

A modern conference room with large windows and a long table. The room is empty, with several office chairs arranged around the table. The view outside the windows shows a cityscape. The image has a warm, orange-toned overlay.

Advanced SQL Topics



Types of MySQL Variables – Local Variables

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```

Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```



Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```

≠



Types of MySQL Variables – Local Variables

scope

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```

≠



Types of MySQL Variables – Local Variables

scope =

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```

≠



Types of MySQL Variables – Local Variables

scope = **visibility**

the region of a computer program where a phenomenon, such as a variable, is considered valid

```
BEGIN  
DECLARE v_avg_salary DECIMAL(10,2);  
SELECT  
    AVG(s.salary)  
INTO v_avg_salary FROM  
    employees e
```

≠

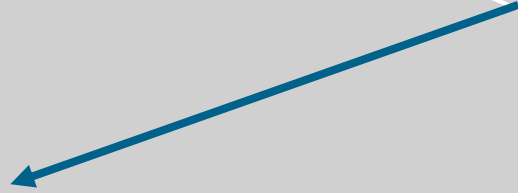


Types of MySQL Variables – Local Variables

MySQL Variables

Types of MySQL Variables – Local Variables

MySQL Variables



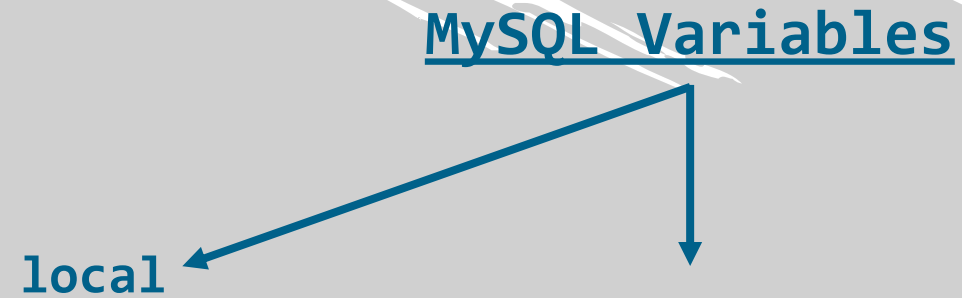
Types of MySQL Variables - Local Variables

MySQL Variables

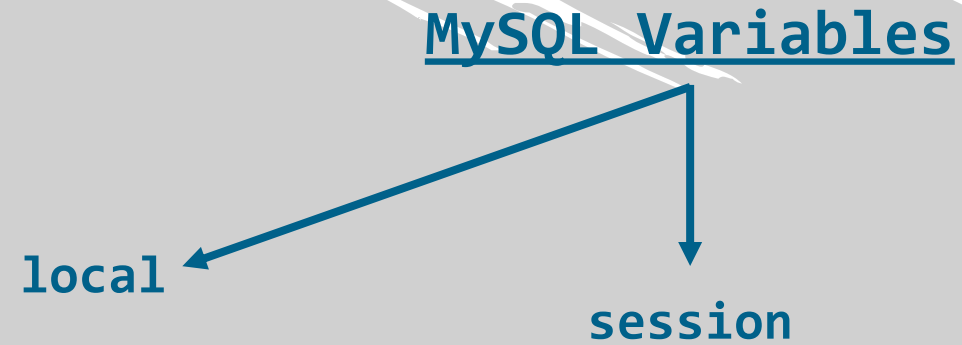
local



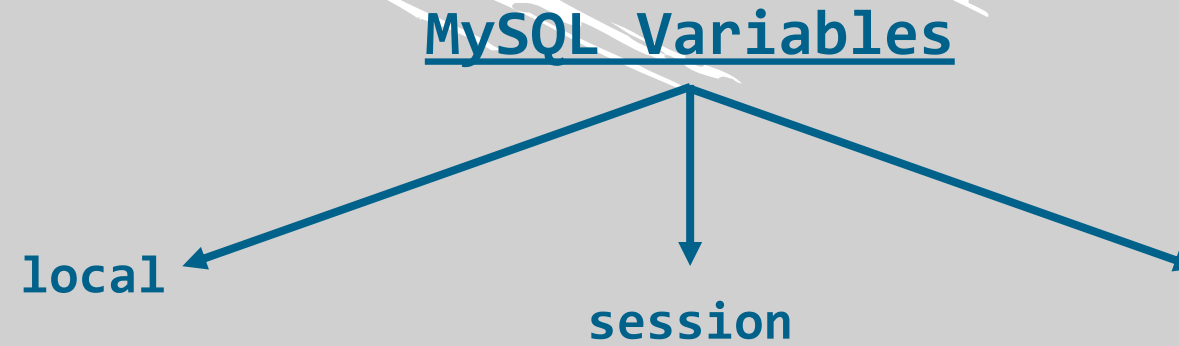
Types of MySQL Variables - Local Variables



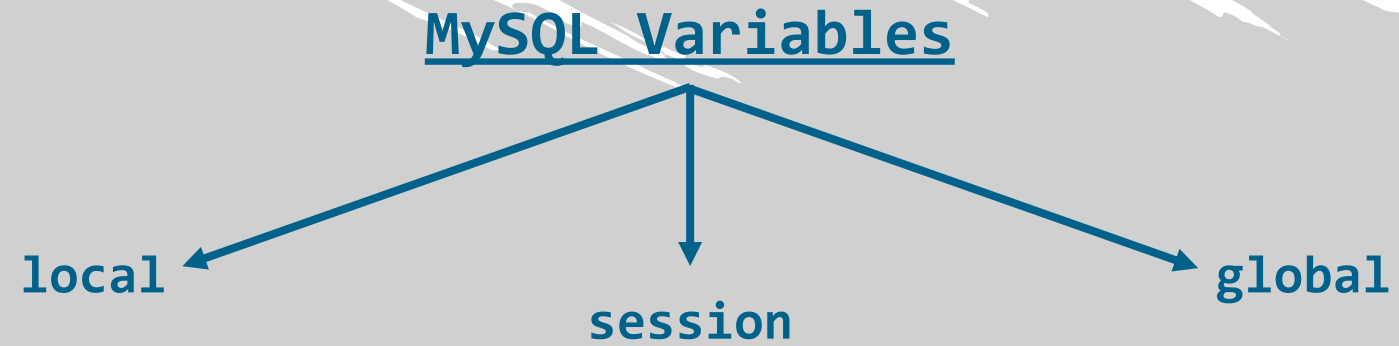
Types of MySQL Variables - Local Variables



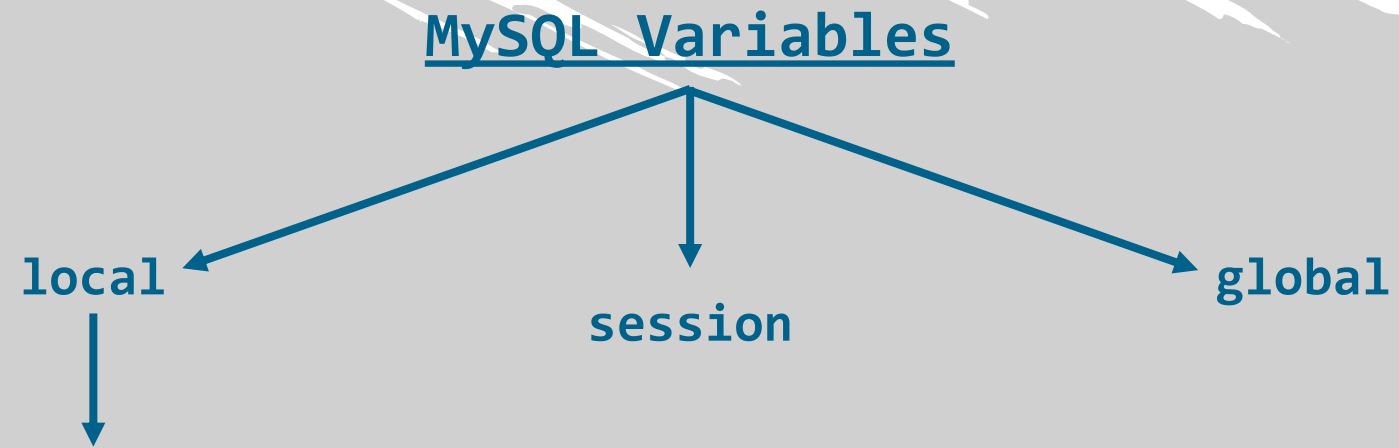
Types of MySQL Variables - Local Variables



