Skrypy do przedmiotu

Bazy danych

Autor: mgr inż. Szymon Guzik

SQL - definicja

SQL (Structured Query Language) jest językiem zapytań służącym do tworzenia i zarządzania bazami danych relacyjnych. Pozwala na wykonywanie operacji takich jak wyszukiwanie, aktualizowanie, usuwanie i dodawanie danych w bazie danych. Jest to jeden z najważniejszych języków programowania używanych w branży IT.

- DDL (Data Definition Language) - język definicji danych, pozwala na tworzenie, modyfikowanie i usuwanie obiektów bazy danych takich jak tabele, indeksy, widoki i procedury składowane. Przykładowe polecenia to: CREATE, ALTER, DROP.

- DML (Data Manipulation Language) - język manipulacji danych, służy do wstawiania, aktualizowania i usuwania danych z bazy danych. Przykładowe polecenia to: SELECT, INSERT, UPDATE, DELETE.

- DCL (Data Control Language) - język kontroli dostępu, umożliwiający użytkownikom określenie, kto ma uprawnienia do wykonywania określonych operacji na bazie danych. Przykładowe polecenia to: GRANT, REVOKE.

- TCL (Transaction Control Language) - język kontroli transakcji, umożliwiający zarządzanie transakcjami w bazie danych, takimi jak zapisywanie zmian, wycofywanie zmian i tworzenie punktów przywracania. Przykładowe polecenia to: COMMIT, ROLLBACK, SAVEPOINT.

- DDL (Data Dictionary Language) - język słownika danych, umożliwiający użytkownikom przeglądanie informacji o obiektach bazy danych, takich jak nazwy tabel, kolumn i indeksów. Przykładowe polecenie to: DESCRIBE.

“Rodzaje” SQL

SQL standardowy (ANSI SQL)
SQL Oracle
SQL Microsoft SQL Server
SQL MySQL
SQL PostgreSQL
SQL SQLite
SQL Sybase
SQL Informix
SQL Firebird
SQL MariaDB.

- SQL Standardowy - jest to jedna z najbardziej popularnych wersji języka SQL, która została zatwierdzona przez ANSI i ISO. Jest używana w wielu bazach danych, takich jak Oracle, MySQL, MS SQL Server i wiele innych.

- MySQL - jest to jeden z najbardziej popularnych systemów zarządzania bazami danych oparty na SQL. Jest to bezpłatne i łatwe w użyciu oprogramowanie, które jest szczególnie popularne w środowiskach webowych.

- Microsoft SQL Server - jest to jedna z najbardziej zaawansowanych i zaawansowanych wersji SQL, która jest szczególnie popularna w środowiskach korporacyjnych i biznesowych. Zawiera szereg funkcji i narzędzi do zarządzania bazami danych, takich jak narzędzia do tworzenia raportów i analizy danych.

- Oracle SQL - jest to jedna z najbardziej zaawansowanych wersji języka SQL, która jest szczególnie popularna w dużych organizacjach i środowiskach biznesowych. Zawiera szereg narzędzi i funkcji do zarządzania bazami danych, takich jak narzędzia do współpracy i integracji z innymi systemami.

- PostgreSQL - jest to jedna z najbardziej zaawansowanych wersji języka SQL, która jest szczególnie popularna w środowiskach webowych i naukowych. Zawiera szereg funkcji i narzędzi do zarządzania bazami danych, takich jak narzędzia do tworzenia raportów i analizy danych.

Tabele

Tables (tabele) w MySQL to struktury danych, które przechowują informacje w postaci wierszy i kolumn. Każdy wiersz odpowiada jednemu rekordowi, a każda kolumna to pole danych w rekordzie.

Klucze

Klucze (keys) w MySQL to specjalne pola w tabeli, które służą do identyfikacji unikalnych rekordów. Istnieją trzy rodzaje kluczy: primary key (klucz główny), foreign key (klucz obcy) i unique key (unikalny klucz).

