

a

Materiały do przedmiotu:

Bazy danych

Laboratorium nr 3:

Podstawy języka SQL: polecenia DDL i DML

Autor:

Alicja Żmudzińska



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Spis treści

Wprowadzenie, tematyka zajęć	3
Cel ćwiczenia/laboratorium	3
Instrukcja laboratoryjna/ćwiczeniowa	3
Ćwiczenia samodzielne	30
Pytania pomocnicze	32
Podsumowanie	32
Literatura	32



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU
SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Podczas zajęć uczestnicy poznają podstawy pracy z relacyjnymi bazami danych i językiem SQL, który jest standardowym narzędziem do tworzenia, modyfikowania i pobierania danych. Punktem wyjścia będzie przykładowa baza Sakila, symulująca wypożyczalnię filmów. Na jej przykładzie krok po kroku omówiony zostanie cały proces: od pobrania i instalacji bazy, poprzez jej wypełnienie danymi, aż po tworzenie własnych zapytań i modyfikowanie struktury. W ramach zajęć wyjaśnione zostanie, czym jest baza danych, jakie są jej rodzaje i w jakich sytuacjach jest wykorzystywana. Następnie uczestnicy wprowadzeni zostaną w świat języka SQL, poznając zarówno polecenia służące do manipulowania danymi (DML), jak i te, które pozwalają definiować strukturę bazy (DDL). Kolejnym krokiem będzie praca z bazą Sakila w środowisku XAMPP i phpMyAdmin – tworzenie nowej bazy, import struktury i danych oraz weryfikacja poprawności importu. Istotną część zajęć stanowią ćwiczenia praktyczne, obejmujące wykonywanie zapytań SELECT z różnymi klauzulami, a także wstawianie, modyfikowanie i usuwanie rekordów. Uczestnicy nauczą się tworzyć i modyfikować tabele, dodawać klucze główne i obce oraz dbać o integralność danych. Na zakończenie przewidziane jest samodzielne rozszerzanie bazy, na przykład poprzez dodanie nowych tabel i tworzenie powiązań między nimi. Celem spotkania jest zdobycie umiejętności praktycznego posługiwania się SQL oraz swobodnego zarządzania relacyjną bazą danych w graficznym środowisku phpMyAdmin, co stanowi doskonałe przygotowanie do pracy z rzeczywistymi systemami bazodanowymi.

Cel ćwiczenia/laboratorium

Celem ćwiczeń jest zapoznanie się z podstawowymi zagadnieniami dotyczącymi baz danych oraz języka SQL. Uczestnicy zajęć nauczą się rozróżniać rodzaje baz danych i zrozumieją, dlaczego SQL jest powszechnie stosowanym językiem do zarządzania nimi. W trakcie pracy praktycznej zostanie wykorzystana przykładowa baza danych Sakila, dzięki której możliwe będzie przećwiczenie zarówno tworzenia i importowania bazy w środowisku phpMyAdmin, jak i wykonywania podstawowych operacji na danych. Studenci będą mieli okazję poznać zapytania SQL związane z manipulowaniem danymi oraz definiowaniem struktury tabel, co pozwoli im lepiej zrozumieć relacje między tabelami oraz praktyczne zastosowanie kluczy obcych w relacyjnych bazach danych.

Instrukcja laboratoryjna/ćwiczeniowa

Co to jest baza danych?

Baza danych (ang. Database) to uporządkowany zbiór danych, który umożliwia ich przechowywanie, organizację i szybki dostęp. Dzięki bazom danych można efektywnie



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



zarządzać dużymi ilościami informacji w różnych systemach, np. w sklepach internetowych, bankach, wypożyczalniach filmów czy portalach społecznościowych.

Rodzaje baz danych:

- Relacyjne (RDBMS) – dane przechowywane w tabelach powiązanych ze sobą (np. MySQL, PostgreSQL, Oracle).
- NoSQL – dane przechowywane w formie dokumentów, klucz-wartość, grafów (np. MongoDB, Redis).

Dlaczego SQL?

SQL (Structured Query Language) to standardowy język do zarządzania relacyjnymi bazami danych.

Dzięki SQL możemy:

- Tworzyć i modyfikować struktury baz danych,
- Wstawiać, modyfikować i usuwać dane,
- Pobierać dane zgodnie z określonymi kryteriami,
- Zarządzać uprawnieniami użytkowników.

SQL jest używany praktycznie w każdej aplikacji, która potrzebuje przechowywać dane w uporządkowany sposób.

Jak uczymy się SQL?

Do nauki SQL warto używać gotowych baz danych testowych, które pozwalają ćwiczyć zapytania DML i DDL na realistycznych przykładach. Jedną z popularnych takich baz jest Sakila, która symuluje działanie wypożyczalni filmów.

Baza sakila:

Baza danych Sakila to przykładowa baza danych, która została stworzona przez MySQL, aby pomóc w nauce i demonstracji możliwości systemu zarządzania bazą danych (DBMS) MySQL. Sakila symuluje środowisko wypożyczalni filmów i zawiera dane dotyczące filmów, aktorów, klientów, pracowników, wypożyczeń, sklepów, oraz inne powiązane informacje. Kluczowe elementy bazy danych Sakila:

Struktura Tabel:

- Film (film): Zawiera informacje o filmach, takie jak tytuł, opis, długość, data wydania, itp.
- Aktor (actor): Przechowuje dane aktorów, w tym imię, nazwisko i inne szczegóły.
- Kategoria (category): Zawiera kategorie filmów, takie jak komedia, dramat, akcja itp.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

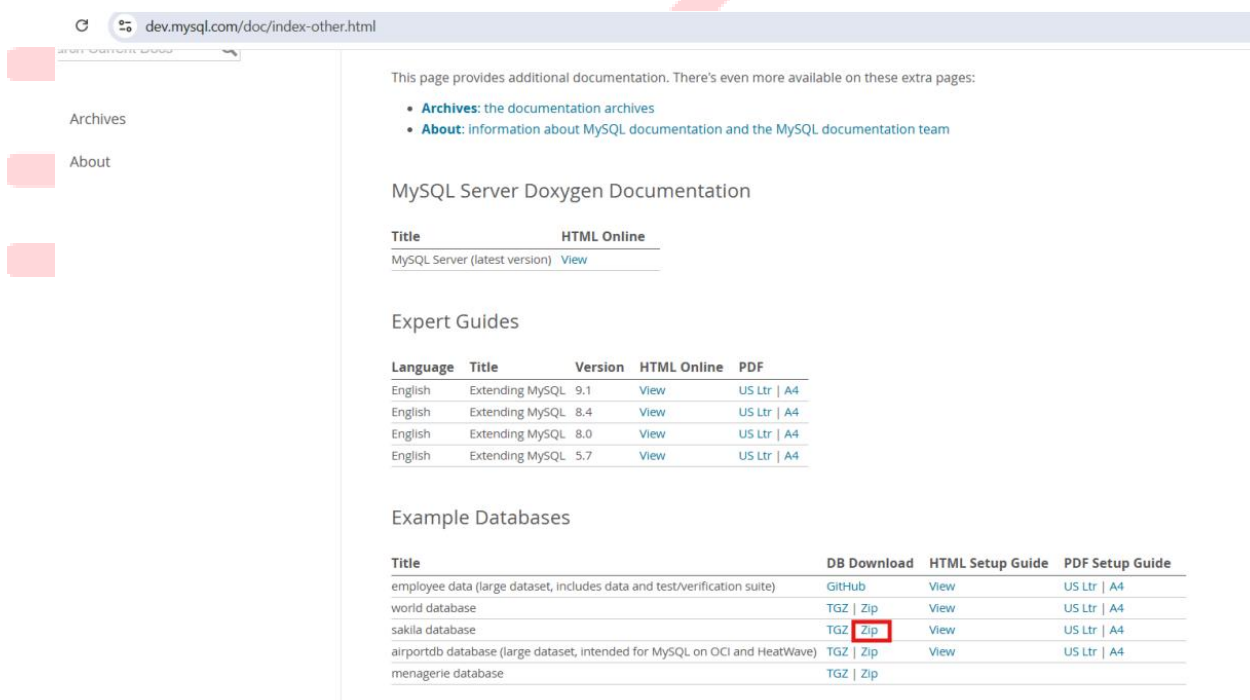
Dofinansowane przez
Unię Europejską



- Klient (customer): Zawiera dane klientów, w tym imię, nazwisko, adres itp.
- Wypożyczenia (rental): Przechowuje informacje o wypożyczeniach filmów przez klientów, w tym daty wypożyczenia i zwrotu.
- Pracownik (staff): Zawiera dane pracowników, którzy zarządzają wypożyczalnią.
- Relacje: Baza danych Sakila zawiera relacje między tabelami, co pozwala na przeprowadzanie skomplikowanych zapytań, takich jak pobieranie informacji o filmach wypożyczanych przez konkretnego klienta lub aktorach występujących w danym filmie.

Zastosowanie: Sakila jest często używana do nauki SQL, testowania funkcji MySQL, a także do prezentacji możliwości systemu w kontekście relacyjnych baz danych. Umożliwia programistom i analitykom przetestowanie i zrozumienie zagadnień związanych z normalizacją danych, tworzeniem zapytań, używaniem kluczy obcych i innych koncepcji baz danych.

Należy ją pobrać ze strony: <https://dev.mysql.com/doc/index-other.html>, tak jak przedstawiono na rysunku 1.



