Projekt bazy danych firmy edukacyjnej

Przedmiot: Podstawy Baz Danych Autorzy: Maciej Nowakowski, Zuzanna Stajniak, Mateusz Lampert

Użytkownicy bazy danych:

1. Administrator
2. Dyrektor Placówki
3. Pracownik biura administracji
4. Pracownik biura dydaktyki
5. Prowadzący
6. Tłumacz
7. Student

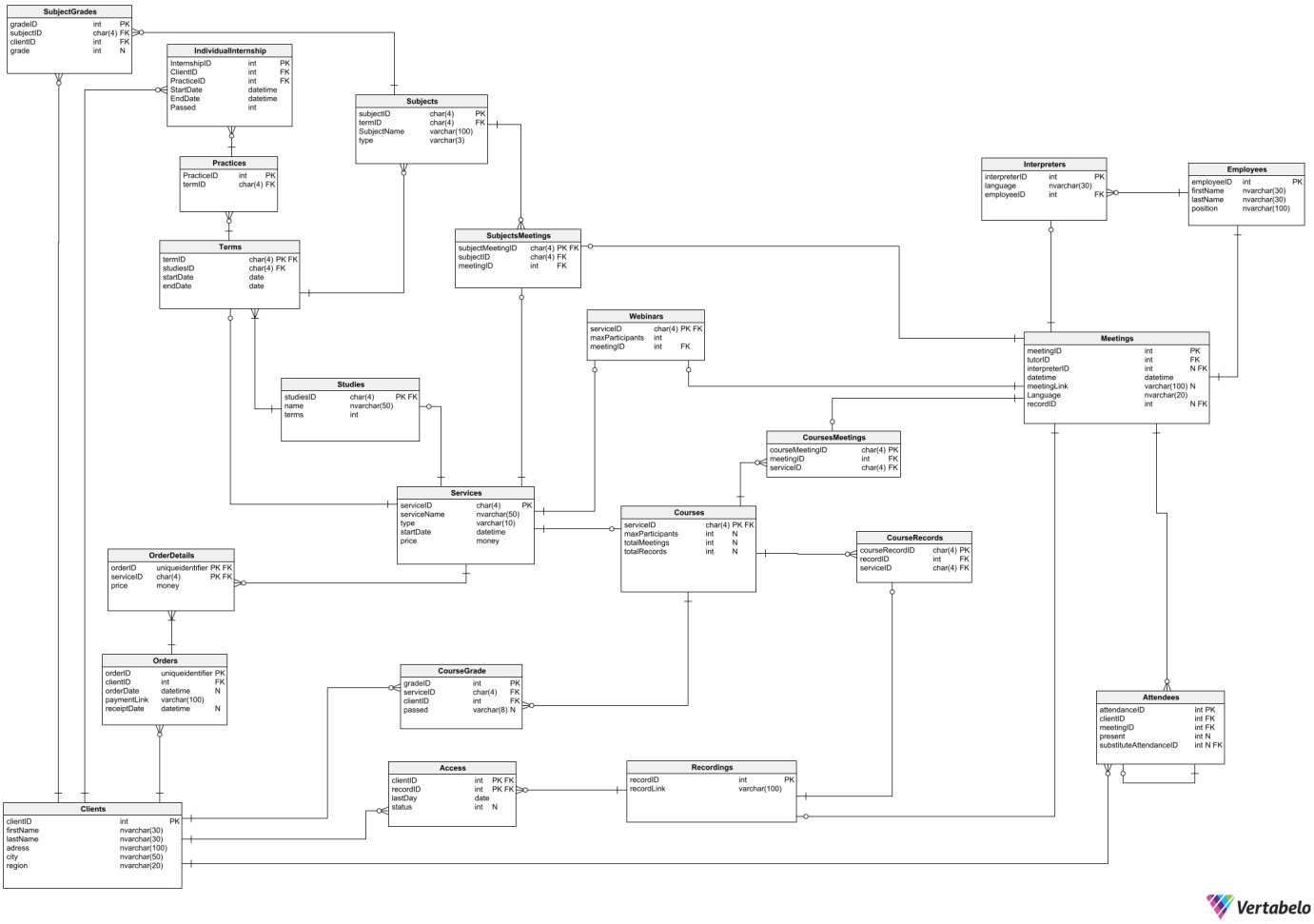
Funkcje Użytkowników:

