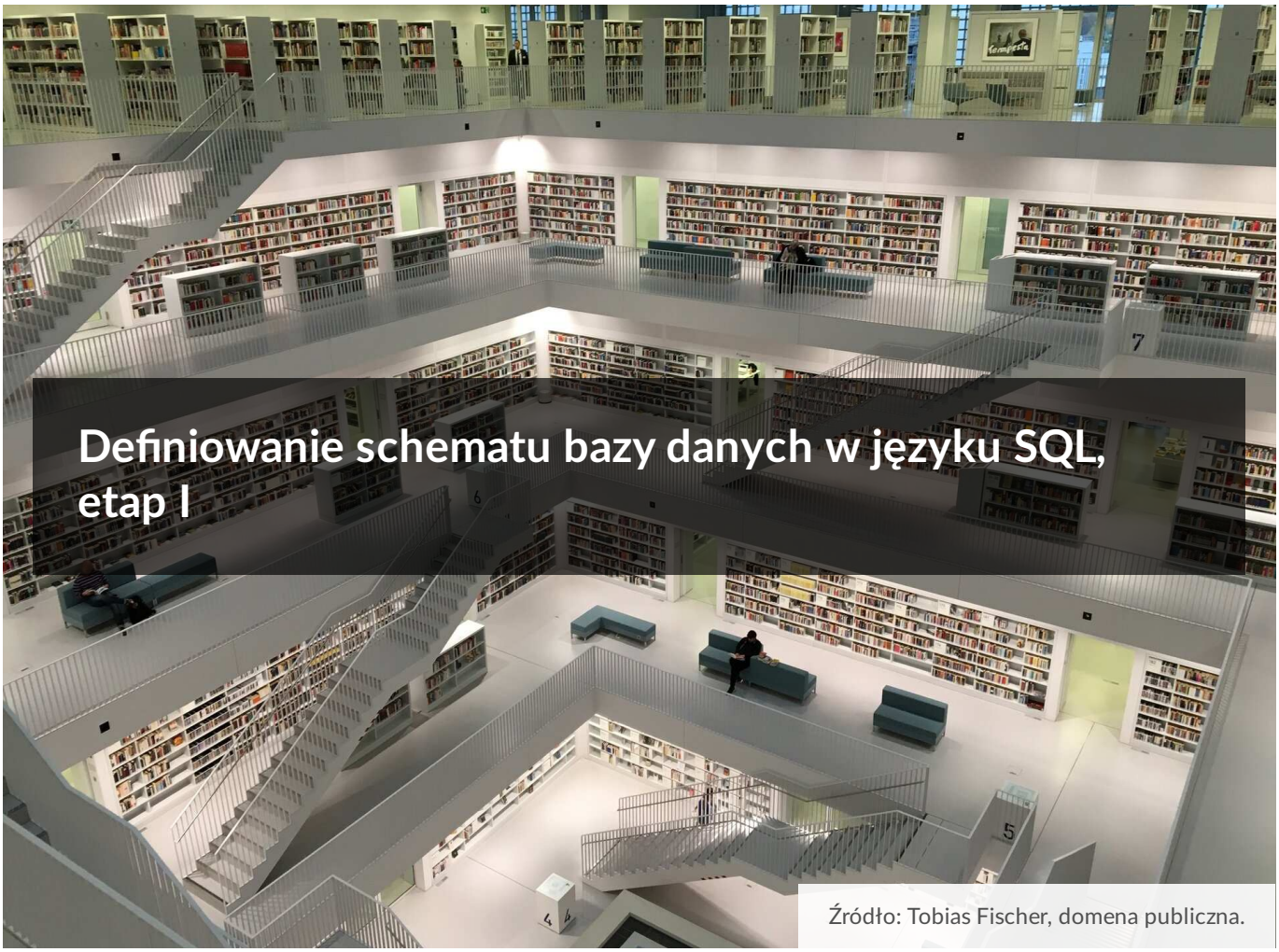
Definiowanie schematu bazy danych w języku SQL, etap I

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela



Definiowanie schematu bazy danych w języku SQL, etap I

Źródło: Tobias Fischer, domena publiczna.

Do zaprojektowania bazy danych można użyć aplikacji z interfejsem okienkowym, będących częściami pakietów biurowych (np. LibreOffice Base, Microsoft Access). Tabele projektujemy, używając kreatorów lub okien projektowych: typy danych i ich ograniczenia wybierane są z rozwijalnych list. Znajomość języka obsługi baz danych nie jest konieczna. W praktyce jednak o wiele szybciej stworzymy bazę danych „bez klikania” za pomocą skryptu. W tym e-materiale dowiesz się, jak projektować bazę danych, poznasz też polecenia języka SQL tworzące tabele.

Więcej informacji o definiowaniu schematu bazy danych w języku SQL znajdziesz w e-materiałach:

- Definiowanie schematu bazy danych w języku SQL, etap II,
- Definiowanie schematu bazy danych w języku SQL, etap III,
- Definiowanie schematu bazy danych w języku SQL, etap IV.

Twoje cele

- Przygotujesz model bazy danych.
- Wyjaśnisz, czym jest schemat bazy danych.
- Przedstawisz zasady tworzenia tabel w bazie danych.
- Wymienisz typy danych i ich ograniczenia.

- Stworzysz w bazie danych tabelę za pomocą języka SQL.

Przeczytaj

Projektowanie bazy

Naszym zadaniem jest zapisanie w bazie danych informacji potrzebnych podczas przyznawania studentom miejsc w akademikach. Oczywiście nie wszystkie informacje będą nas interesować. Potrzebujemy uproszczonego modelu, który będzie opisywał obiekt (studenta) i jego cechy. Ponieważ zajmujemy się relacyjnymi bazami danych, możemy odwołać się do modelu związków-encji, który obiekty świata rzeczywistego określa właśnie mianem encji, a ich powiązania – mianem związków.

Istotne dla nas będą następujące atrybuty encji:

Student
imię
nazwisko
rok_studiów
dochód_na_osobę_w_rodzinie

Danych nie jest dużo – dotyczą one tylko jednego obiektu, więc do przechowywania ich w bazie wystarczy jedna tabela. Możemy ją schematycznie przedstawić tak:

Studenci
id
imie
nazwisko
rok_studiow
dochod
data_a
aktywny

W ten sposób stworzyliśmy najprostszy schemat bazy danych. Warto od razu zwrócić uwagę, że dodaliśmy trzy atrybuty. Pierwszy z nich, czyli unikalny identyfikator obiektu, nazywany najczęściej `id`, jest szczególnie ważny, ponieważ pełni rolę klucza głównego (ang. *primary key*). Drugie dodatkowe pole posłuży do przechowywania czasu dodania (aktualizacji) informacji, a trzecie pozwoli oznaczyć, czy student jest aktywny.

Przykładowa zawartość tabeli wygląda następująco:

Studenci						
id	**imie**	**nazwisko**	**rok_studiow**	**dochod**	**data_a**	**aktywny**
1	Adam	Słodowy	I	2400.00	2019-09-18 12:15:48	Tak
2	Ewa	Kowalska	I	2300.00	2019-09-19 10:20:00	Nie

Definiowanie tabel

Instrukcje manipulujące bazą i tabelami należą do podgrupy poleceń języka SQL, zwanej skrótowo DDL (ang. *Data Definition Language*), czyli językiem definiowania danych. Polecenia z tej grupy to:

- `CREATE` – tworzenie baz, tabel i indeksów,
- `ALTER` – zmiana struktury, np. dodanie kolumny, zmiana typu danych,
- `DROP` – usuwanie bazy, tabeli, indeksu.

