

	<i><b>Bazy Danych laboratorium</b></i>	<b>Laboratorium BD1</b>
--	--------------------------------------------	-----------------------------

## Oracle SQL Developer

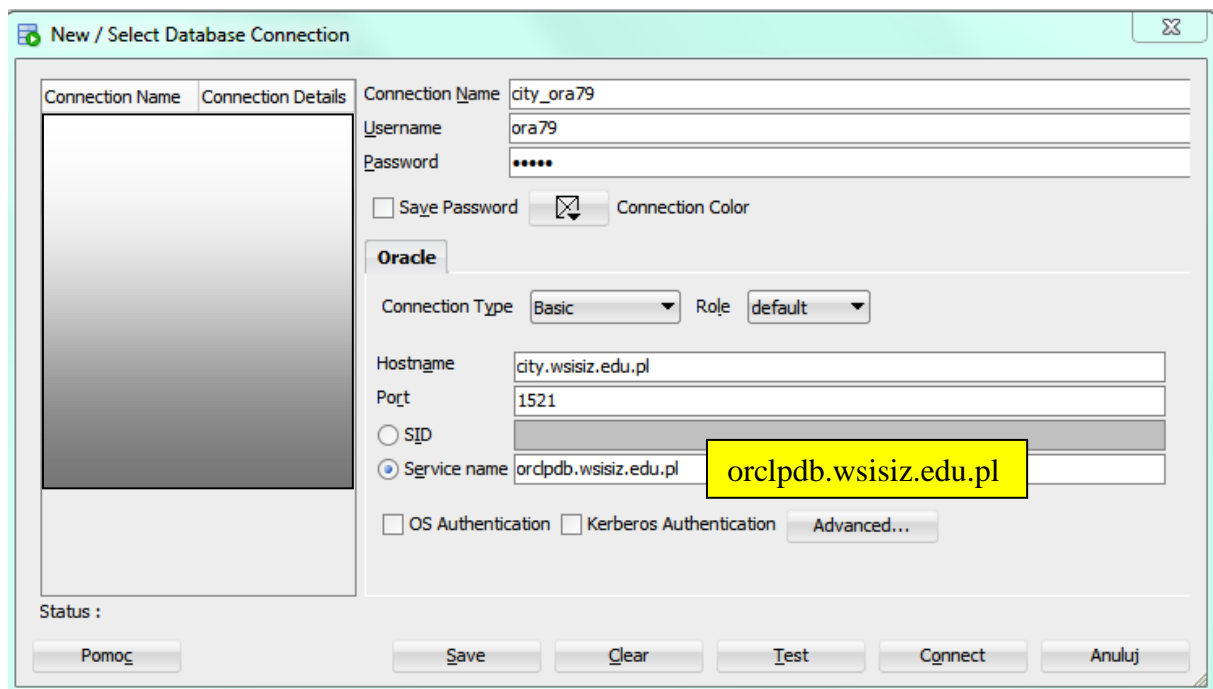
Jest to narzędzie umożliwiające łączenie się z bazami danych na serwerach Oracle, zarządzanie tymi bazami przy pomocy języka SQL oraz programowanie serwerów przy pomocy języka PL/SQL.

Oprogramowanie można pobrać ze strony producenta:

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>

wybierając odpowiednią wersję systemu operacyjnego.

W celu logowania się do bazy danych na określonym serwerze należy zdefiniować połączenie (*Connect*), tak jak na poniższym rysunku:

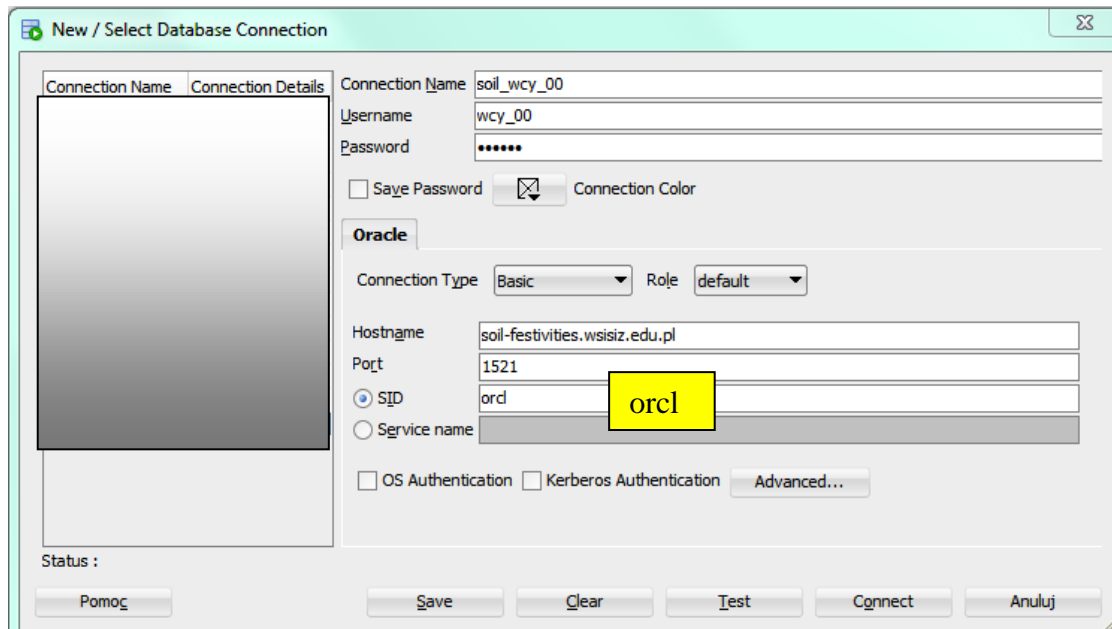


gdzie:

- Connection Name - własna nazwa połączenia,
- Username - nazwa konta (schematu) nadana przez administratora,
- Password - hasło dostępu do konta (schematu),
- Hostname - nazwa komputera, na którym znajduje się serwer Oracle,
- Port - standardowy port Oracle ma wartość 1521,
- SID - nazwa bazy danych Oracle na wskazanym serwerze,
- Service name - nazwa serwisu, przy pomocy którego następuje połączenie z bazą danych.

Drugim sposobem łączenia się z bazą Oracle jest wskazanie nazwy bazy jako *SID* zamiast nazwy serwisu *Service name*.

Przykładowo chcąc zalogować się na inny serwer Oracle ekran definicji połączenia może wyglądać tak:



gdzie zamiast nazwy serwisu należy podać wygenerowaną w procesie instalacji nazwę bazy danych, w tym przypadku *orcl*.

## SQL Plus

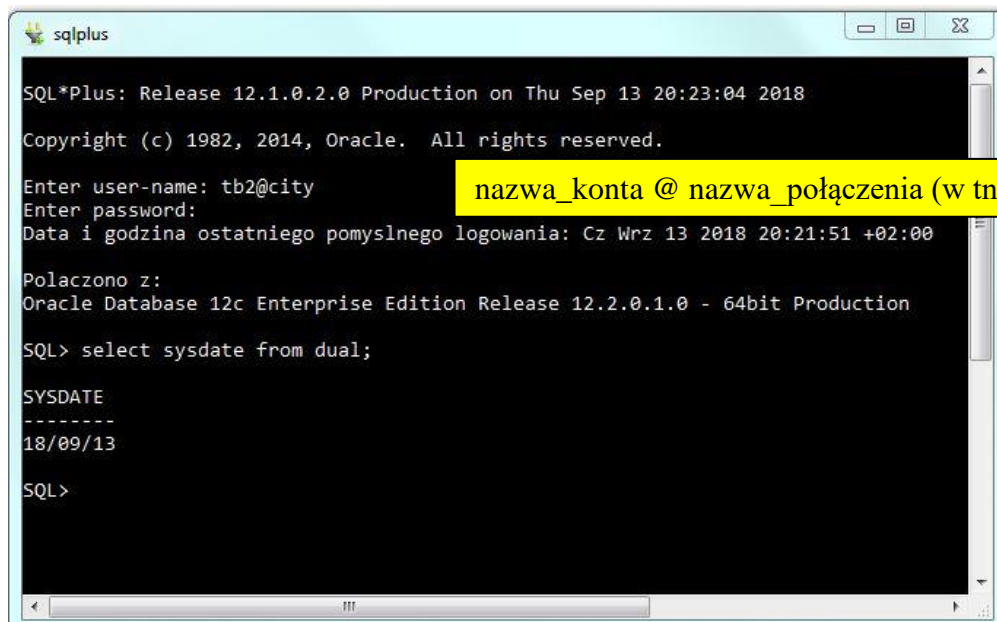
Dodatkowym narzędziem używanym do zarządzania bazą Oracle jest SQL Plus. Szczególnie często jest wykorzystywane do administrowania serwerami Oracle, do wdrażania gotowych rozwiązań oraz do testowania oprogramowania PL/SQL i SQL.

Wymaga zainstalowania oprogramowania Client Oracle na odpowiedni system operacyjny (na przykład <https://www.oracle.com/technetwork/topics/winx64soft-089540.html>) oraz zdefiniowania pliku konfiguracyjnego *tnsnames.ora* zawierającego definicje połączeń z bazami i serwerami Oracle.