1. Administrator
    - Usuwanie nagrań
2. Dyrektor placówki
    - Dodawanie pracowników
    - Indywidualne zmienianie terminów opłat
3. Pracownik biura administracji
    - Generowanie raportów
    - Układanie planu zajęć
    - Zmiana harmonogramu zajęć z przyczyn losowych
4. Pracownik biura dydaktyki
    - Generowanie raportów
    - Generowanie i wysyłanie dyplomu
    - Weryfikowanie czy użytkownik zaliczył dany kurs lub studia
5. Prowadzący
    - Tworzenie sylabusa nowego przedmiotu
    - Tworzenie webinarów
    - Zakładanie kursów
    - Sprawdzanie obecności na stacjonarnych zajęciach
    - Weryfikacja odrabiania nieobecności na studiach.
6. Tłumacz
7. Student
    - Dodawanie usług do koszyka
    - Opłacanie usług w koszyku
    - Zapis na zajęcia do odrabiania nieobecności na studiach
8. Każdy użytkownik
    - Możliwość przeglądania oferty kursów

Funkcje Systemowe:

- Sprawdzanie obecności zdalnej
- Sprawdzanie czy użytkownik ma dostęp do usługi
- Sprawdzenie czy zapis na daną usługę jest możliwy.

Schemat bazy danych



Opis tabel

Clients

Tabela przechowuje podstawowe dane o kliencie. Zawiera identyfikator klienta (clientID), imię oraz nazwisko (firstName, lastName) oraz dane adresowe (adress, city, region).

Klucz główny: clientID

Orders

Tabela przechowuje podstawowe dane o zamówieniu. Zawiera identyfikator zamówienia (orderID), indentyfikator klienta (clientID), datę zamówienia (orderDate), link do płatności (paymentLink) oraz datę przyjęcia płatności (receiptDate).

Klucz główny: orderID

Klucz obcy: clientID (z tabelą clients)

OrderDetails

Tabela przechowuje szczegółowe dane o zamówieniu. Zawiera identyfikator zamówienia (orderID), identyfikator usługi w koszyku (serviceID) oraz cenę za tą usługę (price).

Klucze główne: orderID, serviceID

Klucze obce: orderID (z tabelą **Orders**), serviceID (z tabelą **Services**)

Services

Tabela przechowuje podstawowe dane o dostępnych usługach edukacyjnych. Zawiera identyfikator usługi (serviceID), nazwę usługi (serviceName), typ usługi (type - webinar, kurs, studia, zjazd, pojedyńcze spotkanie), datę rozpoczęcia (startDate) oraz cenę(price).

Klucz główny: serviceID

Studies

Tabela przechowuje podstawowe dane o studiach. Zawiera identyfikator studiów (studiesID), nazwę (name) oraz liczbę zjazdów (terms).

Klucz główny: studiesID

Klucz obcy: studiesID (z tabelą **Services**)

Terms

Tabela przechowuje podstawowe dane o zjeździe na studiach. Zawiera identyfikator zjazdu(termID), identyfikator studiów (studiesID), datę rozpoczęcia i zakończenia(startDate, endDate).

Klucz główny: termID

Klucze obce: studiesID (z tabelą **Studies**), termID (z tabelą **Practices**)

Subjects

Tabela zawiera szczegółowe informacje o przedmiotach realizowanych podczas poszczególnych zjazdów. Zawiera identyfikator przedmiotu(subjectID), identyfikator zjazdu (termID), nazwę przedmiotu (subjectName), kategorię przedmiotu (type)

Klucz główny: subjectID

Klucz obcy: termID (z tabelą **Terms**)

SubjectGrades

Tabela przechowuje informację o ocenach oraz zaliczeniu przedmiotu. Zawiera identyfikator oceny(gradeID), identyfikator przedmiotu (subjectID), identyfikator klienta (clientID) oraz informację o ocenie (grade)

Klucz główny: gradeID

Klucze obce: subjectID (z tabelą **Subjects**), clientID (z tabelą **Clients**)

Employees

Tabela przechowuje podstawowe dane o pracownikach. Zawiera identyfikator pracownika (employeeID), imię oraz nazwisko (firstName, lastName) oraz stanowisko (position).

