Hands-on Lab: Monitoring and Optimizing Your Databases in MySQL

Estimated time needed: 45 minutes

In this lab, you'll learn how to monitor and optimize your database in MySQL with the help of phpMyAdmin.

Objectives

After completing this lab, you will be able to:

1. Maintain the health and performance of your database with phpMyAdmin
2. Identify the difference between an unoptimized and optimized database
3. Optimize your database with phpMyAdmin by applying best practices

Software Used in this Lab

In this lab, you will use MySQL. MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.

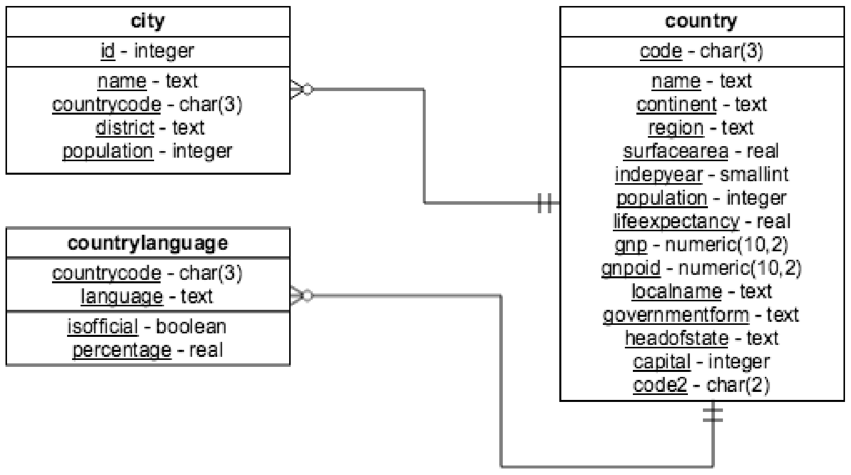
To complete this lab, you will utilize the MySQL relational database service available as part of the IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

The World database used in this lab comes from the following source: https://dev.mysql.com/doc/world-setup/en/ under CC BY 4.0 License with Copyright 2021 - Statistics Finland.

You will use a modified version of the database for the lab, so to follow the lab instructions successfully please use the database provided with the lab, rather than the database from the original source.

The following ERD diagram shows the schema of the World database:



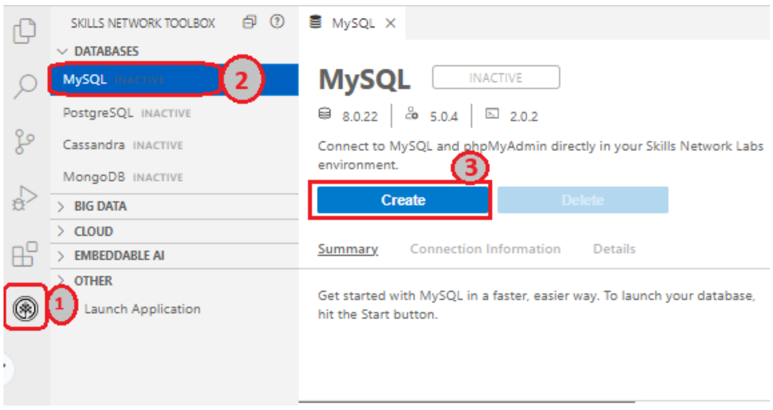
The first row is the table name, the second is the primary key, and the remaining items are any additional attributes.

Exercise 1: Create Your Database

To begin, we'll create the **World** database and load the necessary data.

1. First, we'll launch MySQL. We can do that by navigating to the **Skills Network Toolbox**, selecting **Databases** and then **MySQL**.

   Press **Start**. This will start a session of MySQL in SN Labs.



The **Inactive** label will change to **Starting**. This may take a few moments.

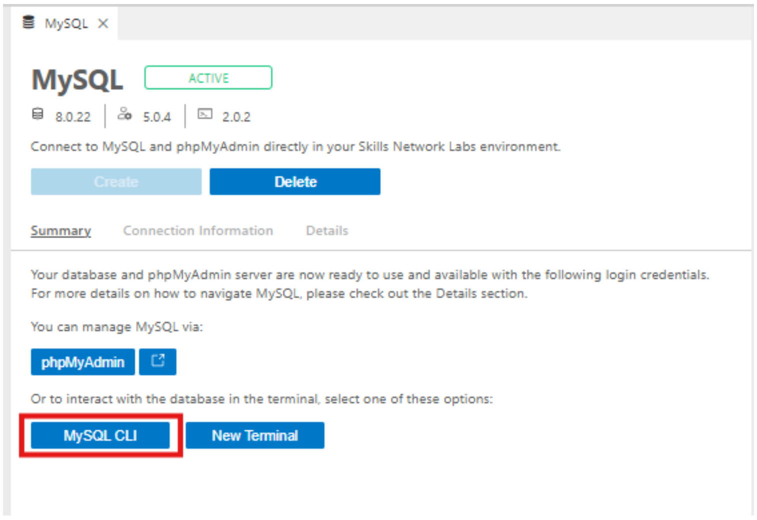
When it changes to **Active**, it means your session has started.

