Materiały do przedmiotu:

Bazy danych

Laboratorium nr 1:

Wprowadzenie do podstawowych zasad działania i organizacji baz danych relacyjnych. Definiowanie encji i związków między nimi

Autor:

Albert Rachwał

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

Spis treści

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

Bazy danych relacyjne stanowią podstawę większości współczesnych systemów informatycznych. Pozwalają na uporządkowane przechowywanie informacji w tabelach, które pozostają ze sobą w określonych relacjach. Dzięki temu możliwe jest efektywne wyszukiwanie, aktualizowanie i kontrolowanie spójności danych. Wprowadzenie do zasad działania baz relacyjnych obejmuje zrozumienie pojęcia encji, atrybutów i związków, które stanowią fundament projektowania i organizacji danych.

Cel ćwiczenia/laboratorium

- Zrozumienie, czym jest baza danych i jej rola w systemach informatycznych.
- Poznanie pojęcia encji jako reprezentacji obiektów świata rzeczywistego.
- Rozpoznawanie atrybutów i ich znaczenia w opisie encji.
- Wyjaśnienie pojęcia relacji między encjami.
- Zrozumienie roli kluczy głównych i obcych.
- Umiejętność wskazania przykładu bazy relacyjnej z życia codziennego.
- Rozumienie różnicy między danymi a informacją.
- Poznanie podstawowych zasad organizacji danych w tabelach.
- Wyjaśnienie roli modelu relacyjnego w informatyce.
- Ćwiczenie identyfikacji encji i relacji w prostych scenariuszach.
- Zrozumienie pojęcia krotki i domeny w kontekście relacyjnych baz danych.
- Omówienie problemu redundancji danych i jej konsekwencji.
- Uświadomienie znaczenia spójności danych.
- Zrozumienie pojęcia schematu bazy danych.
- Wprowadzenie do terminologii używanej w relacyjnych bazach danych.
- Poznanie środowiska pracy phpMyAdmin.
- Poznanie zasady działanie indeksów.

Instrukcja laboratoryjna/ćwiczeniowa

Charakter projektowy przedmiotu

Przedmiot Bazy danych realizowany jest w formie projektowej. Oznacza to, że w trakcie semestru studenci pracują nad własnym projektem, rozwijając go o kolejne elementy i funkcjonalności zgodnie z planem zajęć oraz wskazówkami

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

prowadzącego. Projekt pełni rolę praktycznego narzędzia dydaktycznego – każda nowo poznana koncepcja teoretyczna zostaje wdrożona w postaci rozwiązania bazodanowego, które następnie staje się częścią większej całości.

Podstawą rozpoczęcia pracy projektowej jest wybór przez grupę – składającą się od jednego do trzech studentów – określonego fragmentu rzeczywistości, który będzie odwzorowany w formie bazy danych. Wybrany obszar stanie się punktem wyjścia do stworzenia modelu oraz zaprezentowania możliwości systemu zaprojektowanego przez studentów.

Na tym etapie warto podkreślić wagę kamieni milowych, czyli elementów, które każdy projekt powinien zawierać. Ich brak, niestaranność wykonania, błędy lub nadmierne uproszczenia mogą stanowić podstawę do obniżenia oceny końcowej.

Pomocne dla studentów może być przyjęcie metafory, w której prowadzący pełni rolę klienta „zlecającego" realizację projektu. Grupa projektowa, poszerzając swoją wiedzę w trakcie semestru, prezentuje kolejne postępy, a tym samym potwierdza zdobyte umiejętności i weryfikuje poprawność zaproponowanego rozwiązania.

Celem grupy jest udowodnienie, że przygotowane rozwiązanie:

1. zostało wykonane starannie,
2. spełnia standardy projektowania baz danych,
3. umożliwia zastosowanie szerokiego wachlarza operacji w języku SQL,
4. znajduje praktyczne zastosowanie w wybranym obszarze.

Wymagania końcowego projektu zaliczeniowego

Każdy projekt powinien zawierać następujące elementy:

1. Motywację wykonania bazy danych – uzasadnienie, dlaczego warto zrealizować dany projekt i jakie problemy ma on rozwiązać.
2. Opis słowny wybranej rzeczywistości (np. uczelnia, sklep, szpital).
3. Wyodrębnienie niezbędnych tabel. Ważne jest, aby nazwy tabel zostały wyróżnione pogrubieniem w opisie słownym z punktu drugiego. Odzwierciedla to naturalny proces, w którym baza danych powstaje jako odpowiedź na oczekiwania klienta lub zleceniodawcy, po zebraniu informacji o potrzebach i specyfice organizacji.
4. Określenie atrybutów encji oraz relacji między encjami.

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

5. Opracowanie pełnego diagramu ERD dla bazy danych.
6. Wypełnienie bazy danymi syntetycznymi. Można w tym celu wykorzystać specjalne generatory, np. bibliotekę faker w języku Python lub bibliotekę charlatan w języku R.
7. Opracowanie i prezentacja zróżnicowanych zapytań SELECT, które pozwolą pobierać dane z bazy, wykorzystując możliwości języka SQL omawiane na zajęciach.
8. Opracowanie i prezentacja zapytań modyfikujących dane i strukturę bazy, takich jak edycja, usuwanie rekordów czy zmiany w schemacie tabel.
9. Opracowanie i prezentacja widoków.
10. Opracowanie i prezentacja wyzwalaczy (triggerów).
11. Opracowanie i prezentacja procedur składowanych.
12. Prezentacja procesu zarządzania użytkownikami w systemie bazodanowym, obejmująca tworzenie kont z różnymi zestawami uprawnień oraz testowanie ich działania poprzez uwierzytelnianie i wykonywanie przykładowych operacji.
13. Prezentacja procesu tworzenia kopii zapasowej, importu i eksportu bazy danych.
14. Opracowanie i prezentacja uproszczonej wersji analogicznego systemu w dokumentowej bazie danych. Należy przedstawić różnice między systemem dokumentowym a relacyjnym oraz wskazać, jak baza dokumentowa może zostać wykorzystana – jako uzupełnienie istniejącego rozwiązania o nowe funkcjonalności lub jako alternatywa, w której projekt zrealizowany w bazie relacyjnej zostaje przełożony na język baz dokumentowych.