This page provides additional documentation. There's even more available on these extra pages:

- **Archives:** the documentation archives
- **About:** information about MySQL documentation and the MySQL documentation team

MySQL Server Doxygen Documentation

Title	HTML Online
MySQL Server (latest version)	View

Expert Guides

Language	Title	Version	HTML Online	PDF
English	Extending MySQL	9.1	View	US Ltr A4
English	Extending MySQL	8.4	View	US Ltr A4
English	Extending MySQL	8.0	View	US Ltr A4
English	Extending MySQL	5.7	View	US Ltr A4

Example Databases

Title	DB Download	HTML Setup Guide	PDF Setup Guide
employee data (large dataset, includes data and test/verification suite)	GitHub	View	US Ltr A4
world database	TGZ Zip	View	US Ltr A4
sakila database	TGZ Zip	View	US Ltr A4
airportdb database (large dataset, intended for MySQL on OCI and HeatWave)	TGZ Zip	View	US Ltr A4
menagerie database	TGZ Zip		

Rysunek 1. Strona pobierania przykładowych baz danych MySQL – dokumentacja i linki do plików



Fundusze Europejskie
dla Rozwoju Społecznego



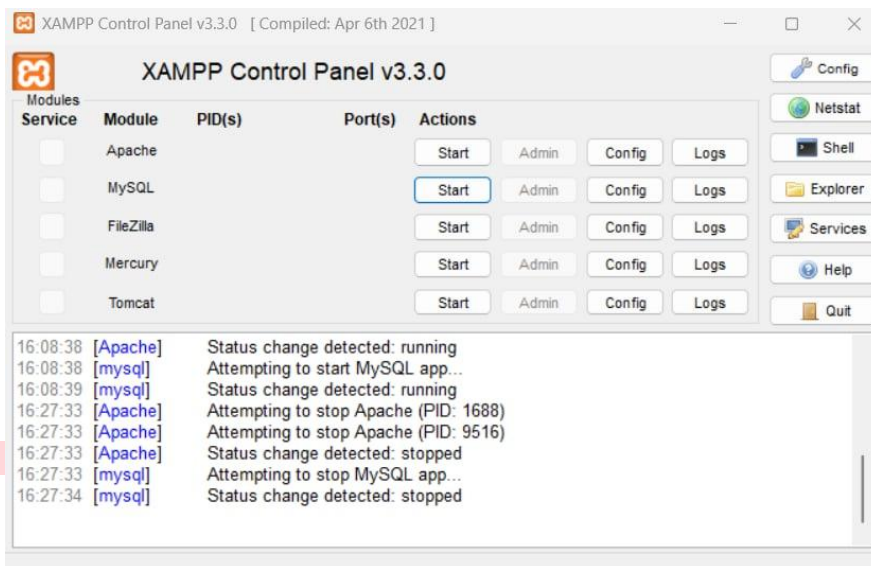
Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



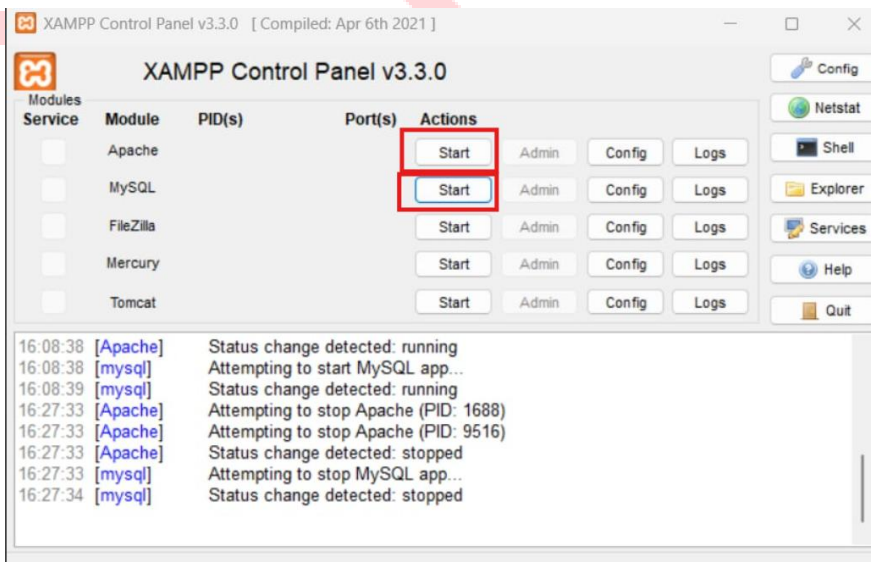
Dodanie bazy sakilka w phpMyAdmin:

W celu dodania bazy danych w phpMyAdmin należy uruchomić XAMPP Control Panel, co zilustrowano na rysunku 2.



Rysunek 2. Panel sterowania XAMPP (XAMPP Control Panel)

Następnie należy wcisnąć przycisk Start w wierszach Apache i MySQL, tak jak przedstawiono na rysunku 3.



Rysunek 3. Uruchamianie usług Apache i MySQL w XAMPP Control Panel



Fundusze Europejskie
dla Rozwoju Społecznego

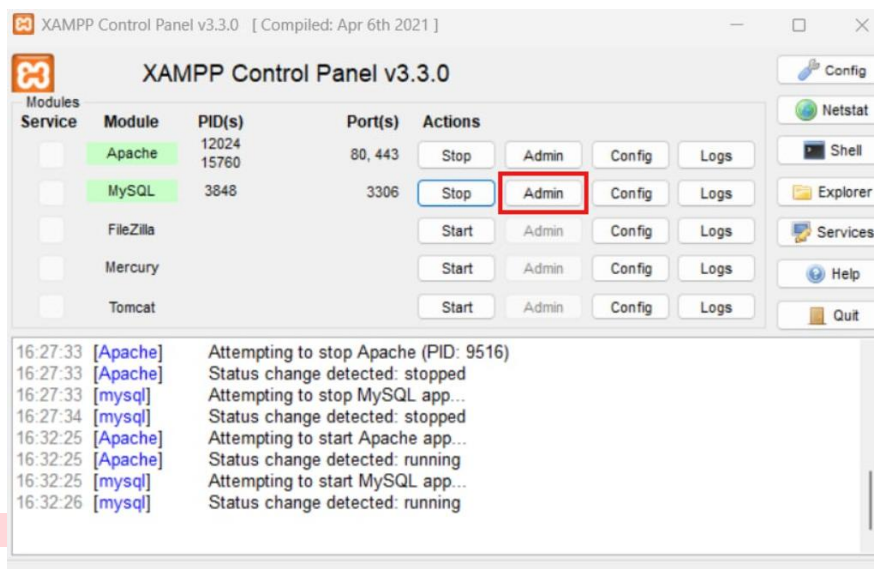


Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

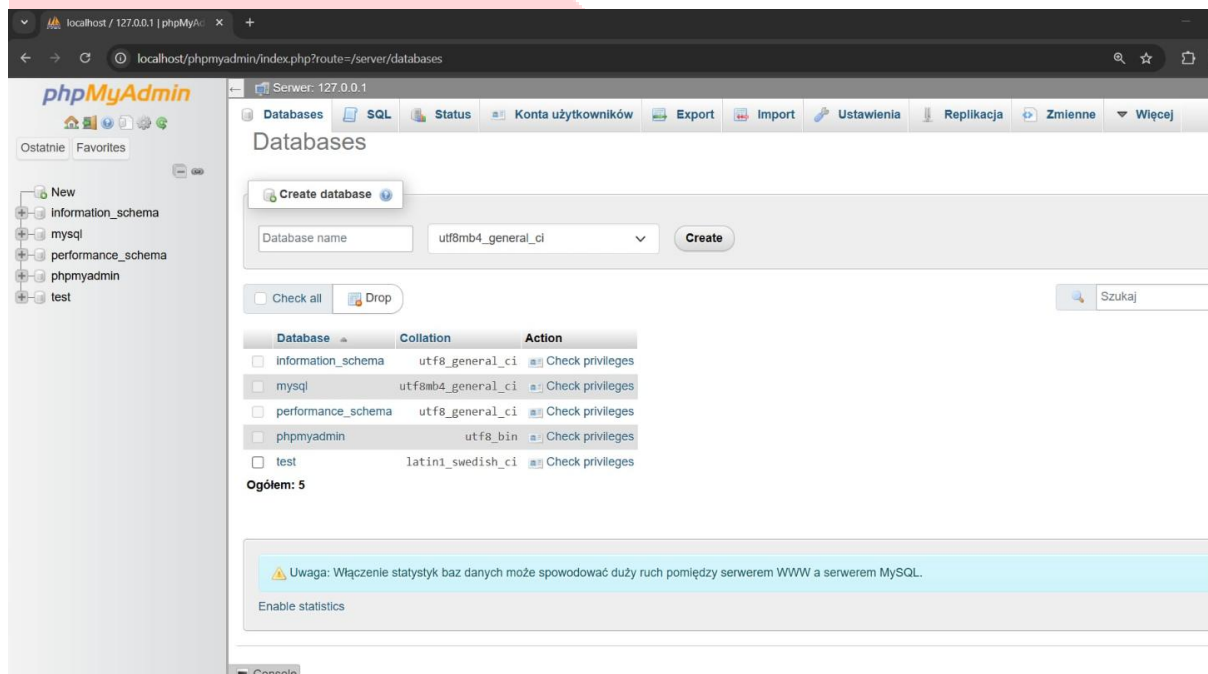


Po uruchomieniu usług przyciski Admin powinny się podświetlić. W celu uruchomienia phpMyAdmin należy kliknąć przycisk Admin w wierszu MySQL, tak jak przedstawiono na rysunku 4.



Rysunek 4. Uruchamianie phpMyAdmin z poziomu XAMPP Control Panel

Po kliknięciu przycisku Admin w panelu XAMPP w przeglądarce otworzy się okno phpMyAdmin, tak jak przedstawiono na rysunku 5.