2. In the menu bar, select **Terminal** > **New Terminal**. This will open the Terminal.

   To download the zip file containing the **.sql** file for the **World** database, copy and paste the following into the Terminal:

   ```
   wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0231EN-SkillsNetwork/datasets/World/world_mysql_s
   ```

3. Next, initiate a MySQL command-line session by clicking **MySQL CLI** from the MySQL tab. Doing this will allow us to create a table in MySQL via the command-line.

   
   MySQL
ACTIVE
8.0.22
5.0.4
2.0.2
Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment.
Create
Delete
Summary
Connection Information
Details
Your database and phpMyAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate MySQL, please check out the Details section.
You can manage MySQL via:
phpMyAdmin
Or to interact with the database in the terminal, select one of these options:
MySQL CLI
New Terminal

4. In the Terminal, create a new database called **world**.

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

5. Now, we'll want to use the newly created database, **world**. How can we go about doing that?

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

6. In order to complete the database creation process, we'll want to execute the MySQL script containing the data that we downloaded. This file will create tables and insert data into those tables.

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

   This may take about thirty seconds or so.

7. With our database created, we'll want to take a look at the tables inside it to get a general idea of the data we have. What tables do we have?

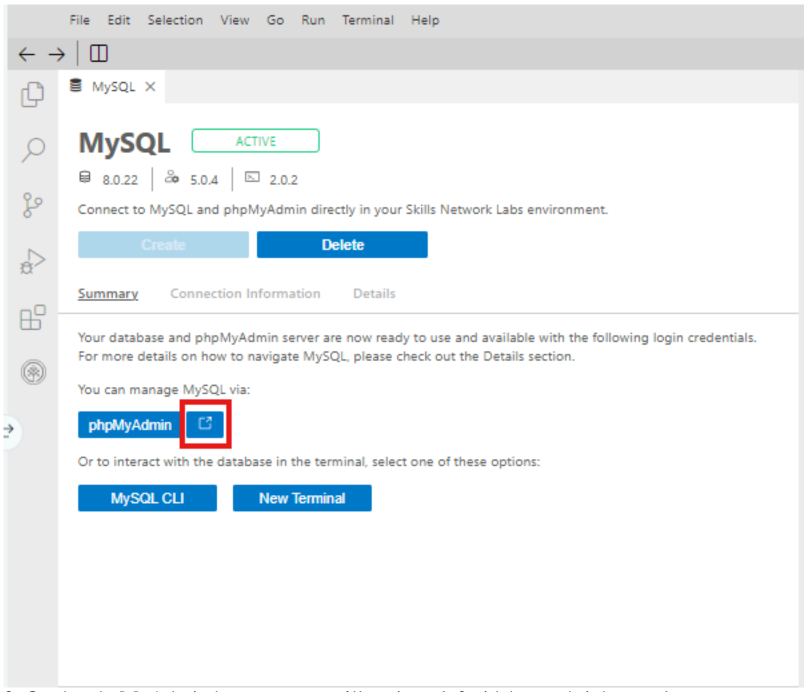
   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

Exercise 2: Monitor Your Database

Database monitoring refers to the tasks related to reviewing the operational status of your database. Monitoring is critical in maintaining the health and performance of your database because it'll help you identify issues in a timely manner. In doing so, you'll be able to avoid problems such as database outages that affect your users.

With phpMyAdmin, we can take a look at how our database is doing and how we can improve it.

1. In the MySQL tab, select button next to **phpMyAdmin** to access the tool in a new tab.



Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment.

Your database and phpMyAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate MySQL, please check out the Details section.

You can manage MySQL via:

Or to interact with the database in the terminal, select one of these options:

2. On the phpMyAdmin homepage, you'll notice a left sidebar and right panel.

On the left sidebar, you'll see all your databases, including the one we just created, **world**.

For now, we'll want to focus on the right panel to see more information about our servers and databases.

Select **Status**.

3. There are several subpages on the **Status** page: **Server**, **Processes**, **Query Statistics**, **All Status Variables**, **Monitor** and **Advisor**. These subpages will give you an inside look at the current processes on your server.