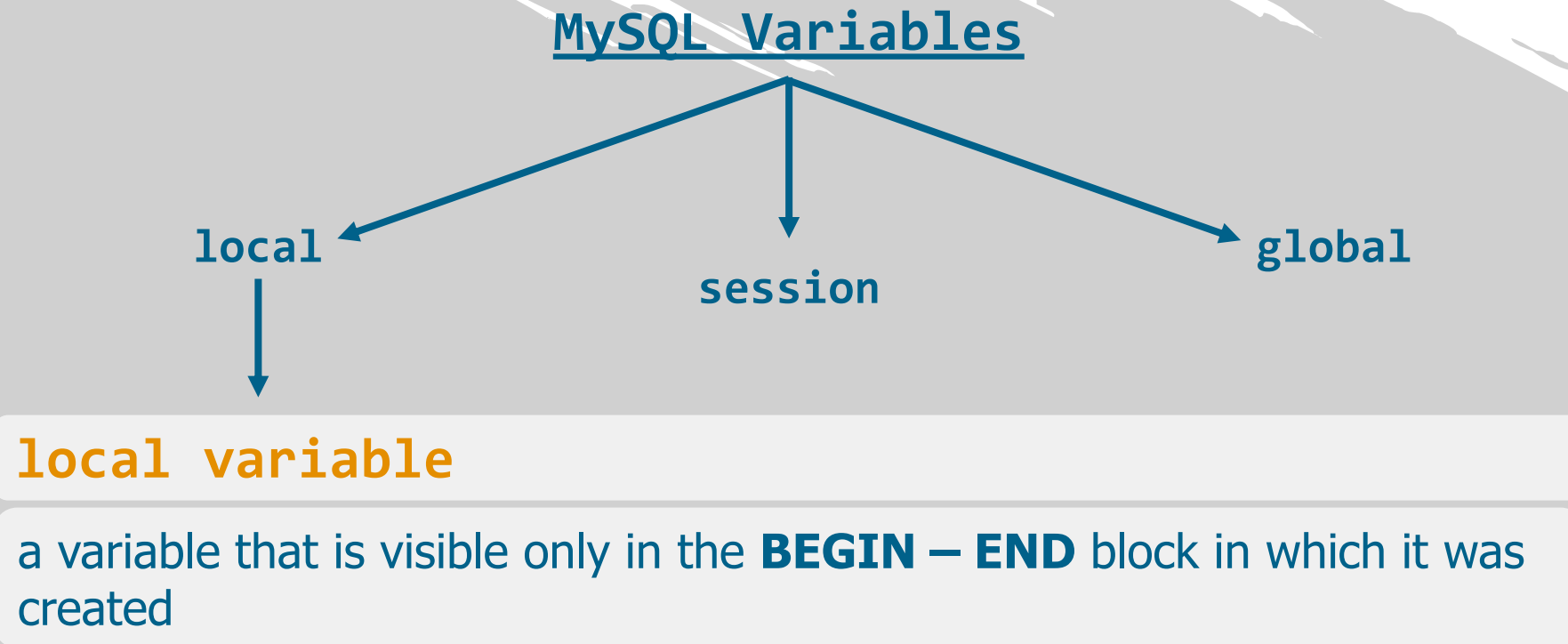
Types of MySQL Variables - Local Variables



Types of MySQL Variables - Local Variables



Types of MySQL Variables – Local Variables



Types of MySQL Variables - Local Variables

- DECLARE is a keyword that can be used when creating *local* variables only
- *v_avg_salary* is visible only in the BEGIN - END block

Session Variables

Session Variables

session

a series of information exchange interactions, or a dialogue, between a computer and a user

Session Variables

session

a series of information exchange interactions, or a dialogue, between a computer and a user

- e.g. a dialogue between the MySQL server and a client application like MySQL Workbench

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1



set up a
connection



Step 1

File Edit View Database Tools Scripting Help

Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

MySQL Connect

Local instance MyS

root

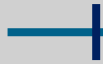
localhost:3306

Filter connections

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1



set up a
connection



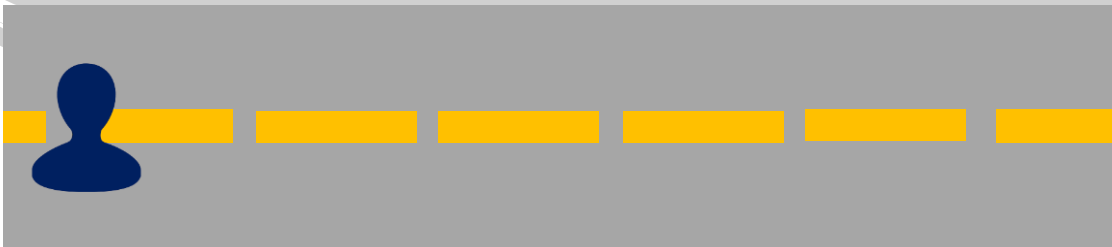
Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1



set up a
connection




Step 2



Welcome to MySQL Workbench

MySQL Workbench is the
create and browse your
design and run SQL queries

[Browse Documentation >](#)

MySQL Connections  


Local instance MySQL57

 root
 localhost:3306

 Filter connections

Connect to MySQL Server

Please enter password for the following service:



Service: Mysql@localhost:3306

User: root

Password:

☐ Save password in vault

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1



set up a
connection



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2



set up a
connection

establish a
connection



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

set up a
connection

establish a
connection



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

set up a
connection



Step 2

establish a
connection



Step 3

the Workbench
interface will
open immediately

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

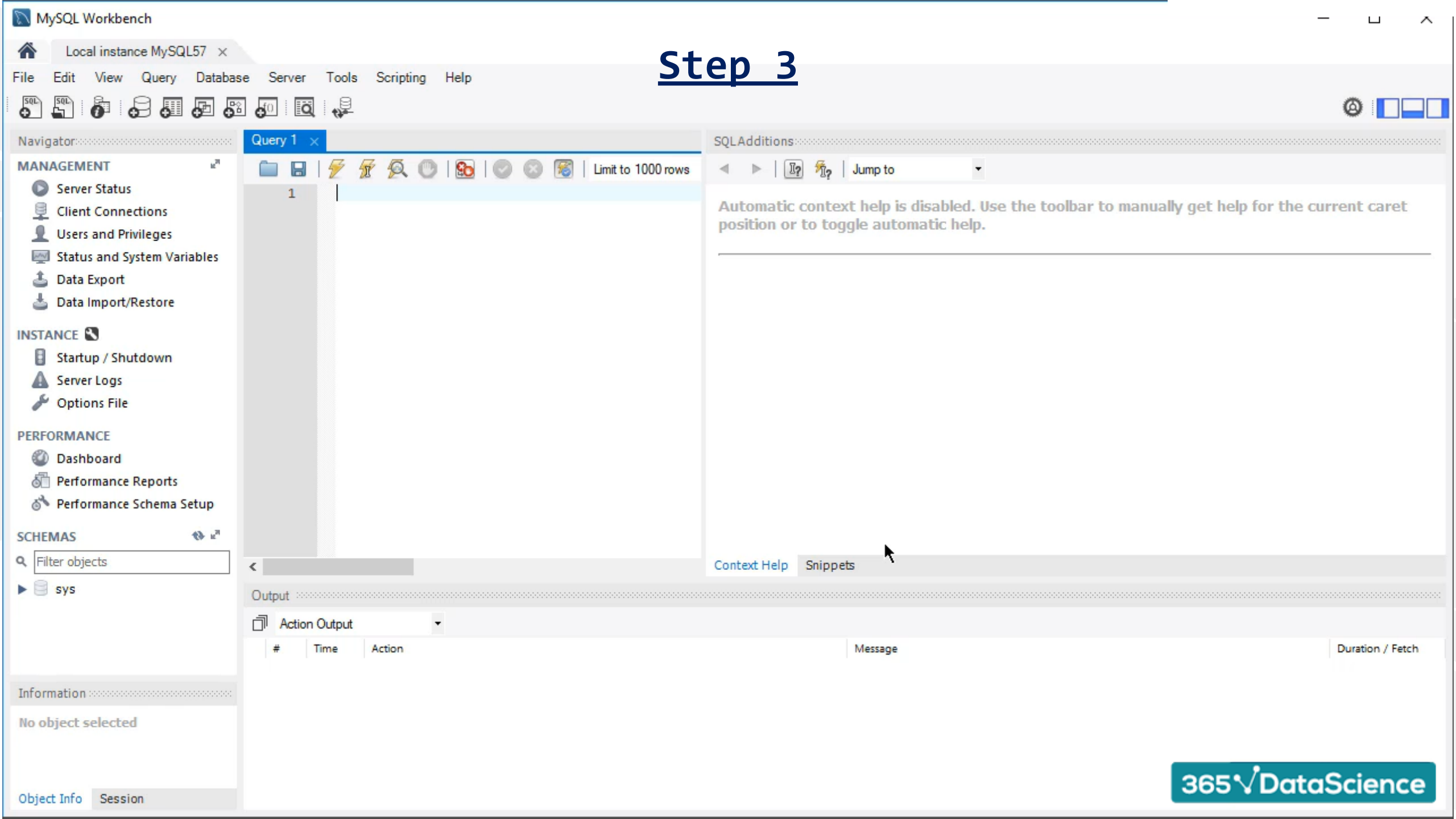
Step 3

set up a
connection

establish a
connection

the Workbench
interface will
open immediately





Step 3

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

sys

Information

No object selected

Object Info Session

Query 1

```
1
```

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Output

Action Output

#	Time	Action	Message	Duration / Fetch
---	------	--------	---------	------------------

Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

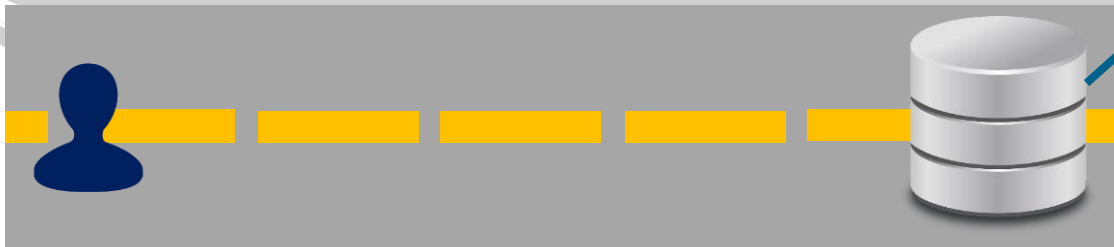
Step 2

Step 3

set up a
connection

establish a
connection

the Workbench
interface will
open immediately



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

set up a
connection

establish a
connection

the Workbench
interface will
open immediately



Session Variables

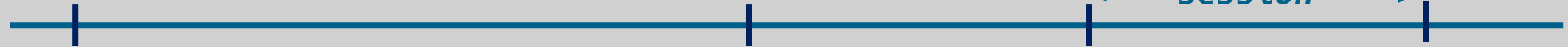
- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

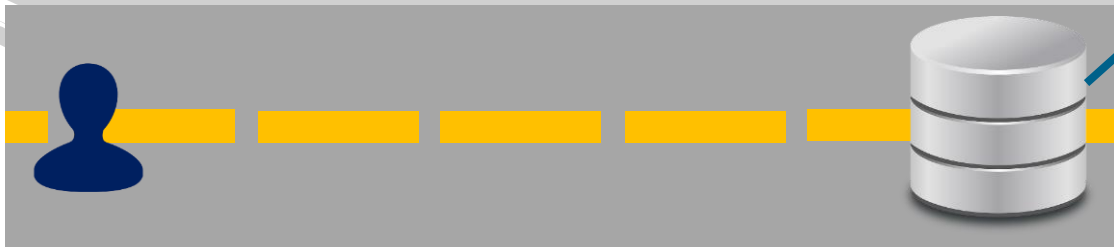
Step 4



set up a
connection

establish a
connection

the Workbench
interface will
open immediately



Session Variables

- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

Step 4



set up a
connection

establish a
connection

the Workbench
interface will
open immediately

end a
connection



Session Variables

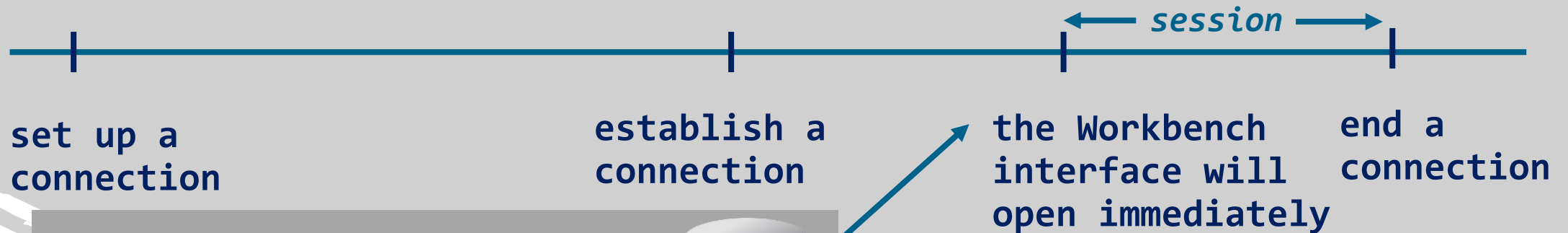
- a *session* begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

Step 4



Session Variables

- a session begins at a certain point in time and terminates at another, later point

Step 1

Step 2

Step 3

Step 4

set up a
connection

establish a
connection

the Workbench
interface will
open immediately

end a
connection



Session Variables

- there are certain SQL objects that are valid for a specific session only

Session Variables

- there are certain SQL objects that are valid for a specific session only

Step 1

set up a
connection



Step 2

establish a
connection



Step 3

the Workbench
interface will
open immediately

← session →

Session Variables

- there are certain SQL objects that are valid for a specific session only

emp_no	from_date	to_date
10001	1986-06-26	9999-01-01
10002	1996-08-03	9999-01-01
10003	1995-12-03	9999-01-01
10004	1986-12-01	9999-01-01
10005	1989-09-12	9999-01-01

Step 1

Step 2

set up a
connection

establish a
connection

the Workbench
interface will
open immediately



Session Variables

- there are certain SQL objects that are valid for a specific session only

emp_no	from_date	to_date
10001	1986-06-26	9999-01-01
10002	1996-08-03	9999-01-01
10003	1995-12-03	9999-01-01
10004	1986-12-01	9999-01-01
10005	1989-09-12	9999-01-01