Fundusze Europejskie
dla Rozwoju Społecznego



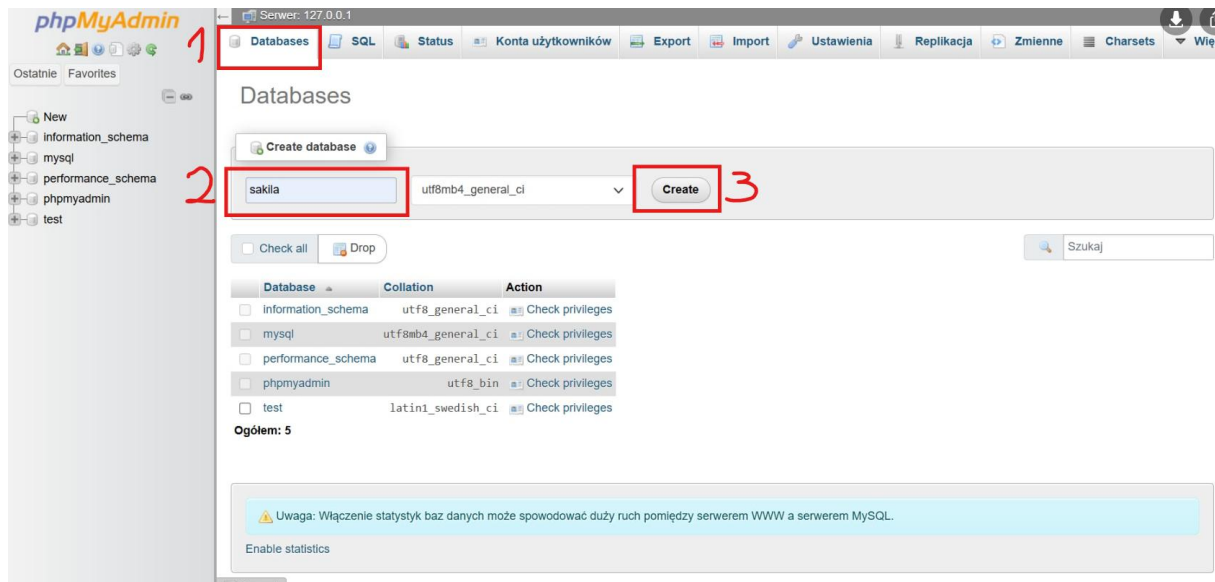
Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



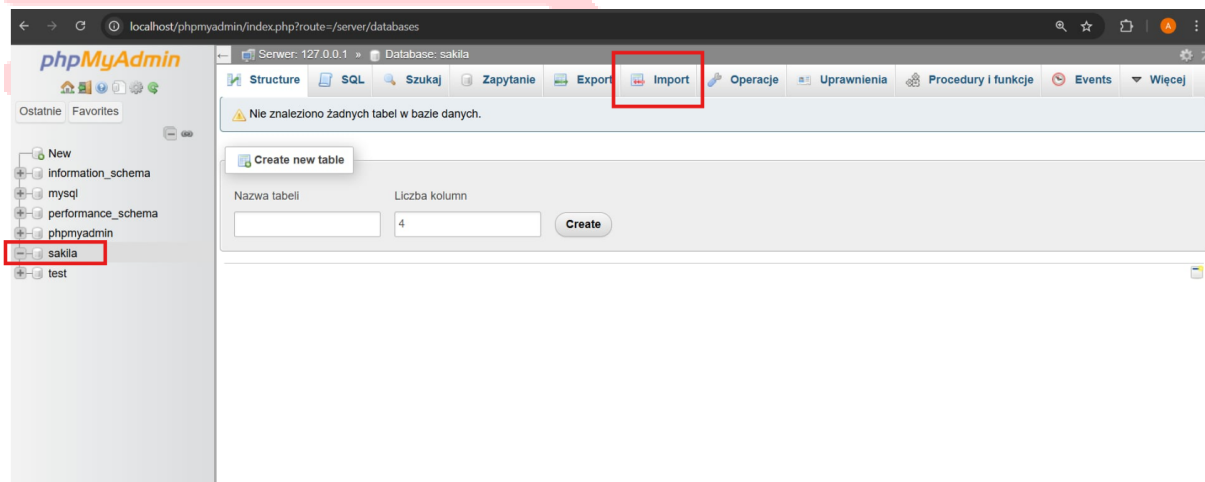
Rysunek 5. Ekran główny phpMyAdmin w przeglądarce

Teraz należy dodać bazę danych sakila. W tym celu w zakładce Databases (1) należy wpisać nazwę bazy (2) i kliknąć przycisk Create (3), tak jak przedstawiono na rysunku 6.



Rysunek 6. Dodanie bazy danych „sakila” w phpMyAdmin

Utworzy się pusta baza danych sakila. Aby dodać do niej dane pobrane wcześniej, należy w menu górnym przejść do zakładki Import, jak pokazano na rysunku 7.



Rysunek 7. Import danych do bazy „sakila” w phpMyAdmin

W zakładce Import kliknij przycisk Wybierz plik, a następnie wskaż wcześniej pobrany plik sakila-schema.sql. staw pozostałe opcje tak, jak przedstawiono na rysunku 8, a następnie naciśnij przycisk Import w celu wczytania struktury bazy danych.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Importowanie do bazy danych "sakila"



File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in `.[format].[compression]`. Example: `.sql.zip`

Browse your computer: (Maksymalny rozmiar: 40MB)

Wybierz plik sakila-schema.sql

Możesz także przeciągnąć i upuścić plik na dowolnej stronie.

Character set of the file:

utf-8

Częściowy import:

☐ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit.
This might be a good way to import large files, however it can break transactions.

Pomiń tę liczbę zapytań (dla SQL), zaczynając od pierwszej:

0

Other options

☐ Enable foreign key checks

Format

SQL

Format-specific options:

Tryb zgodności SQL:

NONE

☐ Do not use AUTO_INCREMENT for zero values

Import

Rysunek 8. Import pliku sakila-schema.sql do bazy danych w phpMyAdmin

Po zakończeniu importu powinno pojawić się okno z komunikatem Import zakończony sukcesem oraz listą wykonanych zapytań, jak pokazano na rysunku 9.



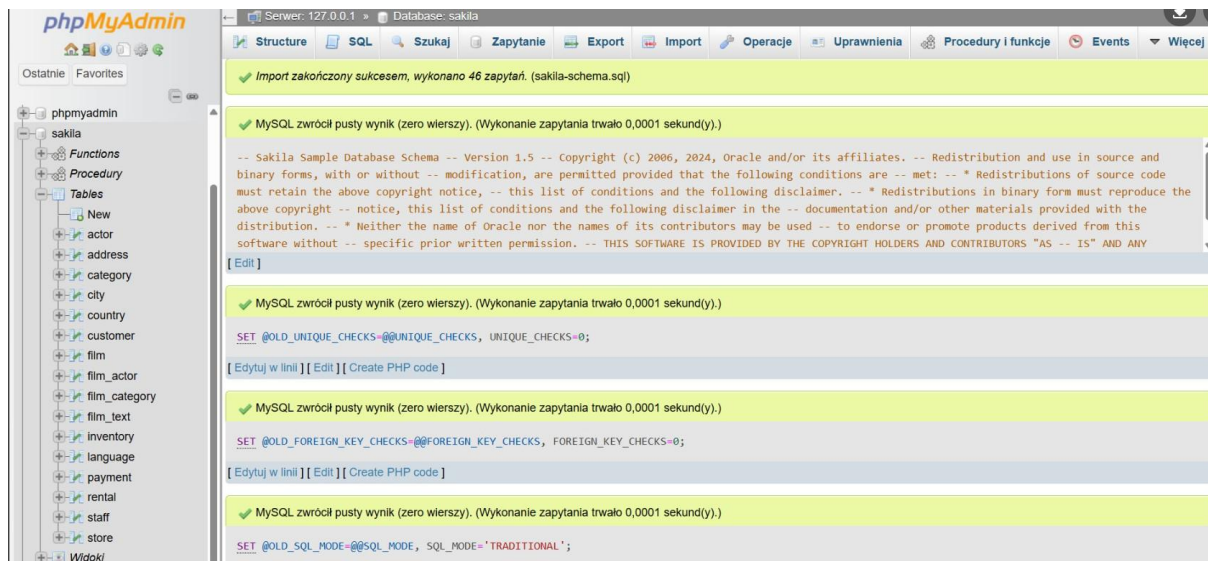
Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

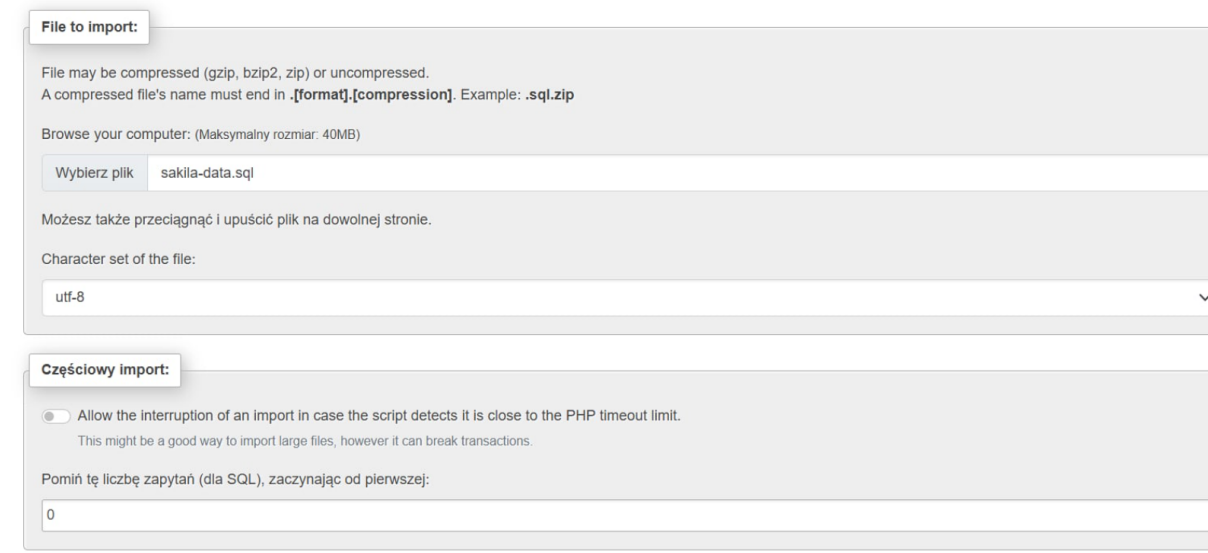




Rysunek 9. Wynik pomyślnego importu struktury bazy danych „sakila” w phpMyAdmin

Po utworzeniu struktury tabel należy wypełnić je danymi. W tym celu wybierz z listy po lewej stronie bazę sakila, przejdź do zakładki Import i tym razem wskaż plik sakila-data.sql, jak przedstawiono na rysunku 10. Pozostałe pola wypełnij tak jak przy imporcie pliku sakila-schema.sql. Następnie kliknij Import.