   Feel free to explore the rest of the subpages to gain a better understanding of the current status of your server.

   If you'd like a refresher on the information provided by each subpage, you can revisit the previous reading, [Monitoring and Optimizing Your Databases in MySQL]().

   In this lab, we'll be taking a look at the **Monitor** subpage.

4. From the available sections, click **Monitor**.

   When you first arrive at this page, you may only have three charts: **Questions**, **Connections / Processes** and **Traffic**. We can add a few more graphs to help track important metrics.

5. Select **Settings** then **Add chart**.

6. In the dialog box, you can choose a chart based on a status variable or select **Preset chart**. In this case, we'll be adding two preset charts.

   First, let's select **System CPU Usage**.

   Press **Add chart to grid**.

   Now, you should see the **System CPU Usage** following chart.

   Notice how there was a spike in your charts. Recall that **Questions** includes any queries run in the background by phpMyAdmin. In this case, the interaction was MySQL bringing up the **System CPU Usage** chart.

7. Repeat this process to add the **System memory** chart.

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

   What happens when we run a query? Let's take a look!

8. On the left sidebar, press the + sign next to the database **world**. Then, right-click **city** and slect **Open link in new tab**.

   Doing this will run a `SELECT * FROM city` query. Notice the spike in the charts. Hovering over the number will provide you more insight into the number of questions, or queries, that are run.

   To confirm that we ran the query, we can navigate to the new tab we opened and take a look at the success message:

   Since phpMyAdmin limits queries to displaying the first 25 rows (similar to the `LIMIT` clause), the query did not take long to load.

The **Status** page is helpful in monitoring the health and performance of your server and database. Particularly with the **Monitor** page's charts, you'll be able to see real-time information on how your databases are doing. If there are unexpected irregularities, such as a sudden spike in queries or a high volume of incoming traffic, then that may point to a problem that needs to be attended to.

Exercise 3: Optimize Your Database

Database optimization refers to maximizing the speed and efficiency of retrieving data from your database. When you optimize your database, you are improving its performance, which can also improve the performance of slow queries. By optimizing your database, you'll improve the experience for both yourself and your users.

We can see this in practice with the following scenario:

Imagine that you are the organizer of a sports tournament with a total of 120 participating teams. There's three pieces of information that you've collected from each team: their (unique) team number, their three-letter team code for the system, and the date of the team's creation.

How can we go about optimizing a database containing that information? Let's take a look!

Task A: Create Your Table

1. To start, we'll create a database in phpMyAdmin.

   On the left sidebar, select the **New** button to create a new database.

2. On the **Databases** page, you can enter a Database name and leave it as the **UTF-8** encoding. UTF-8 is the most commonly used character encoding for content or data.

   Let's call our database **tournament_teams**.

   Press **Create**.

3. Now, we can make a table. Let's call it **teams**.

   Recall the information we want to store in this database: team number, team code and team creation date. We can make three columns for that.

   Press **Go**.

4. We can now add in the column names and data types.

   We'll enter the following information into the table. Below are only the values that you'll need to fill out. Anything else can be left as is:

Name	Type	Length/Values
team_no	INT	
team_code	CHAR	100
creation_date	CHAR	100

   Since the team number is unique, with each of the 120 teams having their own number ranging from 1 to 120, that will be our primary key. We will set that later. It is a numeric data type, so we use **INT** for now.

   The team code and creation date are both set as the **CHAR** data type. Notice that for the **CHAR** type, we must include the length of the value. To be safe, we've set that to 100.

   You might be thinking that these data types aren't the best choices for this database. You'd be correct! Given what we know, there are better choices for each of the entries, but let's first take a look at an unoptimized database and how we can optimize it.

   When all the values are entered, press **Save**.

5. On the **Structure** page, select the checkbox next to **team_no** and click **Primary**.

   This will set **team_no** as the primary key of the table. If you were successful in doing so, you'll receive a success message, like the following:

   Notice that **team_no** was automatically added as a PRIMARY index as well.

6. Let's import the data! This sample teams data has been prepared specifically for this lab and is composed of team numbers, team code names, and the creation date of these teams.

   You can download the file by clicking the following link: tournament_teams_data.sql.

   To import it, navigate to **Import**, then select **Browse** and find the file you just downloaded.