Step 1

set up a
connection



Step 2

establish a
connection



the Workbench
interface will
open immediately

session

end a
connection

Session Variables

- there are certain SQL objects that are valid for a specific session only

Step 1

set up a
connection



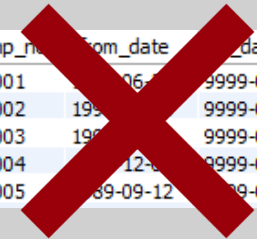
Step 2

establish a
connection



the Workbench
interface will
open immediately

end a
connection

A large red 'X' is drawn over the table, indicating that the data shown is not session-specific or is invalid for a specific session.

emp_id	hire_date	expiry_date
10001	1980-06-06	9999-01-01
10002	1980-06-06	9999-01-01
10003	1980-06-06	9999-01-01
10004	1980-12-06	9999-01-01
10005	1989-09-12	9999-01-01

Session Variables

session variable

a variable that exists only for the session in which you are operating

Session Variables

session variable

a variable that exists only for the session in which you are operating

- it is *defined* on our server, and it lives there

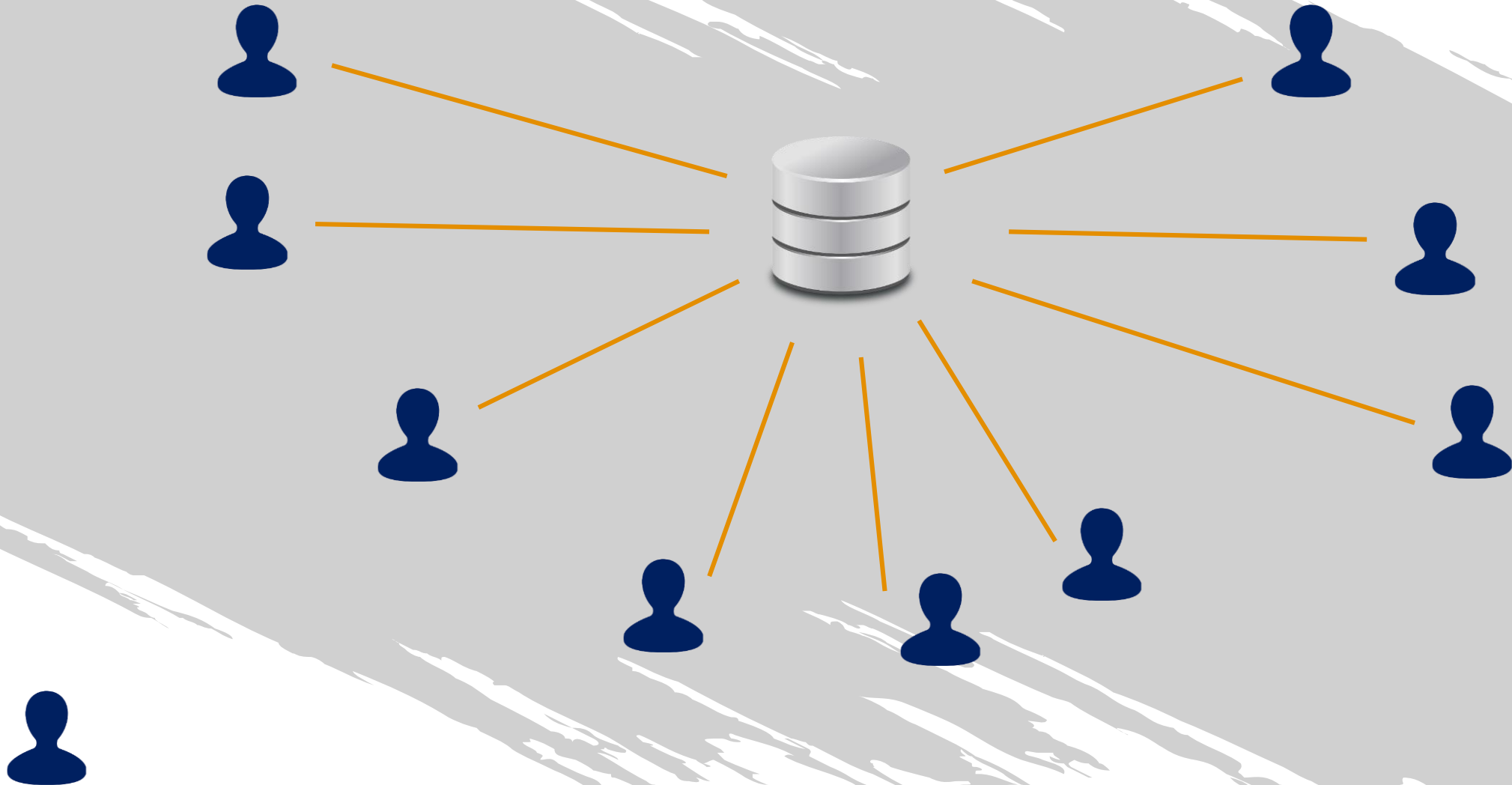
Session Variables

session variable

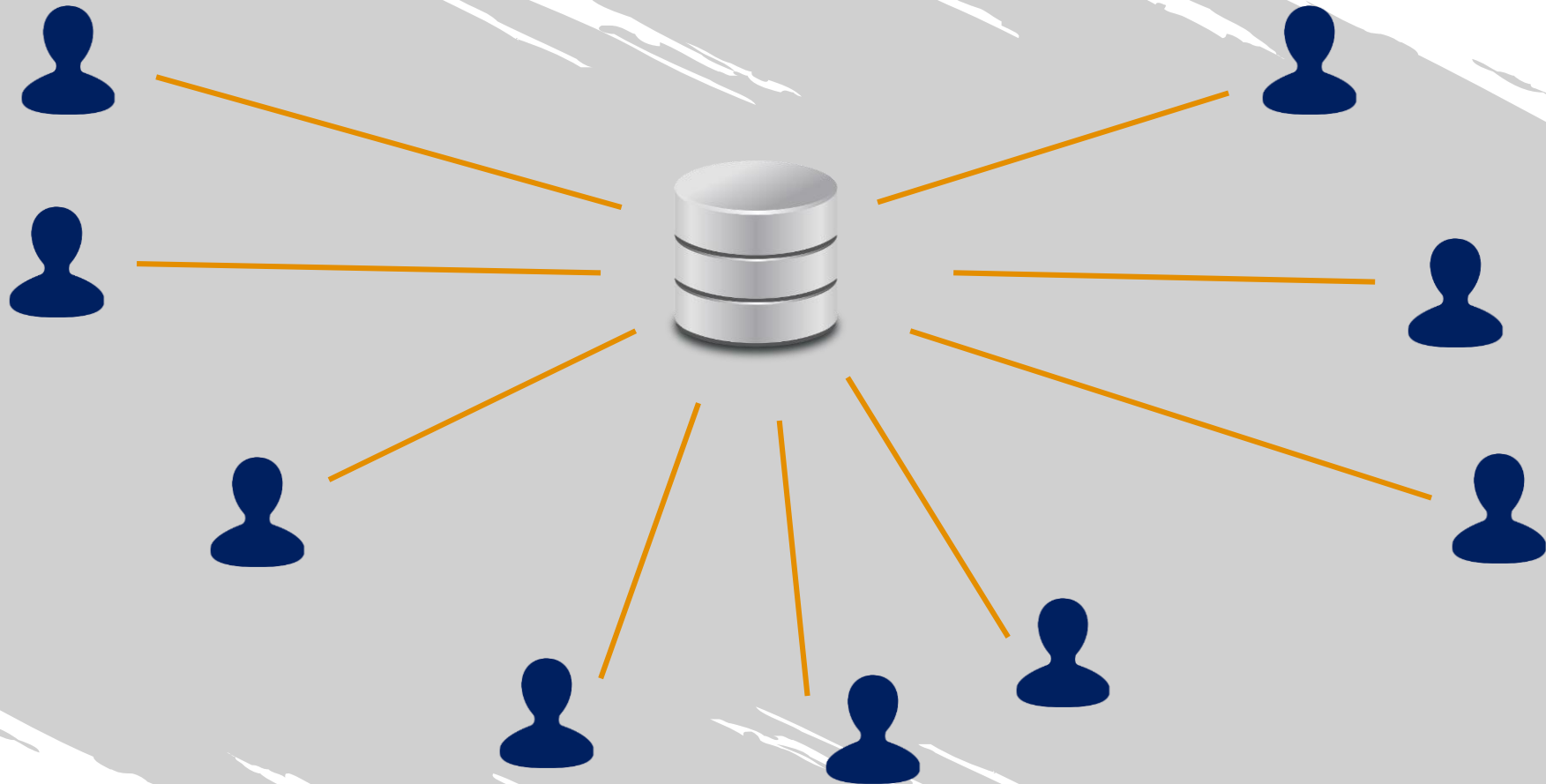
a variable that exists only for the session in which you are operating