Czym są i do czego służą sytemy bazodanowe

Baza danych to zorganizowany zbiór informacji, który umożliwia ich łatwe przechowywanie, wyszukiwanie i modyfikację. Współcześnie korzystamy z systemów zarządzania bazami danych (DBMS – Database Management Systems), które pełnią rolę „pośrednika” między użytkownikiem a danymi. Dzięki DBMS można tworzyć tabele, wprowadzać rekordy, wyszukiwać informacje przy pomocy języka SQL, a także kontrolować uprawnienia i bezpieczeństwo. Już od pierwszego kontaktu ważne jest, aby zrozumieć, że baza danych nie jest tylko prostym magazynem informacji, ale narzędziem wspierającym logikę działania aplikacji i organizacji. W trakcie pracy na zajęciach będziemy korzystać zarówno z interfejsu przeglądarkowego (phpMyAdmin), który pozwala na szybkie i intuicyjne tworzenie tabel i relacji, jak i z

bardziej zaawansowanego środowiska MySQL Workbench, które umożliwia pisanie zapytań w SQL oraz projektowanie diagramów baz danych.

Encja to pojęcie teoretyczne opisujące obiekt, który występuje w rzeczywistości i który chcemy odzwierciedlić w bazie danych. Encją może być na przykład student, książka, zamówienie czy pracownik. Każda encja posiada zestaw cech, które ją opisują – i to właśnie one stają się atrybutami w bazie danych.

Tabela jest praktycznym odpowiednikiem encji w bazie danych relacyjnej. Dane przechowywane są w wierszach (rekordach), a kolumny odpowiadają atrybutom encji. Na przykład tabela Students może zawierać kolumny takie jak id_student, imie, nazwisko, kierunek. Tabela jest więc miejscem, w którym encja znajduje swoje odzwierciedlenie w systemie komputerowym.

Atrybut to pojedyncza cecha encji, przechowywana jako kolumna w tabeli. Dla encji „Student” mogą to być imię, nazwisko czy numer albumu. Każdy atrybut ma swój typ danych (np. liczba całkowita, tekst, data), który określa, jakie wartości można w nim zapisać.

Relacje między między encjami

Relacja określa sposób powiązania encji między sobą. W świecie rzeczywistym obiekty często występują w powiązaniach – pracownik należy do działu, student zapisuje się na kurs, klient składa zamówienie. W modelu relacyjnym takie związki odzwierciedlamy przez klucze główne i obce. Klucz główny jednoznacznie identyfikuje każdy rekord w tabeli, a klucz obcy wskazuje na rekord w innej tabeli, tworząc w ten sposób powiązanie.

Wyróżniamy trzy podstawowe rodzaje relacji:

1. Relacja jeden do jednego (1:1) – każdemu rekordowi w jednej tabeli odpowiada najwyżej jeden rekord w drugiej tabeli. Na przykład każdemu pracownikowi przypisany jest dokładnie jeden identyfikator służbowy.
2. Relacja jeden do wielu (1:N) – jednemu rekordowi w tabeli odpowiada wiele rekordów w innej tabeli. Przykład: jeden autor może napisać wiele książek, ale każda książka ma tylko jednego głównego autora.
3. Relacja wiele do wielu (N:M) – rekordy z jednej tabeli mogą odpowiadać wielu rekordom w drugiej i odwrotnie. Przykład: studenci mogą zapisywać się na

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

wiele kursów, a każdy kurs może być realizowany przez wielu studentów. W relacyjnej bazie danych relacje tego typu wymagają utworzenia dodatkowej tabeli pośredniczącej, w której przechowywane są pary powiązanych identyfikatorów.

Rodzaje kluczy w bazach danych

Klucze w relacyjnych bazach danych to mechanizm pozwalający jednoznacznie identyfikować dane i kontrolować powiązania pomiędzy tabelami. Dzięki nim możemy mieć pewność, że w bazie nie pojawią się sprzeczne informacje oraz że odwołania pomiędzy tabelami zawsze prowadzą do istniejących rekordów.

Klucz główny (PRIMARY KEY) to najważniejszy klucz w każdej tabeli. Określa on atrybut (lub zestaw atrybutów), którego wartości muszą być unikalne i nie mogą być puste. Dzięki temu każdy rekord w tabeli można jednoznacznie zidentyfikować. Typowym przykładem klucza głównego jest numer PESEL w tabeli Pracownicy albo identyfikator studenta w tabeli Studenci. Jeżeli pojedyncza kolumna nie wystarcza do jednoznacznej identyfikacji, można zdefiniować klucz złożony – np. w tabeli zapisów na kursy kluczem może być kombinacja (id_studenta, id_kursu). Oprócz klucza głównego w tabeli mogą istnieć także inne unikalne zestawy atrybutów. Nazywamy je kluczami alternatywnymi. Przykładowo, w tabeli studenci oprócz id_studenta unikalny może być również adres e-mail – wówczas e-mail można potraktować jako klucz alternatywny. W praktyce często spotykamy się też z kluczami sztucznymi (surrogate keys). Są to specjalnie wygenerowane identyfikatory, takie jak licznik automatycznie zwiększający się przy każdym nowym wpisie (AUTO_INCREMENT w MySQL), czy identyfikatory UUID. Klucz sztuczny sam w sobie nie niesie treści biznesowej, ale ułatwia obsługę bazy i zapewnia jednoznaczność.

Drugim ważnym pojęciem jest **klucz obcy** (FOREIGN KEY). To atrybut w jednej tabeli, który wskazuje na klucz główny w innej tabeli. Dzięki kluczom obcym budujemy relacje między encjami. Na przykład w tabeli Zamówienia możemy umieścić kolumnę id_klienta, która będzie kluczem obcym odwołującym się do tabeli Klienci. Dzięki temu każde zamówienie jest przypisane do konkretnego klienta, a baza danych nie pozwoli wprowadzić zamówienia z nieistniejącym identyfikatorem klienta.

Czym jest język SQL oraz plan budowy zapytania

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

Język SQL (Structured Query Language) stanowi podstawowe narzędzie pracy z relacyjnymi bazami danych. Służy zarówno do definiowania struktur (DDL – Data Definition Language), jak i do operowania na danych (DML – Data Manipulation Language). Poniżej przedstawiono zestaw najważniejszych konstrukcji, które stanowią fundament praktycznej pracy z SQL.