Relacje

Relacje (relationships) w MySQL to połączenia pomiędzy tabelami, które określają jak dane w jednej tabeli są powiązane z danymi w innej tabeli. Możliwe relacje to np. jeden do wielu (one-to-many) lub wiele do wielu (many-to-many). Zdefiniowanie relacji pozwala na lepsze zarządzanie danymi i unikanie powielania informacji.

Podstawowe zasady projektowania tabel w MYSQL

- Unikanie duplikacji danych: powinno się unikać duplikacji danych w tabelach, aby uniknąć problemów z aktualizacją i wydajnością.

- Normalizacja danych: normalizacja danych polega na rozdzieleniu danych na kilka tabel, aby uniknąć powtarzalności i ułatwić aktualizację danych.

- Określenie kluczy głównych i zewnętrznych: klucz główny jest unikalnym identyfikatorem rekordów w tabeli, natomiast klucz zewnętrzny jest używany do powiązania tabel.

- Określenie typów danych: należy określić właściwy typ danych dla każdej kolumny w tabeli, aby uniknąć błędów i uzyskać optymalną wydajność.
-
- Indeksowanie: indeksowanie jest procesem tworzenia specjalnej struktury danych, która umożliwia szybsze wyszukiwanie danych.

- Unikanie nadmiernego korzystania z funkcji: nie należy przesadzać z korzystaniem z funkcji, ponieważ może to negatywnie wpłynąć na wydajność.

- Śledzenie zmian: powinno się śledzić zmiany w danych, aby umożliwić łatwe wycofanie nieodpowiednich zmian.

Typy danych w MYSQL

- INT - służy do przechowywania liczb całkowitych.
- VARCHAR - służy do przechowywania tekstu o długości do 255 znaków.
- TEXT - służy do przechowywania tekstu o długości do 65,535 znaków.
- DATE - służy do przechowywania daty.
- TIME - służy do przechowywania godziny.
- DATETIME - służy do przechowywania daty i godziny.
- FLOAT - służy do przechowywania liczb zmiennoprzecinkowych.
- DECIMAL - służy do przechowywania liczb zmiennoprzecinkowych o precyzyjnej wartości.

sql Copy code

```sql
CREATE TABLE students (
  id INT,
  name VARCHAR(255),
  birthday DATE
);
```

Tworzenie i wybór bazy danych

Aby utworzyć bazę danych, należy użyć polecenia CREATE DATABASE nazwa_bazy; Aby wybrać bazę danych, należy użyć polecenia USE nazwa_bazy;

sql Copy code

```sql
CREATE DATABASE nazwa_bazy;
USE nazwa_bazy;
```

Tworzenie tabel

Aby utworzyć tabelę, należy użyć polecenia CREATE TABLE nazwa_tabeli (kolumna1 typ_danych, kolumna2 typ_danych, ...);

```sql
CREATE TABLE nazwa_tabeli (
  kolumna1 typ_danych atrybuty,
  kolumna2 typ_danych atrybuty,
  ...
);
```

Atrybuty kolumn

Atrybuty kolumn to dodatkowe parametry, takie jak NOT NULL, UNIQUE, PRIMARY KEY, AUTO_INCREMENT, itp. które określają jak dane będą przechowywane w kolumnie.

- NOT NULL - kolumna nie może zawierać wartości NULL

```sql
CREATE TABLE students (
  id INT NOT NULL,
  name VARCHAR(255) NOT NULL,
  birthday DATE
);
```

- UNIQUE - każda wartość w kolumnie jest unikalna

```sql
CREATE TABLE osoby (
   id INT NOT NULL,
   imie VARCHAR(50) NOT NULL,
   nazwisko VARCHAR(50) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE
);
```

- PRIMARY KEY - kolumna jest kluczem głównym

```sql
CREATE TABLE students (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  birthday DATE
);
```

