dokodu

MYSQL

JAK ZACZĄĆ?

W KSIĄŻCE

Typy danych
Tworzenie tabel
Zapytanie Select
Warunki
Wykorzystanie funkcji
Złączenia
Grupowanie danych
Podzapytania
Widoki

DOKODU.IT

O Autorze

Rozdział 0: Zaczynamy!

Na wstępie, bardzo dziękuję Ci za okazane zaufanie. W książce tej zajmiemy się tematem baz danych, które stanowią bardzo istotny fundament współczesnego świata informatyki. Niezależnie od tego czy jesteś (lub chcesz pracować) jako programista, tester czy też analityk zawsze temat baz danych będzie Ci niezwykle potrzebny. Dlaczego tak jest? Ponieważ absolutnie wszystkie dane Twojej aplikacji takie jak informacje o użytkownikach, zamówieniach czy nawet zawartość wpisów na bloga gdzieś muszą się znaleźć. Miejscem ich przechowywania są właśnie bazy danych. Bez nich stworzenie aplikacji czy jej przetestowanie jest absolutnie niemożliwe.

Fragmenty tej książki pochodzą z prezentacji kursu o nazwie skumajbazy.pl, który prowadzę. Jeżeli szukasz kompletnej dawki wiedzy poświęconej różnym bazom danych to zachęcam do dołączenia!

Jak wygląda praca z bazami danych?

Aby dobrze zrozumieć, jak przebiega praca, musimy porozmawiać o tym, jak działa podejście klient-serwer. Oznacza to, że do pracy potrzebujemy dwóch stron: serwera, czyli programu bazodanowego działającego na serwerze, oraz klienta, czyli programu, który będzie inicjował taką komunikację. Tematem serwera zajmiemy się już za moment. Na początek porozmawiajmy o kliencie.

To, jakiego klienta używam podczas pracy, zależy od tego, co tak naprawdę chcę osiągnąć. Jeżeli programuję, to po prostu tworzę przeze mnie aplikację, która wysyła polecenia do serwera bazodanowego, a następnie zmienia jego stan - dodaje nowe dane, usuwa je lub zmienia. Jednak bardzo często, szczególnie przy bardziej złożonych zapytaniach, potrzebuję mieć możliwość szybkiego sprawdzenia, czy

przygotowane przeze mnie zapytanie działa. Wówczas znacznie wygodniej jest mi po prostu użyć aplikacji desktopowej, na przykład DBeavera. Dzięki jego pomocy łatwo jest sprawdzić, czy wszystko działa prawidłowo, a później po prostu przenieść zapytanie do aplikacji i odpowiednio przerobić dane wewnątrz niej.

Część druga naszych rozważań dotyczy strony serwera. Tutaj wszystko zależy od tego, jakiej bazy danych używamy. Ta książka skupia się na bazach danych relacyjnych oraz na języku SQL, który służy do pisania zapytań. Więcej na temat rodzajów danych porozmawiamy za moment.

Niech nazwa "serwer" Cię nie zmyli. Nie musisz kupować żadnej dodatkowej usługi. Tak naprawdę do wykonywania wszystkich ćwiczeń wystarczy Twój komputer. Więcej na ten temat opowiem Ci już za moment.

Rozdział 1: Rodzaje Baz Danych

Bazy danych stanowią fundament współczesnego świata informatyki i przechowują w sobie ogromne ilości cennych informacji. W miarę jak technologia rozwija się w zawrotnym tempie, tak samo ewoluują również rodzaje baz danych, które wykorzystujemy do przechowywania i zarządzania danymi. Ten rozdział wprowadzi Cię w fascynujący świat różnych rodzajów baz danych, zrozumiesz, jakie problemy są w stanie rozwiązywać i dlaczego są niezbędne w dzisiejszym środowisku informatycznym.

Istnieją dwie główne kategorie baz danych, które wyłoniły się jako najważniejsze w świecie technologii: relacyjne i nierelacyjne. Bywa, że podział ten jest definiowany również jako bazy danych SQL i NoSQL. To co ważne to fakt, że w tym drugim przypadku NO to skrót od Not Only, wszach nie tylko SQLem człowiek żyje. Jednak w tej publikacji skupimy się wyłącznie na bazach relacyjnych.

Różnica między nimi polega na sposobie organizacji i przechowywania danych, a także na zastosowaniach, do których są najlepiej dostosowane. Oto skrótowe omówienie każdej z tych kategorii:

Bazy Danych Relacyjne

Bazy danych relacyjne są powszechnie stosowane od kilku dziesięcioleci. Ich struktura opiera się na tabelach, które przechowują dane w postaci relacji. Ten rodzaj baz danych używa języka zapytań SQL (Structured Query Language) do manipulacji danymi. Modele relacyjne są idealne dla sytuacji, w których istnieje potrzeba precyzyjnego i strukturalnego przechowywania danych, takich jak dane finansowe, dane klientów czy dane związane z działalnością przedsiębiorstwa.

Jak sobie to wyobrazić? Na pewno zdarzyło Ci się kiedyś przechowywać dane lub choć widzieć na oczy najprostszą nawet tabelę. Nazwane kolumny, a wszystkie Twoje dane są przechowywane w wierszach. To właśnie jest relacja.

Kiedy patrzysz na taką tabelę wiesz, że nawet jeżeli nie wpiszesz wartości do którejś kolumny to po prostu pozostaje ona pusta, ale ten rekord posiada komplet danych.

W przypadku baz danych nierelacyjnych wszystkie przechowywane dane mogą mieć różną strukturę. Zupełnie tak jak gdyby potrzebne było Ci przechowywanie zupełnie nieregularnych danych po prostu przechowywanych w jednej kolekcji.

Bazy danych nierelacyjne

Wraz z rozwojem internetu i nowoczesnych aplikacji, bazy danych nierelacyjne zyskały na popularności. Są one bardziej elastyczne i skalowalne niż ich odpowiedniki relacyjne. Bazy danych nierelacyjne przechowują dane w różnorodnych formatach, takich jak dokumenty, grafy czy kolumny. Są szczególnie przydatne w przypadku dużych ilości danych, które niekoniecznie muszą być znormalizowane, jak na przykład dane użytkowników mediów społecznościowych czy dane z urządzeń internetu rzeczy (IoT).

Być może kojarzysz format pliku .json wyobraź sobie, że posiadasz pliki json opisujące osoby. Niektóre mogą posiadać strukturę zawierającą wyłącznie imiona i nazwiska, inne imiona, nazwiska oraz wiek itd. W ten sposób możesz przechowywać bardzo różne struktury wszystkie nazwane w ten sam sposób.

Rozdział 2: Przygotowanie środowiska pracy

OK! Zaczynajmy. Na początek zapoznajmy z narzędziami których będziemy używali. Na temat DBeavera już Ci wspominałem. Do połączenia z bazą danych przyda Ci się również sam serwer baz danych. Będzie Ci potrzebny loginu, hasło oraz nazwy bazy danych.

W naszym przypadku będziemy korzystali z bazy danych wypożyczalni filmów, o nazwie Sakila. Jest to zupełnie darmowa baza i dane w niej zgromadzone służa wyłącznie celom edukacyjnym. Jeżeli będzie Ci ona potrzebna, możesz ją pobrać tutaj: https://dev.mysql.com/doc/sakila/en/sakila-installation.html

Najlepszym sposobem uruchomienia jej, bez dodatkowych kosztów będzie zainstalowanie sobie MYSQL lokalnie np. za pomocą programu XAMPP, a następnie zaimportowanie danych wg. opisu z powyższego linka.
W ten sposób uzyskasz zupełnie darmowy poligon doświadczalny do pracy z SQLem.

Pamiętaj, że jeżeli stworzenie tej platformy będzie stanowiło dla Ciebie problem to zawsze możesz odwiedzić nasz serwer Discord: https://dokodu.dev/discord by zapytać o pomoc.

Inny sposób na pracę z bazą danych

W mojej pracy bardzo często wykorzystuję narzędzie o nazwie Docker. Jest to to platforma do konteneryzacji aplikacji, która umożliwia izolację i uruchamianie aplikacji w izolowanych środowiskach zwanych kontenerami.