Klucz główny: employeeID

Interpreters

Tabela przechowuje podstawowe dane o tłumaczu. Zawiera identyfikator tłumacza (interpreterID), identyfikator pracownika (employeeID) oraz język, którym się posługuje (language)

Klucz główny: interpreterID

Klucz obcy: employeeID (z tabelą **Employees**)

Meetings

Tabela przechowuje dane o spotkaniu. Zawiera identyfikator spotkania (meetingID), identyfikator tłumacza (interpreterID), datę i czas spotkania (datetime), link do spotkania online (meetingLink), język wykładowy (language) oraz identyfikator nagrania (recordID).

Klucz główny: meetingID

Klucze obce: interpreterID (z tabelą **Interpreters**), recordID (z tabelą **Recordings**), tutorID (z tabelą **Employees**)

Attendees

Tabela przechowuje dane o obecności na spotkaniach. Zawiera identyfikator obecności (attendanceID), identyfikator spotkania (meetingID), identyfikator klienta (meetingID), informację o obecności (present) oraz indentyfikator zajęć na których odrabiano nieobecność (substituteMeetingID).

Klucz główny: attendanceID

Klucz obcy: substituteAttendanceID (z tabelą **Attendees**), meetingID (z tabelą **Meetings**), clientID (z tabelą **Clients**)

Recordings

Tabela przechowuje linki do nagrań. Zawiera identyfikator nagrania (recordID) oraz link do nagrania (recordLInk).

Klucz główny: recordID

Access

Tabela przechowuje dane dostępów do nagrań. Zawiera identyfikator klienta (clientID), identyfikator spotkania (meetingID), datę końca dostępu (lastDay) oraz status obejrzenia (status).

Klucze główne: clientID, recordID

Klucze obce: clientID (z tabelą **Clients**), recordID (z tabelą **Recordings**)

Practices

Tabela przechowuje informacje o praktykach przypisanych do danego semestru. Zawiera identyfikator praktyk (practiceID), identyfikator semestru (termID)

Klucze główne: practiceID

Klucze obce: termID (z tabelą **Terms**)

IndividualInternship

Tabela przechowuje informacje o poszczególnych praktykach odbywanych przez poszczególnych studentów. Zawiera identyfikator pojedyńczych praktyk (InternshipID), identyfikator studenta (clientID), identyfikator praktyk (practiceID), datę rozpoczęcia oraz zakończenia (startDate oraz endDate) oraz informację o zaliczeniu praktyk (passed)

Klucz główny: internshipID

Klucze główne: clientID (z tabelą **Clients**), practiceID (z tabelą **Practices**)

Webinars

Tabela zawiera informację o webinarach. Zawiera identyfikator webinaru (serviceID), maksymalną możliwą ilość uczestników (maxParticipants) oraz identyfikator spotkania, do którego jest przypisany (meetingID)

Klucz główny: serviceID

Klucze obce: serviceID (z tabelą **Services**), meetingID (z tabelą **Meetings**)

Courses

Tabela zawiera informację o kursach. Zawiera identyfikator kursu (serviceID), maksymalną możliwą ilość uczestników (maxParticipants), łączną ilość spotkań na żywo w ramach kursu (totalMeetings), łączną ilość nagrań w ramach kursu (totalRecords)

Klucz główny: serviceID

Klucz obcy: serviceID (z tabelą **Services**)

CoursesMeetings

Tabela zawiera informacje o poszczególnych spotkaniach na żywo w ramach kursu. Zawiera identyfikator spotkania z kursu (courseMeetingID), identyfikator spotkania (meetingID) oraz identyfikator kursu (serviceID)

Klucz główny: courseMeetingID

Klucze obce: meetingID (z tabelą **Meetings**), serviceID (z tabelą **Courses**)

CoursesRecords

Tabela zawiera informacje o poszczególnych nagraniach w ramach kursu. Zawiera identyfikator spotkania nagrania z kursu (courseRecordID), identyfikator nagrania (recordID) oraz identyfikator kursu (serviceID)

Klucz główny: courseRecordID

Klucze obce: recordID (z tabelą **Recordings**), serviceID (z tabelą **Courses**)

CoursesGrades

Tabela zawiera informacje o zaliczeniu kursu. Zawiera identyfikator oceny (gradeID), identyfikator kursu (serviceID), identyfikator klienta (clientID), informację o zaliczeniu (passed).