- AUTO_INCREMENT - kolumna jest automatycznie inkrementowana przy dodawaniu nowych rekordów

```sql
CREATE TABLE osoby (
   id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   imie VARCHAR(50) NOT NULL,
   nazwisko VARCHAR(50) NOT NULL,
   wiek INT NOT NULL
);
```

- FOREIGN KEY - określa, jak tabele są ze sobą powiązane i jaka kolumna jest kluczem obcym w innej tabeli.

```sql
CREATE TABLE students (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  birthday DATE,
  class_id INT,
  FOREIGN KEY (class_id) REFERENCES classes (id)
);
```

- DEFAULT - pozwala na określenie wartości domyślnej dla kolumny

```sql
CREATE TABLE students (
    ID INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    age INT DEFAULT 18
);
```

Indeksy

Indeksy to struktury danych, które umożliwiają szybsze wyszukiwanie danych. Aby utworzyć indeks, należy użyć polecenia CREATE INDEX nazwa_indeksu ON nazwa_tabeli (kolumna);

java Copy code

```
CREATE INDEX nazwa_indeksu ON nazwa_tabeli (kolumna1, kolumna2, ...);
```

Modyfikacja tabel

Aby zmodyfikować tabelę, należy użyć polecenia ALTER TABLE nazwa_tabeli ADD COLUMN kolumna typ_danych; lub ALTER TABLE nazwa_tabeli MODIFY COLUMN kolumna typ_danych;

sql Copy code

```
ALTER TABLE nazwa_tabeli
ADD kolumna3 typ_danych atrybuty;

ALTER TABLE nazwa_tabeli
MODIFY kolumna1 nowy_typ_danych atrybuty;

ALTER TABLE nazwa_tabeli
DROP COLUMN kolumna2;
```

Usuwanie tabel

Aby usunąć tabelę, należy użyć polecenia DROP TABLE nazwa_tabeli;

```sql
DROP TABLE nazwa_tabeli;
```

Umieszczanie danych w bazie MySQL

Umieszczanie danych w bazie MySQL: Aby umieścić dane w bazie danych MySQL, należy użyć instrukcji SQL INSERT. Można w ten sposób wstawić pojedynczy wiersz danych lub wiele wierszy jednocześnie

Instrukcja INSERT TO

Instrukcja INSERT TO jest używana do wstawiania danych do tabeli w bazie danych MySQL.

```sql
INSERT INTO customers (customer_id, customer_name, customer_email)
VALUES (1, 'John Doe', 'john.doe@example.com');
```

Wprowadzanie wielu wierszy

Można wstawić wiele wierszy danych jednocześnie.

```sql
INSERT INTO customers (customer_id, customer_name, customer_email)
VALUES (1, 'John Doe', 'john.doe@example.com'),
       (2, 'Jane Doe', 'jane.doe@example.com'),
       (3, 'Jim Smith', 'jim.smith@example.com');
```

Druga postać instrukcji INSERT

Druga postać instrukcji INSERT umożliwia wstawianie danych do tabeli w bazie danych MySQL, korzystając z danych z innej tabeli.

```sql
INSERT INTO customers (customer_id, customer_name, customer_email)
SELECT id, name, email
FROM temporary_table;
```

Instrukcja SELECT

Jest podstawowym narzędziem do pobierania danych z tabel w języku SQL. Służy do wybierania wierszy i kolumn z bazy danych i tworzenia wyniku zapytania.

```sql
SELECT nazwisko, imie
FROM pracownicy
```

Sortowanie wyników zapytań

Pozwala na uzyskanie wyniku w określonej kolejności. Można to zrobić za pomocą słówka kluczowego ORDER BY i określenia kolumny, po której nastąpi sortowanie.

vbnet Copy code

```
SELECT nazwisko, imie
FROM pracownicy
ORDER BY nazwisko
```

Kryteria pobierania danych

Pozwalają na wybranie tylko tych wierszy, które spełniają określone warunki. Można to zrobić za pomocą słówka kluczowego WHERE i warunku logicznego.