- it is *defined* on our server, and it lives there
- it is *visible* to the connection being used only

Session Variables

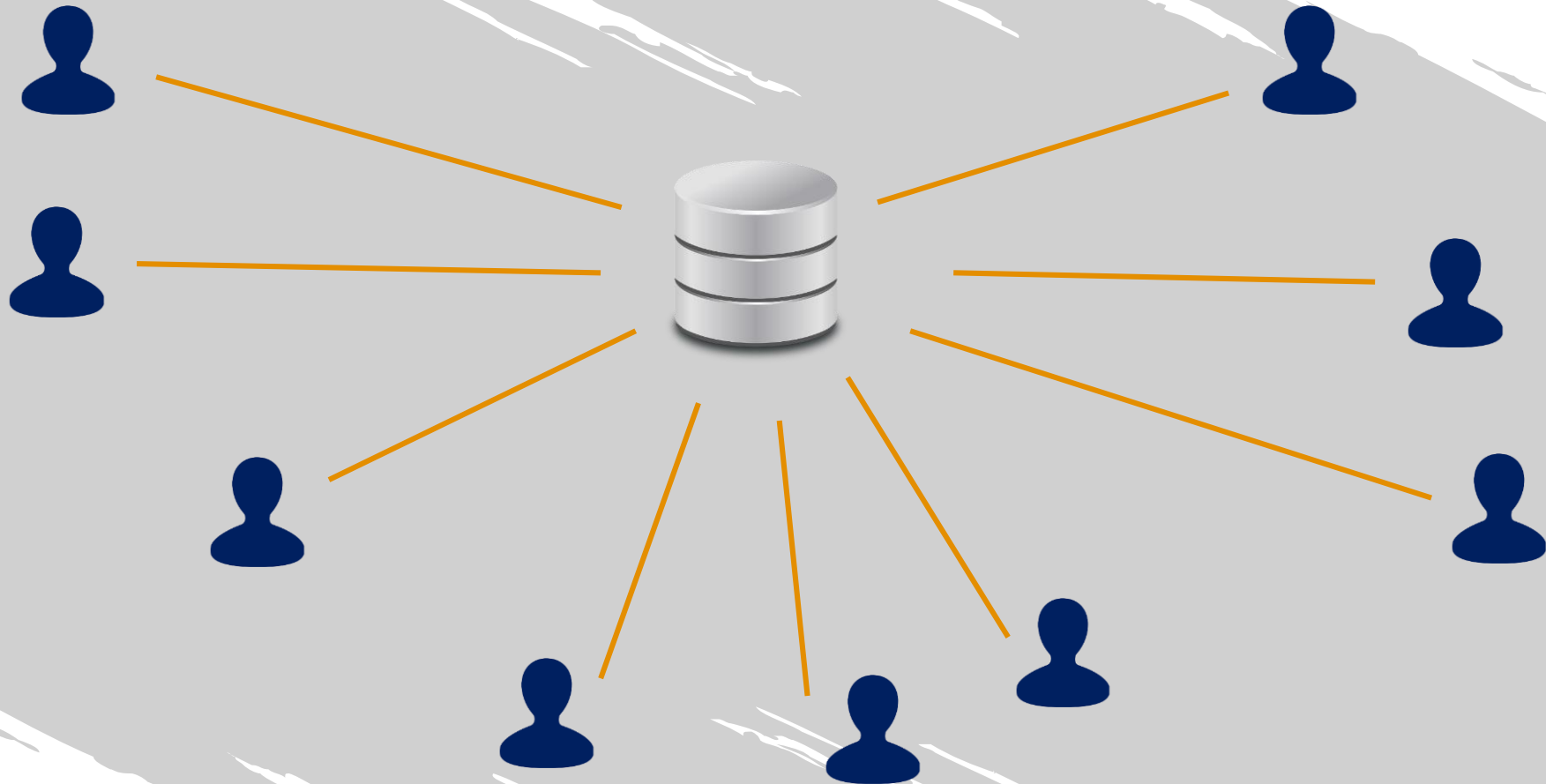


Session Variables



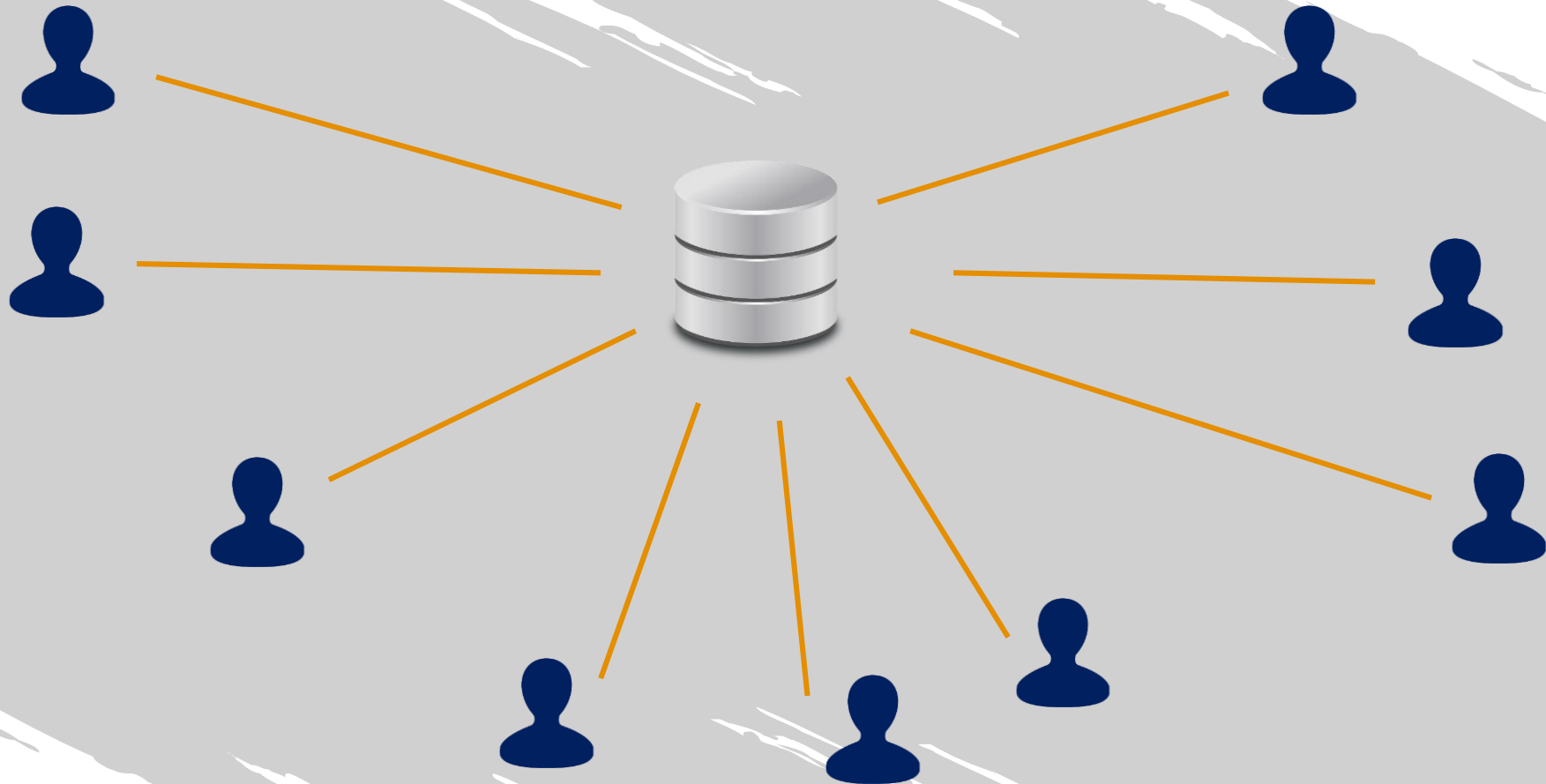
 = 100

Session Variables



 = 100 → connections = 100

Session Variables



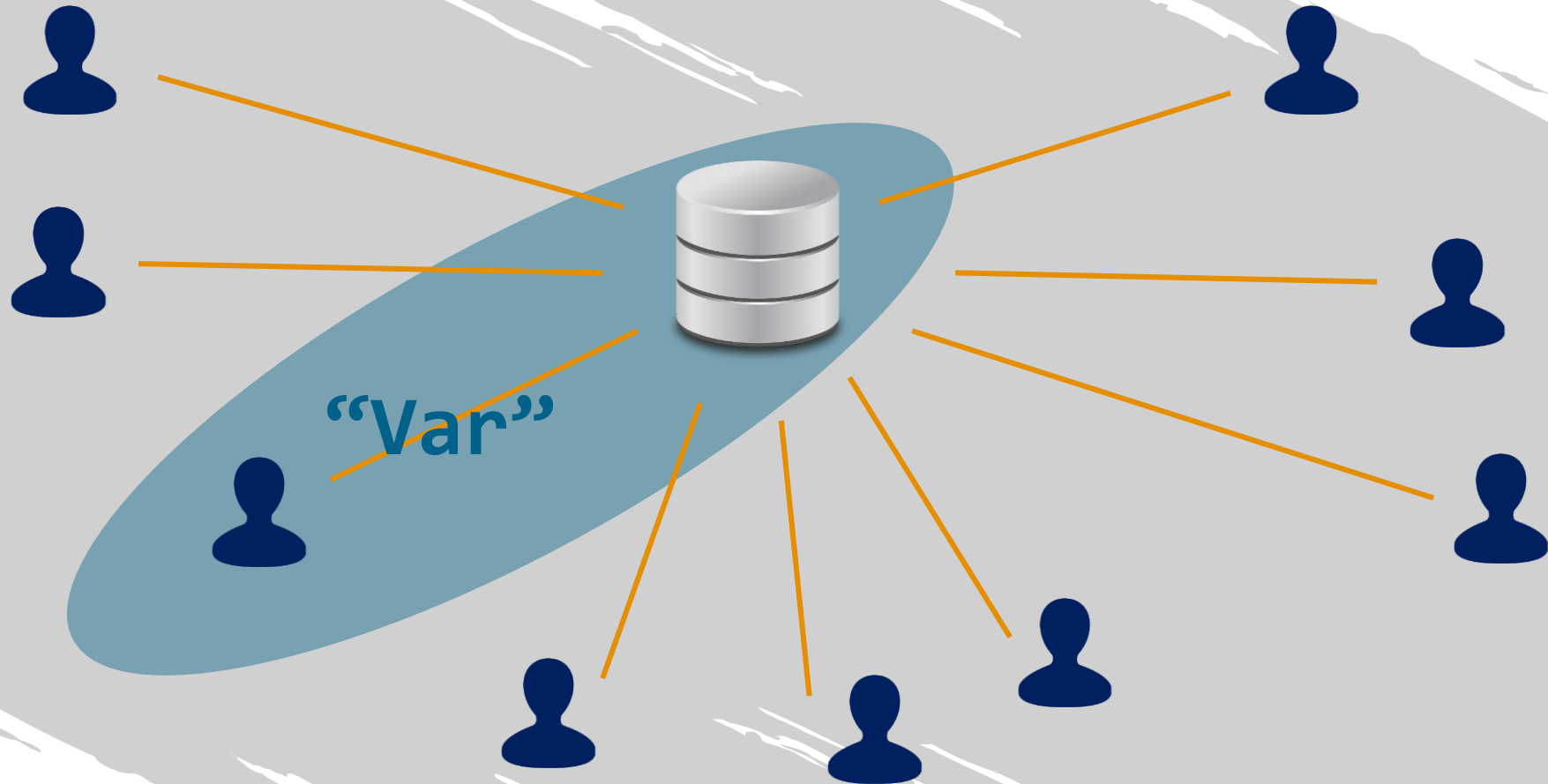
 = 100 → connections = 100 → sessions = 100

Session Variables



 = 100 → connections = 100 → sessions = 100

Session Variables



 = 100 → connections = 100 → sessions = 100

Session Variables



SQL

```
SET @var_name = value;
```

A modern conference room with large windows and a blue tint. The room features a long table, several chairs, and a laptop on the table. The text "Global Variables" is centered in the middle of the image.