Kontenery są przenośnymi jednostkami oprogramowania, które zawierają wszystko, co jest potrzebne do uruchomienia aplikacji, w tym kod źródłowy, zależności i konfigurację. Docker ułatwia deweloperom i administratorom zarządzanie aplikacjami oraz ich wdrożenie na różnych środowiskach, zapewniając spójność i skalowalność. Docker jest często wykorzystywany w środowiskach deweloperskich, testowych i produkcyjnych do automatyzacji procesów wdrażania i zapewnienia izolacji aplikacji.

Dzięki Dockerowi można tworzyć, udostępniać i uruchamiać kontenery z łatwością na różnych platformach.

Jak może Ci się on przydać?

Wystarczy, że zainstalujesz go, a następnie za pomocą instrukcji sporządzonej na tym githubie: https://github.com/sakiladb/mysql Za pomocą jednego polecenia pobierzesz kontener z tą bazą danych i po kilku minutach będziesz mógł się do niej łatwo podłączyć. Jak podłączyć się do bazy danych opisałem w kolejnym rozdziale.

Jak zainstalować Dockera?

Przyznam, że nie jest to najłatwiejszy proces jednak jestem przekonany, że wchodząc w świat Dockera znacząco poszerzysz swoje horyzonty. Jest to narzędzie, które wykorzystuje absolutnie cały świat IT. Proces instalacyjny na Windowsie opisany jest tutaj: https://docs.docker.com/desktop/install/windows-install/

Druga sprawa.. kiedy to piszę jest początek października 2023 roku. Wiem, że trwają pracę nad kursem Dockera oraz filmami wprowadzającymi w ten piękny świat. Jeżeli masz chwilę możesz sprawdzić czy na moim kanale nie pojawiły się filmy, które pomogą Ci w ten świat wejść znacznie łatwiej.

Rozdział 3: Jak podłączyć się do bazy danych?

Podłączać do bazy danych można się na kilka sposobów. Można to zrobić za pomocą dedykowanego programu - klienta np. DBeavera. W ten sposób zarządzanie bazą danych jest wygodniejsze, jednak wymaga instalacji wspomnianego programu.

Druga opcja polega na skorzystaniu z aplikacji webowej, o którą zatroszczy się nasz usługodawca. W tym przypadku wiele zależy od tego do jakiej bazy danych się łączymy. Jeżeli jest to MySQL to najpopularniejszym programem tego typu jest phpMyAdmin, jeśli do PostgreSQL to korzystamy z pgAdmin.

Moim zdaniem w tym starciu zdecydowanie wygrywa DBeaver. Jest to aplikacja, która jest prawdziwym kombajnem umożliwiającym podłączenie się do praktycznie każdej bazy danych w bardzo wygodny i komfortowy sposób.

Ważna uwaga: Niezależnie od tego w jaki sposób zdecydujemy się połączyć z naszymi danymi, wszystkie zapytania które wykonują się pod spodem działają tak samo.

Skąd wziąć DBeavera?

Z Internetu :-) Na stronie https://dbeaver.io/ znajdziesz przycisk “Download” i zależnie od Twojej platformy będziesz mógł pobrać wersję Community oraz PRO. Mówiąc szczerze od kiedy pamiętam korzystam z wersji Community i w zupełności mi ona wystarczała.

Instalacja DBeavera pod Windowsem przypomina instalację absolutnie wszystkich innych programów. Po prostu odpowiednio wiele razy musimy przycisnąć przycisk “Next”. Jeżeli chodzi o pozostałe platformy to również nie sprawiło mi to żadnego problemu.

Podłączenie się do bazy danych

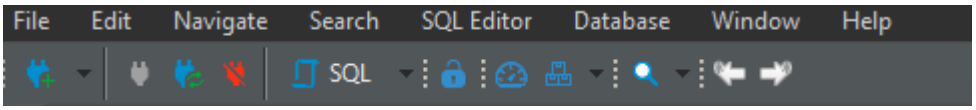
Aby połączyć się z bazą danych będziesz potrzebował czterech ważnych informacji. Są nimi:

- Adres serwera bazy danych, nazywany często również hostem. Może być on udostępniony Ci w dwóch wersjach: adres IP lub adres domenowy. Jeżeli korzystasz z lokalnej bazy danych(zainstalowanej na Twoim komputerze) to możesz wpisać localhost lub 127.0.0.1 to dwa określenia wskazujące na Twój komputer.
- Nazwa użytkownika i hasło, to wartości które zależą od sposobu jak uruchomiłeś bazę danych. Jeżeli wybrałeś ścieżkę z XAMPem to nazwą

użytkownika jest “root”, a hasło jest puste. Jeżeli Twoim wyborem był Docker to zgodnie z informacją na githubie nazwa użytkownika to sakila, a hasło to p_ssW0rd.

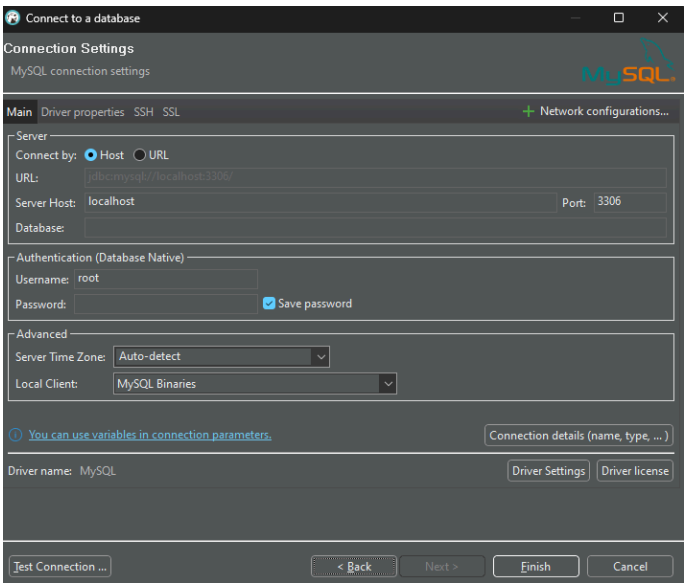
- Nazwa bazy danych i tu również to zależy od sposobu instalacji. Jeżeli Twoim wyborem jest Docker to ta wartość to sakila. Pozostałe nazwy zależą od Twoich ruchów. Jeśli masz problem ze znalezieniem odpowiedzi napisz do mnie na Discordzie https://dokodu.dev/discord

Kolejnym krokiem naszej układanki jest uruchomienie Dbeavera.



Następnie należy kliknąć w ten przycisk, który wygląda jak wtyczka do kontaktu pod menu “File” i zależnie od używanej przez nas bazy danych wybrać odpowiedni sterownik. W naszym przypadku jest to MySQL.

Kolejny ekran polega na zebraniu wprowadzonych danych o których pisałem CI wcześniej.



Kiedy wypełnisz pola “Server Host”, “Database”, “Username” oraz “Password” wciśnij przycisk Test Connection w lewym dolnym rogu. Jeżeli łączysz się z tym typem bazy danych po raz pierwszy to DBeaver pobierze również sterownik potrzebny do nawiązania takiego połączenia po czym otrzymasz dostęp do bazy danych.

Zapewne przebierasz już nogami i chcesz zacząć pisać pierwsze zapytania. Wcalee Ci się nie dziwię. Jednak zanim przejdziemy dalej musimy porozmawiać chwile o kilku bardzo ważnych kwestiach bez których tak naprawdę nie ruszymy dalej.

Rozdział 4: Ważne informacje

Pamiętasz jak mówiłem Ci, że relacyjna baza danych przypomina trochę tabelę? Pójdźmy o krok dalej.. widziałeś kiedyś plik Excela? Jestem przekonany, że kiedyś tak.

Każdy Excelowy plik nazywany jest skoroszytem. W ramach takiego skoroszytu posiadamy Arkusze: Arkusz1, Arkusz2, Arkusz3. Możemy im oczywiście zmieniać nazwy i tak często nazywamy je np. Zamówienia, Klienci, Zwroty, Produkty itp.