sql Copy code

```
SELECT nazwisko, imie
FROM pracownicy
WHERE stanowisko = 'kierownik'
```

Niepowtarzalność wierszy

Można osiągnąć za pomocą słówka kluczowego DISTINCT, które pozwala na wybranie tylko unikalnych wierszy.

```sql
SELECT DISTINCT stanowisko
FROM pracownicy
```

Ograniczenie wyników zapytań

Pozwala na wybranie tylko określonej liczby wierszy. Można to zrobić za pomocą słówka kluczowego LIMIT i podania liczby wierszy, które mają zostać zwrócone.

```sql
SELECT nazwisko, imie
FROM pracownicy
LIMIT 10
```

Modyfikacja i usuwanie danych

MySQL to relacyjna baza danych, w której można modyfikować i usuwać dane.
Aby zmodyfikować dane, należy wykonać komendę "UPDATE", w której podajemy nazwę tabeli, nowe wartości dla wybranych kolumn oraz warunek określający, które rekordy mają zostać zmodyfikowane.
Aby usunąć dane, należy wykonać komendę "DELETE", w której podajemy nazwę tabeli i warunek określający, które rekordy mają zostać usunięte.
Ważne jest, aby dokładnie określić warunek w komendach "UPDATE" i "DELETE", aby uniknąć niepotrzebnego usuwania lub zmiany danych.

Instrukcja UPDATE

Instrukcja UPDATE w języku SQL pozwala na modyfikację danych w istniejących rekordach w tabeli.

sql Copy code

```sql
UPDATE nazwa_tabeli
SET nazwa_kolumny = nowa_wartość
WHERE warunek;
```

sql Copy code

```sql
UPDATE customers
SET address = 'New York'
WHERE customer_id = 1;
```

Usuwanie danych

by usunąć dane z tabeli, można użyć instrukcji DELETE

sql Copy code

```sql
DELETE FROM nazwa_tabeli
WHERE warunek;
```

sql Copy code

```sql
DELETE FROM customers
WHERE customer_id = 1;
```

Złączenia

W MySQL istnieją trzy główne rodzaje złączeń: wewnętrzne (INNER JOIN), zewnętrzne lewe (LEFT JOIN) i zewnętrzne prawe (RIGHT JOIN). Złączenie polega na łączeniu wyników z dwóch lub więcej tabel na podstawie warunków złączenia. Warunki złączenia to kryteria, które określają, jakie wiersze z jednej tabeli powinny być połączone z wierszami z drugiej tabeli.

Możesz użyć kilku warunków złączenia, takich jak równość lub nierówność pomiędzy wartościami kolumn w dwóch tabelach. Możesz również użyć operatorów logicznych, takich jak AND i OR, aby dostosować warunki złączenia.

- Łączenie wyników zapytań - Polega na połączeniu kilku zapytań SQL w jedno, tak aby uzyskać jednolite dane. Łączenie zapytań jest często używane, gdy chcemy połączyć dane z kilku tabel lub wyodrębnić określone dane z bazy danych. Przykładowy kod:

```
SELECT * FROM table1
UNION
SELECT * FROM table2;
```

- Pobieranie danych z wielu tabel - Polega na pobieraniu danych z kilku tabel w bazie danych w jednym zapytaniu. Jest to użyteczne, gdy chcemy połączyć dane z kilku tabel, aby uzyskać pełen obraz danych. Przykładowy kod:

```
SELECT * FROM table1
JOIN table2
ON table1.column_name = table2.column_name;
```

- Złączenia - Polega na połączeniu danych z kilku tabel na podstawie określonego warunku. Istnieją trzy rodzaje złączeń: złączenie wewnętrzne (INNER JOIN), złączenie zewnętrzne lewe (LEFT JOIN) i złączenie zewnętrzne prawe (RIGHT JOIN). Przykładowy kod:

```
SELECT * FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

- Rodzaje warunków złączenia - Warunki złączenia określają, jakie dane są połączone. Istnieją trzy rodzaje warunków złączenia: warunek równości (ON), warunek zastępstwa (USING) i warunek złączenia za pomocą podzapytania (ON subquery). Przykładowy kod:

```
SELECT * FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name
AND table1.column_name2 = 'value';
```

JOINS

W MySQL istnieją 4 główne rodzaje JOIN-ów: INNER JOIN, LEFT JOIN (LEFT OUTER JOIN), RIGHT JOIN (RIGHT OUTER JOIN) i FULL JOIN (FULL OUTER JOIN). Oto przykłady każdego z nich:

INNER JOIN: Zwraca wiersze, które spełniają warunek łączenia z obu tabel.

vbnet Copy code

```vbnet
SELECT *
FROM table1
JOIN table2
ON table1.column = table2.column;
```

LEFT JOIN (LEFT OUTER JOIN): Zwraca wszystkie wiersze z lewej tabeli (table1) oraz odpowiadające im wiersze z prawej tabeli (table2). Jeśli w prawej tabeli nie ma odpowiadającego wiersza, to wartości z kolumn z prawej tabeli będą NULL.

sql Copy code

```sql
SELECT *
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

RIGHT JOIN (RIGHT OUTER JOIN): Zwraca wszystkie wiersze z prawej tabeli (table2) oraz odpowiadające im wiersze z lewej tabeli (table1). Jeśli w lewej tabeli nie ma odpowiadającego wiersza, to wartości z kolumn z lewej tabeli będą NULL.

sql Copy code

```sql
SELECT *
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

FULL JOIN (FULL OUTER JOIN): Zwraca wszystkie wiersze z obu tabel, niezależnie od tego, czy istnieje odpowiadający wiersz w drugiej tabeli. Jeśli w jednej z tabel nie ma odpowiadającego wiersza, to wartości z kolumn tej tabeli będą NULL.

sql Copy code

```sql
SELECT *
FROM table1
FULL JOIN table2
ON table1.column = table2.column;
```

Funkcje agregujące

Funkcje agregujące to funkcje, które pozwalają na grupowanie danych i wyciąganie informacji zbiorczych. W MYSQL dostępne są następujące funkcje agregujące:

- AVG: oblicza średnią wartość z kolumny.
- COUNT: zwraca liczbę wierszy w grupie.
- MAX: zwraca największą wartość z kolumny.
- MIN: zwraca najmniejszą wartość z kolumny.
- SUM: zwraca sumę wartości z kolumny.
- GROUP_CONCAT: zwraca ciąg znaków, który składa się z wartości z kolumny oddzielonych przecinkami.
- BIT_AND: oblicza bitową iloczyn dla wartości w kolumnie.
- BIT_OR: oblicza bitowy sumę dla wartości w kolumnie.
- BIT_XOR: oblicza bitowy XOR dla wartości w kolumnie.
- Funkcje te są używane w zapytaniach SQL, aby grupować i analizować dane w bazie danych. W połączeniu z klauzulami GROUP BY i HAVING pozwalają na bardzo szczegółowe i precyzyjne analizowanie danych.

SUM - zwraca sumę wartości wybranych kolumn:

```sql
SELECT SUM(column_name)
FROM table_name;
```

AVG - zwraca średnią wartość wybranych kolumn:

```sql
SELECT AVG(column_name)
FROM table_name;
```

\

MIN - zwraca najmniejszą wartość wybranych kolumn:

```sql
SELECT MIN(column_name)
FROM table_name;
```

MAX - zwraca największą wartość wybranych kolumn

```sql
SELECT MAX(column_name)
FROM table_name;
```

COUNT - zwraca liczbę wierszy w tabeli:

```sql
SELECT COUNT(column_name)
FROM table_name;
```

GROUP BY - grupuje wyniki według wartości wybranych kolumn

```sql
SELECT column_name1, AVG(column_name2)
FROM table_name
GROUP BY column_name1;
```

Grupowanie danych

Grupowanie danych w MySQL polega na agregacji danych z jednej lub więcej kolumn z jednej lub kilku tabel w celu uzyskania informacji zbiorczych. Można to zrobić za pomocą polecenia GROUP BY.

Zsumowanie ilości sprzedanych produktów dla każdego producenta

```sql
SELECT manufacturer, SUM(quantity)
FROM sales
GROUP BY manufacturer;
```

Wyliczenie średniej ceny produktów dla każdej kategorii.