Importowanie do bazy danych "sakila"



Rysunek 10. Import pliku sakila-data.sql w celu wypełnienia bazy danymi w phpMyAdmin



Fundusze Europejskie
dla Rozwoju Społecznego

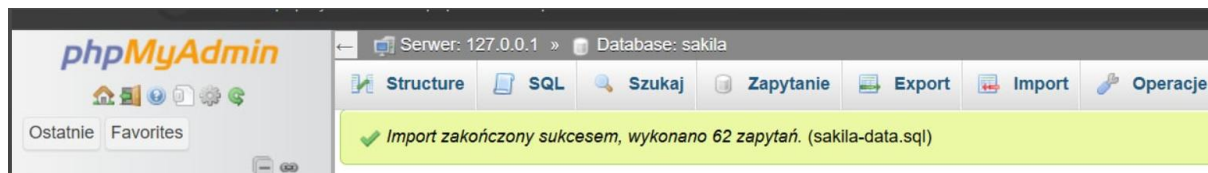


Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

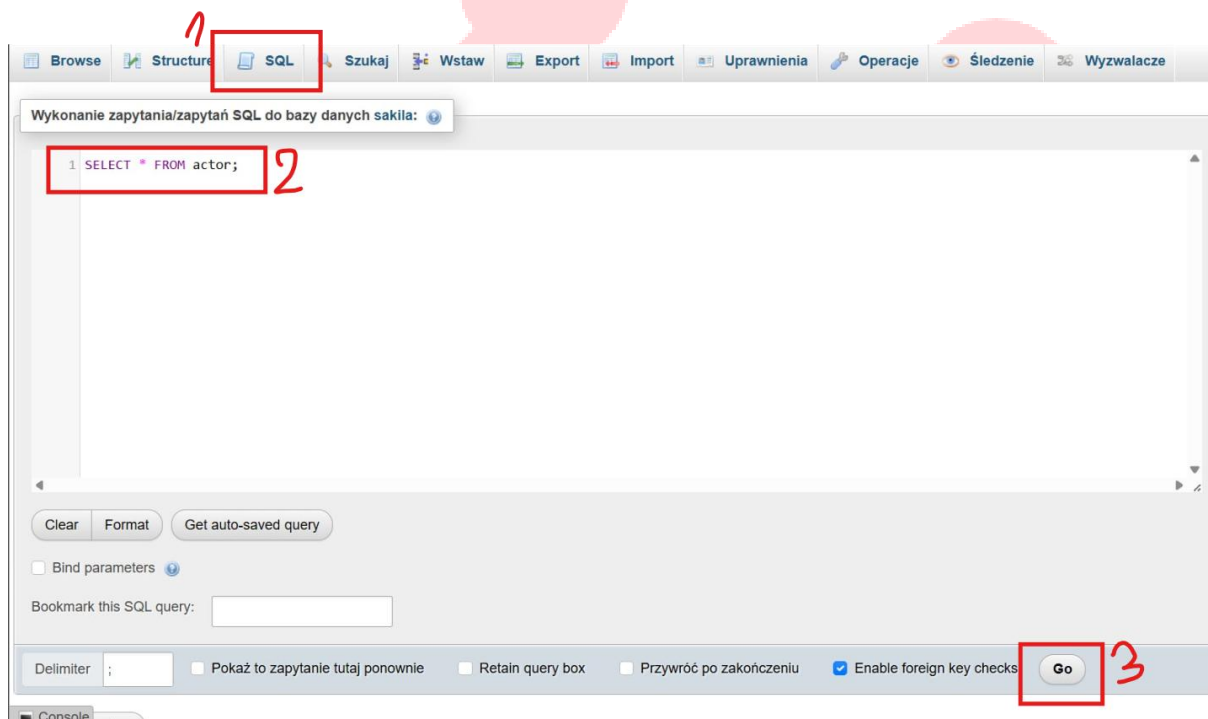


Po zakończeniu importu danych powinien pojawić się komunikat Import zakończony sukcesem, wykonano 62 zapytań (sakila-data.sql), jak pokazano na rysunku 11.



Rysunek 11. Potwierdzenie pomyślnego importu danych do bazy „sakila” w phpMyAdmin

Aby sprawdzić, czy baza danych sakila została poprawnie zaimportowana, przejdź do zakładki SQL (1), wpisz zapytanie `SELECT * FROM aktor;` (2), a następnie kliknij przycisk Go (3), jak przedstawiono na rysunku 12.



Rysunek 12. Wykonanie zapytania SQL w phpMyAdmin w celu sprawdzenia bazy „sakila”

Jeśli baza została poprawnie zaimportowana powinna wyświetlić nam się lista rekordów z tabeli aktor odpowiadająca naszemu zapytaniu:



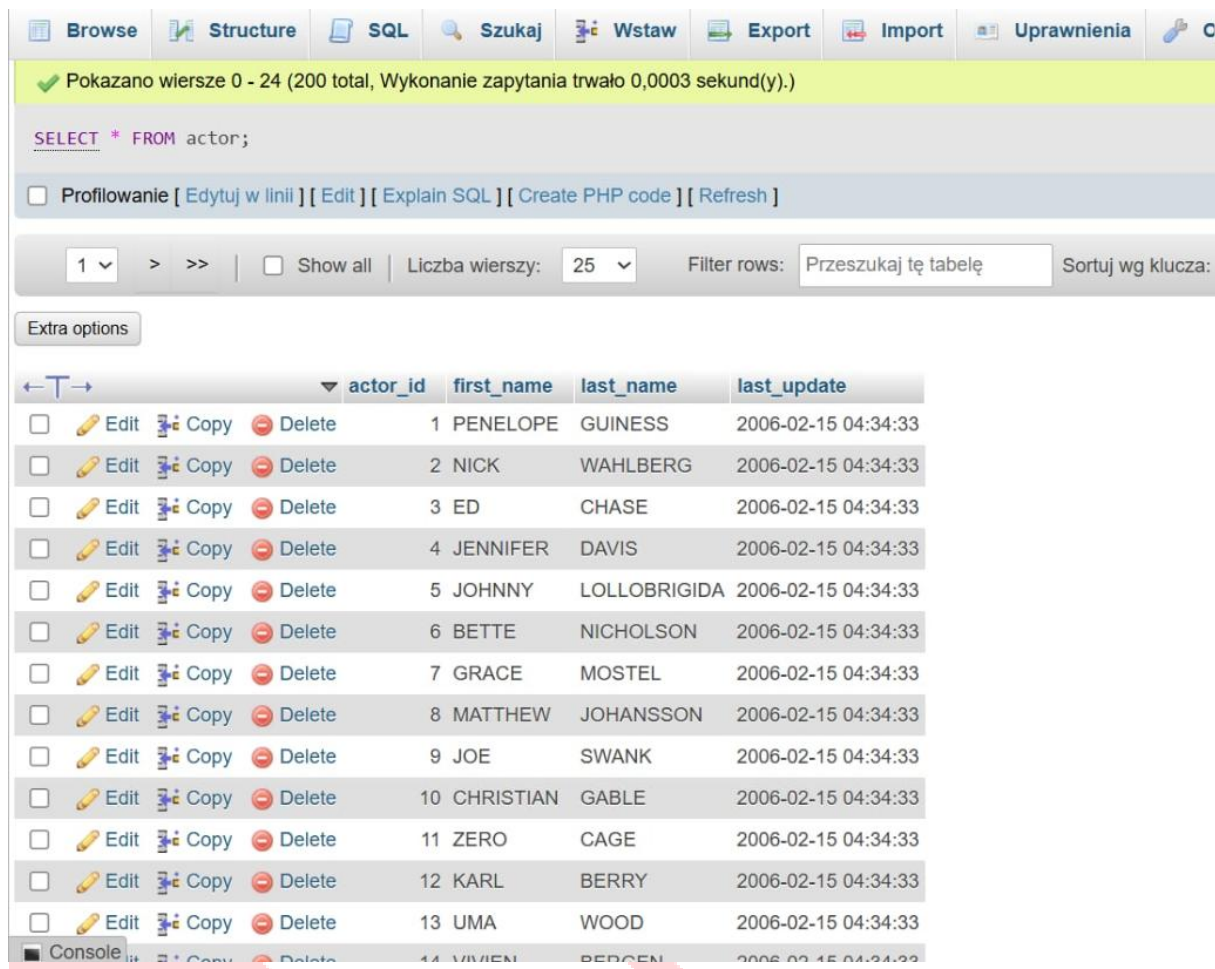
Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską





	actor_id	first_name	last_name	last_update
☐	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
☐	2	NICK	WAHLBERG	2006-02-15 04:34:33
☐	3	ED	CHASE	2006-02-15 04:34:33
☐	4	JENNIFER	DAVIS	2006-02-15 04:34:33
☐	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
☐	6	BETTE	NICHOLSON	2006-02-15 04:34:33
☐	7	GRACE	MOSTEL	2006-02-15 04:34:33
☐	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
☐	9	JOE	SWANK	2006-02-15 04:34:33
☐	10	CHRISTIAN	GABLE	2006-02-15 04:34:33
☐	11	ZERO	CAGE	2006-02-15 04:34:33
☐	12	KARL	BERRY	2006-02-15 04:34:33
☐	13	UMA	WOOD	2006-02-15 04:34:33
☐	14	VIVIEN	BERGEN	2006-02-15 04:34:33

Rysunek 13. Wynik zapytania SQL SELECT * FROM actor w phpMyAdmin

Wprowadzenie do SQL:

SQL (Structured Query Language) to standardowy język zapytań używany do komunikacji z systemami zarządzania bazami danych (DBMS). Dzięki SQL można:

- Tworzyć bazy danych i tabele,
- Wstawiać, aktualizować i usuwać dane,
- Wykonywać zapytania do pobierania danych,
- Zarządzać uprawnieniami użytkowników.

Dla porządku wyróżnia się dwie podstawowe grupy poleceń:

- **DML (Data Manipulation Language)** – manipulacja danymi,



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- **DDL (Data Definition Language)** – definiowanie struktury bazy.