Pisząc zapytanie w języku SQL, konieczne jest zachowanie określonej kolejności klauzul i operatorów. Nawet jeśli w praktyce pewne fragmenty są pomijane (bo nie zawsze wszystkie są potrzebne), logika wykonania zapytania zawsze przebiega w tej samej sekwencji:

- WITH (CTE) – definiowanie wspólnych wyrażeń tablicowych, czyli tymczasowych „mini-tabel", które można wykorzystać w dalszej części zapytania.
- SELECT – wybór kolumn, obliczeń, aliasów, funkcji agregujących i innych elementów, które mają być wyświetlone w wyniku.
- FROM – wskazanie tabel lub podzapytań, z których pobierane są dane.
- JOIN – łączenie danych z wielu tabel na podstawie kluczy i warunków powiązań.
- WHERE – filtrowanie danych na poziomie pojedynczych rekordów.
- GROUP BY – grupowanie rekordów według wskazanych kolumn.
- HAVING – filtrowanie grup po dokonaniu agregacji.
- WINDOW / OVER – funkcje okna, pozwalające na obliczenia w ramach określonego „okna" danych (np. sumy czy średnie w działach).
- UNION / UNION ALL / EXCEPT / INTERSECT – operatory zbiorowe, umożliwiające łączenie wyników kilku zapytań lub wyciąganie części wspólnych czy różnic.
- ORDER BY – sortowanie wyników według wybranych kolumn.
- OFFSET – pominięcie określonej liczby pierwszych wierszy (np. przy paginacji).
- LIMIT – ograniczenie liczby zwróconych rekordów.

Elementy języka SQL

SQL pozwala pobierać, filtrować i prezentować dane w elastyczny sposób. Tworzenie zapytań SQL opiera się na pewnej ustalonej strukturze, w której poszczególne

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

klauzule wykonywane są w określonej kolejności. Całość zaczyna się od SELECT, czyli wyboru kolumn, które mają zostać zwrócone. To tutaj decydujemy, jakie dane będą widoczne w wyniku – mogą to być pojedyncze kolumny, wszystkie dane z tabeli (*), albo bardziej złożone elementy, takie jak aliasy, funkcje agregujące czy wyrażenia na ciągach znaków. Przykładowo możemy pobrać imię, nazwisko oraz pensję pracownika, a jednocześnie zbudować nową kolumnę „Pełne Imię i Nazwisko", łącząc dwie wartości funkcją CONCAT.

Kolejnym krokiem jest FROM, czyli określenie tabeli lub tabel, z których dane mają być pobrane. Tutaj wskazujemy główne źródło danych. W prostych przypadkach będzie to pojedyncza tabela, np. employees, ale nic nie stoi na przeszkodzie, by w tym miejscu pojawiło się podzapytanie zwracające dane pośrednie.

W praktyce bardzo często korzystamy również z JOIN-ów, czyli mechanizmów łączenia danych z różnych tabel. Najczęściej stosowany jest INNER JOIN, który zwraca tylko te rekordy, które mają swoje odpowiedniki w obu tabelach. LEFT JOIN pozwala pobrać wszystkie dane z tabeli lewej, nawet jeśli nie mają dopasowania po stronie prawej. Analogicznie działa RIGHT JOIN. Jeżeli interesują nas wszystkie rekordy niezależnie od dopasowania, możemy użyć FULL JOIN. Istnieje też CROSS JOIN, który generuje iloczyn kartezjański wszystkich wierszy, choć używany jest rzadziej. Dzięki JOIN-om możemy w jednym zapytaniu odwołać się np. do tabeli pracownicy, pensje i działy, łącząc dane według kluczy obcych.

Aby ograniczyć liczbę zwracanych rekordów, stosujemy klauzulę WHERE, która filtruje dane przed wykonaniem grupowania i agregacji. To tutaj umieszczamy warunki logiczne (np. pensja większa niż 50 000, data zatrudnienia po roku 2000) oraz operatory porównania (=, >, <, BETWEEN, IN, LIKE, IS NULL). Klauzula WHERE pozwala nam skoncentrować się tylko na tych danych, które rzeczywiście spełniają interesujące nas kryteria.

Następnym etapem jest GROUP BY, które grupuje dane według wskazanych kolumn. Jest to niezbędne, gdy chcemy wykonywać operacje agregujące, takie jak liczenie średniej, sumy czy liczby rekordów w każdej grupie. Ważna zasada mówi, że kolumny pojawiające się w SELECT muszą być częścią grupowania lub zostać użyte wewnątrz funkcji agregującej.

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

Gdy już pogrupujemy dane, możemy zastosować dodatkowe filtrowanie za pomocą klauzuli HAVING. W przeciwieństwie do WHERE, które działa na poziomie pojedynczych rekordów, HAVING służy do filtrowania całych grup wynikowych. Możemy tu np. odrzucić działy, w których średnia pensja jest niższa niż określona wartość.

Otrzymane wyniki można uporządkować przy użyciu ORDER BY. Ta klauzula określa, według których kolumn i w jakiej kolejności mają zostać posortowane dane. Domyślnie sortowanie odbywa się rosnąco (ASC), ale można wymusić malejącą kolejność (DESC). Nic nie stoi na przeszkodzie, aby sortować po kilku kolumnach jednocześnie, np. najpierw po wysokości pensji, a potem po dacie zatrudnienia.

Jeżeli danych jest bardzo dużo, w wielu sytuacjach przydaje się LIMIT, które ogranicza liczbę zwróconych rekordów. Dzięki temu możemy np. wyświetlić tylko 10 najlepiej zarabiających pracowników, zamiast całej tabeli.

Czasem pojawia się potrzeba łączenia wyników wielu zapytań. W tym celu stosujemy UNION oraz UNION ALL. Pierwsze z nich zwraca unikalne rekordy, eliminując duplikaty, drugie natomiast wyświetla wszystkie dane, w tym powtarzające się wiersze. Istnieje przy tym ważna zasada: liczba i typy kolumn w zapytaniach łączonych przez UNION muszą być zgodne.

Wreszcie, w SQL często korzysta się z podzapytań. Mogą one być nieskorelowane, czyli niezależne od zapytania głównego (np. lista wartości w klauzuli IN), albo skorelowane, czyli odwołujące się do danych aktualnie przetwarzanych przez zapytanie nadrzędne. W tym drugim przypadku podzapytanie wykonywane jest wielokrotnie, np. aby sprawdzić za pomocą EXISTS, czy dany rekord ma powiązane dane w innej tabeli.

Wszystkie te elementy można ze sobą łączyć, tworząc bardziej złożone zapytania. Przykładowo, zapytanie może jednocześnie wybierać kolumny z kilku tabel połączonych JOIN-ami, filtrować dane przy użyciu WHERE, grupować je według działu, stosować agregację, a następnie sortować i ograniczać liczbę wyników.