Każdy arkusz zawiera dane tabelaryczne. Wiem, że nie zawsze tak jest.. ale na potrzeby tej książki załóżmy, że nasze tabele posiadają nazwy kolumn. Zatem w momencie w którym patrzysz na arkusz o nazwie Produkty możesz zauważyć, że kolumny nazywają się dla przykładu: Id, Nazwa, Cena, ilość w magazynie, data przyjęcia itd.. Ta wiedza bardzo ułatwi Ci zrozumienie relacji.

Zależy mi byś zwrócił uwagę na jeszcze jedną kwestię.. każda z tych kolumn ma różne wartości i nie mam tu na myśli tylko treści, które znajdują się np. w kolumnie nazwa, ale również fakt, że każda z tych kolumn posiada różne typy. Niektóre zawierają tekst, inne są liczbami całkowitymi, jeszcze inne posiadają wartość dziesiętną, a coś innego jest jeszcze datą.

Z punktu widzenia Excela takie wartości wynikają głównie z formatowania, bo jeśli pamięć mnie nie myli wszystko w Excelu jest liczbą (tylko różnie sformatowaną) oraz tekstem, ale tym nie musimy się przejmować. W przypadku baz danych określanie odpowiednich typów to niezwykle ważna umiejętność.

Baza danych, tabela, kolumna..

Dobra, skoro choć trochę udało mi się pobudzić Twoją wyobraźnię i widzisz jarzący się różnymi kolorami i danymi arkusz Excela czas wrócić do powodu dla którego czytasz tę książkę. Chcesz nauczyć się korzystać z baz danych, prawda?

Zacznijmy od początku. Podczas połączenia musiałeś w DBeaverze wprowadzić “bazę danych”. Tak naprawdę w serwerze bazodanowym może być bardzo dużo takich baz. Do niektórych możesz mieć dostęp, a inne mogą być dla Ciebie niedostępne.

Wyobraź sobie, że serwer to po prostu komputer stojący gdzieś w świecie. Jednocześnie może z niego korzystać duża liczba użytkowników, natomiast bazy danych często przechowują dość wrażliwe informacje.

Baza danych

Wracając na moment do analogii z Excelem. Pojęcie bazy danych możemy rozumieć jako pojedynczy skoroszyt Excela. Oznacza to, że każda pojedyncza baza danych:

- posiada swoją nazwę
- może składać się z wielu arkuszy(tabel).
- ma odpowiednie kodowanie

Warto też dodać, że baza danych jest to przestrzeń w której przechowujemy różne tabele. Bardzo często kiedy tworzymy pojedynczą stronę internetową to wprowadzamy do niej jedną bazę danych.

Kodowanie

O co chodzi z tym kodowaniem? W skrócie, jeśli zapisujemy dane zawierające polskie znaki to zależy nam na tym aby nie pojawiały nam się konkretne “krzaki”. Tworząc bazę musimy określić jej kodowanie. Chodzi tu o to, że dla komputera jest to bardziej skomplikowane, gdyż każda litera jest rozumiana jako liczba, która w zależności od wybranego kodowania może być różna.

Dlatego wybór odpowiedniego kodowania jest bardzo ważny. Osobiście najczęściej używam utf8mb4. Zmiana rodzaju kodowania po utworzeniu bazy danych bywa kłopotliwa. Dobrze przemyśl więc ten wybór.

Tabela

W naszej analogii, w której porównujemy bazy danych do Excela możemy przyjąć, że pojedyncza tabela jest arkuszem, jedną zakładką w której trzymamy dane.

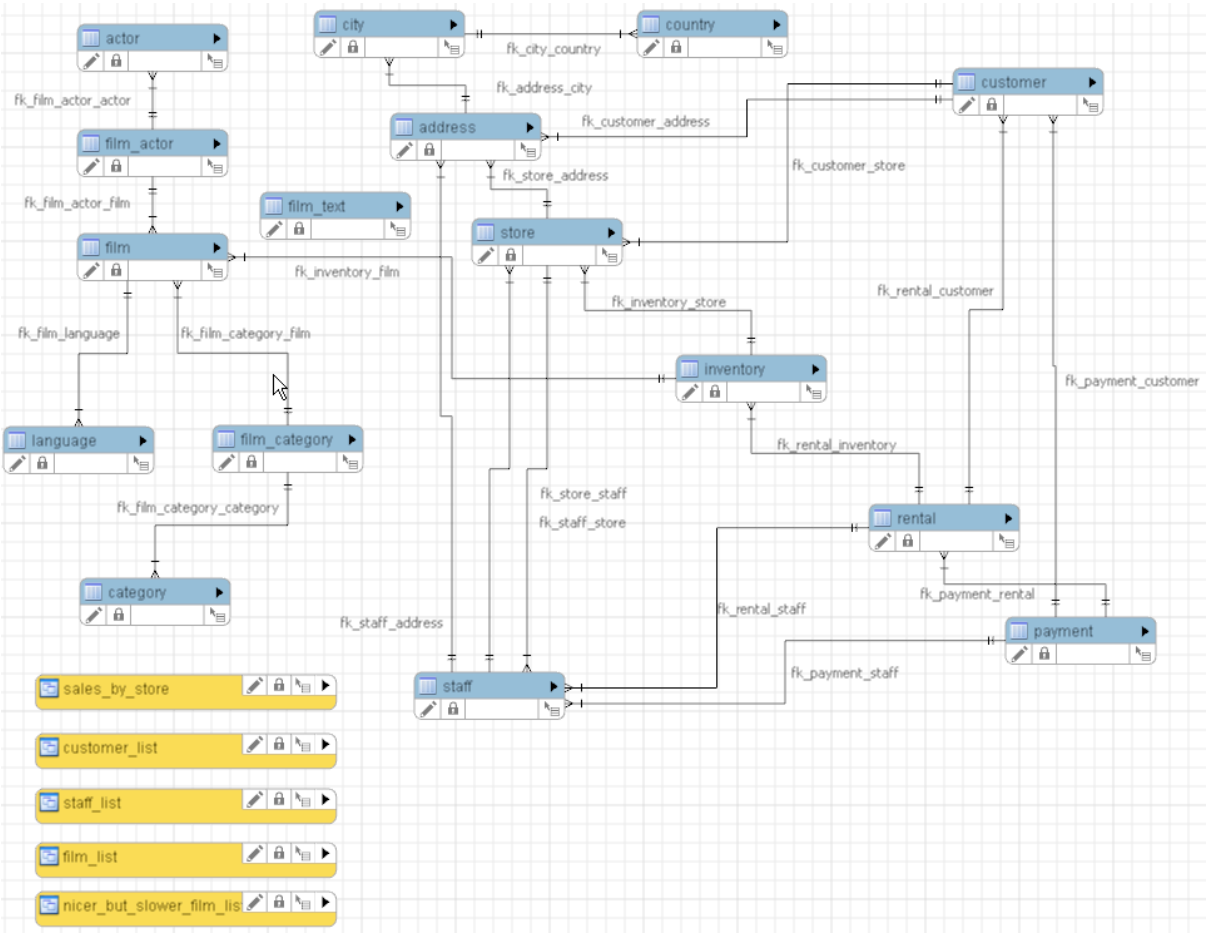
Kiedy myślisz o projekcie wyobrażasz sobie różne obiekty które chcesz przechowywać w bazie danych. Dla przykładu jeżeli myślisz o sklepie internetowym to możesz wyobrazić sobie zamówienia, produkty, użytkowników, utworzone koszyki i inne tego typu elementy.

Każda tabela przechowuje grupę danych opisujących jeden typ takiego obiektu, np. w wypożyczalni filmów mogą być to tabele przechowujące użytkowników, filmy, wypożyczenia, aktorów.

Zasada dotycząca ogólnego nazewnictwa tabel jest następująca: używamy języka angielskiego, zamiast spacji wprowadzamy znak podkreślenia i rzeczowniki zapisujemy w liczbie mnogiej. Odnosząc się do powyższego przykładu moglibyśmy więc wyróżnić tabele: users, movies, rentals, actors.