Ogólna składnia polecenia tworzącego tabelę wygląda następująco:

```
CREATE TABLE nazwa_tabeli (
    nazwa_kolumny_1 typ_danych,
    nazwa_kolumny_2 typ_danych,
    ...
);
```

W definicjach tabel obowiązują określone zasady składniowe:

- definicje wszystkich pól wydzielamy okrągłymi nawiasami,
- definicje pól wewnątrz nawiasów oddzielamy przecinkiem; dla przejrzystości możemy zapisywać je w nowych wierszach,
- instrukcję `CREATE` kończymy średnikiem.

> Ze względów praktycznych warto przemyśleć nazwy kolumn. Np. wielowyrazowe nazwy zawierające małe i wielkie litery, spacje i znaki diakrytyczne możliwe są w programach typu LibreOffice Base lub Microsoft Access. Tymczasem w skryptach i przy wpisywaniu w wiersze poleceń baz danych tego typu, nazwy mogą sprawiać kłopoty ze względu na liczbę znaków potrzebnych do wpisania, konieczność specjalnego traktowania spacji czy problemy z kodowaniem znaków. Dlatego w nazwach tabel, zwłaszcza tych, które będziemy obsługiwać za pomocą skryptów i/lub wiersza poleceń, lepiej unikać polskich znaków, spacji oraz mieszania wielkich i małych liter.

Typy danych

Składnia polecenia tworzącego tabelę jest prosta – poważniejszym wyzwaniem są typy danych, których należy użyć do przechowywania informacji. Od ich właściwego wyboru zależy wydajność bazy, wynikająca między innymi z wielkości pamięci niezbędnej do przechowywania danych, jak i z czasu pracy procesora potrzebnego do wykonania operacji w bazie.

Zacznijmy od przeglądu dostępnych typów danych.

> Przegląd oparto przede wszystkim na dokumentacji systemu bazodanowego MariaDB (otwartoźródłowa kontynuacja bazy MySQL).

Liczby całkowite

Podstawowym typem liczbowym całkowitym jest:

- `INT` – liczba 4-bajtowa, $\langle -2\ 147\ 483\ 648; 2\ 147\ 483\ 647 \rangle$.

Omawiany typ ma wiele odmian różniących się rozmiarem wymaganej pamięci oraz zakresem wartości, np.: `TINYINT`, `SMALLINT`, `MEDIUMINT` czy `BIGINT` w wersji `SIGNED` (ze znakiem, domyślne) lub `UNSIGNED` (bez znaku).

Często używa się uniwersalnego typu `INTEGER`, który reprezentuje wszystkie omawiane typy.

Synonimem typu `TINYINT` jest `BOOLEAN` (`BOOL`), który służy do przechowywania wartości logicznych: `True` (prawda) – czyli wartość różna od zera – oraz `False` (fałsz), czyli wartość zero.

Typ `INTEGER` często występuje jako wartość klucza głównego. Może mieć wtedy atrybut `AUTO_INCREMENT` (baza MariaDB – MySQL) lub `AUTOINCREMENT` (baza SQLite3), który służy do generowania unikalnych identyfikatorów dla rekordów, poprzez zwiększanie poprzedniej wartości o 1. Pierwszy rekord ma zazwyczaj identyfikator 1.

Liczby zmiennoprzecinkowe

Typy zmiennoprzecinkowe także różnią się wielkością wymaganej pamięci oraz zakresem:

- `FLOAT` – liczba 4-bajtowa, pojedynczej precyzji, do 7 cyfr znaczących po separatorze,
- `DOUBLE` – liczba 8-bajtowa, podwójnej precyzji, do 15 cyfr znaczących po separatorze,
- `DECIMAL` – liczba 8-bajtowa, przechowywana w postaci znakowej, maksymalnie 65 cyfr, w tym 30 po separatorze dziesiętnym.

Synonimem typu `DOUBLE` jest `REAL`. Synonimy `DECIMAL` to `DEC`, `NUMERIC` i `FIXED`.

W typach zmiennoprzecinkowych możemy określić liczbę wszystkich cyfr oznaczaną literą `M` oraz dokładność jako liczbę miejsc po separatorze dziesiętnym, oznaczaną jako `D`, np. `DECIMAL(6,3)`.

Typy `FLOAT` i `DOUBLE` zalecane są do przechowywania danych naukowych i wykonywania na nich obliczeń, natomiast dane finansowe powinny być zapisywane w polach typu `DECIMAL`.

Typy tekstowe

- `CHAR (M)` – krótkie ciągi znaków (maks. 255) o stałej długości, ciąg krótszy niż zadeklarowany rozmiar pola uzupełniany jest spacjami;
- `BINARY` – typ podobny do `CHAR`, ale przechowuje wartości binarne;
- `VARCHAR (M)` – tekst o zmiennej długości, maksymalna liczba znaków zależy m.in. od kodowania, np. przy domyślnym kodowaniu UTF-8 może to być 21844 znaków;
- `TEXT (M)` – tekst o maksymalnej długości 65535 znaków;
- `BLOB (M)` – pole tekstowe lub binarne o maksymalnej długości 65535 bajtów.

Opcjonalny parametr `M` pozwala wskazać liczbę znaków przechowywanych w polu. Różnice między typami `VARCHAR` i `TEXT` dotyczą indeksowania, wydajności oraz wykorzystania pamięci podczas zapisu i odczytu danych.

Typ `TEXT` ma odmiany różniące się maksymalną liczbą przechowywanych znaków: `TINYTEXT`, `MEDIUMTEXT`, `LONGTEXT`. Podobnie typ `BLOB`: `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOB`.

Typy daty i czasu

Datę i czas możemy zapisywać i przechowywać w różnych formatach:

- `DATE` – data w formacie YYYY-MM-DD może być zapisywana w różny sposób, np. 'YYYY-M-DD', 'YYMMDD', 'YY*MM*DD',
- `DATETIME` – data i czas wyświetlane w formacie YYYY-MM-DD HH-MM-SS mogą być zapisywane jako znaki lub liczby, np. 'YY-MM-DD HH:MM:SS', YYYYMMDDHHMMSS,
- `TIMESTAMP` – typ wykorzystywany do zapisania czasu dodania lub aktualizacji rekordu; wartością jest liczba sekund, jaka upłynęła od początku epoki Uniksa, to jest od początku roku 1970 UTC; podczas tworzenia lub aktualizacji pola baza danych zapisuje aktualny czas uniksowy.

Jeżeli chcemy wskazać wartość domyślną dla pola `TIMESTAMP` w klauzulach `DEFAULT` i/lub `ON UPDATE`, możemy używać różnych synonimów czasu aktualnego akceptowanego w wybranym systemie bazodanowym, np. `CURRENT_TIMESTAMP`, `CURRENT_TIMESTAMP()`, `NOW`, `NOW()`, `LOCALTIME`, `LOCALTIME()`.

> **Ważne!**
>
> 1. Dobór typu danych w wielu systemach bazodanowych wpływa na wydajność (a nawet możliwość) przeprowadzania różnych operacji.
> 2. Dostępność, nazwy i charakterystyka typów danych zależy od systemu bazodanowego.
>
> **Ciekawostka**
>
> Większość systemów bazodanowych stosuje typowanie statyczne pole – z określonym typem wymaga wartości tego samego typu. SQLite3 dla odmiany stosuje system typowania dynamicznego, w którym ogólne klasy przechowywania (ang. *storage classes*) obejmują bardziej specyficzne typy danych, np. klasa `INTEGER` obejmuje sześć typów liczb całkowitych o różnej wielkości.