Indeksy w bazach danych

Indeks w bazie danych to specjalna struktura, której zadaniem jest przyspieszenie wyszukiwania rekordów w tabeli. Można go porównać do indeksu w książce –

zamiast przeglądać wszystkie strony po kolei, sięgamy od razu do odpowiedniego miejsca. W praktyce oznacza to, że baza danych nie musi analizować każdego wiersza w tabeli przy wykonywaniu zapytania, lecz korzysta z uporządkowanej struktury, najczęściej w postaci drzewa B+. Dzięki temu procesy filtrowania, wyszukiwania i sortowania danych przebiegają znacznie szybciej.

Należy jednak wyraźnie rozróżnić indeksy od kluczy. Klucze (PRIMARY KEY, FOREIGN KEY, UNIQUE) pełnią funkcję reguł integralności. Określają one dopuszczalne wartości w kolumnach i wymuszają poprawność powiązań między tabelami. Na przykład klucz główny gwarantuje unikalność i niepustość identyfikatora, a klucz obcy zapewnia, że dane w jednej tabeli odwołują się wyłącznie do istniejących rekordów w tabeli nadrzędnej. Indeksy natomiast nie stanowią reguły biznesowej – są narzędziem czysto technicznym, które wspiera wydajność działania bazy. Możemy stworzyć indeks na dowolnej kolumnie (np. na kolumnie „nazwisko”), aby przyspieszyć zapytania zawierające filtr WHERE nazwisko = 'Kowalski'.

Warto pamiętać, że wiele kluczy automatycznie powoduje utworzenie indeksu. Zdefiniowanie klucza głównego wiąże się z utworzeniem indeksu unikalnego, podobnie działa klucz UNIQUE. W niektórych systemach również dla kluczy obcych automatycznie sugerowane jest utworzenie indeksu, aby ułatwić sprawdzanie poprawności powiązań. Należy jednak podkreślić, że nie każdy indeks oznacza klucz. Indeks można utworzyć wyłącznie w celu przyspieszenia raportów czy zestawień i w żaden sposób nie wpływa on na poprawność danych.

W praktyce wyróżnia się kilka podstawowych rodzajów indeksów. Indeks jednokolumnowy dotyczy jednej kolumny i przyspiesza zapytania bazujące na jej wartościach. Indeks wielokolumnowy obejmuje kilka kolumn, co jest przydatne przy filtrach i sortowaniach uwzględniających wiele kryteriów jednocześnie. Indeks unikalny zapewnia brak duplikatów w kolumnie, np. w przypadku adresu e-mail użytkownika. Indeksy pełnotekstowe służą do wyszukiwania słów i fraz w dużych tekstach, a indeksy przestrzenne (GIS) wspierają operacje na danych geograficznych, takich jak współrzędne czy kształty geometryczne.

Indeksy są szczególnie przydatne, gdy tabela zawiera dużą liczbę rekordów, gdy często korzystamy z zapytań filtrujących (WHERE), sortujących (ORDER BY) lub łączących (JOIN), a także wtedy, gdy chcemy wymusić unikalność wartości w

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

kolumnie. Nie są jednak pozbawione ograniczeń. Każdy indeks przyspiesza odczyt, ale spowalnia operacje zapisu – każde dodanie, usunięcie lub modyfikacja rekordu wymaga również aktualizacji indeksów. Nadmiar indeksów zwiększa również zużycie pamięci i przestrzeni dyskowej. Źle dobrany indeks, np. na kolumnie, w której większość wartości jest identyczna, może w praktyce nie dawać żadnych korzyści wydajnościowych.

Polecenia SQL związane z indeksami

- Tworzenie indeksu na jednej kolumnie:

  CREATE INDEX idx_nazwisko ON Pracownicy(nazwisko);

- Tworzenie indeksu złożonego (na kilku kolumnach):

  CREATE INDEX idx_nazwisko_imie ON Pracownicy(nazwisko, imie);

- Tworzenie indeksu unikalnego:

  CREATE UNIQUE INDEX idx_email_unique ON Uzytkownicy(email);

- Usuwanie indeksu:

  DROP INDEX idx_nazwisko ON Pracownicy;

- Wyświetlenie indeksów w tabeli (w MySQL)

  SHOW INDEXES FROM Pracownicy;

Rys historyczny najważniejsze formaty przechowywania danych

Zanim zaczniemy pracować z bazami danych, warto najpierw zrozumieć ich pochodzenie, potrzebę powstania oraz to, jak przez lata się rozwijały. Bazy danych nie pojawiły się z dnia na dzień — ich historia sięga dawnych czasów, kiedy ludzie zaczęli prowadzić interesy i chcieli przechowywać informacje o kontrahentach, zbierać podatki, spisy ludności czy inwentarza. Przez całe wieki dane były zapisywane na papierze, w księgach, na kamiennych tabliczkach, czasem na liściach

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

czy pergaminach. Współcześnie wielu z nas pamięta jeszcze czasy sprzed masowej cyfryzacji, gdy dokumentacja, faktury, ewidencje – odbiory, dostawy czy zamówienia – były prowadzone ręcznie i papierowo.

Rozwój technologii komputerowych – początkowo stosowanych głównie w zastosowaniach wojskowych i naukowych – sprawił, że zaczęto dostrzegać, iż komputery mogą znacznie ułatwić, przyspieszyć i uodpornić na błędy procesy związane ze zbieraniem, przechowywaniem i przetwarzaniem danych. W latach sześćdziesiątych XX wieku zaczęły się pojawiać pierwsze systemy komputerowych baz danych, jak systemy hierarchiczne i sieciowe.

Format CSV (Comma-Separated Values) jest jednym z najstarszych i najprostszych sposobów przechowywania danych tabelarycznych w plikach tekstowych. Jego historia sięga lat siedemdziesiątych, kiedy wczesne języki programowania, takie jak Fortran, korzystały z prostych konwencji rozdzielania pól przecinkami. Plik CSV to nic innego jak sekwencja wierszy, w których poszczególne wartości oddzielone są separatorem – najczęściej przecinkiem, choć bywa nim również średnik, tabulacja czy inny znak. Pierwszy wiersz zwykle pełni funkcję nagłówka zawierającego nazwy kolumn. Format ten, dzięki swojej prostocie, stał się powszechny i uniwersalny. Można go łatwo odczytać w prostym edytorze tekstu, w arkuszu kalkulacyjnym czy w bazie danych. CSV sprawdza się doskonale do prostych zestawień, raportów czy eksportu danych między różnymi programami. Jego wadą jest brak możliwości reprezentowania bardziej złożonych struktur, takich jak dane hierarchiczne, brak jednoznacznej specyfikacji typów danych oraz problemy z kodowaniem czy cytowaniem pól zawierających przecinki lub znaki nowej linii. Jest to więc format niezwykle praktyczny, lecz ograniczony do prostych tabel.

