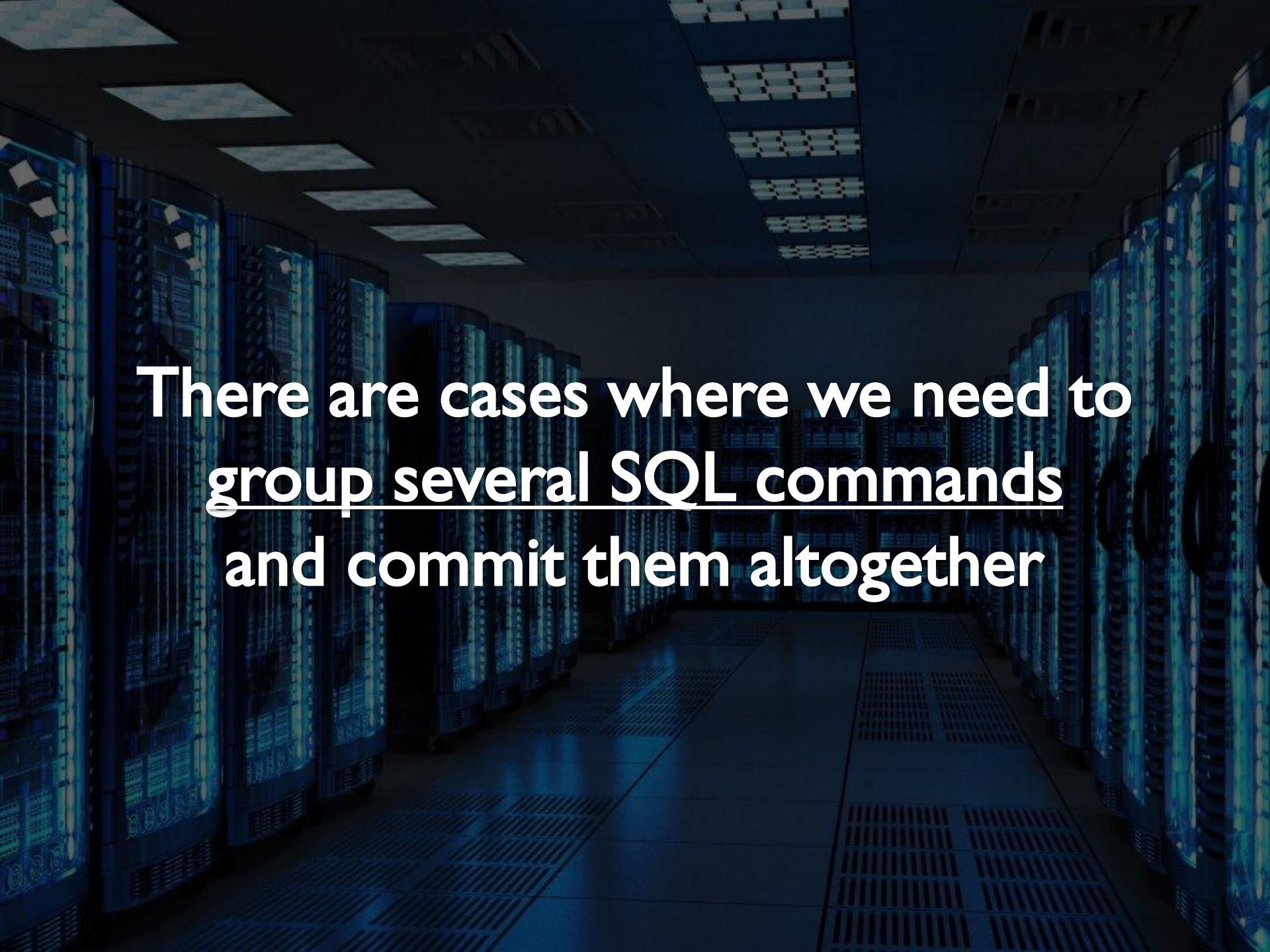
```
// Procedural style  
mysqli->close($mysqli);
```

A perspective view of a server room. On both sides, there are long rows of server racks, each illuminated from within with a bright blue light. The floor is made of dark tiles with metal grilles. The ceiling is white with several rectangular recessed lights. The overall atmosphere is cool and futuristic.