Przykładowy plik konfiguracyjny może wyglądać tak:

```
soil =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = soil-festivities.wsisiz.edu.pl)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = orcl)
  )
)
city =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = city.wsisiz.edu.pl)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = orclpdb.wsisiz.edu.pl)
  )
)
```

Fragment sesji realizowanej przy użyciu SQL Plus:



```
SQL*Plus: Release 12.1.0.2.0 Production on Thu Sep 13 20:23:04 2018
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter user-name: tb2@city
Enter password:
Data i godzina ostatniego pomyslnego logowania: Cz Wrz 13 2018 20:21:51 +02:00

Polaczono z:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> select sysdate from dual;

SYSDATE
-----
18/09/13

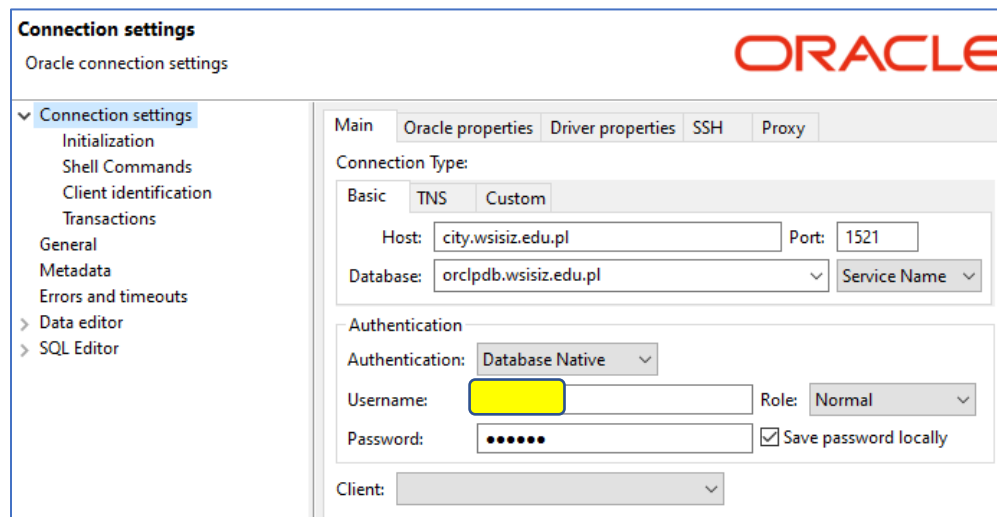
SQL>
```

## DBeaver – Universal Database Manager



Jest to interesujące (darmowe w szerokim zakresie) narzędzie służące do zarządzania różnymi systemami baz danych, nie tylko Oracle. Definiowanie połączeń z serwerami bazodanowymi odbywa się w oparciu o drivery jdbc, które są wbudowane w aplikację.

Szczegóły oraz oprogramowanie można znaleźć na stronie <https://dbeaver.io/>



**Connection settings**  
Oracle connection settings

**ORACLE**

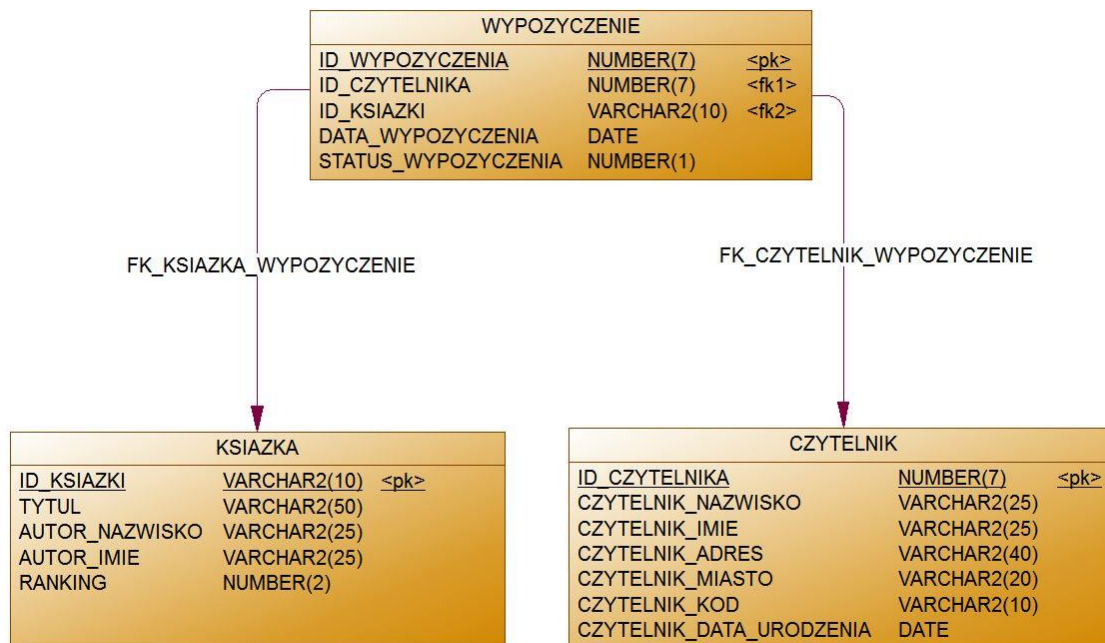
Main | Oracle properties | Driver properties | SSH | Proxy