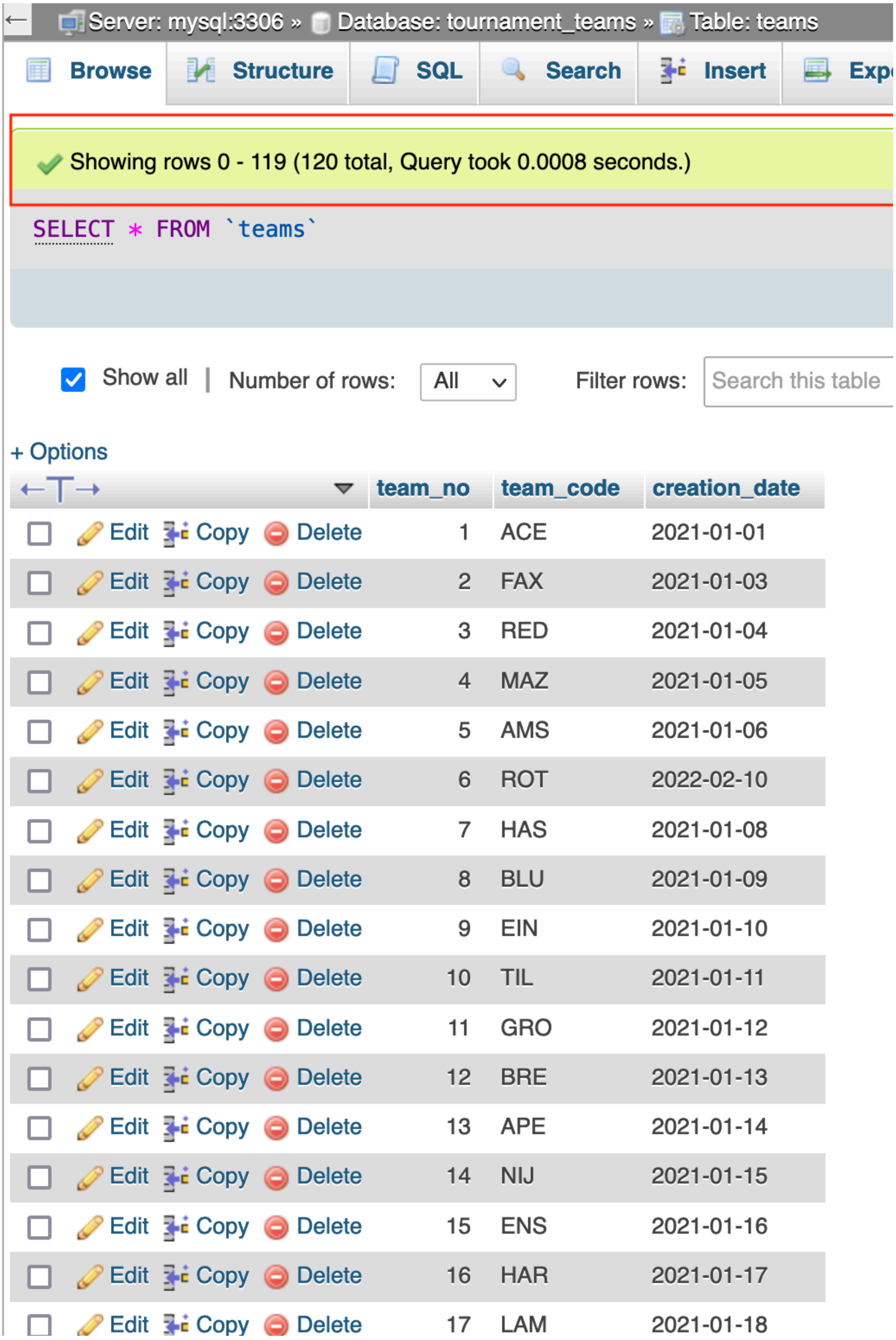
   Press **Go**.

   When your import is successful, you'll see the following message:

7. Now, we can take a look at the imported data by navigating back to **Browse**. This automatically loads a `SELECT * FROM teams` statement, which will load the first 25 rows from the **teams** table.

   The limit is implemented by phpMyAdmin, but you can easily show all rows by checking the **Show all** checkbox.

   You will receive a warning asking if you'd like to see all rows. Since there's only 120 rows in this example, you can click **OK**. In larger datasets, loading all the rows will likely crash your browser.


Server: mysql:3306 » Database: tournament_teams » Table: teams
Browse
Structure
SQL
Search
Insert
Exp

Showing rows 0 - 119 (120 total, Query took 0.0008 seconds.)