XML, czyli eXtensible Markup Language, został opracowany w latach dziewięćdziesiątych przez konsorcjum W3C jako uproszczona i bardziej elastyczna wersja wcześniejszego standardu SGML. Został zaprojektowany do przechowywania i wymiany danych w sposób czytelny zarówno dla ludzi, jak i maszyn. Struktura XML opiera się na elementach oznaczonych tagami otwierającymi i zamykającymi, wewnątrz których mogą znajdować się dane, atrybuty oraz kolejne zagnieżdżone elementy. XML świetnie nadaje się do reprezentowania danych hierarchicznych i złożonych, a dodatkowo pozwala na walidację poprawności dokumentu dzięki

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

wykorzystaniu schematów DTD lub XSD. Format ten znalazł szerokie zastosowanie w integracji systemów, przesyłaniu dokumentów biznesowych, tworzeniu kanałów RSS czy w standardach wymiany danych w administracji publicznej. Jego główną wadą jest duża „ciężkość" – pliki XML bywają rozbudowane i zawierają wiele nadmiarowych znaczników, co zwiększa rozmiar danych i wydłuża czas przetwarzania. Pomimo tego XML pozostaje bardzo istotnym formatem, szczególnie tam, gdzie kluczowa jest precyzja opisu, walidacja i standaryzacja danych.

JSON, czyli JavaScript Object Notation, pojawił się na początku lat dwutysięcznych jako prostsza alternatywa dla XML. Wywodzi się bezpośrednio z notacji obiektowej języka JavaScript, ale szybko stał się niezależnym i uniwersalnym formatem wymiany danych. JSON opiera się na zapisie par klucz–wartość oraz tablic wartości, przy czym dopuszcza typy takie jak liczby, teksty, wartości logiczne czy null. Jego składnia jest niezwykle przejrzysta, zbliżona do sposobu, w jaki programista definiuje struktury w kodzie. Format ten jest lekki, szybki w przetwarzaniu i wyjątkowo popularny w aplikacjach webowych, szczególnie w komunikacji serwer–klient poprzez API. JSON bardzo dobrze wspiera wymianę danych w usługach internetowych, a także stał się podstawowym formatem przechowywania danych w wielu bazach dokumentowych, takich jak MongoDB. Wadą JSON jest brak możliwości bezpośredniego dodawania komentarzy oraz brak wbudowanego mechanizmu walidacji tak precyzyjnego, jak w przypadku XML. Istnieją wprawdzie rozszerzenia takie jak JSON Schema, jednak ich stosowanie nie jest jednolicie ustandaryzowane. Mimo to JSON dzięki prostocie i efektywności stał się najpopularniejszym formatem wymiany danych w nowoczesnych systemach informatycznych.

XLSX to format plików arkuszy kalkulacyjnych wprowadzony przez Microsoft wraz z pakietem Office 2007 jako część standardu Office Open XML. Zastąpił on wcześniejsze pliki w formacie binarnym XLS, wprowadzając strukturę opartą na XML i przechowywaną w skompresowanej paczce ZIP. Każdy plik XLSX składa się z zestawu plików XML, które opisują dane w komórkach, formatowanie, style, formuły, metadane, wykresy czy grafiki. Dzięki temu format XLSX obsługuje wiele arkuszy w jednym pliku, zaawansowane funkcje obliczeniowe, formatowanie warunkowe, wykresy i tabele przestawne. Jego ogromną zaletą jest bogactwo funkcji, które pozwalają nie tylko przechowywać dane, lecz także je analizować i prezentować w

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

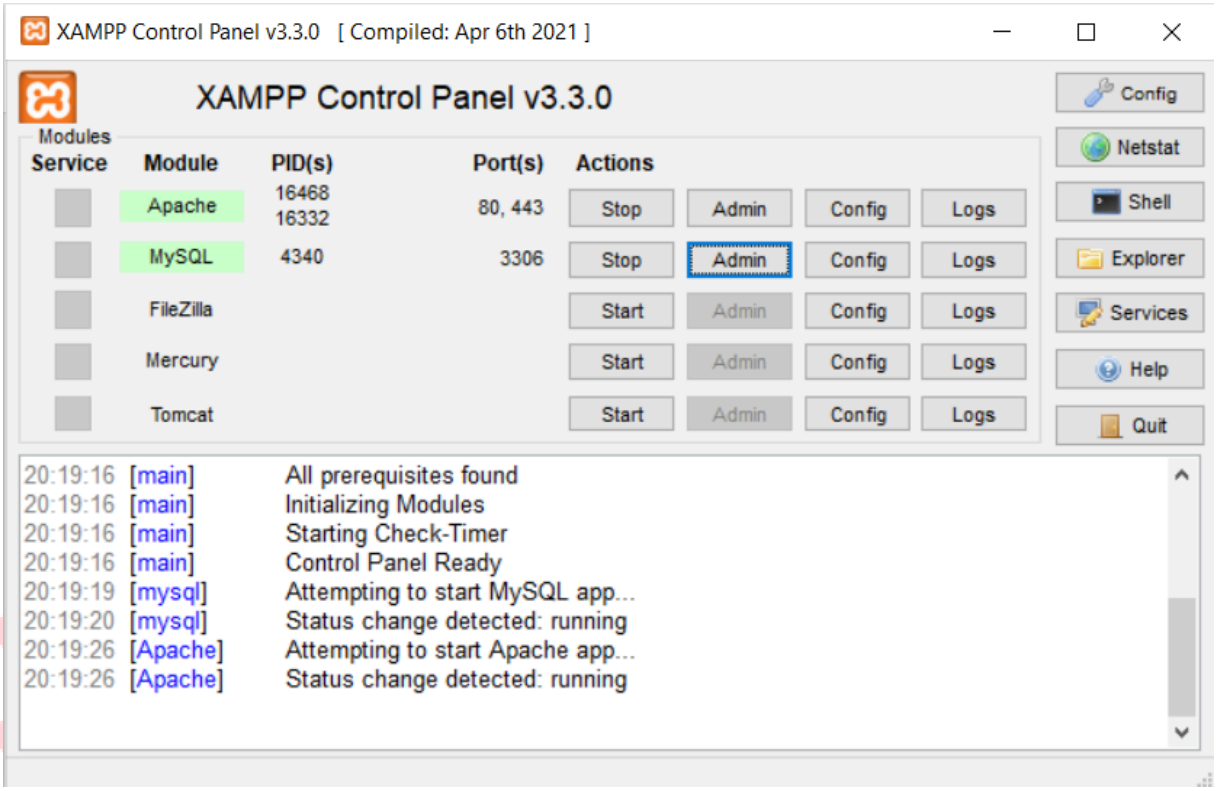
atrakcyjnej formie. XLSX jest także lepiej zabezpieczony przed uszkodzeniami niż wcześniejsze formaty binarne, ponieważ uszkodzenie jednego pliku wewnętrznego nie musi oznaczać utraty całej zawartości. Jako format otwarty (ISO/IEC 29500) zapewnia też kompatybilność z innymi pakietami biurowymi, takimi jak LibreOffice czy Google Sheets. Wadą XLSX jest większa złożoność oraz większe zapotrzebowanie na zasoby przy odczycie i zapisie w porównaniu z prostymi formatami tekstowymi. Pomimo tego XLSX pozostaje standardem de facto w świecie arkuszy kalkulacyjnych i jest niezastąpiony wszędzie tam, gdzie oprócz danych istotna jest również ich analiza i prezentacja.