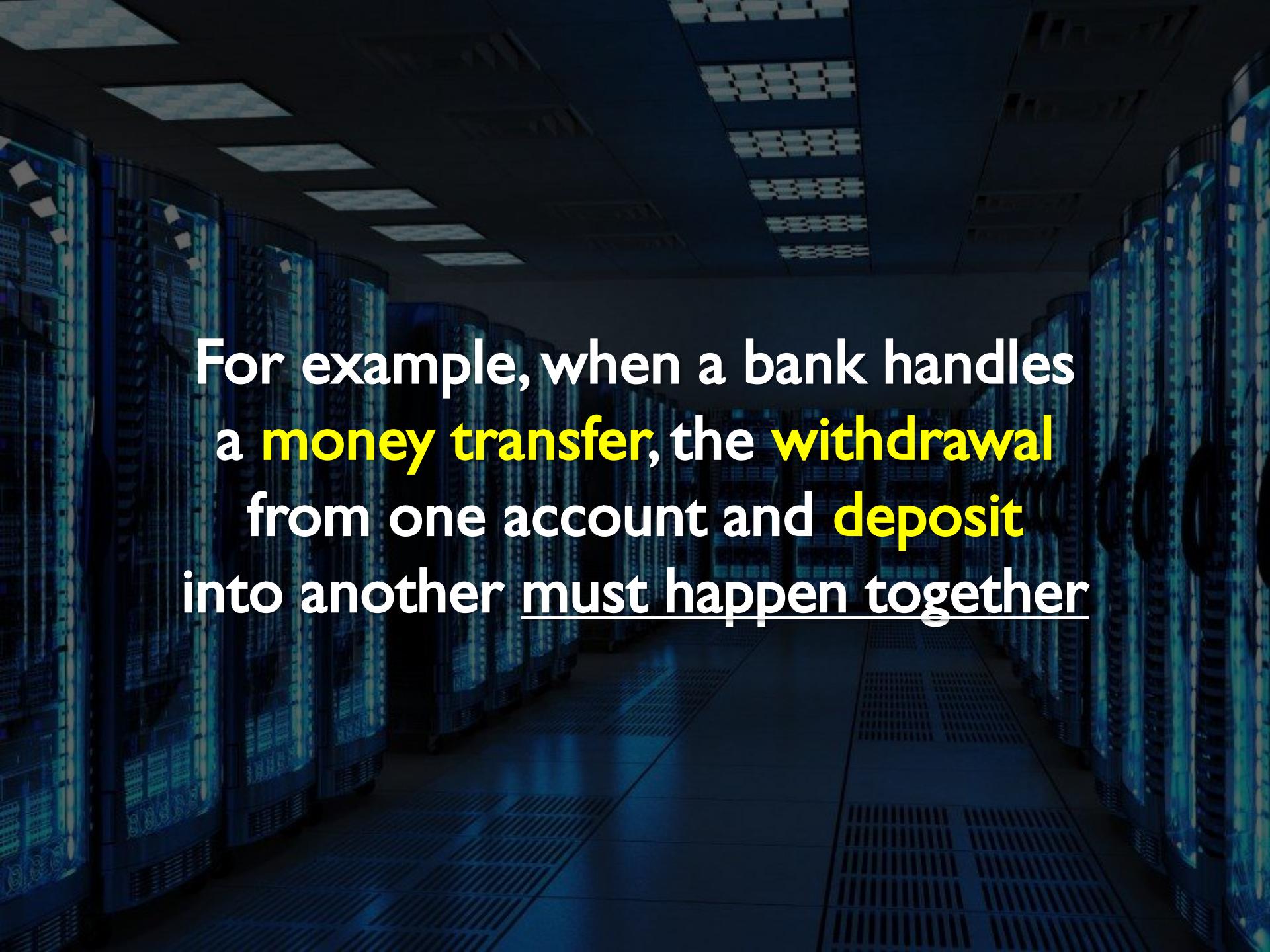
Transactions

The background of the slide is a photograph of a server room. The room is dimly lit, with the primary light source being the blue and white glowing lights from the server racks themselves. These lights create a pattern of vertical and horizontal lines across the dark space. The racks are arranged in two main rows that recede into the distance, creating a sense of depth. The floor is made of polished tiles, reflecting some of the ambient light. The overall atmosphere is one of a high-tech, industrial environment.

By default, each **SQL** command
is committed immediately
to the database

The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks, their front panels illuminated with a bright blue light that casts a glow on the floor and the ceiling. The ceiling features several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial data center.

There are cases where we need to
group several SQL commands
and commit them altogether



For example, when a bank handles a **money transfer**, the **withdrawal** from one account and **deposit** into another **must happen together**

The background of the slide shows a dimly lit server room. On either side, there are long rows of server racks. The screens of the servers are illuminated with various shades of blue and white, creating a pattern of light and shadow. The floor is made of polished tiles, reflecting the ambient light. The ceiling has several rectangular recessed lights. The overall atmosphere is one of a high-tech, industrial environment.