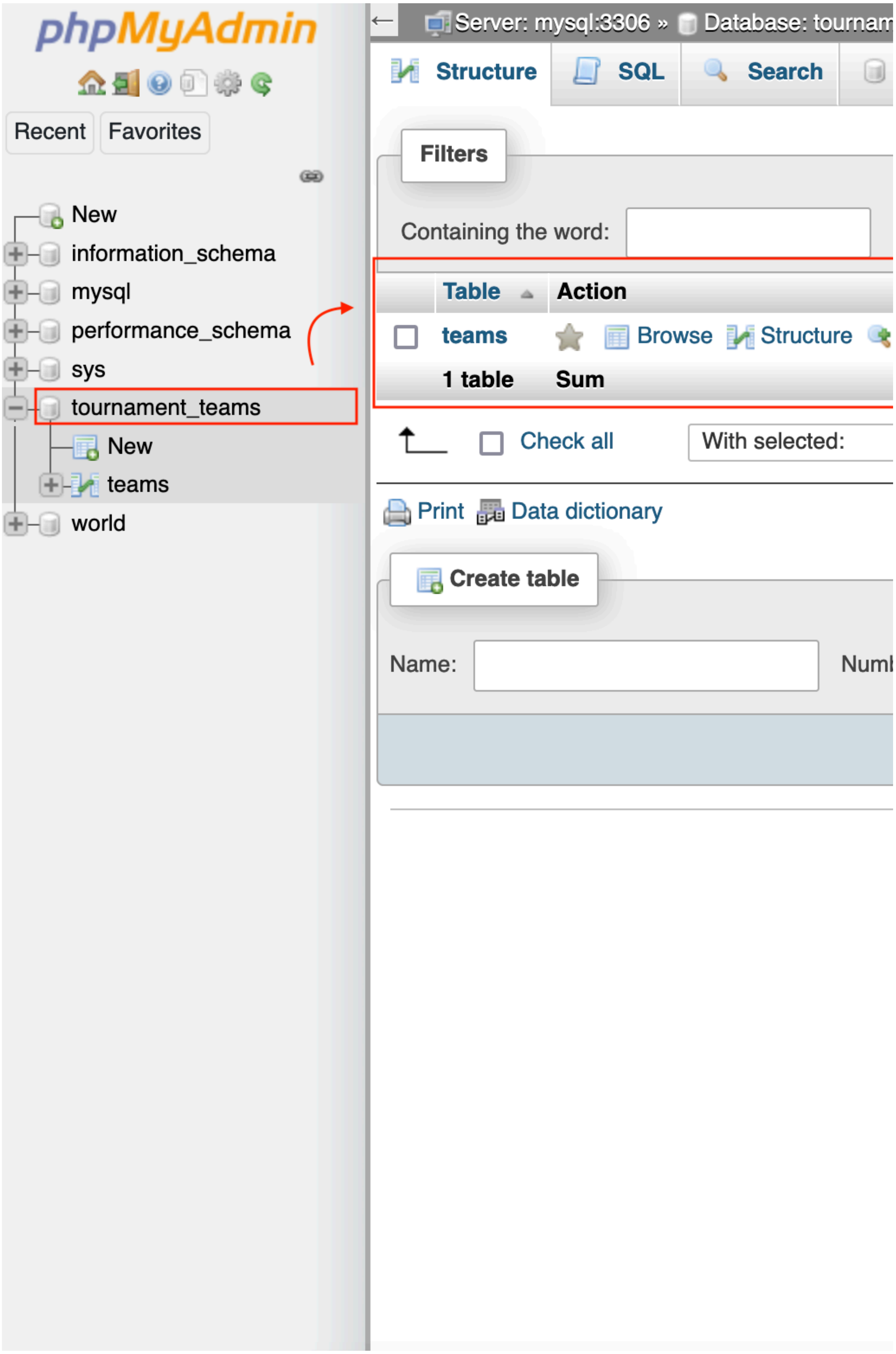
```
SELECT * FROM `teams`
```

Show all | Number of rows: All
Filter rows: Search this table
+ Options

	team_no	team_code	creation_date
Edit Copy Delete	1	ACE	2021-01-01
Edit Copy Delete	2	FAX	2021-01-03
Edit Copy Delete	3	RED	2021-01-04
Edit Copy Delete	4	MAZ	2021-01-05
Edit Copy Delete	5	AMS	2021-01-06
Edit Copy Delete	6	ROT	2022-02-10
Edit Copy Delete	7	HAS	2021-01-08
Edit Copy Delete	8	BLU	2021-01-09
Edit Copy Delete	9	EIN	2021-01-10
Edit Copy Delete	10	TIL	2021-01-11
Edit Copy Delete	11	GRO	2021-01-12
Edit Copy Delete	12	BRE	2021-01-13
Edit Copy Delete	13	APE	2021-01-14
Edit Copy Delete	14	NIJ	2021-01-15
Edit Copy Delete	15	ENS	2021-01-16
Edit Copy Delete	16	HAR	2021-01-17
Edit Copy Delete	17	LAM	2021-01-18

We can now confirm that all 120 rows have been loaded.