Wykorzystanie pakietu XAMPP oraz phpMyAdmin

Aby uzyskać praktyczny obraz tego, do czego może służyć baza danych i jak rozpocząć z nią pracę, warto skorzystać z pakietu XAMPP. Jest to darmowe środowisko, które zawiera w sobie wszystkie niezbędne narzędzia do uruchomienia serwera lokalnego:

- Apache – serwer WWW obsługujący zapytania HTTP,
- MySQL/MariaDB – system zarządzania relacyjną bazą danych,
- PHP – interpreter języka PHP pozwalający uruchamiać aplikacje internetowe,
- phpMyAdmin – przeglądarkowe narzędzie graficzne do zarządzania bazami danych.

Dzięki XAMPP możemy uruchomić lokalny serwer i od razu rozpocząć naukę pracy z bazami danych, bez konieczności skomplikowanej konfiguracji. Po uruchomieniu panelu sterowania XAMPP należy włączyć dwa podstawowe moduły: Apache oraz MySQL i wybrać przycisk Admin. Zostało to przedstawione na rysunku 1. To wystarczy, aby w przeglądarce otworzyć phpMyAdmin i rozpocząć tworzenie pierwszej bazy.



Rysunek 1. Widok programu XAMPP

phpMyAdmin to narzędzie dostępne z poziomu przeglądarki internetowej pod adresem http://localhost/phpmyadmin. Domyślnie logowanie odbywa się przy pomocy użytkownika root, bez hasła. Jest to rozwiązanie wygodne do pracy lokalnej, ale w przypadku faktycznego wdrożenia w przedsiębiorstwie czy pracy w zespole należy utworzyć własnych użytkowników oraz zmodyfikować konfigurację, tak aby uniemożliwić korzystanie z konta root w celach produkcyjnych. Zmiany tego typu można wprowadzać w plikach konfiguracyjnych serwera MySQL.

W tym celu należy edytować plik konfiguracyjny phpMyAdmin znajdujący się w katalogu:

C:\xampp\phpMyAdmin\config.inc.php

W pliku tym należy odnaleźć linijkę:

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

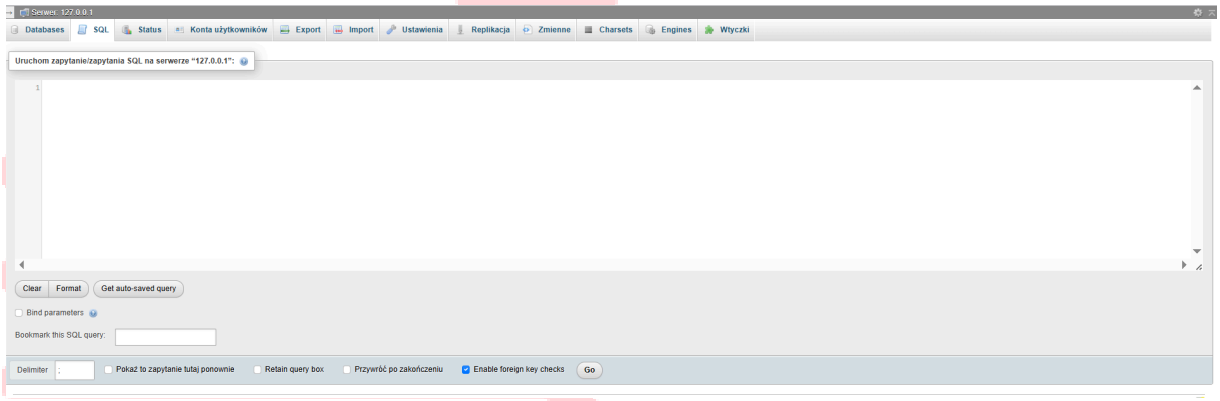
Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

$cfg['Servers'][$i]['auth_type'] = 'config';

i zmienić ją na:

$cfg['Servers'][$i]['auth_type'] = 'cookie';

Ustawienie *auth_type* na wartość cookie powoduje, że przy każdym logowaniu phpMyAdmin poprosi o podanie nazwy użytkownika i hasła. Dzięki temu możemy zalogować się zarówno na konto root, jak i na innych użytkowników utworzonych w systemie MySQL. Po dokonaniu tej zmiany należy zresetować serwer w panelu XAMPP (zatrzymać i ponownie uruchomić moduły Apache oraz MySQL), aby nowe ustawienia zaczęły obowiązywać.



Rysunek 2 phpMyAdmin z otwartą zakładką służącą wywoływaniu kwerend SQL.

Zapoznanie interfejsu phpMyAdmin

Po zalogowaniu do phpMyAdmin widzimy panel główny, w którym znajdują się różne zakładki ułatwiające zarządzanie bazą zostały one przedstawione na rysunku 2:

- Databases (Bazy danych) – umożliwia tworzenie nowych baz i przeglądanie istniejących. To od tej zakładki najczęściej zaczynamy pracę.
- SQL – pozwala ręcznie wpisywać i wykonywać zapytania w języku SQL, co jest kluczowe przy nauce praktycznej składni.
- Status – wyświetla informacje o obciążeniu serwera i statystykach pracy.
- Konta użytkowników – umożliwia dodawanie i zarządzanie użytkownikami baz danych, przypisywanie im haseł i uprawnień.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

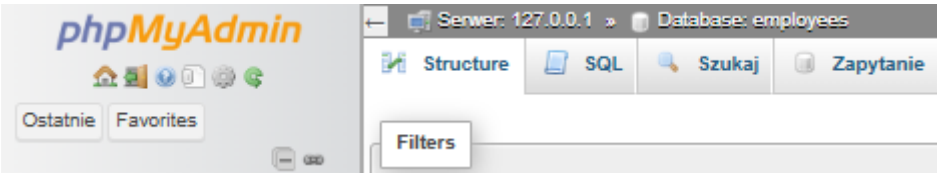
Projekt współfinansowany ze Środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

