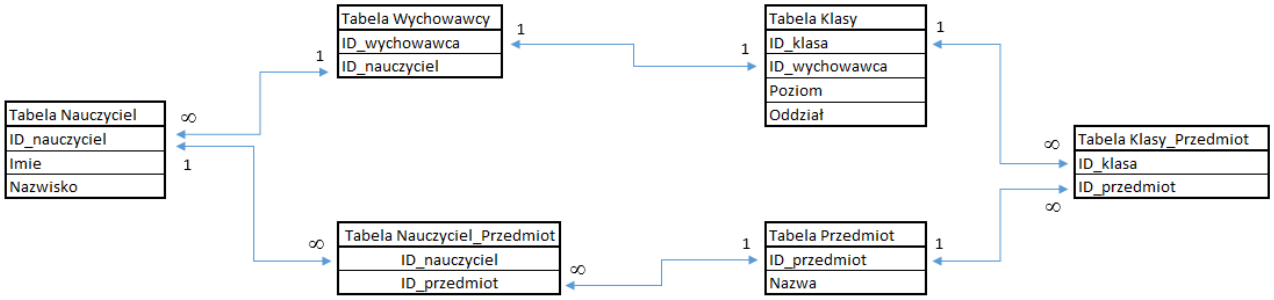
Wykorzystanie powiązań między tabelami

Zanim przejdę do szczegółów dobrze jest zapoznać się z niektórymi pojęciami wykorzystywanymi na etapie projektowania bazy danych. Pierwsza faza jest związana z tworzeniem diagramu związków encji, a encja to jest pewna klasa (zbiór) obiektów występujących w rzeczywistości. Uczeń może być przykładem encji, ale już Franek Kowalski z klasy II A będzie instancją tej encji. Instancja ma atrybuty, czyli właściwości, tj. opis prezentowanej wartości, np. czym ona będzie: liczbą całkowitą, czy ciągiem znaków, czy datą.
Mając encje i ich instancje, należy się zastanowić, który rodzaj bazy danych wybrać. Jeśli zdecydujemy się na relacyjną, czyli np. Microsoft SQL, to pojawia się kolejne pytanie: jakie będą powiązania między tabelami.
Może być powiązanie jeden do jednego (np. Dyrektor szkoły ustali, że dany nauczyciel może być wychowawcą tylko jednej klasy).
Powiązanie jeden do wielu (np. funkcję wychowawcy może pełnić wielu nauczycieli)
Powiązanie wiele do wielu (np. każdy nauczyciel może uczyć wielu przedmiotów, ale też każdy przedmiot może mieć przypisanych wielu nauczycieli).



Warto zauważyć, że w przypadku powiązań wiele do wielu tworzymy tabele pomocnicze.
Taki graficzny sposób powiązań można wykonać w programie Data Modeler (https://www.oracle.com/tools/downloads/sql-data-modeler-downloads.html)

Kiedy na podstawie rozmowy z odbiorcą powstaną już encje i będziemy wiedzieć, które instancje będą one zawierać, a także, jakie będą powiązania między tymi encjami, przychodzi czas na implementację, czyli tworzenie konkretnej bazy danych np. w Microsoft SQL.

Stworzę przykładową bazę Szkoła zawierającą kilka tabel, zgodnie z przedstawionym schematem:

```
CREATE DATABASE Szkola
USE Szkola
```

Najpierw wykonam tabelę z nauczycielami i wypełnię ją odpowiednimi danymi:

```
CREATE TABLE Nauczyciel (
ID_nauczyciel INT NOT NULL PRIMARY KEY,
Imie NVARCHAR(20),
Nazwisko NVARCHAR(30)
)

INSERT INTO Nauczyciel VALUES     (1, 'Adam', 'Nowobilski'),
                                  (2, 'Joanna', 'Trybuła'),
                                  (3, 'Franciszek', 'Mocny'),
                                  (4, 'Stanisław', 'Frączysty'),
                                  (5, 'Adrian', 'Nowakowski'),
                                  (6, 'Sabina', 'Podsiadło'),
                                  (7, 'Beata', 'Tragarz'),
                                  (8, 'Anastazja', 'Wolny'),
                                  (9, 'Zbigniew', 'Bodziony'),
                                  (10, 'Andrzej', 'Piaseczny')
SELECT * FROM Nauczyciel
```