Wybór typów

Ogólna zasada mówi, że do przechowywania informacji wybieramy typ najprostszy, zajmujący najmniej pamięci, który może przechować przewidywany zakres wartości. Na przykład klucze główne w postaci liczb nie będą rosły w nieskończoność, więc zazwyczaj typ `INT` wystarczy – dla małych liczb można użyć typów `TINYINT` lub `SMALLINT`. Kod pocztowy to zdecydowanie typ `CHAR`, natomiast imię czy nazwisko – `VARCHAR` lub `TEXT` w zależności od systemu bazodanowego. W przypadku dat i czasu przetwarzanie `TIMESTAMP` jest efektywniejsze niż `DATETIME`, chociaż częściej stosowany jest ten drugi typ.

> Przedstawione uwagi potraktujmy jako ogólne wskazówki, a użyty typ danych dostosujmy do przewidywanego użycia i konkretnego systemu bazodanowego.

Ograniczenia

Podczas definiowania pól nakładamy na nie ograniczenia (nazywane też właściwościami), które wyznaczają dodatkowe zasady przechowywania wartości w polu, np.:

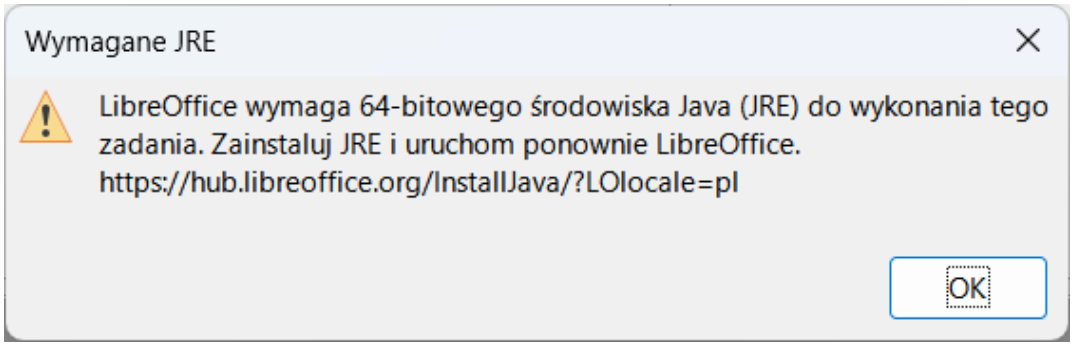
- `PRIMARY KEY` – (klucz główny) wskazuje, że wartość w polu jednoznacznie identyfikuje każdy rekord, czyli nie może się powtarzać; w obrębie tabeli może być tylko jedno takie pole;
- `NOT NULL` – wskazuje, że pole nie może zawierać wartości `NULL`;
- `DEFAULT` – definiuje domyślną wartość pola, np. '' – ciąg pusty;
- `UNIQUE` – oznacza, że wartości w polu nie mogą się powtórzyć, podobnie jak w kluczu głównym, ale w tabeli może być kilka kolumn z tym ograniczeniem;
- `CHECK` – wskazuje, że wszystkie wartości w polu spełniają podany warunek;
- `INDEX` – jest używane do szybkiego wyszukiwania danych w tabeli na podstawie pola z indeksowaniem.

Definiując pola, należy – jeśli to tylko możliwe – określać wartości domyślne. Dzięki temu podczas dodawania danych można pomijać część informacji, co przyśpiesza operację.

> **Ważne!**
>
> Dostępność poszczególnych ograniczeń zależy od systemu bazodanowego.

Baza danych i tabele w LibreOffice Base

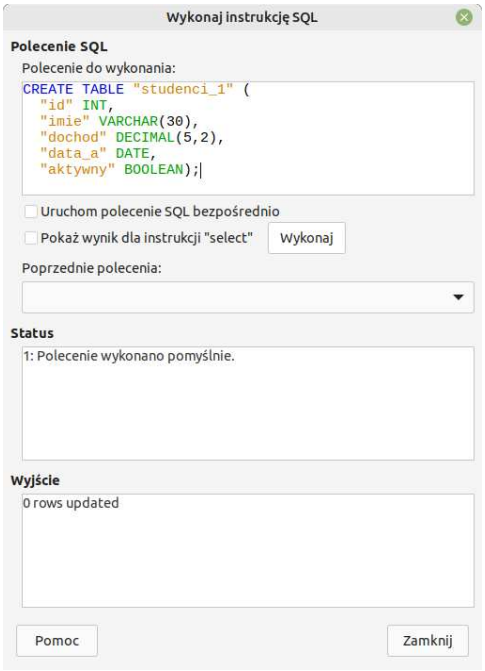
Do utworzenia bazy danych oraz tabel użyjemy programu LibreOffice Base, będącego składnikiem wolnego i darmowego pakietu biurowego LibreOffice. W domyślnej instalacji program używa wbudowanego systemu bazodanowego HSQLDB w wersji 1.8, który do działania wymaga zainstalowanego środowiska uruchomieniowego języka Java (Java Runtime Environment).



> Baza HSQLDB osadzona w LibreOffice Base ma pewne ograniczenia w porównaniu do tej samej bazy zainstalowanej na zewnętrznym serwerze. Najważniejszym jest brak obsługi wielu użytkowników i brak możliwości definiowania ich uprawnień. W kolejnych wersjach pakietu planuje się zastąpienie HSQLDB przez system bazodanowy Firebird. LibreOffice Base może obsługiwać wiele zewnętrznych systemów bazodanowych, w tym MariaDB (MySQL), SQL Server, Firebird czy SQLite3.

Do wykonania podanych poleceń potrzebna będzie nowa baza danych. Podczas jej tworzenia korzystamy z domyślnej opcji `Osadzona HSQLDB`. Baza powinna zostać zapisana w pliku o nazwie np. `studenci.odb`.

Polecenia będziemy wpisywać (lub wklejać) do okna, które wywołujemy, klikając w menu `Narzędzia` | `SQL`.



> **Ważne!**
>
> Podczas używania poleceń SQL należy zwrócić uwagę na dokładność wpisywanych lub wklejanych poleceń. Dotyczy to zwłaszcza typów używanych cudzysłowów oraz stosowania przecinków, nawiasów i średników.

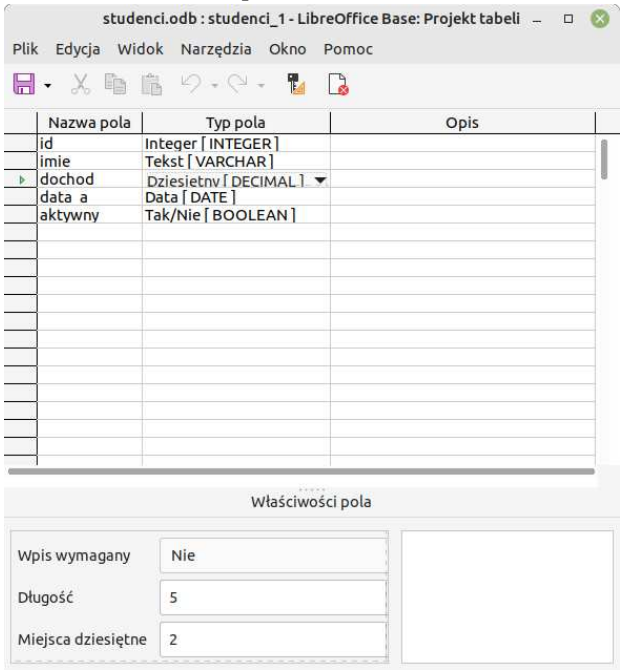
W polu *Polecenie do wykonania* wpisujemy polecenie tworzące tabelę:

> **Przykład 1**

```
CREATE TABLE "studenci_1" (
  "id" INT,
  "imie" VARCHAR(30),
  "dochod" DECIMAL(5,2),
  "data_a" DATE,
  "aktywny" BOOLEAN);
```

> Po zamknięciu okna `Polecenie SQL` wybieramy z menu: `Widok | Odśwież tabelę`, aby zobaczyć utworzoną tabelę.