In a transaction, all SQL commands
must be **executed successfully**,
or else, it should be **rolled back**

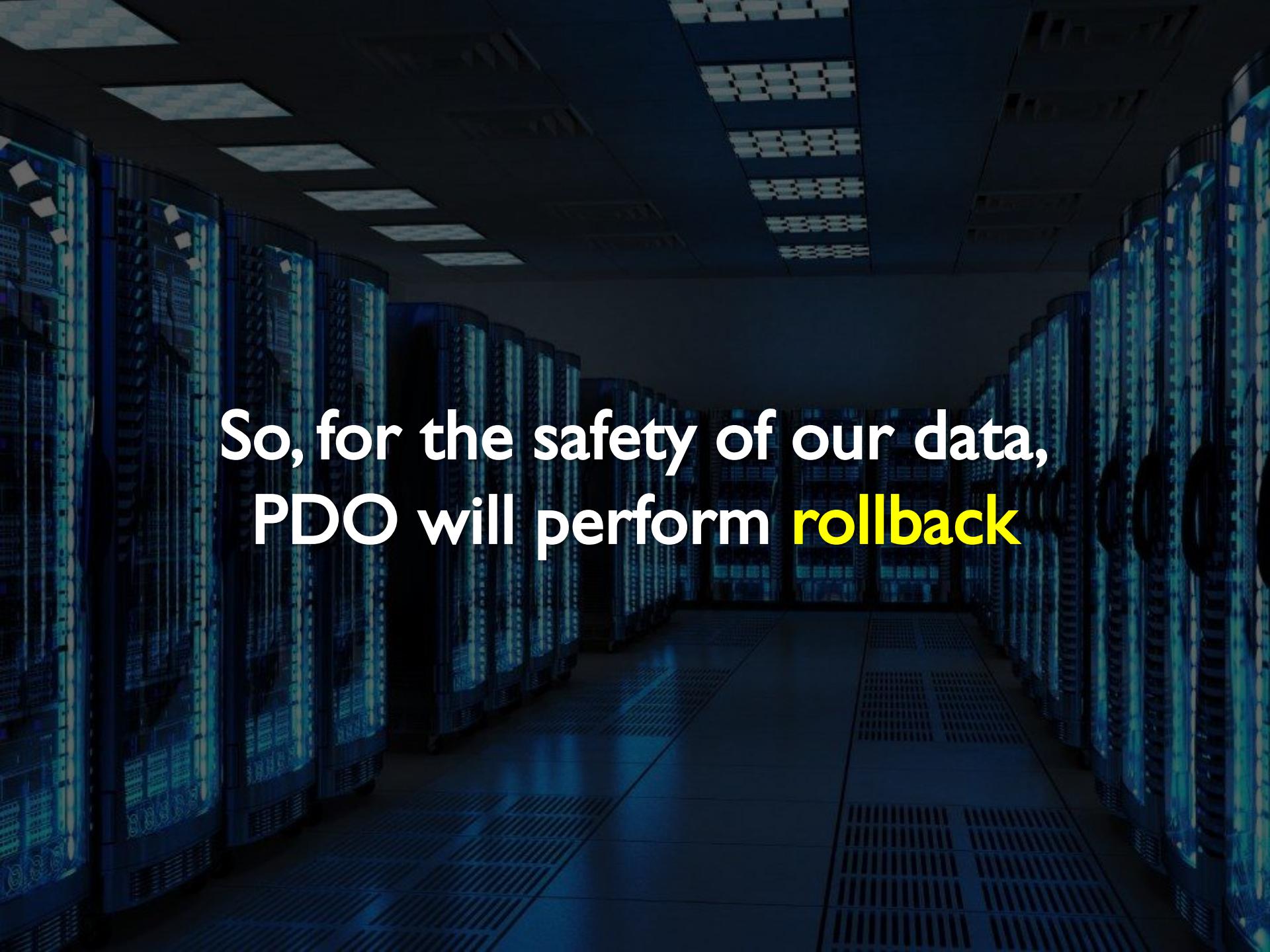
```
try {
    $db->setAttribute(PDO::ATTR_ERRMODE,
        PDO::ERRMODE_EXCEPTION);
    $db->beginTransaction();

    $db->exec("UPDATE account SET ...");
    $db->exec("INSERT INTO account ...");
    $db->exec("DELETE FROM transaction WHERE ...");

    $db->commit();
} catch (Exception $error) {
    $db->rollback();
    echo "Transaction error: ", $error->getMessage();
}
```

A dark server room with rows of server racks. The racks have glowing blue screens displaying binary code. The floor is made of metal grates. The ceiling has several rectangular light fixtures.

If we don't explicitly **commit** our transaction, PDO will assume that something has gone wrong

A dark server room with rows of server racks. The racks are illuminated from within, showing blue lights and circuit boards. The floor has a metal grate. The ceiling has several rectangular light fixtures.

So, for the safety of our data,
PDO will perform **rollback**

A dark, atmospheric photograph of a server room. Rows of server racks are visible, their front panels illuminated with a blue light, showing various ports and status indicators. The floor is made of dark tiles, and the ceiling features several rectangular recessed lights. The overall mood is mysterious and tech-oriented.

`$class->close();`