Connection Type:  
Basic | TNS | Custom

Host: city.wsisiz.edu.pl Port: 1521  
Database: orclpdb.wsisiz.edu.pl Service Name:   
Authentication:  
Authentication: Database Native  
Username: Role: Normal  
Password: Save password locally  
Client:

## Relacyjny model bazy danych – koncepcja

Poniżej został przedstawiony model relacyjny pewnego fragmentu rzeczywistości związanego z działalnością biblioteki opracowany przy pomocy SAP PowerDesigner.<sup>1</sup>



Mamy do czynienia z trzema tabelami, z których dwie **KSIAZKA** i **CZYTELNIK** mają charakter statyczny obrazujący ewidencję książek i czytelników w bibliotece. Trzecia tabela **WYPOZYCZENIE** rejestruje fakty związane z procesem wypożyczenia, a więc ma charakter dynamiczny.

Każda tabela posiada ściśle określoną strukturę, na którą składa się przede wszystkim jej nazwa oraz specyfikacja kolumn. Przykładowo tabela **KSIAZKA** zawiera pięć kolumn o określonych nazwach i typach. Nazwy kolumn muszą być unikalne w ramach definicji tabeli czyli mogą się powtarzać w innych tabelach. Typy kolumn określają rodzaj danych w nich przechowywanych i w wielu przypadkach zależą od konkretnego systemu zarządzania bazą danych, w którym się je implementuje. Nie mniej jednak można wskazać kilka podstawowych typów uniwersalnych, takich jak: *varchar* (lub *char*) określający typy znakowe (np. nazwiska i tytuły)<sup>2</sup>, *number* określający typ liczbowy (np. kwoty lub miary) oraz *date* związany z zapamiętywaniem dat.

Typy znakowe muszą mieć zdefiniowaną skalę czyli swoją długość, np. *varchar(10)* oznacza, że w kolumnie takiego typu można umieścić łańcuch znaków o maksymalnej długości 10 znaków.

Typy liczbowe także mogą mieć zdefiniowaną skalę, np. *number(2)* oznacza możliwość przechowywania liczb całkowitych z zakresu [-99..+99]. Dodatkowo typy liczbowe mogą zawierać precyzję, np. *number(5,2)* oznacza możliwość przechowywania liczb z dokładnością do dwóch miejsc po przecinku [-999.99..+999.99]. Ale również mogą występować typy liczbowe bez skali i precyzji. Należą do nich na przykład *integer* i *float*, chociaż nie jest to zalecane w przypadku definiowania tabel.

Typ *date* umożliwia przechowywanie danych interpretowanych jako daty. Należy pamiętać, że format przechowywanej daty musi być zgodny z formatem daty zdefiniowanym na serwerze bazodanowym, np. jeśli na serwerze data systemowa jest zdefiniowana jako YY/MM/DD, to w takim samym formacie należy datę wprowadzać do tabeli lub używać specjalnych funkcji konwersji.

<sup>1</sup> PowerDesigner firmy SAP (poprzednio Sybase) jest narzędziem służącym do modelowania systemów czyli tworzenia modeli biznesowych, logicznych i fizycznych w różnych środowiskach (Oracle, Sybase, MS Server i wielu innych).

<sup>2</sup> Dla bazy danych Oracle obowiązującym typem znakowym jest *varchar2*.

Każda tabela posiada jedną (na ogół) kolumnę specjalnie wyróżnioną zwaną kluczem głównym.