Tabelę *studenci_1* warto otworzyć w trybie edycji i przejrzeć właściwości utworzonych pól, np. długość pola *imie* lub długość i dokładność pola *dochod*.



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Wykonane polecenie pokazuje, jak tworzyć pola przechowujące podstawowe typy danych. W praktycznych zastosowaniach powinniśmy umieć definiować pole klucza głównego, a także nakładać ograniczenia, w tym definiować domyślne wartości pól. Zobaczmy, jak to robić.

Otwieramy okno `Polecenie SQL` i wpisujemy kod:

> **Przykład 2**

```
CREATE TABLE "studenci" (
  "id" INT IDENTITY PRIMARY KEY,
  "imie" VARCHAR(30) NOT NULL,
  "nazwisko" VARCHAR(30) NOT NULL,
  "rok_studiow" VARCHAR(3) DEFAULT 'I',
  "dochod" DECIMAL(5,2) DEFAULT 0.00,
  "data_a" DATE DEFAULT NOW);
```

Tabela *studenci* utworzona za pomocą przedstawionego polecenia ma klucz główny (`PRIMARY KEY`) w postaci liczby całkowitej, której wartość będzie generowana automatycznie przez bazę, zaczynając od 0. Aby uzyskać taki efekt, musieliśmy użyć klauzuli `IDENTITY` wymaganej przez bazę HSQLDB.

> **Dla zainteresowanych**
>
> Aby identyfikatory w polu klucza głównego zaczynały się od wartości 1, należy użyć definicji: `"id" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 1, INCREMENT BY 1) PRIMARY KEY`.

Ponadto pola tekstowe nie mogą zawierać wartości `NULL`. W trzech ostatnich polach podaliśmy wartości domyślne, przy czym w polu *data_a* użyliśmy funkcji SQL `NOW`, która zwraca aktualny czas.

Sprawdźmy teraz, jak funkcjonują zdefiniowane kryteria. Wykonajmy następujące zapytanie:

> **Przykład 3**
>
> ```
> INSERT INTO studenci(id, imie, nazwisko) VALUES(1, 'Adam', 'Słodowy');
> ```

Przedstawione polecenie warto wykonać dwa razy pod rząd – za drugim razem powinniśmy zobaczyć komunikat: `Violation of unique constraint SYS_PK_49: duplicate value(s) for column(s) "id"`.

> **Przykład 4**
>
> ```
> INSERT INTO "studenci" ("imie", "nazwisko") VALUES('Ewa', 'Kowalska');
> ```

W tym wypadku baza sama dobiera odpowiednią wartość klucza głównego, o czym możemy się przekonać, wydając polecenie:

> **Przykład 5**
>
> ```
> SELECT * FROM "studenci";
> ```

Wyniki zapytania pokazują również, że pola, dla których nie podajemy wartości, przyjmują poprawne wartości domyślne, w tym również pole przechowujące czas.

Baza danych i tabele w Microsoft Access

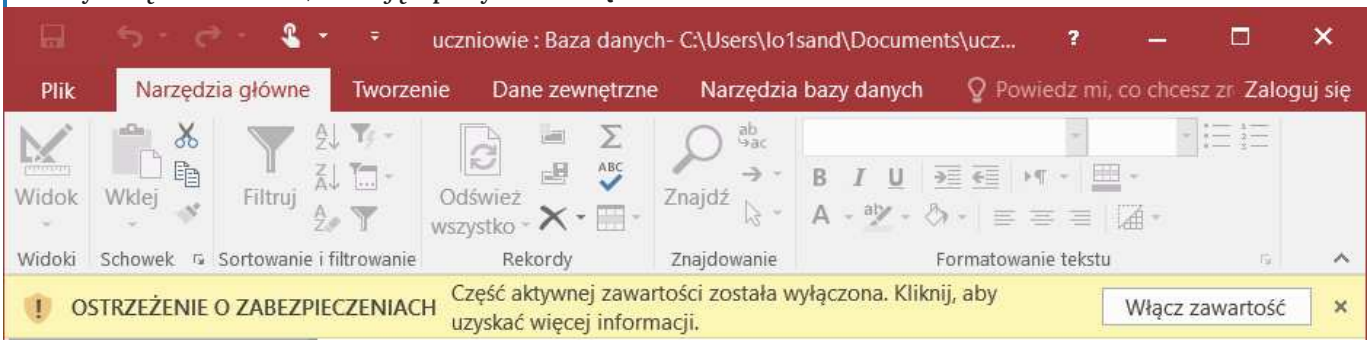
Microsoft Access to zamkniętoźródłowy system obsługi relacyjnych baz danych wchodzący w skład pakietu Microsoft Office. Od wersji 2007 wykorzystuje silnik bazodanowy o nazwie Access Database Engine oraz Microsoft Access SQL, czyli implementację standardu ANSI SQL stworzoną przez Microsoft.

> Między ANSI SQL i Microsoft Access SQL występują różnice w zakresie wspieranych typów danych, nazw typów danych i funkcji oraz składni.

Aby wykonywać polecenia SQL w Microsoft Access, tworzymy pustą bazę danych i zapisujemy w pliku o nazwie `studenci.accdb`. Po otwarciu bazy wybieramy wstążkę `Tworzenie` i klikamy ikonę `Projekt kwerendy`. Zamykamy okno `Pokazywanie tabeli` i klikamy ikonę `SQL`.

Ważne!

Jeżeli zostanie wyświetlone `Ostrzeżenie o zabezpieczeniach`, należy zezwolić na aktywną zawartość, klikając przycisk `Włącz zawartość`.