- Import – pozwala wczytywać dane z plików w różnych formatach (np. CSV, SQL).
- Export – umożliwia zapisanie bazy lub tabeli do pliku, co jest szczególnie ważne w kontekście tworzenia kopii zapasowych.
- Ustawienia (Variables/Settings) – pozwalają dostosować parametry pracy serwera bazy danych.
- Replikacja – moduł służący do konfiguracji replikacji baz danych, wykorzystywany w zaawansowanych wdrożeniach.
- Wtyczki (Plugins) – sekcja umożliwiająca zarządzanie dodatkowymi funkcjonalnościami phpMyAdmin.
- Charsets – ta zakładka dotyczy obsługi zestawów znaków i porównań (collation) w bazie danych. Można tutaj podejrzeć i skonfigurować, jakie kodowanie znaków obsługuje serwer (np. UTF-8, latin1). Jest to szczególnie istotne w przypadku baz zawierających dane w różnych językach – niewłaściwy zestaw znaków może prowadzić do błędnego wyświetlania liter, np. polskich znaków diakrytycznych.
- Engines – zakładka pozwala zobaczyć, jakie silniki bazodanowe są obsługiwane przez serwer MySQL/MariaDB (np. InnoDB, MyISAM, MEMORY). Każdy silnik ma swoje właściwości – np. InnoDB obsługuje transakcje i klucze obce, a MyISAM jest szybszy w niektórych zastosowaniach, ale nie zapewnia integralności referencyjnej. W zakładce Engines można również podejrzeć, jaki silnik jest domyślny dla nowo tworzonych tabel.

Warto zwrócić uwagę, że w prawym górnym rogu znajduje się informacja o tym w jakiej bazie danych oraz w jakiej tabeli obecnie się znajdujemy. Wybranie przycisku Serwer 127.0.0.1 zawsze spowoduje przejście na stronę główną serwera i wyjście z aktywnej bazy. Przycisk strzałki znajdujący się w lewym górnym boku spowoduje zamknięcie lub otwarcie listy bocznej z dostępnymi bazami danych. Elementy te zostały pokazane na rysunku 3.

Mając aktywną bazę danych lub tabele z bazy dostajemy dostęp do dodatkowych zakładek opisanych poniżej:

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



Rysunek 3. Informacja o aktywnej bazie danych

Rysunek 4. Zakładki dostępne dla bazy danych

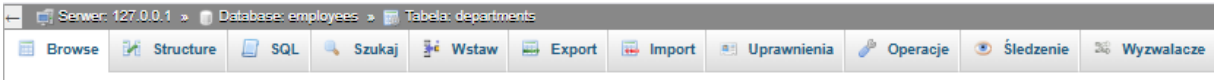
Po wejściu do konkretnej bazy danych w phpMyAdmin oprócz zakładek serwerowych pojawiają się dodatkowe, przedstawione na rysunku 4:

- Structure – pokazuje strukturę bazy, czyli listę tabel wraz z ich podstawowymi informacjami (liczba rekordów, typ silnika, zestaw znaków).
- Szukaj – pozwala wyszukiwać dane w całej bazie.
- Zapytanie (Query) – umożliwia skonstruowanie zapytania w graficznym kreatorze, bez konieczności ręcznego pisania SQL.
- Operacje – umożliwia zarządzanie całą bazą, np. zmianę nazwy, zestawu znaków, a także skopiowanie lub usunięcie bazy.
- Uprawnienia – pozwala ustawić prawa dostępu użytkowników do danej bazy danych.
- Procedury i funkcje – umożliwia zarządzanie obiektami z logiką programistyczną zapisanymi w bazie.
- Events – pozwala definiować i zarządzać zdarzeniami cyklicznymi w bazie (scheduler).
- Wyzwalacze – umożliwia przeglądanie i tworzenie triggerów dla tabel w tej bazie.
- Śledzenie (Tracking) – pozwala monitorować zmiany w strukturze i zapytaniach w bazie.
- Designer – narzędzie graficzne do wizualnego projektowania schematu bazy.
- Central columns - służy do centralnego zarządzania kolumnami, które pojawiają się w wielu tabelach. Dzięki niej można definiować wzorce kolumn, a

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

następnie łatwo dodawać je do różnych tabel bez konieczności każdorazowego ręcznego tworzenia od nowa.



Rysunek 5. Zakładki dostępne po wybraniu tabeli.

Jeżeli wybierzemy konkretną tabelę w bazie, otrzymujemy dodatkowy zestaw zakładek przedstawionych na rysunku 5:

- Browse (Przeglądaj) – umożliwia przeglądanie zawartości tabeli w formie rekordów.
- Structure (Struktura) – pokazuje listę kolumn tabeli wraz z ich typami, kluczami i atrybutami.
- Wstaw – pozwala dodać nowy rekord do tabeli poprzez formularz.
- Szukaj – umożliwia wyszukiwanie rekordów w tabeli z warunkami.
- Export – pozwala wyeksportować zawartość tabeli do różnych formatów (np. SQL, CSV, JSON).
- Import – służy do wczytania danych do tabeli z pliku.
- Uprawnienia – daje możliwość zarządzania dostępem użytkowników do danej tabeli.
- Operacje – pozwala wykonać akcje administracyjne na tabeli, np. zmianę nazwy, silnika, usunięcie czy kopiowanie.
- Śledzenie – podobnie jak w bazie, monitoruje zmiany w tabeli.
- Wyzwalacze – pokazuje, jakie triggery są powiązane z tabelą i pozwala dodawać nowe.

Dane do wykorzystania w trakcie zajęć

Na stronie: https://dev.mysql.com/doc/index-other.html znajdują się przykładowe bazy danych do pobrania, sugerowane do celów edukacyjnych (rysunek 6). W dalszej pracy na zajęciach będzie wykorzystywana baza sakilla, natomiast na potrzeby wprowadzenia w podstawowe koncepty baz danych, zostanie wykorzystana baza

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

world która cechuje się prostotą, ponieważ zawiera tylko 3 tabele: kontynenty, kraje, miasta.