DML – język manipulowania danymi:

DML skupia się na manipulowaniu danymi przechowywanymi w tabelach, a nie na samej strukturze bazy. Instrukcje DML są integralną częścią języka zapytań SQL, używanego do zarządzania relacyjnymi bazami danych. Jest to kluczowy mechanizm dla tworzenia aplikacji, ponieważ pozwala na precyzyjne zarządzanie danymi i zapewnienie ich jakości.

Kluczowe operacje DML:

SELECT – pobieranie danych

Klauzula SELECT, musi znaleźć się w każdym zapytaniu. Bez niej, nie wykonamy żadnej kwerendy, bo to właśnie tutaj, określamy w jaki sposób chcemy oglądać wynik zapytania. Pomimo że występuje zawsze jako pierwsza, jej treść jest przetwarzana jako jedna z ostatnich (jeśli nie korzystamy z sortowania zbioru wynikowego ORDER BY, będzie to faktycznie ostatni krok).

- (1) SELECT
- (2) FROM
- (3) WHERE
- (4) GROUP BY
- (5) HAVING
- (6) ORDER BY

SELECT to komenda używana do pobierania danych z bazy danych. Może zwracać jedną lub więcej kolumn oraz wszystkie lub tylko wybrane rekordy.

Składnia:

```
SELECT [kolumna1, kolumna2, ...]  
FROM [nazwa_tabeli]  
WHERE [warunki]  
ORDER BY [kolumna] [ASC|DESC];
```

SELECT: Określa kolumny, które chcesz pobrać z tabeli. kolumna1, kolumna2, ... – lista kolumn, które chcemy wybrać. Użycie * oznacza, że chcemy wyświetlić wszystkie kolumny.

FROM: Określa tabelę (lub tabele), z których chcesz pobrać dane. nazwa_tabeli – tabela, z której pobieramy dane. **WHERE:** Filtruje wiersze według podanego warunku.

ORDER BY: Sortuje wyniki według określonych kolumn, rosnąco (ASC) lub malejąco (DESC).



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Przykład:

```
SELECT * FROM actor;
```

To zapytanie pobiera wszystkie kolumny i wszystkie wiersze z tabeli actor.

```
SELECT first_name, last_name FROM actor;
```

Pobiera tylko kolumny first_name i last_name z tabeli actor.

INSERT – wstawianie danych:

Klauzula INSERT służy do dodawania nowych wierszy (rekordów) do istniejącej tabeli. Pozwala ona na wprowadzenie danych do tabeli, wypełniając je konkretnymi wartościami lub danymi pobranymi z innej tabeli, tym samym rozszerzając zbiory danych w bazie.

Jak działa INSERT:

1. **Wskazanie tabeli i kolumn:** Określasz, do której tabeli ma zostać dodany nowy rekord, oraz listę kolumn, do których chcesz wstawić dane.
2. **Podanie wartości:** Wskazujesz konkretne wartości, które mają być zapisane w tych kolumnach, lub używasz wartości z innej instrukcji SELECT

Przykład: użycia:

```
INSERT INTO actor (first_name, last_name)  
VALUES ('Jan', 'Kowalski');
```

UPDATE – aktualizacja danych

Instrukcja UPDATE w SQL służy do modyfikowania istniejących danych w tabeli. Pozwala ona na zmianę wartości w jednej lub wielu kolumnach dla wybranych rekordów (wierszy) tabeli, na podstawie określonych kryteriów.

Jak działa UPDATE?

1. Identyfikacja tabeli: Musisz wskazać, w której tabeli chcesz dokonać zmiany, używając nazwy tabeli.
2. Określenie zmian: Za pomocą klauzuli SET wskazujesz, które kolumny chcesz zaktualizować i jakie nowe wartości mają przyjąć.
3. Filtracja wierszy: Klauzula WHERE służy do określenia, które konkretne wiersze w tabeli mają zostać objęte zmianą. Jeśli pominiesz klauzulę WHERE, zmiana zostanie zastosowana do wszystkich rekordów w tabeli.

```
UPDATE actor  
SET last_name = 'Nowak'  
WHERE first_name = 'Jan';
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



DELETE – usuwanie danych

Instrukcja DELETE służy do usuwania jednego lub więcej rekordów (wierszy) z tabeli bazy danych, zachowując jednocześnie strukturę tabeli, jej schemat i ograniczenia. Można usunąć wszystkie rekordy, jeśli klauzula WHERE nie zostanie użyta, lub usunąć tylko te rekordy, które spełniają określony warunek.

```
DELETE FROM actor  
WHERE last_name = 'Nowak';
```

DDL – język definiowania danych:

DDL (Data Definition Language) skupia się na **strukturze bazy danych**, a nie na samych danych. Instrukcje DDL pozwalają tworzyć, modyfikować i usuwać obiekty bazy danych, takie jak tabele, widoki, indeksy czy schematy. Są to podstawowe polecenia używane przy projektowaniu i zarządzaniu architekturą bazy. Polecenia DDL wykonują się natychmiast i najczęściej nie można ich cofnąć (automatycznie wywołują COMMIT).

Kluczowe operacje DDL:

CREATE – tworzenie obiektów

Instrukcja CREATE służy do zakładania nowych obiektów w bazie danych, np. tabel, widoków, indeksów. Najczęściej używana jest do tworzenia tabel, które będą przechowywać dane.

```
CREATE TABLE actor (  
    actor_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50)  
);
```

ALTER – modyfikacja obiektów

Polecenie ALTER służy do zmiany już istniejących obiektów w bazie, np. dodania nowej kolumny do tabeli, zmiany typu danych kolumny czy dodania klucza obcego

```
ALTER TABLE actor  
ADD birth_date DATE;
```

DROP – usuwanie obiektów

Instrukcja DROP całkowicie usuwa obiekt z bazy danych, łącznie z jego strukturą i wszystkimi danymi.

```
DROP TABLE actor;
```

TRUNCATE – czyszczenie tabeli



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Polecenie TRUNCATE usuwa wszystkie dane z tabeli, ale **pozostawia jej strukturę** (czyli kolumny, typy danych i klucze). Jest szybsze od DELETE, bo nie zapisuje każdego usunięcia w logu transakcji.

TRUNCATE TABLE actor;

Przykład 1: Wyświetl strukturę tabeli actor.

DESCRIBE sakila.actor;

Wyjaśnienie:

1. DESCRIBE (albo krócej DESC) – to polecenie w MySQL, które służy do wyświetlania struktury tabeli.
2. sakila – to nazwa przykładowej bazy danych dostarczanej z MySQL. Zawiera ona dane o wypożyczalni filmów.
3. actor – to nazwa tabeli, która w tej bazie przechowuje informacje o aktorach.

Co zwróci polecenie:

Zapytanie SQL zostało wykonane pomyślnie.

```
DESCRIBE sakila.actor;
```

[Edytuj w linii] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
actor_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
first_name	varchar(45)	NO		NULL	
last_name	varchar(45)	NO	MUL	NULL	
last_update	timestamp	NO		current_timestamp()	on update current_timestamp()

Przykład 2: Wyświetl wszystkie kolumny i wiersze z actor.

SELECT * FROM sakila.actor;

Wyjaśnienie krok po kroku:

1. SELECT – służy do wybierania danych z tabeli.
2. * – oznacza „wszystkie kolumny”. W tym przypadku pobierzesz wszystkie kolumny tabeli actor.
3. FROM sakila.actor – wskazuje, że dane mają pochodzić z tabeli actor w bazie danych sakila.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0002 sekund(y).)

```
SELECT * FROM sakila.actor;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> ☐ Show all Liczba wierszy: 25 Filter rows:

Extra options

	actor_id	first_name	last_name	last_update
☐ Edit Copy Delete	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
☐ Edit Copy Delete	2	NICK	WAHLBERG	2006-02-15 04:34:33
☐ Edit Copy Delete	3	ED	CHASE	2006-02-15 04:34:33
☐ Edit Copy Delete	4	JENNIFER	DAVIS	2006-02-15 04:34:33
☐ Edit Copy Delete	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
☐ Edit Copy Delete	6	BETTE	NICHOLSON	2006-02-15 04:34:33
☐ Edit Copy Delete	7	GRACE	MOSTEL	2006-02-15 04:34:33
☐ Edit Copy Delete	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
☐ Edit Copy Delete	9	JOE	SWANK	2006-02-15 04:34:33
☐ Edit Copy Delete	10	CHRISTIAN	GABLE	2006-02-15 04:34:33

Przykład 3: Wyświetl tylko first_name i last_name aktorów.

```
SELECT first_name, last_name FROM sakila.actor;
```

Wyjaśnienie krok po kroku:

1. SELECT first_name, last_name – wybiera tylko kolumny first_name i last_name z tabeli.
 - o W przeciwieństwie do SELECT *, tutaj nie pobieramy wszystkich kolumn, tylko konkretne, które nas interesują.
2. FROM sakila.actor – wskazuje tabelę actor w bazie danych sakila.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0002 sekund(y).)

```
SELECT first_name, last_name FROM sakila.actor;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> | ☐ Show all | Liczba wierszy: 25 Filter rows:

Extra options

	first_name	last_name
☐ Edit Copy Delete	PENELOPE	GUINNESS
☐ Edit Copy Delete	NICK	WAHLBERG
☐ Edit Copy Delete	ED	CHASE
☐ Edit Copy Delete	JENNIFER	DAVIS
☐ Edit Copy Delete	JOHNNY	LOLLOBRIGIDA
☐ Edit Copy Delete	BETTE	NICHOLSON
☐ Edit Copy Delete	GRACE	MOSTEL
☐ Edit Copy Delete	MATTHEW	JOHANSSON
☐ Edit Copy Delete	JOE	SWANK
☐ Edit Copy Delete	CHRISTIAN	GABLE

W zapytaniach SQL często skracamy nazwy tabel i porządkujemy nagłówki wyników, nadając im tzw. aliasy. To po prostu krótsze, tymczasowe etykiety używane wyłącznie w trakcie wykonywania zapytania. Dzięki nim zapis staje się czytelniejszy, a przy złączeniach łatwiej uniknąć niejednoznaczności. Alias tabeli nadajemy po jej nazwie, np. FROM sakila.customer c, a potem odwołujemy się do kolumn przez skrót (c.first_name, c.last_name). Podobnie możemy nadać alias kolumnie, aby nadać sensowny nagłówek w raporcie: SUM(amount) AS suma albo f.title AS tytuł. Warto pamiętać, że w UNION nagłówki wyników biorą się z pierwszego SELECT-a, więc to tam opłaca się dodać aliasy (np. SELECT first_name AS imię FROM ... UNION SELECT first_name FROM ...). W złożonych zapytaniach — zwłaszcza z wieloma JOIN-ami lub podzapytaniem EXISTS — aliasy są praktycznie niezbędne: pozwalają jednoznacznie wskazać, z której tabeli pochodzi dana kolumna i znacząco skracają kod. Same aliasy nie wpływają na wydajność, ale poprawiają czytelność i ułatwiają utrzymanie zapytań.

Przykład 4: Wyświetl imię i nazwisko aktorów z aliasami imię, nazwisko.