```sql
SELECT category, AVG(price)
FROM products
GROUP BY category;
```

Wyliczenie liczby produktów sprzedanych w każdym miesiącu

```sql
SELECT MONTH(date_sold) AS month, SUM(quantity)
FROM sales
GROUP BY MONTH(date_sold);
```

Podzapytania

Podzapytania w MySQL to zapytania wewnątrz innych zapytań. Służą one do pobierania danych z jednej lub kilku tabel i wykorzystywania ich w innym zapytaniu.

Znalezienie imion i nazwisk klientów, którzy zamówili co najmniej jeden produkt z kategorii "Electronics".

sql Copy code

```sql
SELECT first_name, last_name
FROM customers
WHERE customer_id IN (SELECT customer_id
                      FROM orders
                      WHERE product_id IN (SELECT product_id
                                           FROM products
                                           WHERE category = 'Electronics'));
```

Wyliczenie średniej ceny produktów dla każdej kategorii, ale tylko dla kategorii, które posiadają co najmniej jeden produkt.

sql Copy code

```sql
SELECT category, AVG(price)
FROM products
WHERE category IN (SELECT category
                   FROM products
                   GROUP BY category
                   HAVING COUNT(*) >= 1)
GROUP BY category;
```

Podzapytania pozwalają na bardziej złożone manipulacje danymi i są szczególnie przydatne przy łączeniu danych z wielu tabel.

Podzapytania w klauzuli FROM

Podzapytania w klauzuli FROM służą do łączenia danych z wielu tabel w jednym zapytaniu. Są one określane jako podzapytania w klauzuli FROM, ponieważ są umieszczane bezpośrednio w klauzuli FROM głównego zapytania.

Znalezienie nazw produktów i średniej ich ceny w poszczególnych kategoriach.

vbnet Copy code

```vbnet
SELECT p.category, p.product_name, AVG(o.price)
FROM products p
JOIN (SELECT product_id, AVG(price) AS price
      FROM orders
      GROUP BY product_id) o
ON p.product_id = o.product_id
GROUP BY p.category, p.product_name;
```

Podzapytania proste

Podzapytania proste są używane do pobierania danych z jednej lub kilku tabel i wykorzystywania ich w innym zapytaniu. Są one bardzo przydatne w przypadku, gdy chcemy uzyskać informacje zbiorcze, takie jak średnie, sumy lub liczby, na podstawie danych z kilku tabel:

Znalezienie nazw klientów, którzy zamówili produkty z kategorii "Electronics".

sql Copy code

```sql
SELECT first_name, last_name
FROM customers
WHERE customer_id IN (SELECT customer_id
                      FROM orders
                      WHERE product_id IN (SELECT product_id
                                           FROM products
                                           WHERE category = 'Electronics'));
```

Podzapytania skorelowane

Podzapytania skorelowane to podzapytania, w których wynik podzapytania jest używany do filtrowania danych w głównym zapytaniu. Są one bardzo przydatne w przypadku, gdy chcemy uzyskać dane z jednej tabeli na podstawie danych z innej tabeli.

Znalezienie nazw produktów i średniej ich ceny, które zostały sprzedane w ciągu ostatnich 30 dni

sql Copy code