Klucz główny: gradeID

Klucze obce: serviceID (z tabelą **Courses**), clientID (z tabelą **Clients**)

SubjectsMeetings

Tabela zawiera informacje o poszczególnych spotkaniach w ramach przedmiotu. Zawiera identyfikator spotkania w ramach przedmiotu (subjectMeetingID), identyfikator spotkania (meetingID) oraz identyfikator przedmiotu (subjectID)

Klucz główny: subjectMeetingID

Klucze obce: meetingID (z tabelą **Meetings**), subjectID (z tabelą **Subjects**)

Implementacja bazy danych

Do utworzenia bazy danych wraz z wszystkimi połączeniami użyta została następująca implementacja:

```
-- Table: Access
CREATE TABLE Access (
    clientID int  NOT NULL,
    recordID int  NOT NULL,
    lastDay date  NOT NULL,
    status int  NULL,
    CONSTRAINT Access_pk PRIMARY KEY  (clientID,recordID)
);
-- Reference: Access_Clients (table: Access)
ALTER TABLE Access ADD CONSTRAINT Access_Clients
    FOREIGN KEY (clientID)
    REFERENCES Clients (clientID);

-- Reference: access_recordings (table: Access)
ALTER TABLE Access ADD CONSTRAINT access_recordings
    FOREIGN KEY (recordID)
    REFERENCES Recordings (recordID);

-- Table: Attendees
CREATE TABLE Attendees (
    attendanceID int  NOT NULL,
    clientID int  NOT NULL,
    meetingID int  NOT NULL,
    present int  NULL,
    substituteAttendanceID int  NULL,
    CONSTRAINT Attendees_pk PRIMARY KEY  (attendanceID)
);

-- Reference: attendees_attendees (table: Attendees)
ALTER TABLE Attendees ADD CONSTRAINT attendees_attendees
    FOREIGN KEY (substituteAttendanceID)
    REFERENCES Attendees (attendanceID);
```

```
-- Reference: attendees_meeting (table: Attendees)
ALTER TABLE Attendees ADD CONSTRAINT attendees_meeting
    FOREIGN KEY (meetingID)
    REFERENCES Meetings (meetingID);

-- Reference: Attendees_Clients (table: Attendees)
ALTER TABLE Attendees ADD CONSTRAINT Attendees_Clients
    FOREIGN KEY (clientID)
    REFERENCES Clients (clientID);

-- Table: Clients
CREATE TABLE Clients (
    clientID int  NOT NULL IDENTITY(1, 1),
    firstName varchar(30)  NOT NULL,
    lastName varchar(30)  NOT NULL,
    adress varchar(100)  NOT NULL,
    city varchar(50)  NOT NULL,
    region varchar(20)  NOT NULL,
    CONSTRAINT Clients_pk PRIMARY KEY  (clientID)
);

-- Table: CourseGrade
CREATE TABLE CourseGrade (
    gradeID int  NOT NULL IDENTITY(1, 1),
    serviceID char(4)  NOT NULL,
    clientID int  NOT NULL,
    passed varchar(8)  NULL,
    CONSTRAINT CourseGrade_pk PRIMARY KEY  (gradeID)
);
-- Reference: CourseGrade_Courses (table: CourseGrade)
ALTER TABLE CourseGrade ADD CONSTRAINT CourseGrade_Courses
    FOREIGN KEY (serviceID)
    REFERENCES Courses (serviceID);

-- Reference: CourseGrade_Clients (table: CourseGrade)
ALTER TABLE CourseGrade ADD CONSTRAINT CourseGrade_Clients
    FOREIGN KEY (clientID)
    REFERENCES Clients (clientID);

-- Table: CourseRecords
CREATE TABLE CourseRecords (
    courseRecordID char(4)  NOT NULL,
    recordID int  NOT NULL,
    serviceID char(4)  NOT NULL,
    CONSTRAINT CourseRecords_pk PRIMARY KEY  (courseRecordID)
);

-- Reference: Recordings_CourseRecords (table: CourseRecords)
ALTER TABLE CourseRecords ADD CONSTRAINT Recordings_CourseRecords
    FOREIGN KEY (recordID)
    REFERENCES Recordings (recordID);

-- Reference: Courses_CourseRecords (table: CourseRecords)
ALTER TABLE CourseRecords ADD CONSTRAINT Courses_CourseRecords
    FOREIGN KEY (serviceID)
    REFERENCES Courses (serviceID);

-- Table: Courses
CREATE TABLE Courses (
    serviceID char(4)  NOT NULL,
    maxParticipants int  NULL,
    totalMeetings int  NULL,
    totalRecords int  NULL,
    CONSTRAINT Courses_pk PRIMARY KEY  (serviceID)
);

-- Reference: courseDetails_services (table: Courses)
ALTER TABLE Courses ADD CONSTRAINT courseDetails_services
    FOREIGN KEY (serviceID)
    REFERENCES Services (serviceID);

-- Table: CoursesMeetings
CREATE TABLE CoursesMeetings (
    courseMeetingID char(4)  NOT NULL,