Tak jak wspominałem wcześniej. Podczas tego warsztatu będziemy wykorzystywali przykładową bazę danych o nazwie: sakila, która jest udostępniona za darmo na stronie https://dev.mysql.com/doc/sakila/en/.

Schemat tej bazy danych wygląda następująco:



Obrazek pochodzi z https://dev.mysql.com/doc/sakila/en/sakila-structure.html

Typy danych

Dobrze, wiemy już jak wyglądają tabelę czas zająć się kolumnami. Tu jeżeli chodzi o nazewnictwo również króluje język angielski, zamiast spacji używamy podkreślenia.. i na tym mógłbym zakończyć. Gdyby nie fakt, że każda z kolumn posiada również swój typ.

Pamiętasz gdy mówiłem Ci, że niektóre dane mogą być liczbami, tekstem lub datami? Właśnie o to tutaj chodzi. na podstawie typu silnik bazodanowy będzie wiedział jak przechowywać daną wartość oraz jakiego typu operacje możemy na takiej kolumnie wykonywać.

Reasumując: każda kolumna może przechowywać dane różnych typów, np. liczbowy, tekstowy, data. Wybór odpowiedniego typu leży na naszej głowie i wszystko zależy od tego jakie wartości chcemy w konkretnej kolumnie przechowywać.

W zależności od tego czy pracujemy z MySQLem czy z PostgreSQL czy inną bazą danych typy mogą się delikatnie różnić, jednakże wystarczy szybki rzut oka do dokumentacji by w lot przypomnieć sobie jakiego typu danych użyć w danej sytuacji.

Najpopularniejsze typy danych to:

- Liczbowy:
    - INT - liczba całkowita
    - FLOAT - liczba zmiennoprzecinkowa
    - BOOLEAN - wartość logiczna prawda/fałsz
- Tekstowy:
    - VARCHAR - wyrażenie tekstowe od 1 do 255 znaków
    - TEXT - tekst dłuższy niż 255 znaków
- Data i czas
    - DATE - data
    - DATETIME - data i czas

PostgreSQL również wyróżnia różne typy danych, podobnie jak MySQL. Oto podobne zestawienie typów danych w PostgreSQL:

- Liczbowe:
    - INTEGER - liczba całkowita (o różnych rozmiarach, np. INT, SMALLINT, BIGINT)
    - NUMERIC - liczba zmiennoprzecinkowa o dokładności arytmetyki liczby stałoprzecinkowej
- Zmiennoprzecinkowe:
    - REAL - liczba zmiennoprzecinkowa pojedynczej precyzji
    - DOUBLE PRECISION - liczba zmiennoprzecinkowa podwójnej precyzji
- Logiczny: BOOLEAN - wartość logiczna prawda/fałsz
- Tekstowe:
    - CHAR(n) - znak stałej długości, gdzie n to liczba znaków
    - VARCHAR(n) - znak o zmiennej długości, gdzie n to maksymalna liczba znaków
    - TEXT - tekst dłuższy niż VARCHAR
- Data i czas:
    - DATE - data
    - TIME - godzina
    - TIMESTAMP - data i czas
    - INTERVAL - interwał czasowy

Warto nadmienić, że poza typami podstawowymi istnieją również bardziej zaawansowane typy danych, takie jak ARRAY, JSON, UUID i wiele innych, które PostgreSQL obsługuje, co czyni go bardzo elastycznym systemem zarządzania bazami danych.

Dobór odpowiedniego typu danych ma niebagatelny wpływ na przechowywane dane, szybkość zapytań oraz rodzaj bazy danych. Zmiana typu danych, w momencie gdy znajdują się w niej już wpisy jest bardzo ryzykowna i może wiązać się z utratą informacji.

Kolumny

Tworząc nową tabelę musimy dokładnie wiedzieć jakiego rodzaju kolumny(pola) będą nam potrzebne. Weźmy na warsztat naszą tabelę przeznaczoną do trzymania danych o filmach. Struktura tabeli odpowiedzialnej za trzymanie informacji o filmami w sakilli wygląda następująco:

Przykładowo, tabela film może składać się z następujących kolumn:

film_id - INT, NOT NULL, AUTO_INCREMENT

Właściwość NOT NULL oznacza, że pole to musi być zawsze wypełnione. W przypadku w którym użytkownik spróbuje dodać wpis nie uzupełniając pola, które posiada taki atrybut SQL wypisze błąd.

Druga cecha film_id to AUTO_INCREMENT, którą może posiadać tylko jedna kolumna. Oznacza to, że każdy nowy rekord będzie posiadał automatycznie zwiększający się identyfikator, który będzie stanowił unikatowy wyróżnik, czyli wartość na podstawie której bez wątpliwości będziemy mogli wskazać konkretny wiersz w tabeli.

Jak budować zapytania?

Tytularny SQL jest językiem służącym do wykonywania zapytań do bazy danych. Zapytania te służą do pobierania(SELECT), aktualizacji(UPDATE), usuwania(DELETE) bądź dodawania(INSERT) nowych danych.

Zapytania w SQL możemy pisać dla różnych baz danych, do najpopularniejszych należą: Oracle, MySQL i PostgreSQL. Różnice pomiędzy poszczególnymi bazami są

nieznaczne, jednak może zdarzyć się, że niektóre zapytania które znajdziesz poniżej nie będą działały. Na tym poziomie zaawansowania nie powinno się to jednak zdarzyć.

Wszystkie zapytania wprowadzać będziemy z użyciem przeglądarki.

SELECT

Jak sama nazwa wskazuje, SELECT służy do pobierania wyników. Wynikiem może być ilość wierszy lub wynik jakiegoś działania matematycznego, operacji. Najprostsze zapytanie SELECT wygląda następująco:

```
SELECT 2+2
```

Jak łatwo przewidzieć wynikiem tego zapytania będzie liczba 4. W SQLu możesz korzystać ze wszystkich operatorów arytmetycznych i logicznych.

Operatory matematyczne

DZIAŁANIE	OPERATOR
Dodawanie	+
Odejmowanie	-
Mnożenie	*
Dzielenie	/

Oczywiście kolejność działań jest zachowana i aby ją zmienić można używać nawiasów.

Budowa zapytania

Tak jak wspomniałem, SELECT służy głównie do pobierania danych z bazy. Podstawowa forma tego zapytania wygląda następująco:

```
SELECT kolumny FROM tabela WHERE warunki
```

O tyle o ile wyrażenia zapisane na szaro są niezmienne, tak elementy które są czerwone mogą się zmieniać i warto na temat każdego poświęcić chwilę. Do rzeczy.

Kolumny

Tak jak pokazałem powyżej, w tym miejscu mogą znaleźć się różne dane.

- * operator gwiazdki, oznacza on że interesują nas wszystkie kolumny z tabeli. Zostaną one wówczas wypisane w standardowej kolejności.
- kolumny rozdzielone przecinkami podajemy nazwy kolumn, jeżeli nazwa zawiera spacje(co nie jest zbyt dobrym pomysłem) należy pamiętać o tzw. backticku czyli ` (znajduje się on na tym samym klawiszu na klawiaturze co ~ np. SELECT `Dane osobowe`, email FROM ...)
- wyrażenie np. suma dwóch kolumn, lub operacja arytmetyczna.
  np. SELECT wynagrodzenie+premia FROM…
- DISTINCT kolumna, za pomocą tego wyrażenia mamy możliwość pobrania unikatowych wartości z tylko jednej kolumny

Jest jeszcze kilka opcji, jednak na tym etapie wolałbym ich nie wymieniać by nie komplikować sprawy. Warto jednak dodać, że każda kolumna lub operator może posiadać swój alias co często poprawia czytelność zapytania. Aby dodać taki alias, należy za operacją lub kolumną użyć słowa kluczowego AS alias.

```
np. SELECT wynagrodzenie+premia as `Do wyplaty`....
```

Efektem tego zapytania będzie utworzenie nowej kolumny “Do wyplaty”, która będzie zawierała sumę dwóch innych kolumn.