Potem stworzę tabelę z wychowawcami. Zadbam o to, by była możliwość wykorzystywania danych z dwóch utworzonych tabel. W tym celu stworzę dodatkowe pole z kluczem obcym. To będzie taka część wspólna między tymi tabelami, pomost na którym można utworzyć powiązanie. Do tego celu wykorzystuje się polecenie REFERENCES i po tym słowie podaje się nazwę tabeli i pola tej tabeli, które jest kluczem głównym i odpowiada polu, które utworzyliśmy, by powstał odpowiednik – klucz obcy w tabeli Wychowawcy
.

```
CREATE TABLE Wychowawcy (
ID_wychowawca INT NOT NULL PRIMARY KEY,
ID_nauczyciel INT NOT NULL REFERENCES Nauczyciel(ID_nauczyciel)
)
```

```
INSERT INTO Wychowawcy VALUES     (1, 1),
                                  (2, 4),
                                  (3, 5),
                                  (4, 7),
                                  (5, 10)

SELECT * FROM Wychowawcy
```

Do mojego zadania przyda się jeszcze tabela Klasy, która też będzie powiązana z tabelą wychowawcy:

```
CREATE TABLE Klasy (
ID_klasa INT NOT NULL PRIMARY KEY,
ID_wychowawca INT NOT NULL REFERENCES Wychowawcy(ID_wychowawca),
Poziom INT,
Oddzial CHAR(1)
)

INSERT INTO Klasy VALUES   (1, 3, 1, 'A'),
                           (2, 5, 1, 'B'),
                           (3, 1, 2, 'A'),
                           (4, 2, 2, 'B'),
                           (5, 4, 3, 'A')
SELECT * FROM Klasy
```

W klasach realizowane są przedmioty, stąd kolejna tabela o takiej nazwie:

```
CREATE TABLE Przedmiot (
ID_przedmiot INT NOT NULL PRIMARY KEY,
Nazwa VARCHAR(30)
)

INSERT INTO Przedmiot VALUES      (1, 'język polski'),
                                  (2, 'język angielski'),
                                  (3, 'język niemiecki'),
                                  (4, 'historia'),
                                  (5, 'wiedza o społeczeństwie'),
                                  (6, 'geografia'),
                                  (7, 'biologia'),
                                  (8, 'chemia'),
                                  (9, 'fizyka'),
                                  (10, 'matematyka'),
                                  (11, 'wychowanie fizyczne'),
                                  (12, 'podstawy przedsiębiorczości')

SELECT * FROM Przedmiot
```

A przedmioty połączę z tabelą nauczyciel poprzez tabelę pomocniczą, która będzie się składać tylko z dwóch kluczy obcych. Taki zabieg jest dlatego, aby w tabelach Przedmiot lub Nauczyciel nie powtarzać danych. Tabela Nauczyciel_Przedmiot powstaje w przypadku relacji wiele do wielu.
W tabeli Nauczyciel_Przedmiot kluczem głównym jest para, bo tylko ona jest niepowtarzalna.

```
CREATE TABLE Nauczyciel_Przedmiot (
ID_nauczyciel INT NOT NULL REFERENCES Nauczyciel(ID_nauczyciel),
ID_przedmiot INT NOT NULL REFERENCES Przedmiot(ID_przedmiot),
PRIMARY KEY(ID_nauczyciel, ID_przedmiot)
)

INSERT INTO Nauczyciel_Przedmiot VALUES  (1, 2),
                                         (1, 4),
                                         (2, 1),
                                         (3, 7),
                                         (4, 3),
                                         (4, 12),
                                         (4, 5),
                                         (5, 5),
                                         (5, 9),
                                         (6, 10),
                                         (7, 6),
                                         (7, 1),
```

```
                                        (8, 11),
                                        (9, 8),
                                        (10, 2),
                                        (10,7)

SELECT * FROM Nauczyciel_Przedmiot
```

Podobny do poprzedniego jest sposób wykonania tabeli pomocniczej Klasy_Przedmiot:

```
CREATE TABLE Klasy_Przedmiot (
ID_klasa INT NOT NULL REFERENCES Klasy(ID_klasa),
ID_przedmiot INT NOT NULL REFERENCES Przedmiot(ID_przedmiot),
PRIMARY KEY(ID_klasa, ID_przedmiot)
)

INSERT INTO Klasy_Przedmiot VALUES      (1, 1), (1,2), (1,4), (1,6), (1,7), (1,8), (1,9), (1,11),
                                        (2, 1), (2,3), (2,4), (2,6), (2,7), (2,8), (2,9), (2,11),
                                        (3, 1), (3,2), (3,4), (3,5), (3,6), (3,11), (3, 12),
                                        (4, 1), (4,3), (4,4), (4,5), (4,6), (4,11), (4, 12),
                                        (5, 1), (5,2), (5,3), (5,4), (5,5), (5,11), (5, 12)

SELECT * FROM Klasy_Przedmiot
```