    meetingID int  NOT NULL,
    serviceID char(4)  NOT NULL,
    CONSTRAINT CoursesMeetings_pk PRIMARY KEY  (courseMeetingID)
);

-- Reference: Workshops_ServiceDetails (table: CoursesMeetings)
ALTER TABLE CoursesMeetings ADD CONSTRAINT Workshops_ServiceDetails
    FOREIGN KEY (serviceID)
    REFERENCES Courses (serviceID);

-- Reference: Meetings_Workshops (table: CoursesMeetings)
ALTER TABLE CoursesMeetings ADD CONSTRAINT Meetings_Workshops
    FOREIGN KEY (meetingID)
    REFERENCES Meetings (meetingID);

-- Table: Employees
```

```
CREATE TABLE Employees (
    employeeID int  NOT NULL IDENTITY(1, 1),
    firstName varchar(30)  NOT NULL,
    lastName varchar(30)  NOT NULL,
    position varchar(100)  NOT NULL,
    CONSTRAINT Employees_pk PRIMARY KEY  (employeeID)
);

-- Table: IndividualInternship
CREATE TABLE IndividualInternship (
    InternshipID int  NOT NULL IDENTITY(1, 1),
    ClientID int  NOT NULL,
    PracticeID int  NOT NULL,
    StartDate datetime  NOT NULL,
    EndDate datetime  NOT NULL,
    Passed int  NOT NULL,
    CONSTRAINT IndividualInternship_pk PRIMARY KEY  (InternshipID)
);

-- Reference: Clients_IndividualInternship (table: IndividualInternship)
ALTER TABLE IndividualInternship ADD CONSTRAINT Clients_IndividualInternship
    FOREIGN KEY (ClientID)
    REFERENCES Clients (clientID);

-- Reference: IndividualInternship_Practices (table: IndividualInternship)
ALTER TABLE IndividualInternship ADD CONSTRAINT IndividualInternship_Practices
    FOREIGN KEY (PracticeID)
    REFERENCES Practices (PracticeID);

-- Table: Interpreters
CREATE TABLE Interpreters (
    interpreterID int  NOT NULL IDENTITY(1, 1),
    language varchar(30)  NOT NULL,
    employeeID int  NOT NULL,
    CONSTRAINT Interpreters_pk PRIMARY KEY  (interpreterID)
);

-- Reference: interpreters_employees (table: Interpreters)
ALTER TABLE Interpreters ADD CONSTRAINT interpreters_employees
    FOREIGN KEY (employeeID)
    REFERENCES Employees (employeeID);

-- Table: Meetings
CREATE TABLE Meetings (
    meetingID int  NOT NULL,
    tutorID int  NOT NULL,
    interpreterID int  NULL,
    datetime datetime  NOT NULL,
    meetingLink varchar(100)  NULL,
    Language varchar(20)  NOT NULL,
    recordID int  NULL,
    CONSTRAINT Meetings_pk PRIMARY KEY  (meetingID)
);

-- Reference: meeting_interpreters (table: Meetings)
ALTER TABLE Meetings ADD CONSTRAINT meeting_interpreters
    FOREIGN KEY (interpreterID)
    REFERENCES Interpreters (interpreterID);

-- Reference: recordings_meeting (table: Meetings)
ALTER TABLE Meetings ADD CONSTRAINT recordings_meeting
    FOREIGN KEY (recordID)
    REFERENCES Recordings (recordID);


-- Reference: Meetings_Employees (table: Meetings)
ALTER TABLE Meetings ADD CONSTRAINT Meetings_Employees
    FOREIGN KEY (tutorID)
    REFERENCES Employees (employeeID);

-- Table: OrderDetails
CREATE TABLE OrderDetails (
    orderID uniqueidentifier  NOT NULL DEFAULT NEWID(),
    serviceID char(4)  NOT NULL,
    price money  NOT NULL,
    CONSTRAINT OrderDetails_pk PRIMARY KEY  (orderID,serviceID)
);

-- Reference: orderDetails_orders (table: OrderDetails)
ALTER TABLE OrderDetails ADD CONSTRAINT orderDetails_orders
    FOREIGN KEY (orderID)
    REFERENCES Orders (orderID);

-- Reference: orderDetails_services (table: OrderDetails)
ALTER TABLE OrderDetails ADD CONSTRAINT orderDetails_services
    FOREIGN KEY (serviceID)
    REFERENCES Services (serviceID);

-- Table: Orders
CREATE TABLE Orders (
```

```
    orderID uniqueidentifier  NOT NULL,
    clientID int  NOT NULL,
    orderDate datetime  NULL,
    paymentLink varchar(100)  NOT NULL,
    receiptDate datetime  NULL,
    CONSTRAINT Orders_pk PRIMARY KEY  (orderID)
);

-- Reference: orders_clients (table: Orders)
ALTER TABLE Orders ADD CONSTRAINT orders_clients
    FOREIGN KEY (clientID)
    REFERENCES Clients (clientID);

-- Table: Practices
CREATE TABLE Practices (
    PracticeID int  NOT NULL IDENTITY(1, 1),