Tabela

W przypadku tego wyrażenia sprawa będzie prosta, podajemy po prostu nazwę tabeli z której chcielibyśmy pobrać wymienione za SELECTem kolumny.

Zadania

1. Pobierz wszystkich aktorów z tabeli actor, a następnie ogranicz wyświetlane kolumny tylko do imienia i nazwiska.

2. Zamień angielską nazwę kolumny first_name na “Imię” oraz “last_name” na “Nazwisko”.

3. Z tabeli film pobierz tytuły filmów, rental_rate(cenę wypożyczenia) oraz length(długość filmu). Sprawdź jaka jest cena za każdą minutę wypożyczonego filmu (podziel cenę przez długość)

Warunki

Warunki za wherem ograniczają ilość zwracanych wierszy. Budowa tych warunków wymaga dobrej znajomości zasad logiki. Należy pamiętać, że po wpisaniu kwerendy(zapytania) wyświetlą się tylko te wiersze dla których formuła znajdująca się za where będzie prawdziwa.

AND czy OR

Podczas budowania zapytań musimy znać podstawową różnicę pomiędzy koniunkcją(AND) i alternatywą(OR).

Ogólna zasada mówi o tym, że AND zwraca prawdę tylko wtedy gdy wszystkie warunki będą spełnione, natomiast OR zwraca prawdę gdy przynajmniej jeden z warunków będzie spełniony.

Dla przykładu:

Pracownik otrzyma premię jeżeli jego sprzeda 10 jabłek lub 5 arbuzów. Oznacza, że premia będzie przysługiwała zarówno jeżeli spełni pierwszy warunek, drugi warunek lub oba. Jeżeli nie uda mu się osiągnąć żadnego z celów to premii nie będzie.

Jeżeli pomiędzy tymi dwoma warunkami postawimy spójnik i lub oraz to tylko przy spełnieniu obu zasad należna będzie premia.

sprzedał 10 jabłek	sprzedał 5 arbuzów	LUB	ORAZ

TAK	TAK	TAK	TAK
TAK	NIE	TAK	NIE
NIE	TAK	TAK	NIE
NIE	NIE	NIE	NIE

Operatory logiczne

Operatory te są bardzo potrzebne do budowania filtrów. Na pewno większość z nich kojarzysz z lekcji matematyki w szkole z kilkoma drobnymi różnicami.

>	większe niż
<	mniejsze niż
>=	większe lub równe
<=	mniejsze lub równe
=	równe
<> lub !=	nierówne

Zadania

1. Z tabeli filmy pobierz wszystkie filmy, których rating(klasyfikacja) to R(restricted audiences)

2. Pobierz tytuły i opisy filmów, których cena wypożyczenia jest mniejsza od 1

3. Znajdź wszystkie tytuły filmów, które można wypożyczyć na 7 dni

4. Pobierz wszystkie filmy, których długość jest krótsza niż godzina

5. Pobierz tytuły, długość oraz klasyfikację filmów, które są dłuższe niż 2h oraz mają klasyfikację G (general audiences):

6. Pobierz wszystkie filmy o cenie replacement_cost większej lub równej 24.99

Operator IN

W SQL jest jeszcze jeden operator który w znaczny sposób ułatwia pobieranie wierszy, które w określonej kolumnie zawierają jedną z opcji.

Dla przykładu wyobraź sobie, że potrzebujesz pobrać imiona i nazwiska wszystkie studentów, którzy pochodzą z Trójmiasta.

```
SELECT

  imie,

  nazwisko

FROM

  studenci

WHERE

  miasto IN (‘Gdynia’, ‘Gdańsk’, ‘Sopot’);
```

Oczywiście ten sam efekt można uzyskać za pomocą zapytania WHERE wraz z trzema testami połączonymi OR, jednakże taki zapis wygląda dużo mniej elegancko - tj. zamiast:

```
miasto IN (‘Gdynia’, ‘Gdańsk’, ‘Sopot’);
```

można zapisać:

```
miasto = ‘Gdynia’ OR miasto=’Gdańsk’ OR miasto=’Sopot’
```

Widzisz różnicę? Przy większej ilości testów operator IN jest wprost nie do zastąpienia!

Zadania

1. Pobierz imiona i nazwiska aktorów, których imię to Bob lub Helen
2. Przygotuj listę filmów wraz z opisami dla klasyfikacji: R, NC-18 oraz PG-13

Operator tekstowy - LIKE

I na dokładkę jeszcze jeden operator bez którego ciężko jest napisać dowolne zapytanie do poszukiwania fragmentu tekstu. Za jego pomocą możemy zastąpić dowolną ilość znaków, wstawiając w określone miejsce symbol %.

Dla przykładu, jeżeli chcielibyśmy znaleźć wszystkie osoby których nazwisko kończy się na “ska”, moglibyśmy napisać:

```
SELECT nazwisko FROM osoba WHERE nazwisko LIKE “%ska”
```

lub wszystkie osoby, których pierwszą częścią nazwiska jest “Sieradzińsk”, a więc wszystkie spokrewnione ze mną Panie

```
SELECT * FROM osoba WHERE nazwisko LIKE “Sieradzińsk%”
```

Sieradzińskiemu

Sieradzińskiej

Natomiast jeżeli chcesz zastąpić tylko jeden znak to możesz użyć symbolu _. W ten sposób np. Na_kowskiej będzie dopasowaniem dla Pani Zofii Nałkowskiej oraz błędnego nazwiska Naukowskiej.

ZADANIA

1. Przygotuj listę tytułów filmów oraz dodatków(special_features) które w kolumnie special_features posiadają “Behind the Scenes”
2. Przygotuj listę filmów, których tytuły zaczynają się od wyrażenia “RUN%”

Zmiana kolejności

Do sortowania danych służy instrukcja ORDER BY, którą umieszcza się za blokiem WHERE. Składania tego fragmentu jest następująca:

```
ORDER BY {kolumna} {kolejnosc}
```

gdzie

- kolumna - nazwa kolumny według której chcemy sortować dane
- kolejność - informacja o tym, czy chcemy te dane sortować malejąco czy rosnąco.

Jeżeli chcesz jednocześnie sortować po kilku kolumnach, np. po nazwisku, a w przypadku powtarzających się danych używać imienia to oddziel takie pary przecinkiem.

```
ORDER BY nazwisko ASC, imie ASC
```

Ograniczenia ilości wierszy

Za ograniczenie ilości wierszy odpowiada instrukcja LIMIT, która może być używana w dwóch wersjach.

LIMIT 5 - w ten sposób wyświetlone zostanie tylko 5 pierwszych rekordów

lub

LIMIT 10, 5 - w ten sposób zostanie pominięte 10 pierwszych wierszy i wyświetlone 5, tak jak w poprzednim przykładzie.

Podsumowanie

Wszystkie te informacje powinny wystarczyć do przygotowania zapytania, które pozwoli pobrać wszystkie rekordy(wiersze) z jednej tabeli, które spełniają konkretne warunki.

```
SELECT {co} FROM {tabela} WHERE {warunki} ORDER BY {} LIMIT {}
```

```
SELECT {co} FROM {tabela} ORDER BY.. LIMIT...
```

ZADANIA

1. Posortuj listę aktorów według nazwiska malejąco.
2. Który film jest najdłuższy, a który najkrótszy?
3. Posortuj listę filmów według ceny zwrotu, a następnie według długości rosnąco

4. Posortuj listę filmów według długości, a następnie ceny zwrotu rosnąco

Funkcje

W świecie SQL istnieje bardzo wiele funkcji, które mogą ułatwić pracę. Niektóre z nich służą do pobierania przydatnych informacji, inne do pracy z tekstem czy też operacji na dacie i czasie. Z listy wszystkich dostępnych funkcji wybrałem te, które przydają mi się najczęściej i pokrótce je opisałem. Pełną listę funkcji znajdziesz tutaj: https://dev.mysql.com/doc/refman/8.0/en/functions.html

TEKST

CONCAT

Pierwsza z funkcji, z której dosyć często korzystam pozwala na łączenie różnych wyrażeń tekstowych ze sobą. Np.

```
SELECT CONCAT(imie, nazwisko) FROM student
```

Połączy wszystkie imiona z nazwiskami i zwróci je połączone w jednej kolumnie.