Jesteśmy już w posiadaniu takiej prostej przykładowej bazy danych, ale to wystarczy, by móc zaprezentować zapytanie skierowane do bazy w taki sposób, żeby zwrócone zostały dane z dwóch tabel:

Przykładowe zadania:

1. Pokaż imiona i nazwiska wychowawców klas

```
SELECT N.Imie, N.Nazwisko, K.Poziom, K.Oddzial
FROM Nauczyciel N JOIN  Wychowawcy W
ON N.ID_nauczyciel = W.ID_nauczyciel
JOIN Klasy K
ON W.ID_wychowawca = K.ID_wychowawca
```

2. Pokaż przedmioty i imiona i nazwiska nauczycieli którzy ich uczą

```
SELECT P.Nazwa, N.Imie, N.Nazwisko
FROM Przedmiot P JOIN Nauczyciel_Przedmiot NP
ON P.ID_przedmiot = NP.ID_przedmiot
JOIN Nauczyciel N
ON NP.ID_nauczyciel = N.ID_nauczyciel
ORDER BY Nazwa ASC
```

3. Sprawdź, ile przedmiotów uczy każdy z nauczycieli

```
SELECT N.Nazwisko, N.Imie,
COUNT(P.Nazwa) AS [Ilosc przedmiotow]
FROM Przedmiot P JOIN Nauczyciel_Przedmiot NP
ON P.ID_przedmiot = NP.ID_przedmiot
JOIN Nauczyciel N
ON NP.ID_nauczyciel = N.ID_nauczyciel
GROUP BY Nazwisko, Imie
ORDER BY [Ilosc przedmiotow] ASC
```

4. Zbuduj tabelę, w której będzie zestawienie wszystkich przedmiotów nauczanych w każdej klasie

```
SELECT K.Poziom, K.Oddzial, P.Nazwa
FROM Klasy K JOIN Klasy_Przedmiot KP
ON K.ID_klasa = KP.ID_klasa
JOIN Przedmiot P
ON KP.ID_przedmiot = P.ID_przedmiot
ORDER BY Poziom, Oddzial  ASC
```

5. Sprawdź, kto jest wychowawcą w klasie 2A

```
SELECT K.Poziom, K.Oddzial, N.Imie, N.Nazwisko
```

```
FROM Nauczyciel N JOIN  Wychowawcy W
ON N.ID_nauczyciel = W.ID_nauczyciel
JOIN Klasy K
ON W.ID_wychowawca = K.ID_wychowawca
WHERE Poziom = 2 AND Oddzial = 'A'
```

6. Sprawdź, kto uczy języka polskiego

```
SELECT P.Nazwa, N.Imie, N.Nazwisko
FROM Przedmiot P JOIN Nauczyciel_Przedmiot NP
ON P.ID_przedmiot = NP.ID_przedmiot
JOIN Nauczyciel N
ON NP.ID_nauczyciel = N.ID_nauczyciel
WHERE Nazwa = 'język polski'
```

7. Sprawdź, w których klasach jest nauczana geografia

```
SELECT K.Poziom, K.Oddzial, P.Nazwa
FROM Klasy K JOIN Klasy_Przedmiot KP
ON K.ID_klasa = KP.ID_klasa
JOIN Przedmiot P
ON KP.ID_przedmiot = P.ID_przedmiot
WHERE Nazwa = 'geografia'
ORDER BY Poziom, Oddzial  ASC
```

Na koniec jeszcze ciekawostka. Jeśli nie chcielibyśmy otrzymać wszystkich danych w tabeli, a jedynie kilka, możemy skorzystać z następującego zapytania:

```
SELECT TOP 3 * FROM Nauczyciel
ORDER BY Nazwisko
```

Można też ograniczyć ilość wyświetlanych kolumn:

```
SELECT TOP 3 Poziom, Oddzial FROM Klasy
ORDER BY Poziom, Oddzial
```

Gdyby zawartość tych kolumn była jednakowa np. w kolumnie poziom, po której porządkujemy dane, to otrzymamy dodatkowo jeszcze te rekordy, które zawierają jednakowe dane:

```
SELECT TOP 3 WITH ties Poziom, Oddzial FROM Klasy
ORDER BY Poziom
```