```
SELECT first_name AS imię, last_name AS nazwisko
FROM sakila.actor;
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Wyjaśnienie krok po kroku:

1. `SELECT first_name AS imie` – pobiera kolumnę `first_name`, ale w wyniku nadaje jej nową nazwę: `imie`.
2. `last_name AS nazwisko` – pobiera kolumnę `last_name` i zmienia jej nagłówek w wynikach na `nazwisko`.
3. `FROM sakila.actor` – określa tabelę `actor` w bazie `sakila`.

Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0002 sekund(y).)

```
SELECT first_name AS imie, last_name AS nazwisko FROM sakila.actor;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> | ☐ Show all | Liczba wierszy: 25 | Filter rows:

Extra options

	imie	nazwisko
☐ Edit Copy Delete	PENELOPE	GUINNESS
☐ Edit Copy Delete	NICK	WAHLBERG
☐ Edit Copy Delete	ED	CHASE
☐ Edit Copy Delete	JENNIFER	DAVIS
☐ Edit Copy Delete	JOHNNY	LOLLOBRIGIDA
☐ Edit Copy Delete	BETTE	NICHOLSON
☐ Edit Copy Delete	GRACE	MOSTEL
☐ Edit Copy Delete	MATTHEW	JOHANSSON
☐ Edit Copy Delete	JOE	SWANK
☐ Edit Copy Delete	CHRISTIAN	GABLE

Przykład 5: Wyświetl pełne imię i nazwisko w jednej kolumnie `pelne_nazwisko`.

```
SELECT CONCAT(first_name, ' ', last_name) AS pelne_nazwisko
FROM sakila.actor;
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

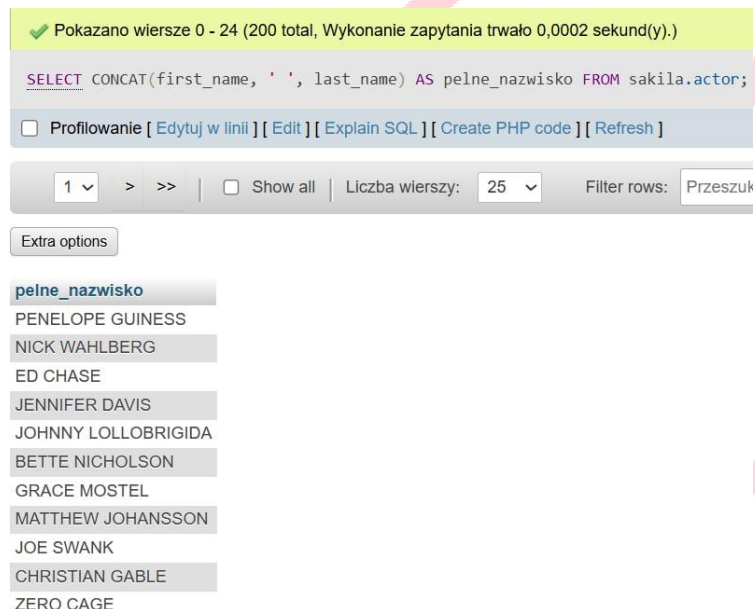
Dofinansowane przez
Unię Europejską



Wyjaśnienie krok po kroku:

1. `CONCAT(first_name, ' ', last_name)`
 - Funkcja `CONCAT` w SQL łączy kilka wartości w jeden ciąg znaków.
 - Tutaj łączymy imię (`first_name`), spację ' ' i nazwisko (`last_name`) w jedno pole tekstowe.
2. `AS pelne_nazwisko`
 - Nadaje wynikowej kolumnie nazwę `pelne_nazwisko`, dzięki czemu nagłówek tabeli jest czytelny.
3. `FROM sakila.actor`
 - Określa tabelę, z której pobieramy dane.

Co zwróci zapytanie:



✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0002 sekund(y).)

```
SELECT CONCAT(first_name, ' ', last_name) AS pelne_nazwisko FROM sakila.actor;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | ☐ Show all | Liczba wierszy: 25 | Filter rows: Przeszuk

Extra options

pelne_nazwisko
PENELOPE GUINNESS
NICK WAHLBERG
ED CHASE
JENNIFER DAVIS
JOHNNY LOLLOBRIGIDA
BETTE NICHOLSON
GRACE MOSTEL
MATTHEW JOHANSSON
JOE SWANK
CHRISTIAN GABLE
ZERO CAGE

Przykład 6: Wyświetl tytuły i długości filmów dłuższych niż 120 min.

```
SELECT title, length
FROM sakila.film
WHERE length > 120;
```

Wyjaśnienie krok po kroku:

1. `SELECT title, length`
 - Wybiera kolumny `title` (tytuł filmu) i `length` (długość filmu w minutach).



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



2. FROM sakila.film

- Określa tabelę film w bazie sakila.

3. WHERE length > 120

- Filtruje dane – w wyniku pojawiają się tylko filmy dłuższe niż 120 minut.
- > oznacza „większe niż”. Można też używać <, >=, <=, <> itd.

Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (457 total, Wykonanie zapytania trwało 0,0010 sekund(y).)

```
SELECT title, length FROM sakila.film WHERE length > 120;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | ☐ Show all | Liczba wierszy: 25 Filter rows:

Extra options

	title	length
☐ Edit Copy Delete	AFRICAN EGG	130
☐ Edit Copy Delete	AGENT TRUMAN	169
☐ Edit Copy Delete	ALAMO VIDEOTAPE	126
☐ Edit Copy Delete	ALASKA PHANTOM	136
☐ Edit Copy Delete	ALI FOREVER	150
☐ Edit Copy Delete	ALLEY EVOLUTION	180
☐ Edit Copy Delete	AMERICAN CIRCUS	129
☐ Edit Copy Delete	ANALYZE HOOSIERS	181
☐ Edit Copy Delete	ANONYMOUS HUMAN	179
☐ Edit Copy Delete	ANTITRUST TOMATOES	168

Przykład 7: Wyświetl first_name, last_name, actor_id i posortuj rosnąco po actor_id, a w ramach tego malejąco po last_name.

```
SELECT first_name, last_name, actor_id
FROM sakila.actor
ORDER BY actor_id ASC, last_name DESC;
```

Wyjaśnienie krok po kroku:



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



1. `SELECT first_name, last_name, actor_id`
 - Wybiera kolumny: imię, nazwisko i identyfikator aktora.
2. `FROM sakila.actor`
 - Określa tabelę actor w bazie sakila.
3. `ORDER BY actor_id ASC, last_name DESC`
 - Sortuje wyniki według kilku kryteriów:
 - `actor_id ASC` – rosnąco po identyfikatorze aktora.
 - `last_name DESC` – malejąco po nazwisku (dla rekordów o tym samym `actor_id`).
 - `ASC` oznacza ascending (rosnąco), a `DESC` descending (malejąco).
 - Można sortować po jednej lub wielu kolumnach w różnym porządku.

Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0003 sekund(y).) [actor_id: 1... - 25...] [last_name: GUINNESS... - BLOOM...]

```
SELECT first_name, last_name, actor_id FROM sakila.actor ORDER BY actor_id ASC, last_name DESC;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> ☐ Show all | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Extra options

	first_name	last_name	actor_id
☐ Edit Copy Delete	PENELOPE	GUINNESS	1
☐ Edit Copy Delete	NICK	WAHLBERG	2
☐ Edit Copy Delete	ED	CHASE	3
☐ Edit Copy Delete	JENNIFER	DAVIS	4
☐ Edit Copy Delete	JOHNNY	LOLLOBRIGIDA	5
☐ Edit Copy Delete	BETTE	NICHOLSON	6
☐ Edit Copy Delete	GRACE	MOSTEL	7
☐ Edit Copy Delete	MATTHEW	JOHANSSON	8
☐ Edit Copy Delete	JOE	SWANK	9
☐ Edit Copy Delete	CHRISTIAN	GABLE	10

Przykład 8: Wyświetl kraje rosnąco alfabetycznie.