**Klucz główny (Primary Key)** – jedna lub kilka kolumn w tabeli, na podstawie których system zarządzania bazą danych kontroluje dane podczas ich wprowadzania. W tabeli nie można zapisać dwóch wierszy mających takie same wartości w kolumnach klucza głównego. Przykładowo w tabeli CZYTELNIK nie można wprowadzić danych dwóch czytelników mających tę samą wartość w kolumnie ID\_CZYTELNIKA. Można wprowadzić dwa takie same nazwiska, lecz różniące się ID\_CZYTELNIKA. **Klucz główny zapewnia, że w tabeli nie ma dwóch takich samych wierszy (zapewnia niepowtarzalność danych w tabeli).**

Klucz główny może być kluczem złożonym składającym się z kilku kolumn tabeli. W takim przypadku kombinacja wartości kolumn wchodzących w skład klucza głównego musi być niepowtarzalna.

Na powyższym diagramie kolumny będące kluczami głównymi oznaczone są atrybutem <pk>. Brak jest klucza złożonego, ale przy niewielkiej modyfikacji modelu można taki klucz zdefiniować. W tabeli WYPOZYCZENIE zamiast kolumny ID\_WYPOZYCZENIA będącej kluczem głównym można zastosować złożony klucz główny składający się z trzech kolumn: ID\_CZYTELNIKA, ID\_KSIAZKI oraz DATA\_WYPOZYCZENIA, a kolumnę ID\_WYPOZYCZENIA usunąć. Oznaczałoby to, że nie jest możliwe, aby ten sam czytelnik tę samą książkę mógł wypożyczyć więcej niż raz tego samego dnia.

Kolejnym ważnym pojęciem związanym z relacyjnym modelem jest klucz obcy.

**Klucz obcy (Foreign Key)** – definiuje relację między dwiema tabelami zapewniając integralność danych. Przykładowo w tabeli KSIAZKA są trzy wiersze: (1, W pustyni i w puszczy,...), (2, Ogniem i mieczem, ....) i (3, Potop....). Rejestrując fakt wypożyczenia książki przez zarejestrowanego czytelnika czyli wprowadzając wiersz do tabeli WYPOZYCZENIE nie można podać w tym wierszu na pozycji ID\_KSIAZKI wartości 4, gdyż książki o takim identyfikatorze nie ma w tabeli KSIAZKA. Najpierw należy do tabeli KSIAZKA wprowadzić tę pozycję (4, Pan Wołodyjowski....), a następnie ją wypożyczyć wprowadzając stosowny wiersz do tabeli WYPOZYCZENIE. Mówimy, że dane w bazie danych są zintegrowane czyli spójne. Zapewnia to klucz obcy.

Podobnie nie można wypożyczyć żadnej książki czytelnikowi, który nie jest zarejestrowany.

Na powyższym diagramie klucze obce zaznaczone są jako:

- <fk1> - klucz obcy między tabelami WYPOZYCZENIE i CZYTELNIK (relacja FK\_CZYTELNIK\_WYPOZYCZENIE),
- <fk2> - klucz obcy między tabelami WYPOZYCZENIE i KSIAZKA (relacja FK\_KSIAZKA\_WYPOZYCZENIE).

Należy zwrócić uwagę, że dwie kolumny, poprzez które tabele są w relacji do siebie mają te same nazwy (ale mogą być różne) i te same typy. Przykładem jest kolumna ID\_CZYTELNIKA, która występuje w tabelach CZYTELNIK (jako klucz główny) i WYPOZYCZENIE i w obu przypadkach jest tego samego typu.

## DDL<sup>3</sup> – zdania SQL tworzące i modyfikujące strukturę bazy danych

Podstawowymi zdaniami z tej grupy zdań SQL są zdania *create*, *alter*, *rename* oraz *drop*.

Zdanie *create* służy do tworzenia wszelkich obiektów bazodanowych, a w szczególności tabel.

Ogólna postać zdania:

```
create table nazwa_tabeli ( kol_ a          typ_danych Primary Key,
                           kol_ b          typ_danych,
                           .....
                           kol_ n          typ_danych,
                           Foreign Key (kol_b) References nazwa_innej_tabeli (kol_c_innej_tabeli)
                           );
```

<sup>3</sup> DDL – Data Definition Language

Sposoby definiowania klucza głównego:

1. Na poziomie kolumny:

```
.....
kol_a    typ_danych Primary Key,
.....
```

2. Na poziomie tabeli:

```
.....
kol_a    typ_danych,
.....
kol_ostatnia    typ_danych,
Primary Key(kol_a),
.....
```

3. Klucz złożony (tylko na poziomie tabeli):

```
.....
kol_a    typ_danych,
kol_b    typ_danych,
.....
kol_ostatnia    typ_danych,
Primary Key(kol_a, kol_b),
.....
```

Definiowanie klucza obcego:

Jest kilka sposobów definiowania klucza obcego:

1. Zapis na końcu definicji struktury tabeli:

```
create table nazwa_tabeli ( kol_ a  typ_danych Primary Key,
                           kol_ b  typ_danych,
                           .....
                           kol n    typ_danych,
Foreign Key (kol_b) References nazwa_innej_tabeli (kol_c_innej_tabeli)
);
```

przy czym:

- w momencie jego definiowania tabela *nazwa\_innej\_tabeli* musi istnieć w schemacie,
- *kol\_c\_innej\_tabeli* jest kluczem głównym w tabeli *nazwa\_innej\_tabeli*.