```sql
SELECT product_name, AVG(price)
FROM products p
JOIN (SELECT product_id, AVG(price) AS price
      FROM orders
      WHERE date_sold >= DATE_SUB(NOW(), INTERVAL 30 DAY)
      GROUP BY product_id) o
ON p.product_id = o.product_id
```

Podzapytania w instrukcjach aktualizujących dane

Możliwe jest użycie podzapytań w instrukcjach aktualizujących dane w MySQL. Służą one do pobierania danych z jednej tabeli i aktualizowania innej tabeli na ich podstawie.

sql Copy code

```sql
UPDATE table1
SET column1 = (SELECT column2 FROM table2 WHERE table2.id = table1.id)
WHERE EXISTS (SELECT 1 FROM table2 WHERE table2.id = table1.id);
```

W powyższym przykładzie aktualizujemy wartość kolumny "column1" w tabeli "table1" na podstawie wartości kolumny "column2" w tabeli "table2", gdzie wartość kolumny "id" jest taka sama w obu tabelach.

Perspektywy - widoki

MySQL Views to virtual tables, które reprezentują wynik wybranego zestawu danych z jednej lub kilku tabel. Mogą być używane do uzyskania bardziej złożonego widoku na dane, które zostały złożone z kilku tabel w bazie danych.

Tworzenie widoku

```sql
CREATE VIEW sales_view AS
SELECT customers.name, orders.order_date, SUM(order_details.quantity * order_detail
FROM customers
INNER JOIN orders ON customers.customer_id = orders.customer_id
INNER JOIN order_details ON orders.order_id = order_details.order_id
GROUP BY customers.name, orders.order_date;
```

Użycie widoku

```sql
SELECT * FROM sales_view;
```

Wynik powyższego zapytania będzie zawierać wynik wybranego zestawu danych z trzech tabel customers, orders i order_details. Zestaw danych będzie zawierał nazwę klienta, datę zamówienia i łączną sprzedaż dla każdej transakcji.

Transakcje

Transakcje w systemach baz danych to grupa jednoznacznie powiązanych operacji, które są wykonywane jako jedna całość. Transakcja zaczyna się od rozpoczęcia i trwa do momentu zakończenia lub wycofania. W przypadku błędu w trakcie transakcji, wszystkie wprowadzone zmiany są cofane, aby zachować integralność danych w bazie danych.

```sql
START TRANSACTION;
```

Instrukcja transakcji w MySQL pozwala na objęcie instrukcji jedną transakcją.

```sql
START TRANSACTION;
UPDATE tabela SET kolumna = wartość WHERE warunek;
COMMIT;
```

```sql
BEGIN TRANSACTION;
INSERT INTO table1 (column1, column2) VALUES (value1, value2);
INSERT INTO table2 (column1, column2) VALUES (value3, value4);
COMMIT;
```

Wycofanie transakcji pozwala na anulowanie wprowadzonych zmian. Przykład kodu wycofywania transakcji w MySQL

```sql
START TRANSACTION;
UPDATE tabela SET kolumna = wartość WHERE warunek;
ROLLBACK;
```

```sql
BEGIN TRANSACTION;
INSERT INTO table1 (column1, column2) VALUES (value1, value2);
INSERT INTO table2 (column1, column2) VALUES (value3, value4);
ROLLBACK;
```

Izolacja transakcji polega na odizolowaniu transakcji od innych transakcji, tak aby jedna transakcja nie miała wpływu na inne. W MySQL można ustawić poziom izolacji transakcji za pomocą instrukcji SET TRANSACTION ISOLATION LEVEL.

```sql
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
START TRANSACTION;
```

```sql
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
SELECT * FROM table1 WHERE column1 = value1;
SELECT * FROM table2 WHERE column2 = value2;
COMMIT;
```

Więzy integralności

Integralność danych to zasada bazodanowa, która zapewnia, że dane w bazie danych są spójne i aktualne. Integralność danych jest zachowana przez wprowadzenie ograniczeń i reguł, które nie pozwalają na dodanie, modyfikację lub usunięcie danych, które byłyby sprzeczne z regułami.

sql Copy code