ZADANIA

1. Zwróć listę połączonych imion i nazwisk klientów (tabela customer). Czy potrafisz sprawić aby imię i nazwisko było rozdzielone spacją?

LEFT

Funkcja ta pozwala na pobranie dowolnej liczby znaków od lewej strony. W połączeniu z funkcją CONCAT można np. przygotować listę inicjałów.

ZADANIA

1. Przygotuj listę zawierającą inicjały wszystkich aktorów, np. Woody Hoffman - W.H.

LENGTH

Funkcja ta daje odpowiedź na pytanie jak długie jest dane wyrażenie tekstowe.

Dla przykładu SELECT LENGTH(‘Schibsted’) zwróci 9.

ZADANIA

1. Znajdź najdłuższy i najkrótszy tytuł filmu
2. Znajdź aktora lub aktorkę którego połączone imię i nazwisko jest najdłuższe, a następnie najkrótsze

LOWER

Funkcja ta zamienia wszystkie litery w danym wyrażeniu na małe

ZADANIA

1. Przygotuj listę tytułów filmów małymi literami

REPLACE

Za pomocą tej funkcji można zmienić dowolny ciąg znaków w wyrażeniu na dowolną inną wartość.

```
SELECT REPLACE(email, ‘@’, ‘[at]’) FROM pracownicy
```

Zwróci nam listę maili w postaci kacper[at]sieradzinski.pl

ZADANIA

1. Znajdź wszystkie tytułu i opisy filmów, które w opisie zawierają słowo “Drama”, następnie zamień to słowo na Comedy.

LICZBY

Funkcje z tej grupy służą do wykonywania obliczeń/zaokrągleń oraz są wykorzystywane przy operacji agregowania danych.

COUNT

Odpowiada na pytanie ile jest wierszy spełniających podany warunek. Dla przykładu, zapytanie:

```
SELECT COUNT(*) FROM pracownicy WHERE miasto=”Gdynia”
```

zwróci odpowiedź ilu jest pracowników pochodzących z Gdyni. Funkcja ta potrafi odpowiedzieć jedynie na jedno zapytanie, bez wyszczególniania części składowych.

SUM

Funkcja bardzo zbliżona do funkcji COUNT, z tą różnicą że zamiast liczyć wiersze, sumowana jest określona kolumna.

```
SELECT SUM(wynagrodzenie) from pracownicy WHERE dzial=”finanse”
```

MAX i MIN

Działają analogicznie do poprzednich funkcji, z tą różnicą, że zwracają największą lub najmniejszą wartość z zakresu.

```
SELECT MIN(wiek) FROM osoby WHERE wyksztalcenie IN (‘srednie’ , ‘wyzsze’);
```

ROUND

Niech rzuci kamieniem ten kto nigdy nie potrzebował zaokrąglać danych. Funkcja ROUND wykonuje dla nas zaokrąglenie matematyczne. Potrafi zaokrąglić określoną liczbę do wskazanej przez nas liczby miejsc po przecinku.

```
SELECT ROUND(wynagrodzenie_roczne/12, 2) FROM pracownicy
```

DATA

Data w SQL pełni bardzo ważną rolę. Często potrzebujemy nie tylko pobierać dane które mieszczą się w określonym przedziale dat, ale często potrzebujemy też mieć możliwość wykonywania na datach różnych operacji, np. szukamy odpowiedzi jaka data będzie tydzień później lub kiedy będzie następny dzień roboczy po jakiejś dacie. We wszystkich tego typu problemach pomagają nam funkcje.

NOW

Funkcja ta zwraca bieżącą datę i aktualny czas. Wypróbuj:

```
SELECT NOW()
```

EXTRACT

Funkcja ta pozwala wyciągać dowolny fragment daty lub czasu. Dla przykładu, jeżeli potrzebna jest Ci informacja o bieżącej godzinie lub dniu możesz to zrobić tak:

```
SELECT EXTRACT(DAY FROM NOW())
```

analogicznie można pobierać dowolny fragment daty lub czasu korzystając z określeń:

- MICROSECOND
- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR
- SECOND_MICROSECOND
- MINUTE_MICROSECOND
- MINUTE_SECOND
- HOUR_MICROSECOND
- HOUR_SECOND
- HOUR_MINUTE
- DAY_MICROSECOND
- DAY_SECOND
- DAY_MINUTE
- DAY_HOUR
- YEAR_MONTH

LOGICZNE

IF

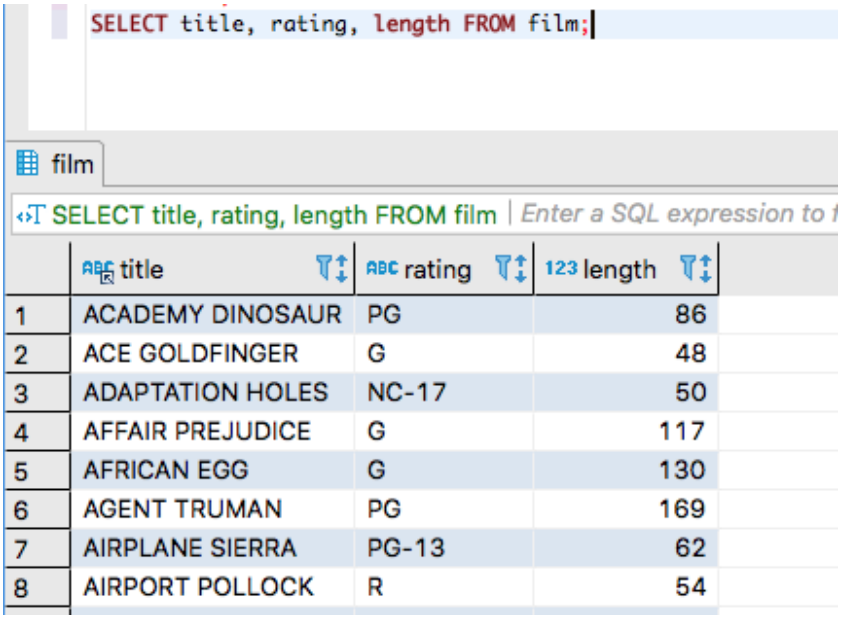
Funkcja pozwala na wzbogacanie zwracanych danych w zależności od wartości innych pól. Dla przykładu, posiadamy w bazie informacje o wieku danej osoby i

chcielibyśmy aby zwracane dane miały “TAK” lub “NIE” w kolumnie odpowiedzialnej za pełnoletniość.

```
SELECT imie, wiek, IF(wiek>=18, “TAK”, “NIE”) as pełnoletni FROM studenci
```

Grupowanie danych

W przypadku prawie wszystkich zagadnień, które do tej pory omawialiśmy stosowałem analogię do Excela. Tak będzie i tym razem, gdyż wielu osobom program ten jest dobrze znany. Analogia brzmi: Group by to narzędzie, które rozwiązuje podobne zadania co sumy częściowe w Excelu. Pomogło?



Na powyższym obrazku widzisz proste zapytanie w którym pobraliśmy sobie tytuły filmów ich klasyfikację oraz długość. Jak odpowiedzieć na pytanie ile jest filmów

poszczególnych kategorii? Sama lista poszczególnych kategorii była omawiana i robi się to tak, dla przypomnienia:

```
SELECT DISTINCT rating FROM film
```

Jednak w żadnym stopniu nie pokaże nam to ilości filmów z poszczególnych rodzajów. A co gdyby dało się zgrupować wszystkie rekordy po jakiejś kolumnie i wykonać agregacje? Ano można, a robi się to tak:

```
SELECT rating, COUNT(*) FROM film GROUP BY rating;
```

W zapytaniu pojawiło nam się kilka nowych rzeczy, przejdźmy po nich kolejno zaczynając od końca.