2. Definicja na poziomie kolumny:

```
create table nazwa_tabeli (.....,
                           kol_ b  typ_danych References nazwa_innej_tabeli (kol_b_innej_tabeli),
                           .....
                           kol n    typ_danych
);
```

3. Przy użyciu zdania *alter* (przykład poniżej).

Język SQL posiada odpowiednie konstrukcje zdaniowe (*alter*, *rename*, *drop*) do modyfikacji już zaprojektowanego i zaimplementowanego modelu danych. Modyfikacje te mogą polegać na usuwaniu, zmianie nazwy kolumn i tabel, dodawaniu kolumn, zmianie typów danych istniejących kolumn, jak również definiowaniu referencji czyli kluczy obcych. Poniżej przedstawione zostały na przykładach podstawowe zdania języka SQL umożliwiające modyfikacje modelu danych.

Usunięcie kolumny z tabeli:

```
alter table KSIAZKA  
drop column RANKING;
```

Dodanie kolumny do tabeli:

```
alter table KSIAZKA  
add RANKING varchar2(3);
```

Modyfikacja istniejącej kolumny w tabeli (typu danych):

```
alter table KSIAZKA  
modify RANKING varchar2(5);
```

Zmiana nazwy kolumny w tabeli:

```
alter table KSIAZKA  
rename column RANKING to OCENA;
```

Zmiana nazwy tabeli:

```
rename KSIAZKA to KSIAZKA_ZMIANA;
```

Zdefiniowanie referencji:

```
alter table WYPOZYCZENIE  
add constraint FK_CZYTELNIK_WYPOZYCZENIE foreign key (ID_CZYTELNIKA)  
references CZYTELNIK (ID_CZYTELNIKA);
```

przy czym zakłada się, że obie tabele (WYPOZYCZENIE i CZYTELNIK) istnieją w schemacie.

Usunięcie referencji:

```
alter table WYPOZYCZENIE  
drop constraint FK_CZYTELNIK_WYPOZYCZENIE;
```

Usunięcie tabeli z bazy danych:

```
drop table KSIAZKA_ZMIANA;
```

## DML<sup>4</sup> – zdania SQL umożliwiające wprowadzanie, aktualizacje i usuwanie danych z tabel

Podstawowymi zdaniami tej grupy są zdania: *insert*, *update* oraz *delete*.

Zdanie *insert* służy do wprowadzania wierszy do tabeli. Jest kilka sposobów użycia tego zdania:

1. Wprowadzanie danych w kolejności zgodnej ze strukturą danych:

```
insert into KSIAZKA  
(ID_KSIAZKI, TYTUL, AUTOR_NAZWISKO, AUTOR_IMIE, RANKING)
```

---

<sup>4</sup> DML – Data Manipulation Language

*values*

(15, 'Pan Wołodyjowski', 'Sienkiewicz', 'Henryk', 10);

2. Uproszczony wariant sposobu pierwszego:

*insert into* KSIAZKA

*values*

(15, 'Pan Wołodyjowski', 'Sienkiewicz', 'Henryk', 10);

3. Wprowadzanie danych w kolejności niezgodnej ze strukturą danych:

*insert into* KSIAZKA

(ID\_KSIAZKI, AUTOR\_NAZWISKO, AUTOR\_IMIE, TYTUL, RANKING)

*values*

(15, 'Sienkiewicz', 'Henryk', 'Pan Wołodyjowski', 10);

4. Wprowadzanie niepełnych danych:

a).

*insert into* KSIAZKA

(ID\_KSIAZKI, AUTOR\_NAZWISKO, TYTUL)

*values*

(15, 'Sienkiewicz', 'Pan Wołodyjowski');

b).

*insert into* KSIAZKA

*values*

(15, 'Pan Wołodyjowski', 'Sienkiewicz', ' ', 10);

W tym ostatnim przypadku dane są wprowadzane nie do wszystkich kolumn tabeli (porównaj ze strukturą tabeli KSIAZKA). W wariancie 4a wyspecyfikowane są wybrane kolumny tabeli i odpowiadające im wartości, a w wariancie 4b wyspecyfikowane są domyślnie wszystkie kolumny, ale we frazie *values* brak danych reprezentowany jest przez pusty parametr (' ').