```
SELECT country
FROM sakila.country
ORDER BY country ASC;
```

Wyjaśnienie krok po kroku:



Fundusze Europejskie
dla Rozwoju Społecznego



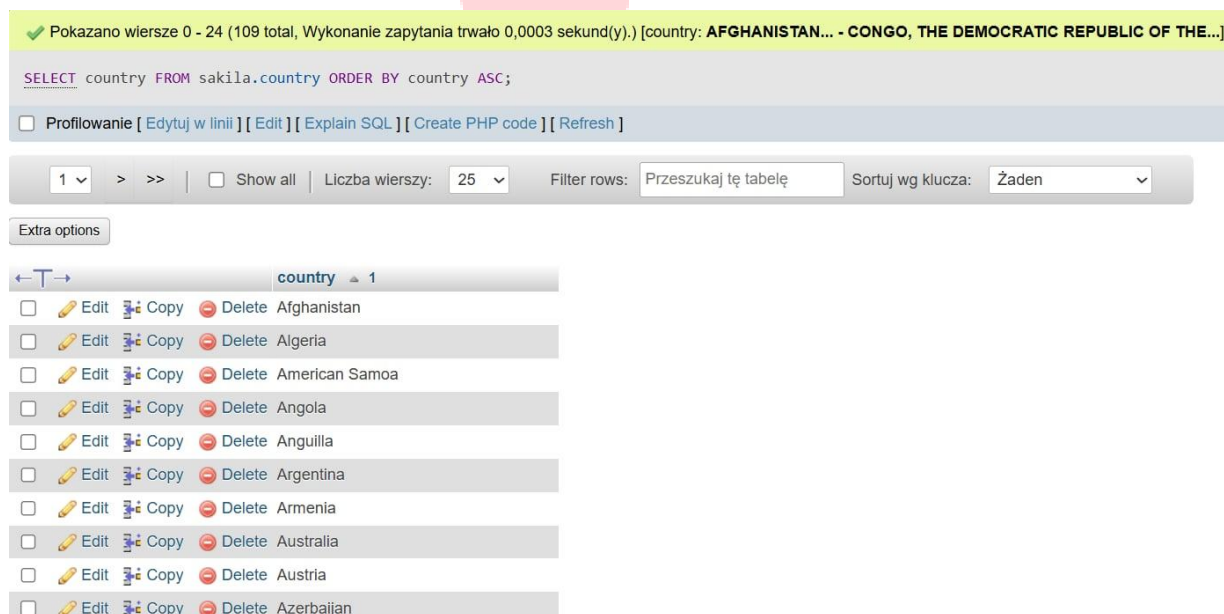
Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



1. SELECT country
 - Pobiera tylko kolumnę country z tabeli country.
2. FROM sakila.country
 - Określa tabelę country w bazie sakila.
3. ORDER BY country ASC
 - Sortuje wyniki rosnąco (ASC), czyli alfabetycznie od A do Z.
 - ASC jest domyślny, więc można go pominąć: ORDER BY country; da taki sam efekt.

Co zwróci zapytanie:



✓ Pokazano wiersze 0 - 24 (109 total, Wykonanie zapytania trwało 0,0003 sekund(y).) [country: AFGHANISTAN... - CONGO, THE DEMOCRATIC REPUBLIC OF THE...]

SELECT country FROM sakila.country ORDER BY country ASC;

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | ☐ Show all | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Extra options

country 1

☐	Edit	Copy	Delete	Afghanistan
☐	Edit	Copy	Delete	Algeria
☐	Edit	Copy	Delete	American Samoa
☐	Edit	Copy	Delete	Angola
☐	Edit	Copy	Delete	Anguilla
☐	Edit	Copy	Delete	Argentina
☐	Edit	Copy	Delete	Armenia
☐	Edit	Copy	Delete	Australia
☐	Edit	Copy	Delete	Austria
☐	Edit	Copy	Delete	Azerbaijan

Przykład 9: Wyświetl last_name, last_update, actor_id i actor_id + 100 jako id_po_zmianie.

```
SELECT last_name AS nazwisko,
       last_update AS data_aktualizacji,
       actor_id AS id,
       actor_id + 100 AS id_po_zmianie
FROM sakila.aktor;
```

Wyjaśnienie krok po kroku:

1. last_name AS nazwisko
 - Pobiera kolumnę last_name i nadaje jej nagłówek nazwisko.
2. last_update AS data_aktualizacji



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- Pobiera kolumnę last_update i nadaje jej nagłówek data_aktualizacji.
- 3. actor_id AS id
 - Pobiera kolumnę actor_id i nadaje jej nagłówek id.
- 4. actor_id + 100 AS id_po_zmianie
 - Wykonuje prostą operację matematyczną – dodaje 100 do wartości actor_id i zapisuje wynik w kolumnie o nagłówku id_po_zmianie.
- 5. FROM sakila.actor
 - Określa tabelę actor w bazie sakila.

Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 24 (200 total, Wykonanie zapytania trwało 0,0002 sekund(y).)

```
SELECT last_name AS nazwisko, last_update AS data_aktualizacji, actor_id AS id, actor_id + 100 AS id_po_zmianie FROM sakila.actor;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> ☐ Show all Liczba wierszy: 25 Filter rows: Przeszukaj tę tabelę Sortuj wg klucza: Żaden

Extra options

		nazwisko	data_aktualizacji	id	id_po_zmianie
☐	Edit Copy Delete	GUINNESS	2006-02-15 04:34:33	1	101
☐	Edit Copy Delete	WAHLBERG	2006-02-15 04:34:33	2	102
☐	Edit Copy Delete	CHASE	2006-02-15 04:34:33	3	103
☐	Edit Copy Delete	DAVIS	2006-02-15 04:34:33	4	104
☐	Edit Copy Delete	LOLLOBRIGIDA	2006-02-15 04:34:33	5	105
☐	Edit Copy Delete	NICHOLSON	2006-02-15 04:34:33	6	106
☐	Edit Copy Delete	MOSTEL	2006-02-15 04:34:33	7	107
☐	Edit Copy Delete	JOHANSSON	2006-02-15 04:34:33	8	108
☐	Edit Copy Delete	SWANK	2006-02-15 04:34:33	9	109
☐	Edit Copy Delete	GABLE	2006-02-15 04:34:33	10	110

Przykład 10: Pokaż 10 pierwszych aktorów wg actor_id.

```
SELECT *
FROM sakila.actor
ORDER BY actor_id
LIMIT 10;
```

Wyjaśnienie krok po kroku:

1. SELECT *
 - Pobiera wszystkie kolumny z tabeli actor.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



2. FROM sakila.actor
 - Określa tabelę actor w bazie sakila.
3. ORDER BY actor_id
 - Sortuje wyniki rosnąco według kolumny actor_id.
 - ASC jest domyślne, więc nie trzeba go pisać.
4. LIMIT 10
 - Ogranicza wynik do pierwszych 10 rekordów po posortowaniu.
 - Przydatne przy dużych tabelach lub do podglądu danych.

Co zwróci zapytanie:

✓ Pokazano wiersze 0 - 9 (10 total, Wykonanie zapytania trwało 0,0003 sekund(y).) [actor_id: 1... - 10...]

```
SELECT * FROM sakila.actor ORDER BY actor_id LIMIT 10;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

	actor_id	first_name	last_name	last_update
☐ Edit Copy Delete	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
☐ Edit Copy Delete	2	NICK	WAHLBERG	2006-02-15 04:34:33
☐ Edit Copy Delete	3	ED	CHASE	2006-02-15 04:34:33
☐ Edit Copy Delete	4	JENNIFER	DAVIS	2006-02-15 04:34:33
☐ Edit Copy Delete	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
☐ Edit Copy Delete	6	BETTE	NICHOLSON	2006-02-15 04:34:33
☐ Edit Copy Delete	7	GRACE	MOSTEL	2006-02-15 04:34:33
☐ Edit Copy Delete	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
☐ Edit Copy Delete	9	JOE	SWANK	2006-02-15 04:34:33
☐ Edit Copy Delete	10	CHRISTIAN	GABLE	2006-02-15 04:34:33

Przykład 11: Pomiń 20 pierwszych aktorów i pokaż kolejnych 10.

```
SELECT *
FROM sakila.actor
ORDER BY actor_id
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



LIMIT 10 OFFSET 20;

Wyjaśnienie krok po kroku:

1. SELECT *
 - Pobiera wszystkie kolumny z tabeli actor.
2. FROM sakila.actor
 - Określa tabelę actor w bazie sakila.
3. ORDER BY actor_id
 - Sortuje wyniki rosnąco według kolumny actor_id.
4. LIMIT 10 OFFSET 20
 - LIMIT 10 – pobiera maksymalnie 10 rekordów.
 - OFFSET 20 – pomija pierwsze 20 rekordów w wynikach.
 - Innymi słowy, pobieramy rekordy od 21 do 30 w kolejności rosnącej według actor_id.

Co zwróci zapytanie:

✓ Pokazano wiersze 20 - 29 (10 total, Wykonanie zapytania trwało 0,0003 sekund(y).) [actor_id: 21... - 30...]

```
SELECT * FROM sakila.actor ORDER BY actor_id LIMIT 10 OFFSET 20;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

	actor_id	first_name	last_name	last_update
☐ Edit Copy Delete	21	KIRSTEN	PALTROW	2006-02-15 04:34:33
☐ Edit Copy Delete	22	ELVIS	MARX	2006-02-15 04:34:33
☐ Edit Copy Delete	23	SANDRA	KILMER	2006-02-15 04:34:33
☐ Edit Copy Delete	24	CAMERON	STREEP	2006-02-15 04:34:33
☐ Edit Copy Delete	25	KEVIN	BLOOM	2006-02-15 04:34:33
☐ Edit Copy Delete	26	RIP	CRAWFORD	2006-02-15 04:34:33
☐ Edit Copy Delete	27	JULIA	MCQUEEN	2006-02-15 04:34:33
☐ Edit Copy Delete	28	WOODY	HOFFMAN	2006-02-15 04:34:33
☐ Edit Copy Delete	29	ALEC	WAYNE	2006-02-15 04:34:33
☐ Edit Copy Delete	30	SANDRA	PECK	2006-02-15 04:34:33



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Przykład 12: Dodaj aktora Jan Kowalski.

```
INSERT INTO sakila.actor (first_name, last_name)
VALUES ('Jan', 'Kowalski');
```

Wyjaśnienie krok po kroku:

1. **INSERT INTO sakila.actor (first_name, last_name)**

- Określa tabelę actor, do której chcemy dodać rekord.
- W nawiasach podajemy **kolumny**, które chcemy wypełnić (tutaj first_name i last_name).

2. **VALUES ('Jan', 'Kowalski')**

- Podaje wartości dla wybranych kolumn:
 - first_name = 'Jan'
 - last_name = 'Kowalski'

Co zrobi zapytanie:

- Dodaje nowy rekord do tabeli actor.
- Kolumna actor_id zostanie automatycznie wypełniona (bo jest auto_increment).
- Kolumna last_update zostanie automatycznie ustawiona na aktualną datę i czas.

Przykład 13: Zmień nazwisko Jan Kowalski na Kowalski-Nowy.