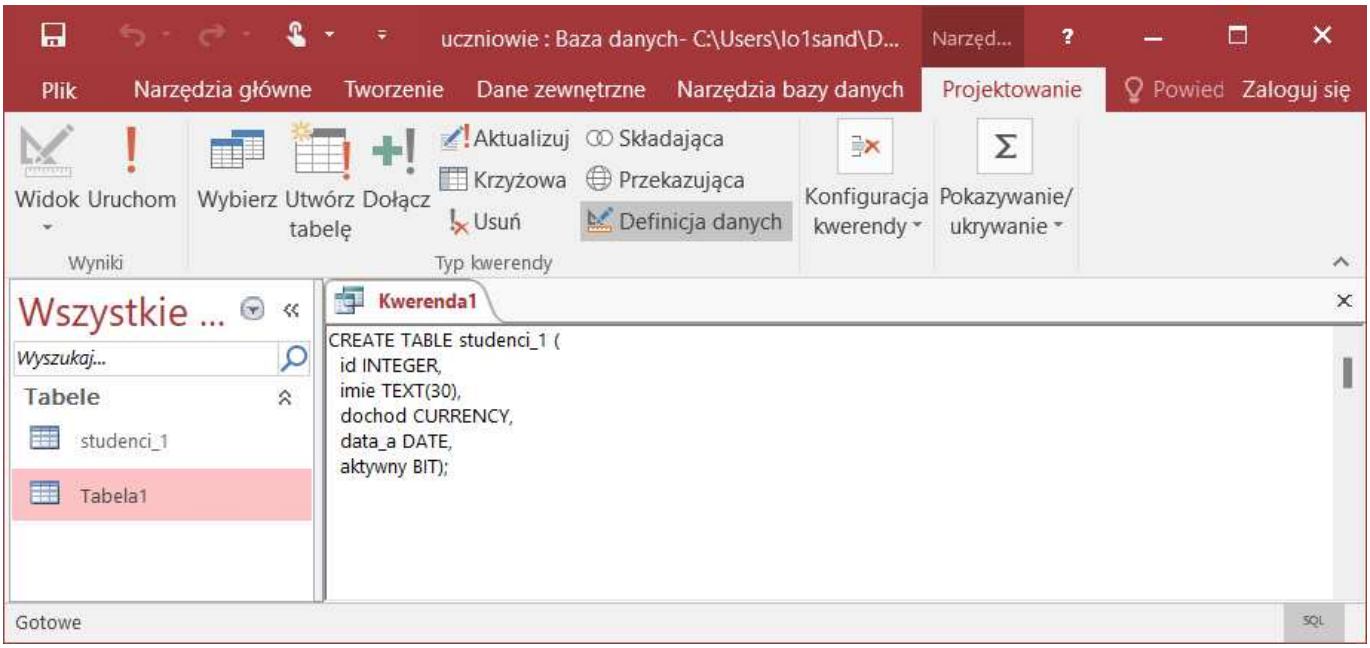


Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

W oknie kwerendy wpisujemy następujące polecenia:

Przykład 6

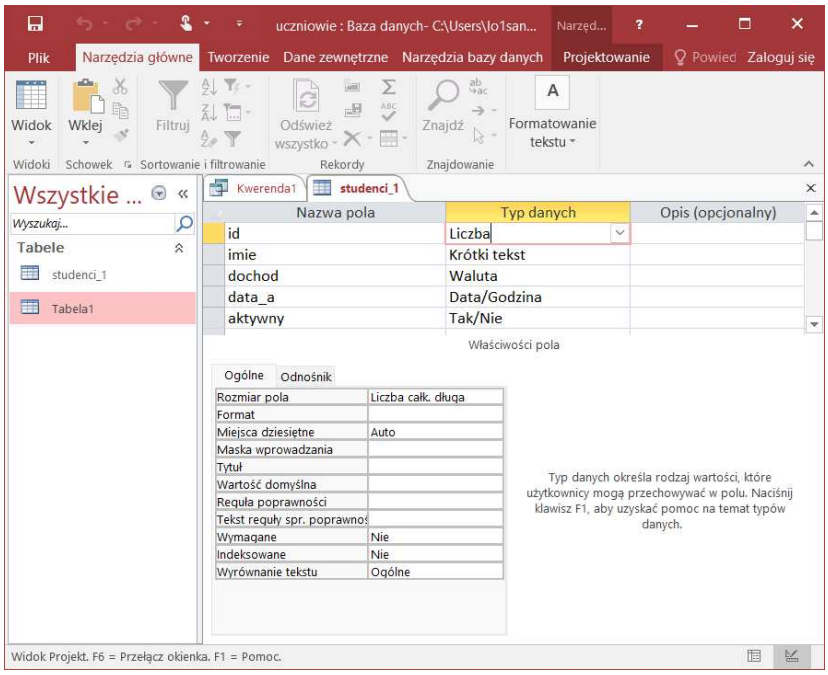
```
CREATE TABLE studenci_1 (
  id INTEGER,
  imie TEXT(30),
  dochod CURRENCY,
  data_a DATETIME,
  aktywny BIT);
```



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Wykonamy zapytanie, klikając ikonę `Uruchom` (z charakterystycznym czerwonym wykrzyknikiem). Po wykonaniu kwerendy na liście tabel powinna pokazać się nazwa `studenci_1`.

Utworzoną tabelę wyświetlimy w widoku projektu:



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Microsoft Access wykorzystuje następujące typy danych:

- `TEXT(n)` – typ przeznaczony dla krótkich tekstów do 255 znaków; w nawiasie podajemy długość pola, w widoku projektu to `Krótki tekst`;
- `CURRENCY` – typ do przechowywania wartości waluty; nie występuje w ANSI SQL, w widoku projektu to `Waluta`;
- `DATETIME` – typ przeznaczony na przechowywanie daty i czasu, w widoku projektu to `Data/Godzina`;
- `BIT` – typ do przechowywania wartości `Prawda/Fałsz`, odpowiada typowi BOOLEAN, w widoku projektu to pole `Tak/Nie`.

Wykonajmy teraz zapytanie, którego celem jest utworzenie tabeli *studenci* zawierającej klucz główny, a także ograniczenia `NOT NULL` nałożone na pola:

Przykład 7

```
CREATE TABLE studenci (
  id AUTOINCREMENT PRIMARY KEY,
  imie TEXT(30) NOT NULL,
  nazwisko TEXT(30) NOT NULL,
  rok_studiow TEXT(3),
  dochod CURRENCY,
  data_a DATE,
  aktywny BIT);
```

Po wykonaniu kwerendy w widoku projektu tabeli *studenci* możemy sprawdzić, że polu *id* został przypisany typ `INTEGER`, chociaż nie podaliśmy go wprost w zapytaniu. Z kolei dla pól *imie* i *nazwisko*, do których dodaliśmy ograniczenia, opcja `Wymagane` została ustawiona na `Tak`.

> Wykonywanie zapytań DDL w oknie projektowania kwerendy ma spore ograniczenia. Np. obecność klauzuli `DEFAULT` w zapytaniu powoduje wyświetlenie komunikatu o błędzie składniowym, podobnie próba określania precyzji przy użyciu notacji `DECIMAL(6,3)`.

Spróbujmy sprawdzić działanie nałożonych ograniczeń. Wykonajmy następujące zapytanie dwa razy:

> **Przykład 8**

```
1 INSERT INTO studenci (id, imie, nazwisko, data_a) VALUES (1, 'Ewa',
```

Po pierwszym wykonaniu nie zobaczymy żadnego komunikatu, a w tabeli zostanie utworzony nowy rekord. Za drugim razem powinniśmy zobaczyć komunikat: „Program Microsoft Access nie może dołączyć wszystkich rekordów w kwerendzie dołączającej" oraz mało zrozumiałe wyjaśnienie powodów błędu.

Taki sam komunikat zobaczymy, gdy spróbujemy dodać rekord bez podania wartości wymaganych pól, np.:

> **Przykład 9**

```
1 INSERT INTO studenci (imie, data_a) VALUES ('Adam', NOW());
```

Oznacza to, że ograniczenia `NOT NULL` działają we właściwy sposób.

Słownik

> **encja**
>
> (ang. *entity*) schematyczny opis rzeczywistych lub wymyślonych obiektów posiadających te same cechy, zwane atrybutami lub własnościami
> **klucz główny**
>
> pole (lub kombinacja pól) zawierające niepowtarzalne wartości, jednoznacznie identyfikujące rekordy tabeli; najczęściej zawiera kolejne wartości liczbowe automatycznie tworzone przez aparat bazy danych
> **model związków-encji**
>
> (ang. *entity-relationship model*) logiczny model bazy danych opisujący obiekty świata rzeczywistego jako encje (ang. *entities*), a powiązania między nimi jako związki (ang. *relationships*)
> **ograniczenia**
>
> (ang. *constraints*) dodatkowe warunki, które musi spełniać wartość przechowywana w kolumnie, np. `NOT NULL`
> **typ danych**
>
> określa strukturę, rodzaj i zakres danych, które można przechowywać w danej kolumnie

Prezentacja multimedialna

Polecenie 1

Zapoznaj się z prezentacją. Wykorzystaj przedstawione wskazówki do stworzenia za pomocą języka SQL bazy danych, która będzie przechowywała wskazane informacje.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Przygotujmy bazę danych do przechowywania informacji o potencjalnych pracownikach.