Zdanie *update* służy do modyfikacji pojedynczego wiersza lub grupy wierszy już istniejących w tabeli.

Przykładowo zdanie:

*update* KSIAZKA

*set* RANKING = 9

*where* ID\_KSIAZKI = 5;

spowoduje, że w tabeli KSIAZKA egzemplarz biblioteczny o numerze 5 otrzyma ocenę rankingową 9. Ponieważ ID\_KSIAZKI jest kluczem głównym tabeli KSIAZKA, więc jest pewność, że tylko jedna pozycja książkowa zostanie tym zdaniem zmodyfikowana.

Zdanie

*update* KSIAZKA

*set* RANKING = 9

*where* AUTOR\_NAZWISKO = 'Kraszewski';

spowoduje, że wszystkie książki Kraszewskiego otrzymają ranking 9, gdyż w kolumnie AUTOR\_NAZWISKO zapis 'Kraszewski' może powtarzać się wielokrotnie, nie jest ona bowiem kolumną klucza głównego. Natomiast zdanie:

*update* KSIAZKA

*set* RANKING = 9;

spowoduje, że wszystkie pozycje książkowe w tabeli KSIAZKA otrzymają ranking 9.



Zdanie

```
update KSIAZKA
  set AUTOR_NAZWISKO ='Sienkiewicz',
      AUTOR_IMIE = 'Henryk'
  where TYTUL = 'W pustyni i w puszczy';
```

umożliwia modyfikacje danych w kilku kolumnach jednocześnie.

Zdanie *delete* służy do usuwania pojedynczego wiersza lub grupy wierszy już istniejących w tabeli.

Przykładowo zdanie:

```
delete WYPOZYCZENIA
  where DATA_WYPOZYCZENIA < '2021/01/01';
```

spowoduje, że z tabeli WYPOZYCZENIA zostaną usunięte wszystkie informacje o wypożyczeniach dokonanych do końca 2020 roku.

Zdanie

```
delete WYPOZYCZENIA
  where ID_WYPOZYCZENIA = 4;
```

spowoduje usunięcie jednego wiersza z tabeli, gdyż ID\_WYPOZYCZENIA jest kluczem głównym.

Zdanie

```
delete WYPOZYCZENIA;
```

usunie z tabeli WYPOZYCZENIA wszystkie wiersze.

Z używaniem zdań języka DML (*insert*, *update* i *delete*) nierozdzielnie związane są dwa zdania: *commit* (zatwierdzenie transakcji) i *rollback* (cofnięcie transakcji).

Poniżej zostanie przedstawiony scenariusz użycia obu tych zdań:

1. W tabeli WYPOZYCZENIA znajduje się pewna liczba wierszy (np. 20),
2. Użycie zdania:

```
delete WYPOZYCZENIA;
```

spowoduje usunięcie z tabeli wszystkich wierszy,

3. Użycie zdania *rollback* cofa skutki działania wszystkich zdań DML od ostatniego *commit* lub od początku sesji, w tym przypadku powyższego zdania czyli nadal jest 20 wierszy,
4. Użycie zdania:

```
delete WYPOZYCZENIA
  where DATA_WYPOZYCZENIA > '2021/03/01';
```

spowoduje usunięcie pewnej grupy wierszy (np. 5),

5. Użycie zdania *commit* zatwierdzi powyższe zdanie i w tabeli WYPOZYCZENIA na trwałe będzie 15 wierszy.
6. Użycie zdania *rollback* nic już nie zmieni, gdyż między ostatnim zdaniem *commit* a obecnym *rollback* nic się nie wydarzyło.

Należy pamiętać, że zdania *create...*, *alter...*, *drop...*, *rename...* generują w niejawnym sposób zatwierdzenie transakcji czyli *commit*.

Dlatego, jeśli w powyższym scenariuszu po realizacji punktu 2, a przed realizacją punktu 3 wykonane zostanie, na przykład, zdanie:

```
create table tmp_tabela  
(  
    iden varchar2(3) primary key  
    , opis varchar2(100)  
);
```

to nastąpi zatwierdzenie kasowania zawartości tabeli WYPOZYCZENIE i dalsze działania na podstawie scenariusza nie mają sensu.