```sql
CREATE TABLE orders (
  order_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT NOT NULL,
  order_date DATE NOT NULL,
  CONSTRAINT fk_customer_id FOREIGN KEY (customer_id)
    REFERENCES customers (customer_id)
    ON DELETE CASCADE
);
```

Definiowanie klucza obcego:
Klucz obcy jest to kolumna lub kolumny w jednej tabeli, które są powiązane z kolumnami w innej tabeli. Klucz obcy jest używany do zapewnienia integralności danych w bazie danych, zapobiegając dodawaniu, modyfikacji lub usuwaniu danych, które byłyby sprzeczne z regułami.

sql Copy code

```sql
CREATE TABLE customers (
  customer_id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL
);

CREATE TABLE orders (
  order_id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT NOT NULL,
  order_date DATE NOT NULL,
  CONSTRAINT fk_customer_id FOREIGN KEY (customer_id)
    REFERENCES customers (customer_id)
    ON DELETE CASCADE
);
```

Dodawanie i usuwanie więzów:

Wiązania pozwalają na dodawanie i usuwanie danych z jednej tabeli, które automatycznie aktualizują odpowiednie dane w innej tabeli. Są to reguły, które zapewniają integralność danych w bazie danych.

sql Copy code

```sql
ALTER TABLE orders
ADD CONSTRAINT fk_customer_id FOREIGN KEY (customer_id)
REFERENCES customers (customer_id)
ON DELETE CASCADE;
```

sql Copy code

```sql
ALTER TABLE Orders
ADD CONSTRAINT FK_Orders_Customers
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);

ALTER TABLE Orders
DROP CONSTRAINT FK_Orders_Customers;
```

Opis silników baz danych

- MySQL to popularny system zarządzania bazami danych relacyjnych, który jest szeroko stosowany w aplikacjach internetowych. Jest on darmowy i łatwy w instalacji oraz konfiguracji, co sprawia, że jest popularny wśród programistów.

- PostgreSQL to system zarządzania bazami danych relacyjnych, który jest szeroko stosowany w aplikacjach biznesowych. Jest on wolnym oprogramowaniem i posiada szerokie możliwości, takie jak obsługa danych geograficznych i wysokiej dostępności.

- SQLite to system zarządzania bazami danych, który jest wbudowany w wiele aplikacji i urządzeń mobilnych. Jest on bardzo mały i lekki, co sprawia, że jest idealny do zastosowań z wymaganiami ograniczenia miejsca i zasobów.

- Microsoft SQL Server (MS SQL) to system zarządzania bazami danych, który jest oferowany przez Microsoft. Jest on szeroko stosowany w aplikacjach biznesowych i posiada szerokie możliwości, takie jak rozbudowane narzędzia analityczne i wysoka dostępność.

- Oracle to system zarządzania bazami danych, który jest szeroko stosowany w dużych i zaawansowanych aplikacjach biznesowych. Jest on oferowany przez firmę Oracle i posiada szerokie możliwości, takie jak wysoka dostępność, wysoka wydajność i obsługa wielu platform.

	MySQL	PostgreSQL	SQLite	MS SQL	Oracle
Dostępność	Open source	Open source	Open source	Proprietary	Proprietary
Skalowalność	Średnia	Wysoka	Niska	Wysoka	Wysoka
Wsparcie	Duże	Duże	Małe	Duże	Duże
Funkcjonalność	Średnia	Wysoka	Niska	Wysoka	Wysoka
Bezpieczeństwo	Średnie	Wysokie	Niskie	Wysokie	Wysokie
Wydajność	Wysoka	Wysoka	Niska	Wysoka	Wysoka
Ceny licencji	Darmowe	Darmowe	Darmowe	Opłacalne	Opłacalne
Typ bazy danych	Relacyjna	Relacyjna	Relacyjna	Relacyjna	Relacyjna
Obsługiwane języki	PHP, Java	C, Perl, Python	C	T-SQL	PL/SQL
Platformy	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS, iOS, Android	Windows	Windows, Linux, Unix, zCloud