8. On the left sidebar, select the **tournament_teams** database.


phpMyAdmin
Recent
Favorites
New
information_schema
mysql
performance_schema
sys
tournament_teams
New
teams
world
Server: mysql:3306 » Database: tournam
Structure
SQL
Search
Filters
Containing the word:
Table
Action
teams
Browse
Structure
1 table
Sum
Check all
With selected:
Print
Data dictionary
Create table
Name:
Numb

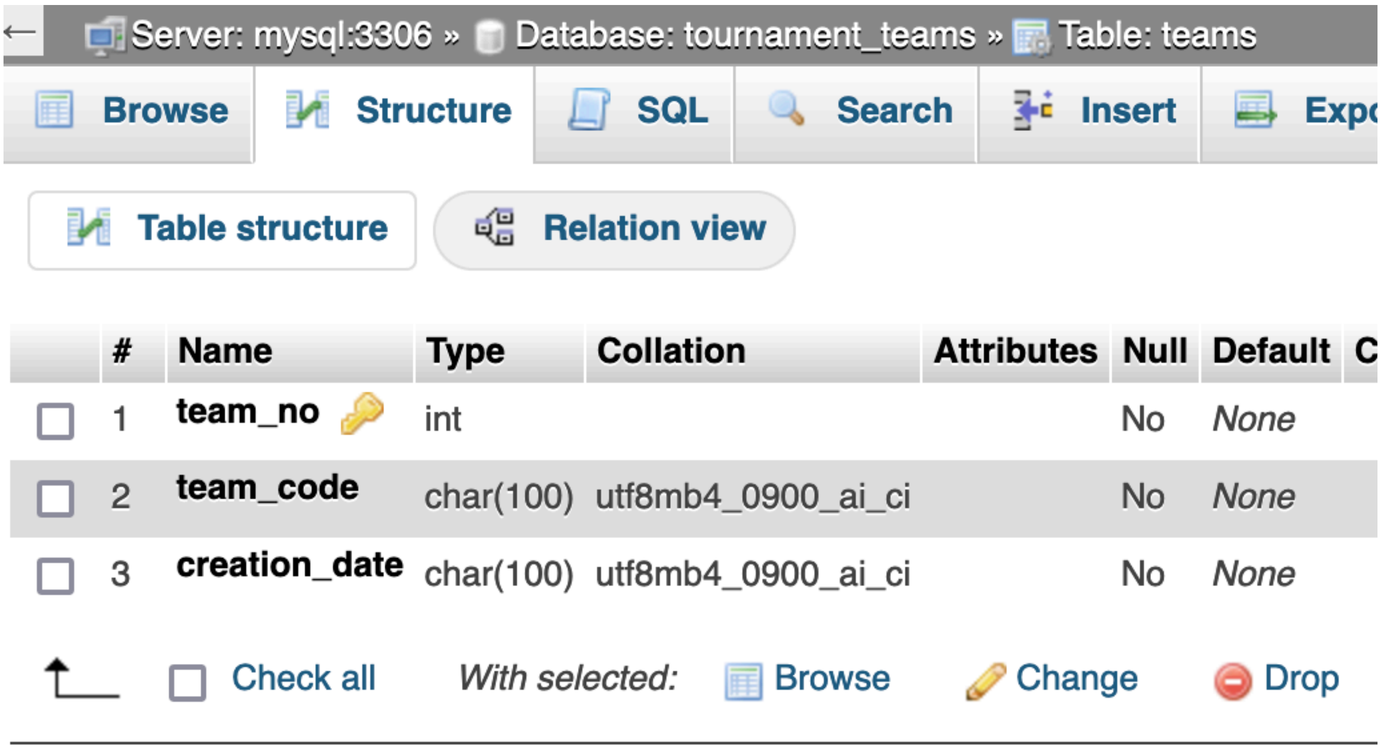
With this view, you can see all the tables in the database, including the **teams** table that we just created. As you can see, the size of the database is currently 64 KiB, that is, 64 kilobytes. While this is a small size, do remember that this would not be the case with larger datasets as this sample table only has 120 entries.

Keep this number in mind as we start optimizing this table.

Task B: Optimize Your Data Types

1. Under **Action** for the **teams** table, select **Structure**. This will bring you back to the **Structure** page.

   Next to **team_no**, click **Change**. This function is helpful when you need to make adjustments to a column, for any reason.