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 6. Źródła do pobrania baz danych używanych na zajęciach dostępne na stronie: https://dev.mysql.com/doc/index-other.html

Ćwiczenia samodzielne

1. Zainstaluj pakiet XAMPP i uruchom podstawowe moduły: Apache oraz MySQL.
2. Otwórz phpMyAdmin z poziomu przeglądarki (http://localhost/phpmyadmin) i zapoznaj się z jego interfejsem.
3. Pobierz przykładową bazę danych world ze strony podanej w instrukcji.
4. Załaduj bazę world do phpMyAdmin. Przetestuj różne metody importu:
   - wgranie pliku SQL w zakładce Import,
   - ręczne wklejenie kodu SQL do zakładki SQL,
   - użycie terminala (np. mysql -u root -p < world.sql).
5. Przejrzyj strukturę bazy world i wypisz nazwy wszystkich dostępnych tabel.
6. Zidentyfikuj klucze główne i obce w każdej z tabel (podpowiedź: zwróć uwagę na oznaczenia w zakładce Structure).
7. Dla tabeli City wskaż wszystkie atrybuty wraz z ich typami danych i krótko opisz, do czego mogą służyć.
8. Napisz proste zapytanie SELECT, które pobierze nazwy wszystkich państw z tabeli Country.
9. Utwórz kilka przykładowych zapytań SELECT:
   - Wyświetl wszystkie rekordy z tabeli City – użyj SELECT * FROM City;.
   - pobierz miasta o populacji większej niż 5 milionów,
   - Pobierz tylko nazwy miast z tabeli City (kolumna Name).
   - Wyświetl nazwy miast oraz liczbę ich mieszkańców (Name, Population).

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

- wypisz wszystkie kraje z kontynentu „Europe”,
- Pobierz wszystkie kraje z kontynentu „Europe” – SELECT * FROM Country WHERE Continent = 'Europe';.
- Wyświetl wszystkie miasta, które mają populację większą niż 1 000 000 mieszkańców.
- Pokaż państwa, które mają powierzchnię mniejszą niż 50 000 km².
- Pobierz wszystkie miasta z kraju „Poland” (kolumna CountryCode = 'POL').
- Wyświetl nazwy miast, które zaczynają się na literę „A” (użyj LIKE 'A%').
- Pokaż miasta, których nazwy kończą się na literę „n” (LIKE '%n').
- Wyświetl miasta, których nazwy zawierają literę „burg”.
- Pobierz kraje, które należą do kontynentów: Europa lub Azja (użyj IN).
- Wyświetl wszystkie miasta o populacji pomiędzy 500 000 a 1 000 000 mieszkańców (operator BETWEEN).
- Wyświetl miasta, dla których nie podano populacji (Population IS NULL).
- Pokaż listę wszystkich kontynentów bez powtórzeń (użyj DISTINCT Continent).
- Wyświetl 10 pierwszych miast z tabeli City (LIMIT 10).
- Wyświetl 10 największych miast na świecie według populacji (ORDER BY Population DESC LIMIT 10).
- Wyświetl 10 najmniejszych miast na świecie według populacji (ORDER BY Population ASC LIMIT 10)

10. Napisz zapytanie UPDATE, które zmieni nazwę wybranego miasta na inną (np. zaktualizuj nazwę miasta testowego).
11. Napisz zapytanie DELETE, które usunie wybrany rekord z tabeli City. Upewnij się, że rozumiesz konsekwencje usuwania danych.
12. Dodaj nowy rekord do tabeli City za pomocą zapytania INSERT – wybierz dowolne miasto, określ nazwę, identyfikator kraju i liczbę mieszkańców.
13. Usuń całą bazę world z poziomu phpMyAdmin.
14. Zadanie projektowe (praca w grupach)
    - Wybierzcie w grupie projektowej fragment rzeczywistości (np. uczelnia, biblioteka, sklep internetowy, przychodnia lekarska), który znacie z własnych doświadczeń.
    - Napiszcie wstęp teoretyczny zawierający motywację podjęcia tematu – jakie problemy ma rozwiązać baza danych i dlaczego warto ją stworzyć.
    - Opracujcie wstępny opis słowny bazy, w którym wyodrębnione zostaną najważniejsze encje (tabele) i ich atrybuty.

Pytania pomocnicze

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

1. Czym jest baza danych?
2. Jakie są główne zalety stosowania baz danych?
3. Czym jest encja?
4. Co oznacza atrybut encji?
5. Na czym polega relacja między encjami?
6. Czym różni się klucz główny od klucza obcego?
7. Podaj przykład relacji jeden-do-wielu.
8. Podaj przykład relacji wiele-do-wielu.
9. Co to jest krotka w tabeli?
10. Jak rozumiesz pojęcie schematu bazy danych?
11. Dlaczego redundancja danych jest problematyczna?
12. Czym różni się dane od informacji?
13. Podaj przykład encji z życia codziennego.
14. Jakie są główne elementy modelu relacyjnego?
15. Jakie znaczenie ma spójność danych?

Podsumowanie

W trakcie pierwszego laboratorium studenci poznali podstawowe pojęcia związane z relacyjnymi bazami danych, takie jak encja, atrybut, tabela, relacja oraz klucze główne i obce, co pozwoliło im zrozumieć fundamenty modelowania danych. Ćwiczenie umożliwiło zapoznanie się ze środowiskiem XAMPP i narzędziem phpMyAdmin, które stanowią praktyczną bazę do dalszej pracy nad projektami i nauki języka SQL. Zrealizowane zadania pozwoliły osiągnąć cele dydaktyczne, takie jak umiejętność tworzenia i przeglądania baz danych, rozpoznawania struktury tabel oraz interpretowania relacji między encjami. Ważnym wnioskiem jest uświadomienie sobie, że baza danych nie jest jedynie zbiorem informacji, lecz systemem, który dzięki spójności i odpowiedniemu modelowaniu wspiera realne procesy techniczne i organizacyjne.

Literatura

1. Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th ed.). Pearson.
2. Date, C. J. (2019). Database Design and Relational Theory: Normal Forms and All That Jazz (2nd ed.). O'Reilly Media.

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

3. Hernandez, M. J. (2013). Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design (3rd ed.). Addison-Wesley.
4. Coronel, C., & Morris, S. (2018). Database Systems: Design, Implementation, & Management (13th ed.). Cengage Learning.
5. Watt, A., & Eng, A. (2020). Database Design (2nd Edition). BCcampus Open Textbook. Dostępne online: https://open.umn.edu/opentextbooks/textbooks/database-design-2nd-edition
6. Relacje w bazie danych: Understanding Relational Databases: A Beginner's Guide - Essential SQL: https://www.essentialsql.com/understanding-relational-databases-a-beginners-guide/ (data ostatniego dostępu 25.09.25)

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