GROUP BY rating - Wykonując większość SELECTów mamy możliwość grupowania danych po dowolnej kolumnie. Efektem tego będzie scalenie wierszy wyniku.

COUNT(*) - To jedna z funkcji agregujących, musisz wiedzieć, że w SQL jest ich całkiem sporo. Postaram się większość z nich opisać poniżej. Na chwilę obecną zapamiętaj, że COUNT(*) liczy wszystkie zwrócone wiersze w ramach danej grupy.

UWAGA! zwróć uwagę, że w miejscu w zapytaniu gdzie jest lista kolumn zniknął nam tytuł. Po zastosowaniu grupowania na jakiejś tabeli, na liście kolumn mogą występować tylko te nazwy kolumn które zostały wymienione podczas grupowania.

Aliasy

Oczywiście możemy również zamienić brzydką etykietę kolumny z COUNT(*) na coś innego aby to osiągnąć wystarczy użyć polecenia “AS”.

```
SELECT rating, COUNT(*) as number_of_votes FROM film GROUP BY rating;
```

Taki alias może być również przydatny podczas sortowania gdyż aby posortować kategorie filmów po ilości głosów wystarczy dodać:

```
ORDER BY number_of_votes
```

na końcu powyższego zapytania.

Grupowanie po kilku polach

SQL poza tworzeniem agregatów dla grup, pozwala nam również na tworzenie podgrup. Może brzmieć to dosyć skomplikowanie, ale skomplikowane nie jest.

Załóżmy, że masz taką tabelę:

data_zdarzenia	**imie_pracownika**	**zadanie**
2018-01-01	Zbyszek	Malowanie
2018-01-01	Zbyszek	Szpachlowanie
2018-01-01	Czarek	Malowanie
2018-01-01	Czarek	Malowanie
2018-01-02	Czarek	Szpachlowanie
2018-01-02	Zbyszek	Malowanie

Nasi pracownicy wykonują drobne prace wykończeniowe. Chcielibyśmy dowiedzieć się kto pracuje więcej.

```
SELECT imie_pracownika, COUNT(*) FROM zadania GROUP BY imie_pracownika
```

Dostaniemy w ten sposób listę imion pracowników oraz ilość operacji, które wykonali.

imie_pracownika	**COUNT(*)**
Zbyszek	3
Czarek	3

Jeżeli jednak zależy nam na tym aby dowiedzieć się jakie wykonywali zadania potrzebujemy dodać drugie kryterium grupowania. Aktualizujemy nasze zapytanie:

```
SELECT
    imie_pracownika,
    zadanie,
    COUNT(*)
FROM zadania
GROUP BY imie_pracownika, zadanie
```

Jeżeli wszystko pójdzie zgodnie z planem to powinniśmy zobaczyć widok jak poniżej:

imie_pracownika	**zadanie**	**COUNT(*)**
Zbyszek	Malowanie	2
Zbyszek	Szpachlowanie	1
Czarek	Malowanie	2
Czarek	Szpachlowanie	1

W ten sposób zobaczymy zarówno imiona pracowników, zadania jak i ilość wykonywanych przez nich czynności. Minusem tego rozwiązania jest to, że tracimy informację mówiącą o tym ile razy pracownik wykonywał którekolwiek zadanie.

Podzapytania

```
SELECT

  rating,

  rental_rate,

  (SELECT COUNT(*) FROM film f2 WHERE f1.rating=f2.rating),

  COUNT(*) as ilosc

FROM film f1

GROUP BY rating, rental_rate

ORDER BY ilosc DESC;
```

Lista funkcji agregujących

Na chwilę obecną w absolutnie każdym zapytaniu jako funkcji agregującej używaliśmy wyrażenia COUNT(*). Napisałem wyżej, że funkcja ta daje nam odpowiedź o tym ile jest jakichkolwiek wierszy w ramach danej grupy. Gdzie gwiazdka oznacza wszystkie kolumny.

Naturalnie rzecz biorąc nie jest to jedyna funkcja agregująca jakiej możemy używać, jest ich dużo dużo więcej. Co ważne, w większości przypadków zamiast * wpisanej w nawiasach będziemy mieli nazwę kolumny. Dla przykładu:

```
SELECT rating, SUM(length), COUNT(*) FROM film GROUP BY rating
```

Odpowie nam na pytanie jaka jest łączna długość SUM(length) oraz ile jest wszystkich filmów COUNT(*) w bazie pogrupowanych po kolumnie rating GROUP BY rating.

Oprócz COUNT i SUM dostępne są też takie funkcje jak:

- MAX

- MIN
- AVG
- GROUP_CONCAT

Plus kilka innych związanych ze statystyką. Ich listę znajdziesz tutaj: https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html

GROUP_CONCAT

To chyba jedyna funkcja, której działanie mogło nie być dla Ciebie oczywistę na podstawie nazwy. Dlatego też prosty przykład. Wróćmy na moment do naszych pracowników:

data_zdarzenia	**imie_pracownika**	**zadanie**
2018-01-01	Zbyszek	Malowanie
2018-01-01	Zbyszek	Szpachlowanie
2018-01-01	Czarek	Malowanie
2018-01-01	Czarek	Malowanie
2018-01-02	Czarek	Szpachlowanie
2018-01-02	Zbyszek	Malowanie

Pytanie brzmi:

W jakich dniach w pracy był Zbyszek, a w jakich Czarek?

Jeżeli wykonam grupowanie na podstawie imienia pracownika i do listy kolumn dodam data_zdarzenia to dostanę błąd gdyż kolumna data_zdarzenia nie jest wykorzystywana w grupowaniu. Zamiast tego chcielibyśmy aby SQL wyświetlił nam odpowiedź na to pytanie w jednym polu. Robi się to tak:

```
SELECT

    imie_pracownika,

    GROUP_CONCAT(data_zdarzenia) as powtorki,

    GROUP_CONCAT(DISTINCT data_zdarzenia) as unikaty,

FROM zadania

GROUP BY imie_pracownika
```

W ten sposób SQL będzie wiedział, że jeżeli dla danego pracownika występuje kilka dat_zdarzenia to jego zadaniem jest połączenie tych danych i rozdzielenie ich przecinkiem.

Dobrym pomysłem jest też dodanie DISTINCT bezpośrednio w funkcji GROUP_CONCAT. W ten sposób nasza lista dni nie będzie zawierała duplikatów.

imie_pracownika	powtorki	unikaty
Zbyszek	2018-01-01, 2018-01-01, 2018-01-02	2018-01-01,2018-01-02
Czarek	2018-01-01 2018-01-01 2018-01-02	2018-01-01,2018-01-02

Różnica pomiędzy COUNT(*) a COUNT(kolumna)

Bardzo często podczas szkoleń padało to pytanie. Dlaczego czasem piszemy COUNT z gwiazdką, a czasem piszemy nazwę kolumny. Różnica polega na tym, że COUNT(*) zwraca nam liczbę absolutnie wszystkich wierszy, natomiast COUNT(kolumna) zwraca liczbę tych gdzie kolumna zawiera jakąkolwiek wartość.

Zadania

1. Jaki jest średni czas trwania filmów z poszczególnych kategorii?
2. Przygotuj listę zawierającą kategorie, najdłuższy czas i najkrótszy czas filmu. MAX() MIN()

HAVING

Zanim wyjaśnimy sobie jakie zadania “załatwia” nam HAVING należy podsumować całą wiedzę zdobytą do tej pory.

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
- LIMIT

Kolejność znana prawda? Nie jest przypadkowa. Można sobie wyobrazić, że każdy kolejny fragment zapytania działa na tym co zwróciło nam zapytanie poprzednie. Czy można użyć zapytania WHERE jeżeli nie podaliśmy we FROM nazwy tabeli? ORDER BY jest na samym końcu gdyż jego zadaniem jest wykonanie sortowania wszystkiego tego co zwróciła nam wcześniejsza część.

OK.. to teraz zwróćmy uwagę na ten fragment.