	#	Name	Type	Collation	Attributes	Null	Default	C
☐	1	team_no	int			No	*None*	
☐	2	team_code	char(100)	utf8mb4_0900_ai_ci		No	*None*	
☐	3	creation_date	char(100)	utf8mb4_0900_ai_ci		No	*None*	

2. In this editor, let's change the **Type** to one that would be better suited for this column.

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

3. Change the data type for **team_code**!

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

4. Finally, let's change the **creation_date**.

   ▶ Hint (Click Here)
   ▶ Solution (Click Here)

5. After updating your data types, your **Structure** page should look like the following:

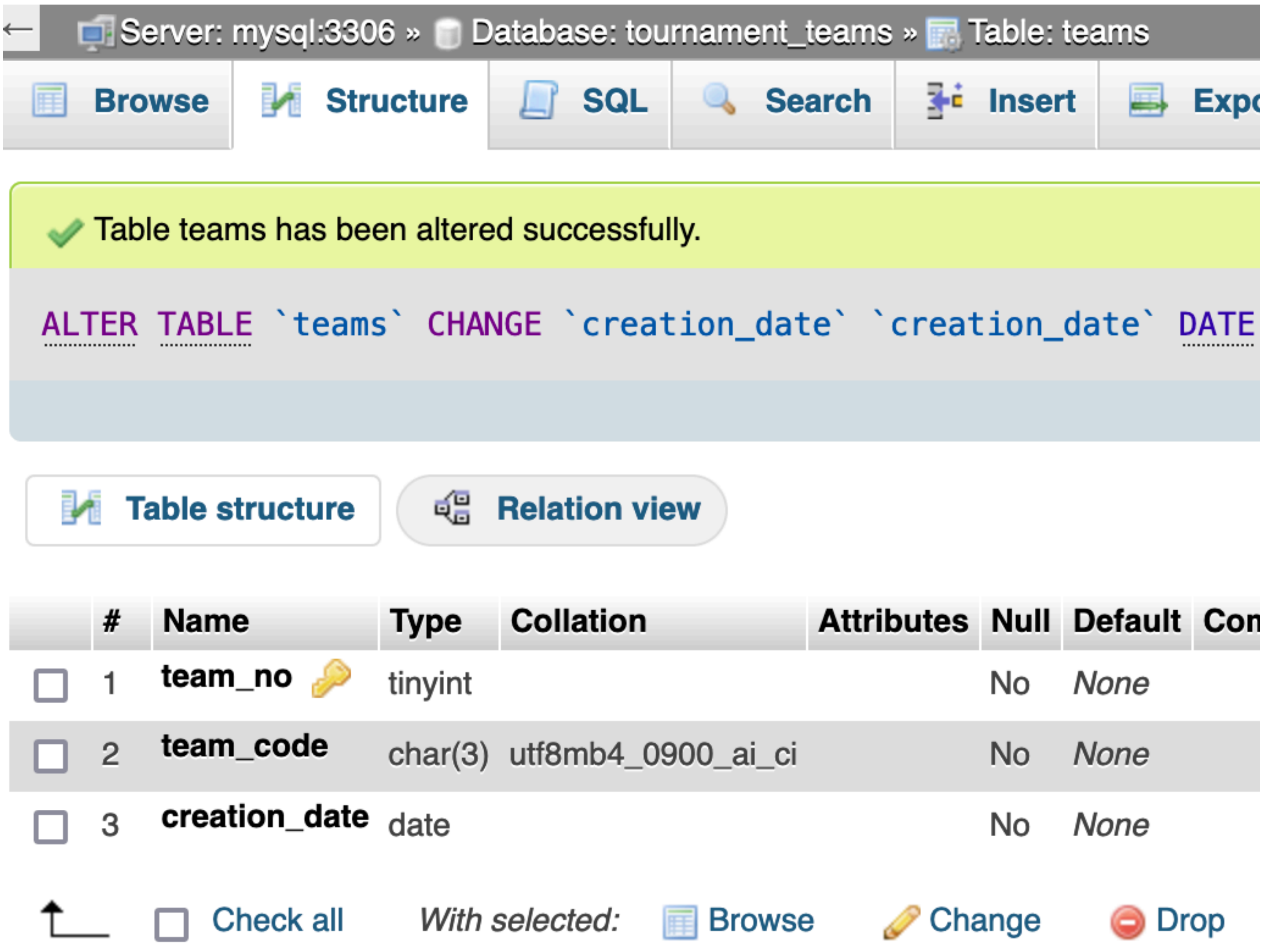

Server: mysql:3306 » Database: tournament_teams » Table: teams

Table teams has been altered successfully.

```
ALTER TABLE `teams` CHANGE `creation_date` `creation_date` DATE
```

#	Name	Type	Collation	Attributes	Null	Default	Con
1	team_no	tinyint			No	*None*	
2	team_code	char(3)	utf8mb4_0900_ai_ci		No	*None*	
3	creation_date	date			No	*None*	

6. We can look at our rows by clicking **Browse**.

   As you can see, we still have all 120 entries. Great! So, let's take a look at the size of this table.