Bazę i tabelę utworzymy w programie LibreOffice Base lub Microsoft Access, używając języka SQL.

Uwaga: zapytania budujemy krok po kroku przy użyciu składni SQL ANSI, nie należy ich testować w trakcie pracy, ponieważ dopiero na końcu dostosujemy je do dialektu obsługiwanego przez LibreOffice Base oraz Microsoft Access i wykonamy.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Dane będziemy przechowywać w jednej tabeli *pracownicy*. W schemacie uwzględniamy następujące pola:

Pracownicy
imie
nazwisko

Pracownicy
pesel
plec
zatrudniony
wynagrodzenie
data_zatr

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Płeć można odczytać, analizując numer PESEL – teoretycznie moglibyśmy nie uwzględniać dodatkowego pola w tabeli. Warto jednak to zrobić, aby usprawnić wyszukiwanie pracowników według płci.

Typ danych przechowywanych w polach *imie*, *nazwisko*, *pesel* i *plec* to tekst. W polu *zatrudniony* zapisywać będziemy wartość „Tak”/ „Nie”. Pole *wynagrodzenie* ma przechowywać liczby rzeczywiste, natomiast pole *data_zatr* datę i czas.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Zapisujemy początkowe, najprostsze zapytanie:

```
CREATE TABLE pracownicy (
    imie VARCHAR,
    nazwisko VARCHAR,
    pesel VARCHAR,
    plec VARCHAR,
    zatrudniony BOOLEAN,
```

```
    wynagrodzenie DECIMAL,
    data_zatr DATETIME
);

```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

W tabeli znajduje się dużo pól tekstowych typu VARCHAR. Ponieważ wiemy, ile znaków zajmą dane w niektórych polach, możemy je zoptymalizować. Określamy również dokładność pola *wynagrodzenie*. Klauzula przyjmuje postać:

```
CREATE TABLE pracownicy (
    imie VARCHAR(20),
    nazwisko VARCHAR(30),
    pesel CHAR(11),
    plec CHAR(1),
    zatrudniony BOOLEAN,
    wynagrodzenie
DECIMAL(8,2),
    data_zatr DATETIME
);

```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Nie zdefiniowaliśmy dotąd żadnych ograniczeń. W takiej sytuacji zastosowane zostaną domyślne ustawienia, co może niekorzystnie wpłynąć na wielkość obszaru zajętego przez przechowywane dane na dysku, a także na

wydajność operacji wykonywanych na tabeli. Spróbujmy sprecyzować właściwości pól.

```
CREATE TABLE pracownicy (
    imie VARCHAR(20) NOT NULL,
    nazwisko VARCHAR(30) NOT NULL,
    pesel CHAR(11) NOT NULL,
    plec CHAR(1),
    zatrudniony BOOLEAN,
    wynagrodzenie DECIMAL(8,2),
    data_zatr DATETIME
);

```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

7

W tabeli brak klucza głównego. Mamy co prawda kolumnę *pesel*, którą moglibyśmy intuicyjnie potraktować jako klucz główny, jednak po pierwsze w tych numerach też zdarzają się błędy, po drugie przeszukiwanie tak dużych wartości trwałoby zdecydowanie dłużej niż np. liczb całkowitych. Dodajmy więc pole *id* – klucz główny, w którym wartości będą generowane przez bazę automatycznie:

```
CREATE TABLE pracownicy (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
... );

```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Przejdźmy do sposobu oznaczania płci. Przyjęta definicja zakłada wykorzystanie pojedynczego znaku. W niektórych systemach bazodanowych możemy używać typu wyliczeniowego, dzięki któremu podajemy wartości mogące wystąpić w danym polu. Np. w bazie SQLite3 definicja byłaby następująca:

```
    plec CHAR(1) CHECK (plec IN ('K', 'M', 'N'))
```

Ograniczenie CHECK oraz operator IN pozwalają sprawdzić, czy wartość pola pasuje do któregokolwiek elementu z listy znaków podanych w nawiasach.

Z kolei w systemie MariaDB (MySQL) napisalibyśmy:

```
    plec ENUM('K', 'M', 'N')
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

W LibreOffice Base, który wykorzystuje system bazodanowy HSQLDB w wersji 1.8, nie możemy użyć typu wyliczeniowego. Podobnie w Microsoft Access klauzula CHECK nie jest poprawnie interpretowana w widoku SQL.

Wykorzystamy więc inną możliwość. Tworzymy tabelę słownikową, w której pole przechowujące literowe oznaczenia płci jest

kluczem głównym. Dodatkowo w tabeli możemy przechować opis płci.

```
CREATE TABLE slownik_plec
(
    id CHAR(1) PRIMARY KEY,
    opis VARCHAR(20)
);

```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

W kolejnym kroku tworzymy relację między tabelą *pracownicy* i tabelą słownikową. Dzięki temu w polu *plec* tabeli *pracownicy* można wpisać tylko oznaczenia literowe występujące w polu *id* tabeli słownikowej. Do polecenia tworzącego tabelę *pracownicy* dodajemy klauzulę:

```
CREATE TABLE pracownicy (
    ...
    FOREIGN KEY (plec) REFERENCES
slownik_plec(id)
    ON UPDATE CASCADE
);

```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

11

Wszędzie, gdzie jest to możliwe i uzasadnione, powinniśmy zdefiniować wartości domyślne – po to, żeby można było przyśpieszyć wprowadzanie znaczących danych.