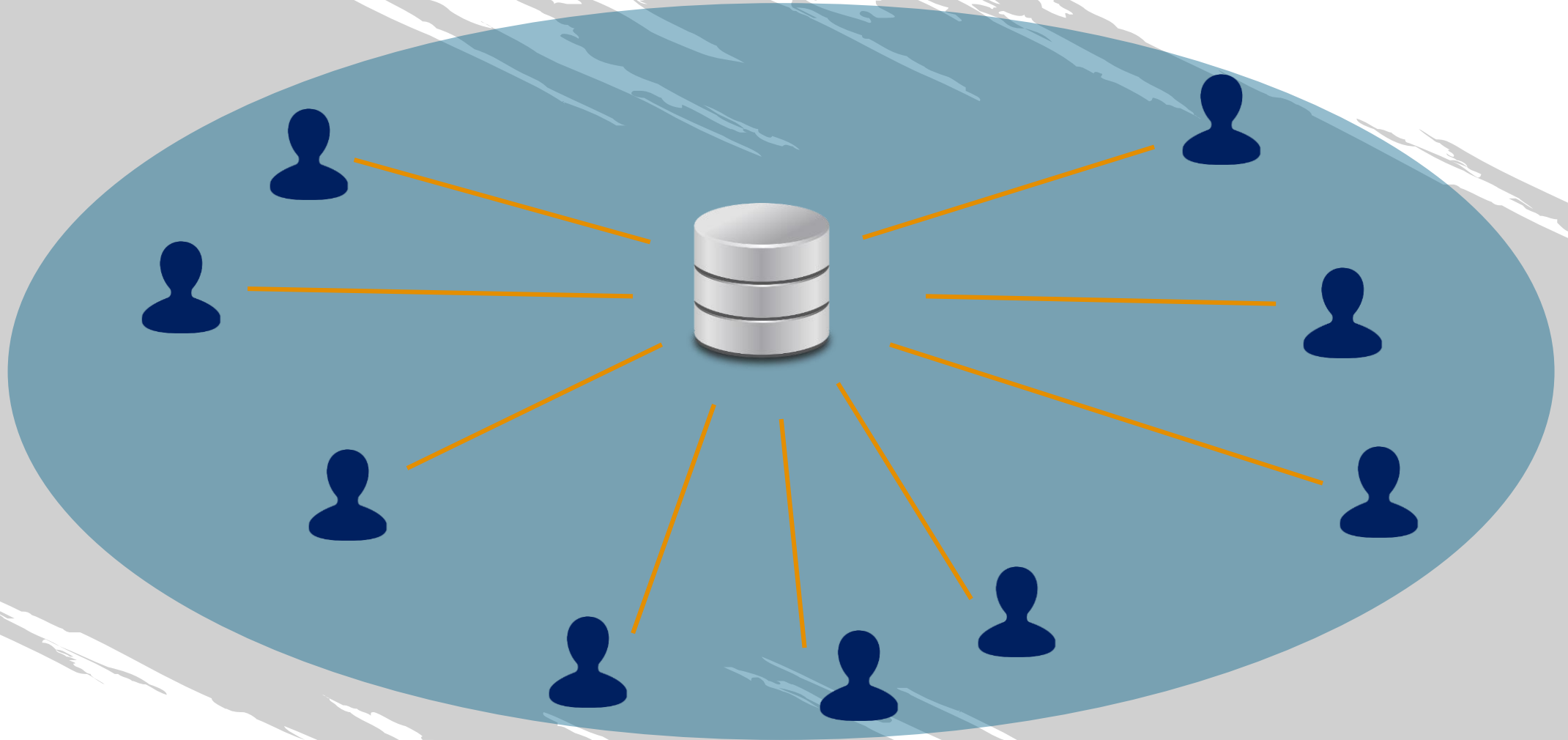
Global Variables

Global Variables

global variables

global variables apply to all connections related to a specific server

Global Variables



Global Variables



Global Variables



SQL

```
SET GLOBAL var_name = value;
```

Global Variables



SQL

```
SET GLOBAL var_name = value;
```



SQL

```
SET @@global.var_name = value;
```

Global Variables

- you cannot set just any variable as global

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables
-

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables
-

```
.max_connections()
```

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables
-

`.max_connections()`

`.max_join_size()`

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables
-

`.max_connections()`

- indicates the *maximum number of connections* to a server that can be established at a certain point in time

`.max_join_size()`

Global Variables

- you cannot set just any variable as global
 - a specific group of *pre-defined variables* in MySQL is suitable for this job. They are called system variables
-

`.max_connections()`

- indicates the *maximum number of connections* to a server that can be established at a certain point in time

`.max_join_size()`

- sets the *maximum memory space* allocated for the joins created by a certain connection

A modern conference room with large windows and a blue tint. The room features a long wooden table, several black office chairs, and a laptop on the table. The windows offer a view of a cityscape. The text "User-Defined vs System Variables" is overlaid in the center.

User-Defined vs System Variables

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined

system

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined

variables that can be *set by the user manually*

system

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

user-defined

variables that can be *set by the user manually*

system

variables that are *pre-defined* on our system – the MySQL server

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓		
system			

- local variables can be user-defined only

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓		
system	✗		

- local variables can be user-defined only

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓		
system	✗		✓

- only system variables can be set as global

User-Defined vs System Variables

- variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓		✗
system	✗		✓

- only system variables can be set as global

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓		✗
system	✗		✓

- both user-defined and system variables can be set as session variables

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓	✓	✗
system	✗	✓	✓

- both user-defined and system variables can be set as session variables

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓	✓	✗
system	✗	✓	✓

- both user-defined and system variables can be set as session variables
there are limitations to this rule!

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓	✓*	✗
system	✗	✓*	✓

- both user-defined and system variables can be set as session variables
there are limitations to this rule!

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓	✓*	✗
system	✗	✓*	✓

- some of the system variables can be defined as global only; they cannot be session variables

User-Defined vs System Variables

variables in MySQL can be characterized according to the way they have been created

	local	session	global
user-defined	✓	✓*	✗
system	✗	✓*	✓

- some of the system variables can be defined as global only; they cannot be session variables (e.g. `.max_connections()`)

A modern office interior with large windows and a long conference table. The room is brightly lit, and the view outside the windows shows a cityscape. The text "MySQL Indexes" is overlaid in the center of the image.

MySQL Indexes

MySQL Indexes



MySQL Indexes



MySQL Indexes

- the index of a table functions like the index of a book

MySQL Indexes

- the index of a table functions like the index of a book
 - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index

MySQL Indexes

- the index of a table functions like the index of a book
 - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index
- your datasets will typically contain 100,000+ or even 1,000,000+ records

MySQL Indexes

- the index of a table functions like the index of a book
 - data is taken from a column of the table and is stored in a certain order in a distinct place, called an index
- your datasets will typically contain 100,000+ or even 1,000,000+ records
 - the *larger* a database is, the *slower* the process of finding the record or records you need

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table



SQL

```
CREATE INDEX index_name  
ON table_name (column_1, column_2, ...);
```

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table



SQL

```
CREATE INDEX index_name  
ON table_name (column_1, column_2, ...);
```

the parentheses serve us to indicate the column names on which our search will be based

MySQL Indexes

- we can use an index that will increase the speed of searches related to a table



SQL

```
CREATE INDEX index_name  
ON table_name (column_1, column_2, ...);
```

these must be fields from your data table you
will search frequently

MySQL Indexes

- composite indexes

MySQL Indexes

- composite indexes
 - applied to *multiple* columns, not just a single one

MySQL Indexes

- composite indexes
 - applied to *multiple* columns, not just a single one



SQL

```
CREATE INDEX index_name  
ON table_name (column_1, column_2, ...);
```

MySQL Indexes

- composite indexes
 - applied to *multiple* columns, not just a single one



SQL

```
CREATE INDEX index_name  
ON table_name (column_1, column_2, ...);
```

- carefully pick the columns that would optimize your search!