7. Return to the **tournament_teams** database in the left sidebar.

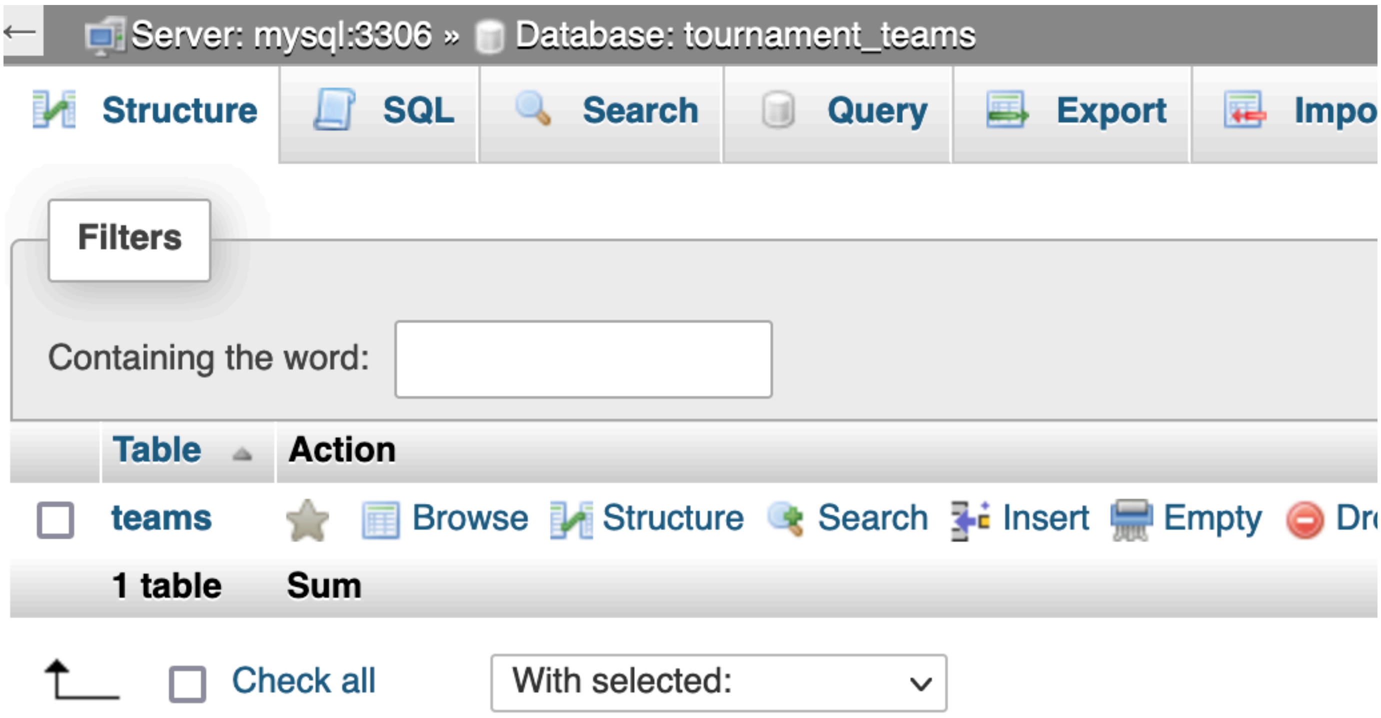

Server: mysql:3306 » Database: tournament_teams

Table	Action
teams	Browse Structure Search Insert Empty Dr
1 table	Sum

Upon first glance, we can see that the size of the database has reduced to 16 KiB, four times smaller than the original unoptimized database!

8. We can take optimization a step further by selecting the **teams** table and, in the dropbox, selecting **optimize table**. This is equivalent to performing the `OPTIMIZE TABLE` function in the command-line interface. This command reorganizes the table data and indexes to reduce storage and make accessing the table more efficient. The exact changes made to the table depend on the storage engine in use.

With the InnoDB engine, which is what we are currently using, the table is actually rebuilt, with the index statistics updated to free unused space.

That is what the returned table tells us: InnoDB “optimizes” tables by recreating and analyzing them, and that the optimization had been completed.

Conclusion

Congratulations! You now know how to monitor and optimize your database with the help of the handy tool, phpMyAdmin. Now, you can apply what you have learned to any database that you create or modify in the future.

Author(s)

Kathy An

Other Contributor(s)

Rav Ahuja