```
CREATE TABLE pracownicy (
    ...
    zatrudniony BOOLEAN
DEFAULT 0,
    wynagrodzenie
DECIMAL(8,2) DEFAULT 0,
    data_zatr DATETIME
DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE slownik_plec
(
    ...
    opis VARCHAR(20)
DEFAULT 'nieznana'
);


```

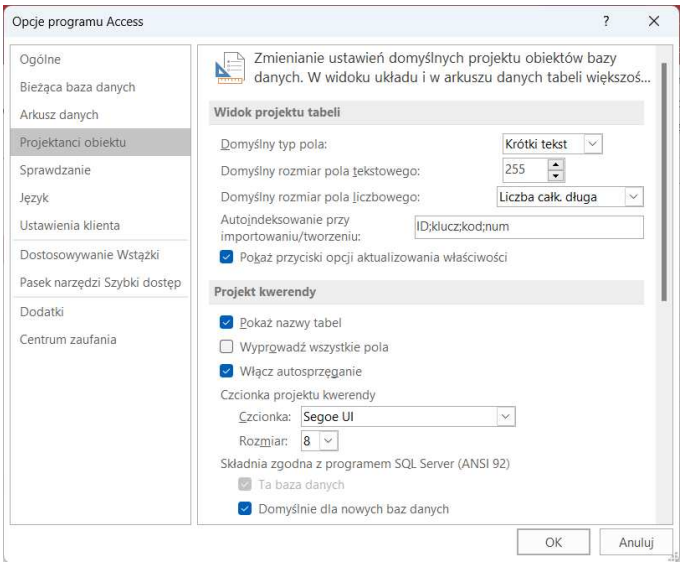
12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Przystępujemy do wykonania omawianych zapytań.

Tworzymy i zapisujemy na dysku w pliku `pracownicy.odb` pustą bazę danych LibreOffice Base. Wybieramy polecenie `Narzędzia | SQL` a w polu `Polecenie do wykonania` wprowadzamy przygotowane kwerendy.



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

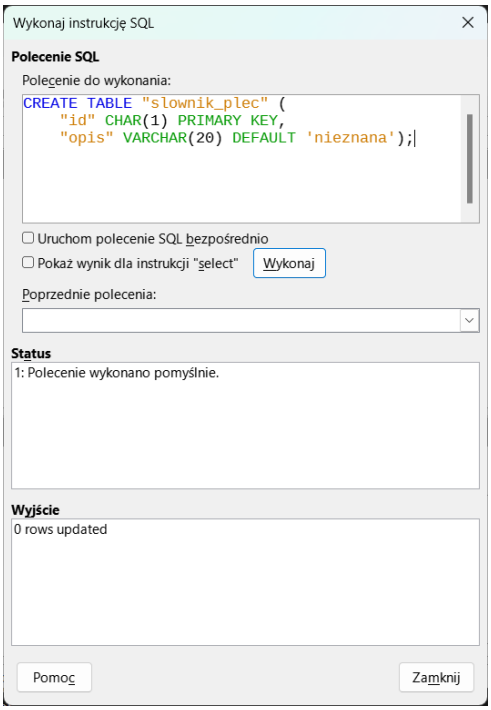
Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Program Microsoft Access domyślnie używa dialektu Microsoft Access SQL, w którym niektóre typy danych i klauzule nie są interpretowane, np. `DECIMAL` czy `DEFAULT`. Aby można było ich użyć, po uruchomieniu programu wybieramy `Plik | Opcje`, następnie wybieramy opcję `Projektanci obiektu` i pod napisem `Składnia zgodna z programem SQL Server (ANSI 92)` zaznaczamy opcję `Domyślnie dla nowych baz danych`.

Następnie tworzymy i zapisujemy nową bazę danych `pracownicy.accdb`. Wybieramy wstążkę `Tworzenie`, dalej `Projekt kwerendy`, zamykamy okno `Pokazywanie tabeli` i klikamy ikonę `SQL`.

14



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

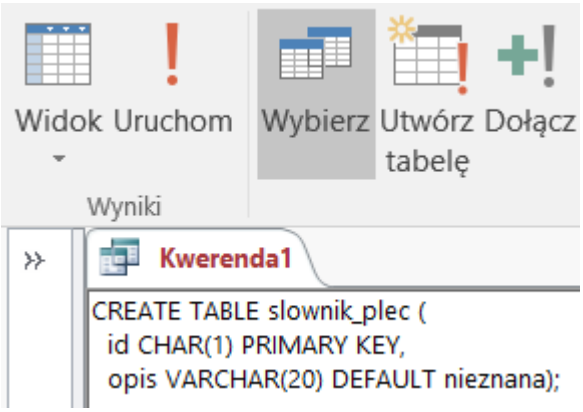
Na początku należy utworzyć tabelę *slownik_plec*, ponieważ musi ona istnieć w bazie, zanim odwołamy się do niej później, kiedy będziemy tworzyli relację.

W LibreOffice Base:

```
1 CREATE TABLE
"slownik_plec" (
2     "id" CHAR(1) PRIMARY
KEY,
3     "opis" VARCHAR(20)
DEFAULT 'nieznana');
4
```

Aby zobaczyć utworzoną tabelę, wybieramy `Widok | Odśwież tabele`.

15



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

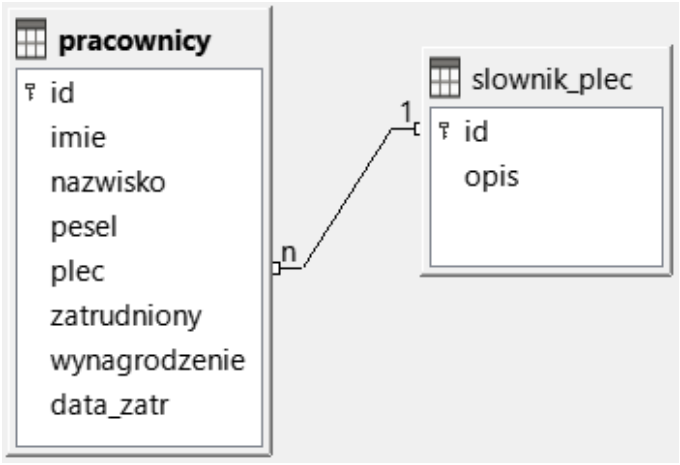
https://zpe.gov.pl/b/P1B6dtEh6

W Microsoft Access:

```
CREATE TABLE slownik_plec
(
  id CHAR(1) PRIMARY KEY,
  opis VARCHAR(20) DEFAULT
nieznana);

```

16



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

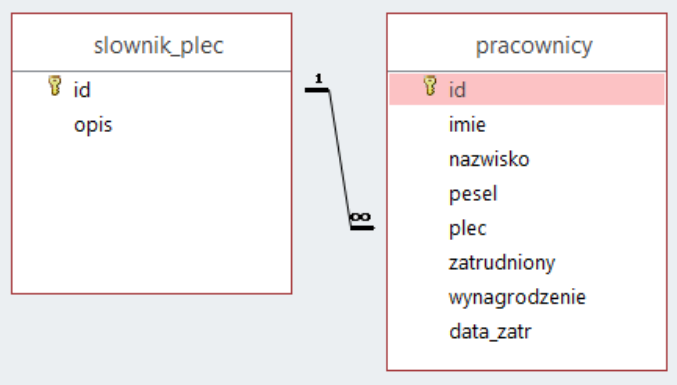
https://zpe.gov.pl/b/P1B6dtEh6

Teraz możemy utworzyć tabelę *pracownicy*.

W LibreOffice Base:

```
CREATE TABLE "pracownicy"
(
    "id" INTEGER IDENTITY
PRIMARY KEY,
    "imie" VARCHAR(20) NOT
NULL,
    "nazwisko" VARCHAR(30)
NOT NULL,
    "pesel" CHAR(11) NOT
NULL,
    "plec" CHAR(1),
    "zatrudniony" BOOLEAN
DEFAULT 0,
    "wynagrodzenie"
DECIMAL(8,2) DEFAULT 0,
    "data_zatr" DATETIME
DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY ("plec")
REFERENCES "slownik_plec"
("id")
    ON UPDATE CASCADE
);
```

Wybieramy polecenie `Narzędzia | Relacje` i powinniśmy zobaczyć tabele połączone relacją.



Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

W Microsoft Access wykonujemy zapytanie:

```
CREATE TABLE pracownicy (
    id AUTOINCREMENT
PRIMARY KEY,
    imie VARCHAR(20) NOT
NULL,
    nazwisko VARCHAR(30)
NOT NULL,
    pesel CHAR(11) NOT
NULL,
    plec CHAR(1),
    zatrudniony BIT
DEFAULT 0,
    wynagrodzenie
DECIMAL(8,2) DEFAULT 0,
    data_zatr DATETIME
DEFAULT NOW(),
    FOREIGN KEY (plec)
REFERENCES
slownik_plec(id)
    ON UPDATE CASCADE
);
```

Ekwiwalentem typu `BOOLEAN` jest typ `BIT`, używamy również funkcji `NOW()` do zdefiniowania domyślnej wartości dla pola z datą i czasem.

Podobnie jak w LibreOffice Base można sprawdzić utworzoną relację: wybieramy `Narzędzia baz danych | Relacje`.

18

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Sprawdzimy, czy tabele zostały poprawnie zdefiniowane. Spróbujmy dodać do nich

przykładowe dane.

W LibreOffice Base:

```
INSERT INTO "slownik_plec"
VALUES ('K', 'kobieta');
INSERT INTO "slownik_plec"
VALUES ('M', 'mężczyzna');

```

W Microsoft Access zapytania wykonujemy pojedynczo:

```
INSERT INTO slownik_plec
VALUES ('K', 'kobieta');
INSERT INTO slownik_plec
VALUES ('M', 'mężczyzna');

```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Następnie dodajemy dane pracowników:

W LibreOffice Base:

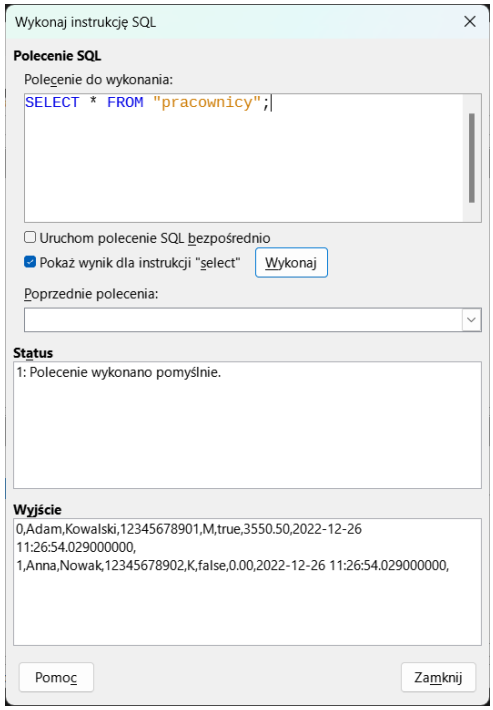
```
INSERT INTO "pracownicy"
("imie", "nazwisko",
"pesel", "plec",
"zatrudniony",
"wynagrodzenie")
VALUES ('Adam',
'Kowalski', '12345678901',
'M', 1, 3550.50);
INSERT INTO "pracownicy"
("imie", "nazwisko",
"pesel", "plec",
"zatrudniony",
"wynagrodzenie")
VALUES ('Anna', 'Nowak',
'12345678902', 'K', 0, 0);
```

```
5
```

W Microsoft Access zapytania wykonujemy pojedynczo:

```
1 INSERT INTO pracownicy
(imie, nazwisko, pesel,
plec, zatrudniony,
wynagrodzenie)
2 VALUES ('Adam',
'Kowalski', '12345678901',
'M', 1, 3550.50);
3 INSERT INTO pracownicy
(imie, nazwisko, pesel,
plec, zatrudniony,
wynagrodzenie)
4 VALUES ('Anna', 'Nowak',
'12345678902', 'K', 0, 0);
5
```

20



Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

Na koniec sprawdzamy, czy możemy odczytać wprowadzone dane:

W LibreOffice Base w oknie `Wykonaj instrukcję SQL` zaznaczamy opcję `Pokaż wyniki dla instrukcji "select"` i wpisujemy:

```
SELECT * FROM
"pracownicy";

```

id	imie	nazwisk	pesel	plec	zatrudniony	wynagrodzenie	data_zatr
1	Adam	Kowalski	12345678901	M	-1	3550,5	26.12.2022 11:19:04
2	Anna	Nowak	12345678902	K	0	0	26.12.2022 11:19:14
(Nowy)					0	0	26.12.2022 11:19:29

21

Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1B6dtEh6

W Microsoft Access:

```
SELECT * FROM pracownicy;

```

Warto zauważyć, że wartości typu „Tak”/ „Nie” w wynikach kwerendy wyświetlane są jako -1 dla „Tak” i 0 dla „Nie”.

Polecenie 2

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w prezentacji.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Uzupełnij zdanie.

Informacje na temat tabel, związków między nimi, pól, typów i ograniczeń to: _____

Ćwiczenie 2

Uzupełnij zdanie właściwym fragmentem wybranym spośród podanych opcji.

Do utworzenia bazy danych lub tabeli używamy klauzuli: _____

`ALTER` `CREATE` `DROP` `SELECT INTO`

Ćwiczenie 3

Wskaż właściwe dokończenia zdania. Zaznacz wszystkie prawidłowe odpowiedzi.
Określenie typu danych wartości przechowywanych w kolumnie...

- [ ] decyduje o relacjach między tabelami.
- [ ] determinuje wielkość potrzebnego miejsca na przechowywanie danych.
- [ ] wpływa na szybkość wykonywania zapytań.
- [ ] nie ma wpływu na działanie bazy.

Ćwiczenie 4

Wskaż, jakiego typu danych użyjesz do przechowywania danych finansowych.

- ◯ `FLOAT`
- ◯ `DECIMAL`
- ◯ `BIGINT`

Ćwiczenie 5

Zaznacz wszystkie poprawne odpowiedzi. Wskaż typy danych, w których można określić opcjonalny rozmiar.

- [ ] `BINARY`
- [ ] `DECIMAL`
- [ ] `CHAR`
- [ ] `INTEGER`

Ćwiczenie 6

Dokończ zdanie.

Daty i czas można zapisywać w bazie danych...

- ( ) tylko jako znaki.
- ( ) tylko jako liczby.
- ( ) jako liczby lub znaki.

Ćwiczenie 7

Dokończ zdanie.

Aby zagwarantować, że w kolumnie przechowującej tekst nie powtórzą się wartości, w definicji typu użyjemy ograniczenia...

- ( ) `UNIQUE`.
- ( ) `DEFAULT`.
- ( ) `AUTOINCREMENT`.

Ćwiczenie 8

Zaznacz poprawną odpowiedź. Wskaż, jakiego ograniczenia użyjemy w definicji pola tekstowego, w którym nie może się znajdować ciąg pusty.

- ◯ `CHECK (pole<>'')`
- ◯ `NOT NULL`
- ◯ `NOT ''`

Dla nauczyciela

Autor: Robert Bednarz
Przedmiot: Informatyka

Temat: Definiowanie schematu bazy danych w języku SQL, etap I

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:
> > >
> > > > d) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie,

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przygotujesz model bazy danych.
- Wyjaśnisz, czym jest schemat bazy danych.
- Przedstawisz zasady tworzenia tabel w bazie danych.
- Wymienisz typy danych i ich ograniczenia.
- Stworzysz w bazie danych tabelę za pomocą języka SQL.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie MySQL w wersji 8.0 (lub nowszej).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Definiowanie schematu bazy danych w języku SQL, etap I". Nauczyciel prosi uczniów o zapoznanie się z multimedium w sekcji „Prezentacja multimedialna".

Faza wstępna:

1. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę

o przedstawienie najważniejszych informacji z sekcji „Przeczytaj”.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna”. Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium. Zapisują ewentualne problemy i pytania, po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1–4 w sekcji „Sprawdź się”, a następnie dzielą się wynikami swojej pracy z kolegą lub koleżanką.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie”. W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 5–8 z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla systemu MySQL 8.0 (lub nowszej wersji).

Wskazówki metodyczne:

- Multimedium w sekcji „Prezentacja multimedialna” można potraktować jako zadanie domowe dotyczące analizy problemu zawartego w temacie „Definiowanie schematu bazy danych w języku SQL, etap I”.