## Zadania do samodzielnego wykonania:

### Działanie na modelu Biblioteka

1. Utworzyć w swoim schemacie tabele modelu relacyjnego *Biblioteka* przy pomocy skryptu *lab\_BD1\_ver2.sql* metodą „@c:\temp\lab\_BD1\_ver2.sql”. Wypełnić tabele przykładowymi danymi wykorzystując wszystkie postacie zdanie *insert*. Układ danych powinien zapewnić obserwację relacji 1:N czyli jedna książka była wypożyczona wiele razy oraz jeden czytelnik kilka razy przychodził do biblioteki w celu wypożyczenia książek.
2. Na tak skonfigurowanym modelu z danymi dokonać modyfikacji struktury poprzez zapewnienie poprawności rejestrowania wypożyczeń. Struktura tabeli WYPOZYCZENIA dopuszcza możliwość rejestrowania transakcji bez określenia numeru wypożyczonej książki. Należy to zmienić.
3. Dokonać modyfikacji danych w dwóch wariantach. Pierwszy - modyfikacji podlega jeden wiersz (modyfikacja w oparciu o klucz główny), a drugi - modyfikacji podlega zbiór wierszy, ale nie wszystkie.
4. Zaprezentować skuteczność lub nie kasowania wiersza nadrzędnego w dwóch wariantach. Pierwszy - wiersz nadrzędny jest w relacji z wierszami podrzędnymi, a drugi - nie jest.

### Tworzenie nowego modelu

1. Zaprojektować prosty model bazy danych składający się z trzech tabel i odzwierciedlający ewidencję studentów uczelni z uwzględnieniem podziału na rodzaj studiowania i kierunki studiów.

Tabela BD1\_Student – zawierająca dane studenta,

Tabela BD1\_Kierunki\_Studiow – zawierająca nazwy kierunków  
(Informatyka, Psychologia,.....),

Tabela BD1\_Rodzaj\_Studiow – zawierająca nazwy sposobów studiowania  
(Dzienne, Zaoczne, Indywidualne, Podyplomowe, MBA).

Tabele BD1\_Kierunki\_Studiow oraz BD1\_Rodzaj\_Studiow są klasycznymi tabelami słownikowymi. Na ogół struktura takich tabel zawiera dwie kolumny: kod i opis:

```
(  
    kod varchar2(3) primary key  
    ,opis varchar2(100)  
);
```

Tabele takie często są używane w systemach bazodanowych do przechowywania danych stałych, na przykład mogą to być zbiory nazw województw, nazw walut, komórek organizacyjnych itp. Kolumna stanowiąca klucz główny może być typu `varchar2` (ale zdecydowanie krótsza niż kolumna stanowiąca opis obiektu) lub typu numerycznego.

Tabelę `BD1_Student` należy traktować jako tabelę dynamiczną (transakcyjną).

2. Zdefiniować klucze główne oraz klucze obce określające relacje między tabelami:

Relacje: student studiuje na określonym kierunku,  
student studiuje określonym sposobem studiowania.

3. Na tak zbudowanym modelu przećwiczyć wprowadzanie danych do tabel, ich modyfikacje oraz usuwanie na różne sposoby (zdania *insert*, *update*, *delete*).
4. Przećwiczyć skuteczność działania klucza obcego w celu utrzymania spójności bazy danych (np. poprzez wprowadzenie do ewidencji studenta, który studiuje na kierunku, którego brak w ewidencji kierunków oraz skasowanie z tabeli kierunków studiów pozycji, w przypadku gdy na ten kierunek są zapisani studenci).
5. Zmodyfikować model bazy danych poprzez wprowadzenie do jednej z tabel nowej kolumny (np. do tabeli kierunków studiów kolumnę określającą datę uruchomienia danego kierunku) i uzupełnić tę kolumnę danymi (zrobić to na dwa sposoby: wszystkie wiersze na raz oraz wybiórczo).
6. Sprawdzić czy jest możliwość zmiany struktury tabeli poprzez zmianę wielkości wybranej kolumny w przypadku, gdy jest ona wypełniona (np. kolumna `Nazwisko` w tabeli `BD1_Student`). Należy zwiększyć rozmiar kolumny np. do 100 i zmniejszyć np. do 5.
7. Utworzyć dwa skrypty. Pierwszy (np. *create\_BD1.sql*) zawierający zdania tworzące obiekty bazodanowe i zdania wprowadzające testowe dane do tabel oraz drugi (np. *drop\_BD1.sql*) usuwający wszystkie tabele opracowanego modelu bazy danych.
8. Usunąć jedną z referencji między tabelami i przetestować tego skutki poprzez wprowadzanie danych umożliwiających osiągnięcie przez bazę danych stanu niespójności. Odpowiednimi zdaniami SQL doprowadzić z powrotem do spójności bazy danych.