    termID char(4)  NOT NULL,
    CONSTRAINT Practices_pk PRIMARY KEY  (PracticeID)
);

-- Reference: Practices_Terms (table: Practices)
ALTER TABLE Practices ADD CONSTRAINT Practices_Terms
    FOREIGN KEY (termID)
    REFERENCES Terms (termID);

-- Table: Recordings
CREATE TABLE Recordings (
    recordID int  NOT NULL IDENTITY(1, 1),
    recordLink varchar(100)  NOT NULL,
    CONSTRAINT Recordings_pk PRIMARY KEY  (recordID)
);

-- Table: Services
CREATE TABLE Services (
    serviceID char(4)  NOT NULL,
    serviceName varchar(50)  NOT NULL,
    type varchar(10)  NOT NULL,
    startDate datetime  NOT NULL,
    price money  NOT NULL,
    CONSTRAINT Services_pk PRIMARY KEY  (serviceID)
);

-- Table: Studies
CREATE TABLE Studies (
    studiesID char(4)  NOT NULL,
    name varchar(50)  NOT NULL,
    terms int  NOT NULL,
    CONSTRAINT Studies_pk PRIMARY KEY  (studiesID)
);

-- Reference: studies_services (table: Studies)
ALTER TABLE Studies ADD CONSTRAINT studies_services
    FOREIGN KEY (studiesID)
    REFERENCES Services (serviceID);

-- Table: SubjectGrades
CREATE TABLE SubjectGrades (
    gradeID int  NOT NULL IDENTITY(1, 1),
    subjectID char(4)  NOT NULL,
    clientID int  NOT NULL,
    grade int  NULL,
    CONSTRAINT SubjectGrades_pk PRIMARY KEY  (gradeID)
);

-- Reference: Clients_Grades (table: SubjectGrades)
ALTER TABLE SubjectGrades ADD CONSTRAINT Clients_Grades
    FOREIGN KEY (clientID)
    REFERENCES Clients (clientID);

    -- Reference: grades_Course (table: SubjectGrades)
ALTER TABLE SubjectGrades ADD CONSTRAINT grades_Course
    FOREIGN KEY (subjectID)
    REFERENCES Subjects (subjectID);

-- Table: Subjects
CREATE TABLE Subjects (
    subjectID char(4)  NOT NULL,
    termID char(4)  NOT NULL,
    SubjectName varchar(100)  NOT NULL,
    type varchar(3)  NOT NULL,
    CONSTRAINT Subjects_pk PRIMARY KEY  (subjectID)
);

-- Reference: Terms_Subjects (table: Subjects)
ALTER TABLE Subjects ADD CONSTRAINT Terms_Subjects
    FOREIGN KEY (termID)
    REFERENCES Terms (termID);

-- Table: SubjectsMeetings
CREATE TABLE SubjectsMeetings (
```

```
    subjectMeetingID char(4)  NOT NULL,
    subjectID char(4)  NOT NULL,
    meetingID int  NOT NULL,
    CONSTRAINT SubjectsMeetings_pk PRIMARY KEY  (subjectMeetingID)
);

-- Reference: Meetings_Lectures (table: SubjectsMeetings)
ALTER TABLE SubjectsMeetings ADD CONSTRAINT Meetings_Lectures
    FOREIGN KEY (meetingID)
    REFERENCES Meetings (meetingID);

-- Reference: Lectures_Services (table: SubjectsMeetings)
ALTER TABLE SubjectsMeetings ADD CONSTRAINT Lectures_Services
    FOREIGN KEY (subjectMeetingID)
    REFERENCES Services (serviceID);

-- Reference: Subjects_Lectures (table: SubjectsMeetings)
ALTER TABLE SubjectsMeetings ADD CONSTRAINT Subjects_Lectures
    FOREIGN KEY (subjectID)
    REFERENCES Subjects (subjectID);

-- Table: Terms
CREATE TABLE Terms (
    termID char(4)  NOT NULL,
    studiesID char(4)  NOT NULL,
    startDate date  NOT NULL,
    endDate date  NOT NULL,
    CONSTRAINT Terms_pk PRIMARY KEY  (termID)
);

-- Reference: Services_Terms (table: Terms)
ALTER TABLE Terms ADD CONSTRAINT Services_Terms
    FOREIGN KEY (termID)
    REFERENCES Services (serviceID);

-- Reference: conventions_studies (table: Terms)
ALTER TABLE Terms ADD CONSTRAINT conventions_studies
    FOREIGN KEY (studiesID)
    REFERENCES Studies (studiesID);

-- Table: Webinars
CREATE TABLE Webinars (
    serviceID char(4)  NOT NULL,
    maxParticipants int  NOT NULL,
    meetingID int  NOT NULL,
    CONSTRAINT Webinars_pk PRIMARY KEY  (serviceID)
);

-- Reference: Meetings_Webinars (table: Webinars)
ALTER TABLE Webinars ADD CONSTRAINT Meetings_Webinars
    FOREIGN KEY (meetingID)
    REFERENCES Meetings (meetingID);

-- Reference: Webinars_Services (table: Webinars)
ALTER TABLE Webinars ADD CONSTRAINT Webinars_Services
    FOREIGN KEY (serviceID)
    REFERENCES Services (serviceID);

-- End of file.
```