- WHERE
- GROUP BY

- HAVING

Czyli, w pierwszej kolejności na dane nakładany jest warunek WHERE, następnie dane które zwraca nam WHERE są pogrupowane, a potem mamy możliwość użycia HAVING, czyli ta część zapytania pozwala nam na ograniczenie danych które zwraca GROUP BY.

Dla przykładu:

```
SELECT rating, COUNT(*) as number_of_votes FROM film GROUP BY rating;
```

To zapytanie już wykonywaliśmy. Zmiana, którą chcemy wprowadzić ma na celu ograniczenie ilości zwracanych danych tylko do tych gdzie ilość jest większa od 200.

```
SELECT rating, COUNT(*) as number_of_votes FROM film GROUP BY rating
HAVING COUNT(*) > 200;
```

W żaden sposób za pomocą WHERE tego nie wykonamy, gdyż nie wie on jeszcze, że planujemy zastosowanie jakichkolwiek GROUP.

JOINY - złączenia

Dobra, temat GROUP mamy za sobą czas dowiedzieć się jak możemy łączyć ze sobą dane pomiędzy różnymi tabelami. Jednak zanim przejdziemy do praktyki przyda nam się trochę teorii.

Jak łączone są ze sobą dane?

Możesz zauważyć, że w każdej tabeli którą zawiera Sakila, każdy rekord posiada tzw. klucz główny czyli jakieś id. Dla przykładu w tabeli film mamy film_id jako klucz główny oraz language_id jako informacje o tym jaki język jest w tym filmie używany.

Jeżeli sprawdzisz strukturę tabeli language to okaże się, że tam również znajduje się language_id i tak właśnie można dane ze sobą wiązać.

Budowa joina w zapytaniu jest następująca

```
SELECT....

FROM tabela_pierwsza

LEFT JOIN tabela_druga ON tabela_pierwsza.kolumna = tabela_druga.kolumna
```

W przykładzie użyłem LEFT JOIN, jednak rodzajami joinów zajmiemy się niżej. Jak widzisz musimy określić jakie tabele ze sobą łączymy oraz wskazać jakie kolumny odpowiadają za wykonanie tego połączenia.

To co znajduje się za słowem ON wygląda następująco: tabela<KROPKA>kolumna. Zrobiłem tak dlatego, że bardzo często w tabelach nazwy kolumn się powtarzają, a baza danych sama się nie domyśli o którą kolumnę nam chodzi.

Rodzaje Joinów

Przejdźmy dalej, joiny mają swoje rodzaje i zależnie od tego jakiego typu joina użyjemy dostaniemy zupełnie inne dane. Wyobraźmy sobie, że

```
MAGAZYN   -------- >  SKLEP
```

Po lewej stronie mamy MAGAZYN, a po prawej SKLEP jeżeli użyjemy:

LEFT JOIN

Otrzymamy listę wszystkich produktów, które są w magazynie, nawet jeżeli nie są dostępne w sklepie.

RIGHT JOIN

Otrzymamy listę wszystkich produktów, które są w sklepie, nie zostaną wyświetlone natomiast te które są tylko w magazynie

INNER JOIN

Otrzymamy listę tych produktów które zarówno są w magazynie jak i w sklepie, nie będzie wyświetlany żaden produkt bez dopasowania

Asocjacje

Bardzo często słyszę stwierdzenie, że baza danych jest relacyjna ponieważ pomiędzy tabelami zachodzą relacje. Absolutnie tak nie jest, to co zachodzi pomiędzy tabelami to asocjacje, które również mają swoje odmiany.

Jeden do jednego

Jeden rekord w tabeli A odpowiada dokładnie jednemu rekordowi w tabeli B.

Np. każdy użytkownik posiada jeden wpis dotyczący jego stanu konta, który jest przechowywany w innej tabeli.

Jeden do wielu

Jeden rekord w tabeli A posiada wiele rekordów powiązanych z tabelą B

Np. każda kategoria w sklepie internetowym posiada wiele produktów, ale jeden produkt może znajdować się tylko w jednej kategorii.

Wiele do wielu

Jeden rekord w tabeli A może być powiązany z wieloma rekordami w tabeli B.

Np. W naszej bazie jeden aktor gra w wielu filmach, ale każdy film ma także wielu aktorów.

I działa to inaczej niż rozwiązanie, które widzieliśmy do tej pory gdyż do wykonania powiązania wiele do wielu potrzebujemy tzw. tabeli pośredniczącej.

FILM ----> FILM_ACTOR ---> ACTOR

Czyli tak naprawdę nasza asocjacja wiele do wielu jest po prostu dwiema relacjami jeden do wielu.. :) Zobacz jak to działa w praktyce. Chciałbym wyświetlić imiona i nazwiska aktorów oraz filmy w których grali. Nie interesują mnie aktorzy którzy nie grali w żadnym filmie, ani filmy które nie zaangażowały żadnego aktora. Jakiego typu JOINA użyje? Zacznę od tabeli łączącej dwie tabele i zrobię na nim LEFT JOINA. Oczywiście możesz mieć inny pomysł na to zapytanie, nie wahaj się spróbować je wykonać, być może będzie bardziej czytelne!

```
SELECT

       actor.first_name,

       actor.last_name,

       film.title

FROM

        film_actor

LEFT JOIN actor ON film_actor.actor_id = actor.actor_id

LEFT JOIN film ON film_actor.film_id = film.film_id;
```

Aliasy

Bardzo często nazwy tabel są długie lub kilka razy próbujemy łączyć się z tą samą tabelą wówczas z pomocą przychodzą nam aliasy, ale inne niż te które już te omawialiśmy. Przykładowe zapytanie z wykorzystaniem aliasów wygląda następująco:

```
SELECT

       a.first_name,

       a.last_name,

       f.title

FROM

        film_actor fa

LEFT JOIN actor a ON fa.actor_id = a.actor_id

LEFT JOIN film f ON fa.film_id = f.film_id;
```

W tym konkretnym przypadku wydaje mi się, że nazwy tabel były ok, po pierwsze były krótkie po drugie bywa, że aliasy zaciemniają obraz całego zapytania gdyż są nieintuicyjne.

Np.

p.id, a.id, i.id ??

Dużo ładniej

product.id, actor.id, item.id

Wniosek: Jak ze wszystkim, dobrze że jest ale trzeba zawsze stosować z umiarem, pamiętając o tym by “wygodne skróty” nie zaprowadziły nas w pole.

Zadania

1. Przygotuj listę klientów składającą się z first_name, last_name oraz adresu. Będzie potrzebne powiązanie tabeli customer z address.

2. Przygotuj ranking 10 aktorów którzy zagrali w największej liczbie filmów.
3. Przygotuj ranking 10 filmów w których zagrało najwięcej aktorów.
4. Przygotuj listę tytułów filmów wraz z kategoriami(tabela category), ale w taki sposób aby kategorie były rozdzielone przecinkami obok tytułów filmów
5. Na podstawie tabeli rental sprawdź który film jest najchętniej wypożyczany, a w osobnym zapytaniu sprawdź który najmniej.

Widoki

Na zakończenie ostatni temat dzisiejszego spotkania. Często tworzymy skomplikowane zapytanie do którego chcielibyśmy mieć stosunkowo szybki dostęp. Możemy to osiągnąć za pomocą narzędzia jakim są widoki.

```
CREATE VIEW AS ... zapytanie
```

Tylko i aż tyle. Jeżeli tworzymy jakiś widok, za którym będzie znajdowało się zapytanie, to następnie możemy pobrać z niego dane wykonując

```
SELECT * FROM nazwa_widoku
```

Usuwanie widoków robimy za pomocą

```
DROP VIEW nazwa_widoku
```

Podsumowanie

To było około 6 godzin wspólnych zajęć. Mam nadzieję, że masz poczucie, że był to czas owocny i jeszcze kiedyś się zobaczymy :) Jeżeli interesuje Cię przeprowadzenie podobnego szkolenia w Twojej firmie skontaktuj się ze mną poprzez moje social media: https://sieradzinski.pl