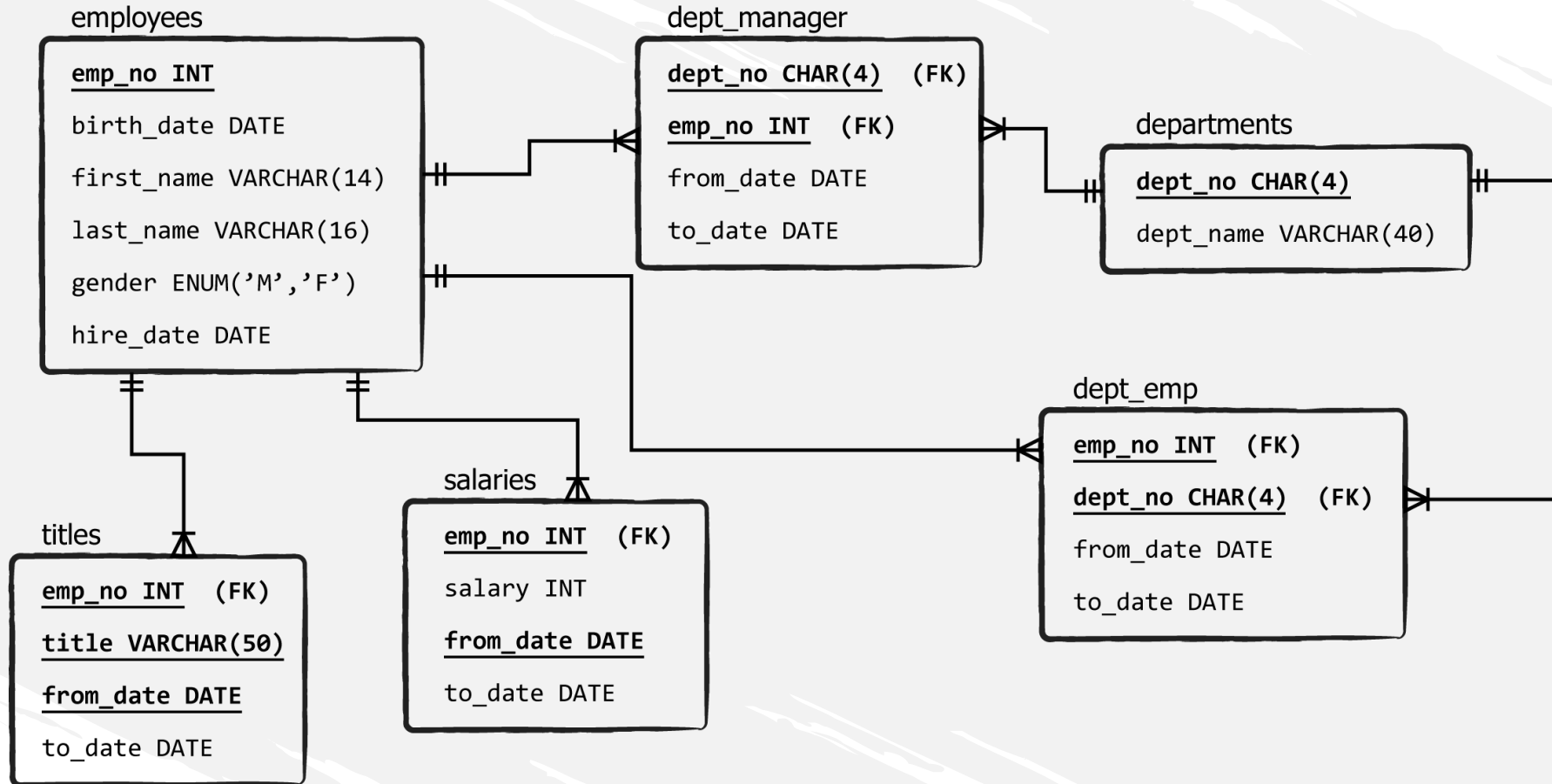
MySQL Indexes

- *primary* and *unique keys* are MySQL indexes

MySQL Indexes

- *primary* and *unique keys* are MySQL indexes
 - they represent columns on which a person would typically base their search

MySQL Indexes



MySQL Indexes

- SQL specialists are always aiming for a good balance between the *improvement of speed search* and the *resources used for its execution*

MySQL Indexes

- SQL specialists are always aiming for a good balance between the *improvement of speed search* and the resources used for its execution

<u>small datasets</u>	the costs of having an index might be higher than the benefits

MySQL Indexes

- SQL specialists are always aiming for a good balance between the improvement of speed search and the *resources used for its execution*

<u>small datasets</u>	the costs of having an index might be higher than the benefits
<u>large datasets</u>	a well-optimized index can make a <i>positive</i> impact on the search process

A modern conference room with large windows and a long table. The room is empty, with several chairs arranged around the table. The view outside the windows shows a cityscape. The image has a blue tint and a stylized, torn-paper-like border.

The CASE Statement

Condition #1

Condition #1



Condition #1



Condition #1



Output #1

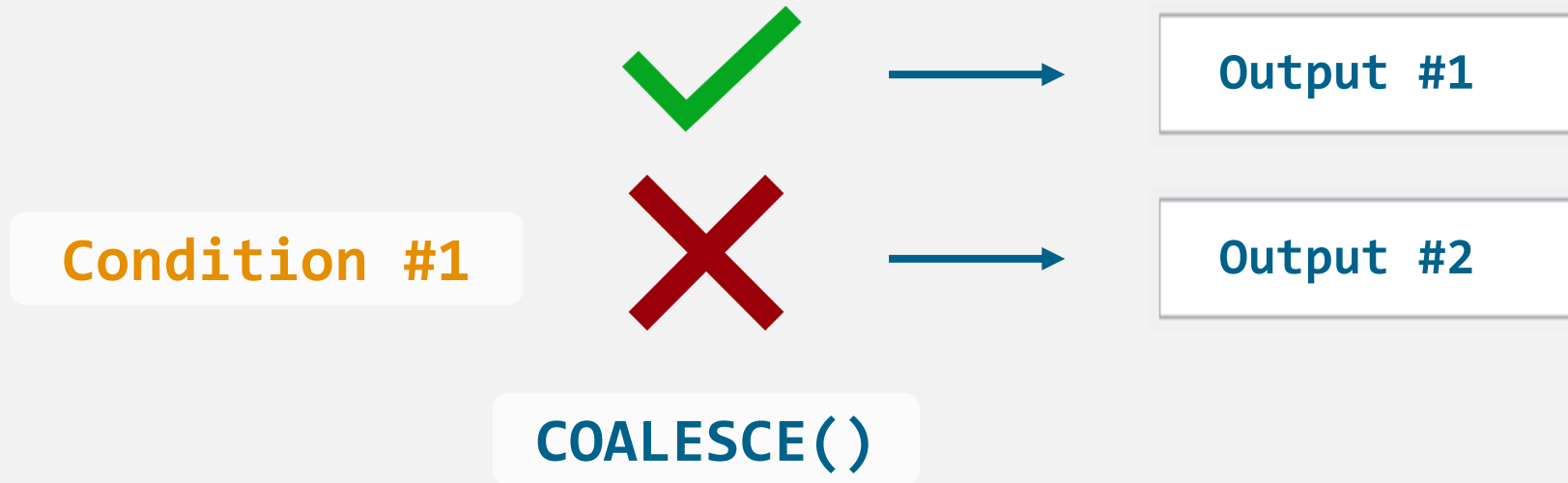
Condition #1



Output #1



Output #2



Condition #1



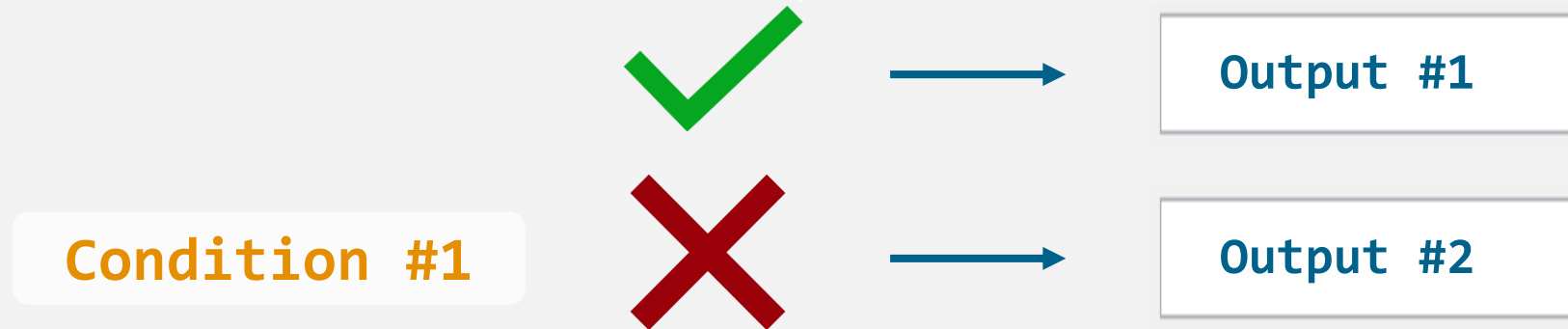
Output #1



Output #2

COALESCE()

IFNULL()



COALESCE()

IFNULL()

The CASE Statement

THE CASE STATEMENT



SQL

```
SELECT
    column_name(s)
    CASE
        WHEN condition_1 THEN result_1
        WHEN condition_2 THEN result_2
        ...
        ELSE
    END AS
FROM
    table_name;
```


THE CASE STATEMENT



SQL

```
SELECT
    column_name(s)
    CASE condition_field
        WHEN condition_field_value_1 THEN result_1
        WHEN condition_field_value_2 THEN result_2
        ...
    ELSE
    END AS
FROM
    table_name;
```