Widoki

Przykładowa implementacja widoku, który dla każdej dostępnej usługi zwraca informację o łącznych przychodach z jej tytułu:

```
-- widok zwracający przychody dla kazdego oferowanego serwisu
create view RaportFinansowy as
    select serviceName, t.serviceID, przychody
    from Services as s
    join (select serviceID, sum(price) as przychody from OrderDetails
    group by serviceID) as t on t.serviceID=s.serviceID
```

Procedury

Przykładowa implementacja procedury dodające nowej klienta:

```
Procedura dodająca nowego klienta. Jako argumenty przyjmuje imię, nazwisko, adres, miasto oraz województwo

CREATE PROCEDURE addNewClient
    @p_firstName NVARCHAR(30),
    @p_lastName NVARCHAR(30),
    @p_address NVARCHAR(100),
    @p_city NVARCHAR(50),
    @p_region NVARCHAR(20)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO Clients (firstName, lastName, address, city, region)
    VALUES (@p_firstName, @p_lastName, @p_address, @p_city, @p_region);
END;
```

Przykładowa implementacja prodecury dodającej nowego pracownika:

```
Procedura dodająca nowego pracownika. Jako argumenty przyjmuje imię, nazwisko oraz stanowisko pracy.

CREATE PROCEDURE addNewEmployee
    @p_firstName NVARCHAR(30),
    @p_lastName NVARCHAR(30),
    @p_position NVARCHAR(100)
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO Employees(firstName,lastName,position)
    VALUES (@p_firstName, @p_lastName, @p_position);
END;
```