```
UPDATE sakila.actor
SET last_name = 'Kowalski-Nowy'
WHERE first_name = 'Jan' AND last_name = 'Kowalski';
```

Wyjaśnienie krok po kroku:

1. **UPDATE sakila.actor**

- Określa tabelę actor, w której chcemy zmienić dane.

2. **SET last_name = 'Kowalski-Nowy'**

- Określa nową wartość dla kolumny last_name.
- W tym przypadku zmieniamy nazwisko z Kowalski na Kowalski-Nowy.

3. **WHERE first_name = 'Jan' AND last_name = 'Kowalski'**



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- Filtruje rekordy, które mają być zmienione.
- Aktualizacja dotyczy tylko aktora o imieniu Jan i nazwisku Kowalski.
- Bardzo ważne: bez WHERE wszystkie rekordy w tabeli zostałyby zmienione!

Co robi zapytanie:

Znajdzie wszystkie rekordy spełniające warunek (`first_name = 'Jan'` i `last_name = 'Kowalski'`) i zmieni ich nazwisko na Kowalski-Nowy.

Przykład 14: Usuń aktora Jan Kowalski-Nowy.

```
DELETE FROM sakila.actor  
WHERE first_name = 'Jan' AND last_name = 'Kowalski-Nowy';
```

Wyjaśnienie krok po kroku:

1. `DELETE FROM sakila.actor`
 - Określa tabelę `actor`, z której chcemy usunąć rekordy.
2. `WHERE first_name = 'Jan' AND last_name = 'Kowalski-Nowy'`
 - Filtruje rekordy do usunięcia.
 - Usunięty zostanie tylko aktor o imieniu Jan i nazwisku Kowalski-Nowy.
 - Bardzo ważne: bez WHERE wszystkie rekordy w tabeli zostałyby usunięte!

Co robi zapytanie:

- Znajdzie rekordy spełniające warunek i je usunie.

Przykład 15: Utwórz w bazie danych `sakila` nową tabelę o nazwie `favorite_actor`, która będzie przechowywać informacje o ulubionych aktorach użytkowników. Tabela powinna zawierać:

- kolumnę `favorite_id` typu `INT` z automatycznym zwiększaniem wartości (`AUTO_INCREMENT`), pełniącą rolę klucza głównego;
- kolumnę `actor_id` typu `INT`, która nie może być pusta i powinna stanowić klucz obcy wskazujący na kolumnę `actor_id` w tabeli `actor`;
- kolumnę `comment` typu `VARCHAR(100)`, w której można umieścić krótki komentarz.

```
CREATE TABLE sakila.favorite_actor (
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



```
favorite_id INT AUTO_INCREMENT PRIMARY KEY,  
actor_id INT NOT NULL,  
comment VARCHAR(100),  
FOREIGN KEY (actor_id) REFERENCES sakila.actor(actor_id)  
);
```

Wyjaśnienie krok po kroku:

1. CREATE TABLE sakila.favorite_actor
 - Tworzy nową tabelę favorite_actor w bazie sakila.
2. favorite_id INT AUTO_INCREMENT PRIMARY KEY
 - favorite_id – kolumna typu liczbowego (INT).
 - AUTO_INCREMENT – numer rośnie automatycznie przy dodawaniu kolejnych rekordów.
 - PRIMARY KEY – oznacza, że jest to klucz główny tabeli (unikalny identyfikator rekordu).
3. actor_id INT NOT NULL
 - Kolumna do przechowywania identyfikatora aktora.
 - NOT NULL – nie może być pusta.
4. comment VARCHAR(100)
 - Kolumna tekstowa, w której można dodać komentarz (maks. 100 znaków).
5. FOREIGN KEY (actor_id) REFERENCES sakila.actor(actor_id)
 - Tworzy klucz obcy, który wskazuje na kolumnę actor_id w tabeli actor.
 - Zapewnia integralność danych – nie można dodać do favorite_actor aktora, którego nie ma w tabeli actor.

Co robi zapytanie:

- Tworzy tabelę do przechowywania ulubionych aktorów z komentarzami.
- Każdy rekord ma swój unikalny identyfikator (favorite_id) i odnosi się do istniejącego aktora (actor_id).

Przykład 16: Dodaj kolumnę date_added z domyślną bieżącą datą/czasem.

```
ALTER TABLE sakila.favorite_actor  
ADD date_added DATETIME DEFAULT NOW();
```



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Wyjaśnienie krok po kroku:

1. ALTER TABLE sakila.favorite_actor
 - Określa tabelę favorite_actor, którą chcemy zmodyfikować.
2. ADD date_added DATETIME
 - Dodaje nową kolumnę date_added typu DATETIME do tabeli.
3. DEFAULT NOW()
 - Ustawia domyślną wartość kolumny na aktualną datę i czas w momencie wstawiania rekordu.
 - Dzięki temu, jeśli nie podamy wartości dla date_added przy wstawianiu rekordu, system automatycznie zapisze datę i godzinę dodania.

Co zrobi zapytanie:

- Tabela favorite_actor zyska nową kolumnę date_added.
- Każdy nowo dodany rekord automatycznie zapisze datę i godzinę utworzenia, jeśli nie zostanie podana wartość ręcznie.

Przykład 17: Usuń tabelę favorite_actor.

```
DROP TABLE sakila.favorite_actor;
```

Wyjaśnienie krok po kroku:

1. DROP TABLE sakila.favorite_actor
 - Usuwa całkowicie tabelę favorite_actor z bazy sakila.
 - Tabela wraz ze wszystkimi rekordami, strukturą kolumn, kluczami i indeksami **zostaje trwale usunięta**.

Ważne informacje:

- Operacja jest **nieodwracalna** – po wykonaniu zapytania dane zostają utracone.
- Należy upewnić się, że naprawdę chcemy usunąć tabelę, szczególnie jeśli zawiera ważne dane.
- Jeśli tabela była powiązana kluczem obcym z innymi tabelami, MySQL może zgłosić błąd lub automatycznie usunąć zależności w zależności od ustawień (CASCADE).

Ćwiczenia samodzielne



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



1. Wyświetl wszystkie kolumny z sakila.category.
2. Wyświetl first_name, last_name z sakila.actor.
3. Wyświetl first_name jako imię i last_name jako nazwisko (aliasy).
4. Wyświetl title i release_year z sakila.film.
5. Posortuj aktorów rosnąco po last_name.
6. Posortuj aktorów malejąco po last_name.
7. Posortuj klientów rosnąco po last_name, a następnie po first_name.
8. Wyświetl klientów ze sklepu o store_id = 2, posortowanych malejąco po create_date.
9. Wyświetl tytuły i długości filmów, które mają length > 120.
10. Wyświetl klientów z create_date >= '2006-01-01', posortowanych po create_date.
11. Wyświetl kraje (country) rosnąco alfabetycznie.
12. Wyświetl filmy (title, rental_rate) oraz dodatkową kolumnę rental_rate * 1.2 jako „podwyżka_20”.
13. Wyświetl aktorów o actor_id < 50 (pokaż first_name, last_name, actor_id).
14. Wyświetl title, release_year filmów, których rok produkcji jest < 2000 lub > 2002.
15. Wyświetl 10 pierwszych wierszy z sakila.actor wg actor_id.
16. Pomiń 20 pierwszych aktorów i pokaż kolejnych 10 (LIMIT/OFFSET).
17. Wyświetl first_name, last_name, actor_id, sortując rosnąco po actor_id, a w ramach tego malejąco po last_name.
18. Wyświetl last_name, last_update, actor_id oraz actor_id + 100 jako id_po_zmianie.
19. Wyświetl wszystkie kolumny z sakila.category dla wierszy z category_id > 5.
20. Wyświetl wszystkie kolumny z sakila.country dla wierszy z country_id > 50.
21. Wstaw nowego aktora ('Anna', 'Nowak') do sakila.actor.
22. Wstaw dwóch aktorów jednocześnie (dowolne poprawne dane).
23. Zmień nazwisko wybranego aktora (podaj warunek po actor_id).
24. Dodaj sufiks '*' do last_name aktorów o actor_id < 10.
25. Usuń aktora o podanym actor_id (upewnij się, że istnieje).
26. twórz tabelę p3_notes z kolumnami: note_id INT AUTO_INCREMENT (PK), note_text VARCHAR(200) NOT NULL.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



27. Dodaj do p3_notes kolumnę created_at typu DATETIME z domyślną wartością bieżącą.
28. Zmień typ note_text na VARCHAR(255).
29. Zmień nazwę tabeli p3_notes na p3_notes_archive.
30. Usuń tabelę p3_notes_archive.

Pytania pomocnicze

1. Czym jest baza danych i do czego służy?
2. Jakie są podstawowe rodzaje baz danych i czym się różnią?
3. Dlaczego SQL jest ważny w pracy z relacyjnymi bazami danych?
4. Do czego służy przykładowa baza Sakila i co zawiera?
5. Jakie kroki należy wykonać, aby zaimportować bazę Sakila w phpMyAdmin?
6. Na czym polegają operacje DML i DDL w SQL?
7. Jakie są przykłady poleceń DML i do czego się ich używa?
8. Jakie są przykłady poleceń DDL i jakie mają zastosowanie?
9. Czym różnią się instrukcje DELETE i TRUNCATE?
10. Dlaczego relacje między tabelami są istotne w relacyjnych bazach danych?

Podsumowanie

Bazy danych to uporządkowane zbiory informacji, które umożliwiają ich przechowywanie, organizację i szybki dostęp. Kluczową rolę w pracy z nimi odgrywa język SQL, pozwalający tworzyć struktury baz, wstawiać, modyfikować i usuwać dane oraz pobierać je zgodnie z określonymi kryteriami. Przykładowa baza Sakila, symulująca działanie wypożyczalni filmów, stanowi narzędzie do nauki i ćwiczeń, dzięki któremu można praktycznie poznać podstawowe operacje DML i DDL. Zrozumienie tych mechanizmów pozwala lepiej opanować relacyjny model baz danych i stanowi fundament dalszej pracy z systemami informatycznymi.

Literatura

Karwin, Bill, i in. Antywzorce języka SQL : jak uniknąć pułapek podczas programowania baz danych. Wyd. Helion, 2012.

<https://dev.mysql.com/doc/sakila/en/>

<https://www.w3schools.com/sql/>



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

