Kolokwium II grupa A

W odpowiedziach do zadań nie trzeba używać nazwy schematu - przyjmuje się, że *search_ path* jest ustawiony na *kolokwium1*.

1. *(5p.)* AGH projektuje nowy system do obługi akademików. Narysuj diagram związków encji według poniższego opisu. Pogrubione słowa oznaczają encje, które powinny znaleźć się na diagramie.

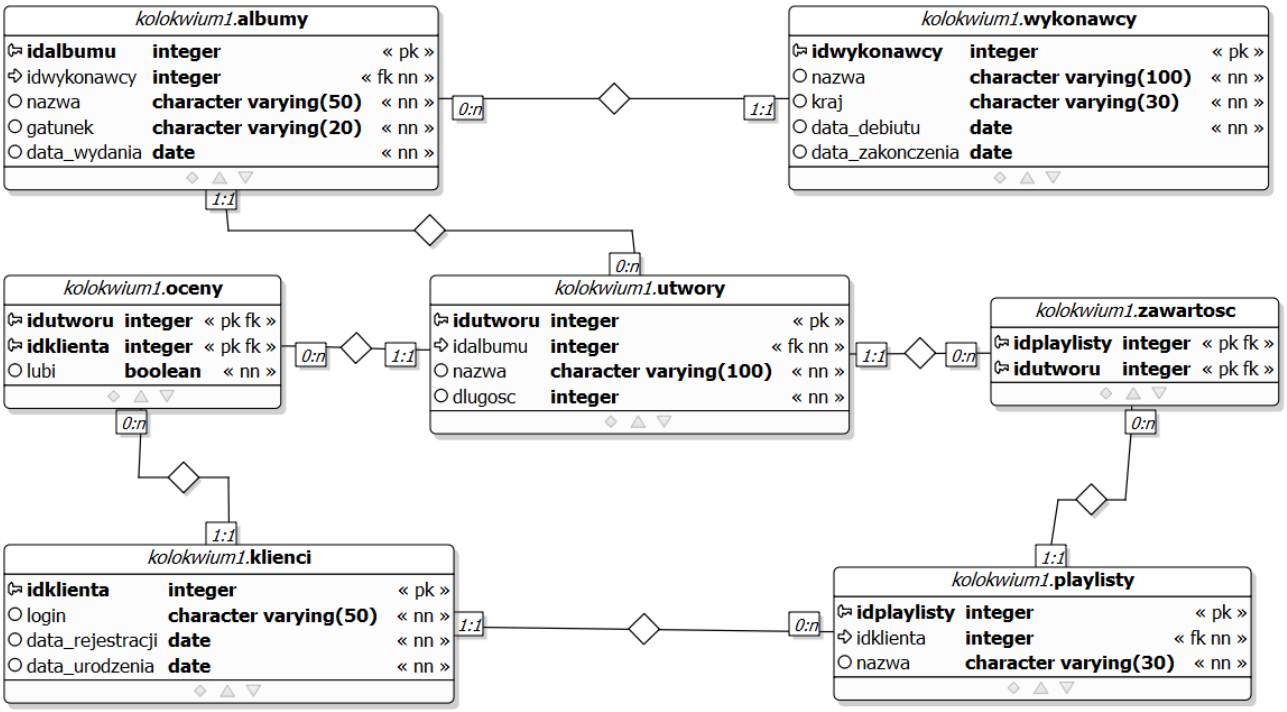
   - Każdy **akademik** mieści co najmniej 30 **pokojów**.
   - Każdy **student** może mieć (lub nie mieć) **rezerwację** na pokój.
   - Na każdy pokój mogą być dokonane maksymalnie 3 rezerwacje.
   - W każdym akademiku pracuje co najmniej 20 **pracowników**. Hierarchia pracowników może mieć strukturę drzewa (np. jeden pracownik może podlegać innemu pracownikowi).

2. *(5p.)* Dana jest relacja R o schemacie H = {A,B,C,X,Y,Z} oraz zbiór zależności funkcyjnych
   $F = \{\{A, B\} \to \{C, Y\}, \{A, X\} \to Y, Y \to X, Y \to \{B, C\}, A \to Z\}$
   Sprawdź czy dekompozycja:
   H_1 = {A,Y}, H_2 = {B,Y,Z}, H_3 = {A,C,X,Z} jest dekompozycją bezstratną.

3. *(5p.)* Napisz zapytania SQL tworzące w bazie tabele *albumy* i *wykonawcy* (patrz: załączony schemat). Zadbaj o utworzenie właściwych kluczy głównych i kluczy obcych (mogą być częścią zapytań CREATE lub stanowić odrębne zapytania typu ALTER). Dodatkowo nałóż ograniczenia na kolumny, aby login klienta miał co najmniej 5 znaków oraz gatunek albumu mógł przyjmować tylko jedną z wartości: 'Rock', 'Pop', 'Metal'.

4. *(5p.)* Korzystając z operatorów *all* oraz *any* (obu) napisz zapytanie SQL pobierające z bazy ID wszystkich playlist, dla których wszystkie znajdujące się na nich utwory są dłuższe niż 300 sekund oraz co najmniej jeden z ich utworów należy do gatunku 'Pop'.

5. *(5p.)* Napisz funkcję o nazwie *uzupelnij_ playliste*, która przyjmuje trzy argumenty: *idplaylisty_ od* (int), *idplaylisty_ do* (int), *polub* (boolean). Funkcja skopiuje z playlisty *idplaylisty_ od* do playlisty *idplaylisty_ do* utwory, które nie występują na tej drugiej. Jeżeli parametr *polub* jest równy TRUE to dla skopiowanych utworów funkcja doda oceny pozytywne (lubi = TRUE), wystawione przez właściciela drugiej playlisty, ale tylko jeśli jeszcze nie mają od niego ocen. Funkcja zwraca tabelę zawierającą wszystkie utwory (wiersze z tabeli *utwory*) znajdujące się na playliście *idplaylisty_ do* po operacji kopiowania.



Kolokwium II grupa B

W odpowiedziach do zadań nie trzeba używać nazwy schematu - przyjmuje się, że *search_ path* jest ustawiony na *kolokwium1*.

1. *(5p.)* Spółka Fit4You projektuje system dla cateringu dietetycznego. Narysuj diagram związków encji według poniższego opisu. Pogrubione słowa oznaczają encje, które powinny znaleźć się na diagramie.

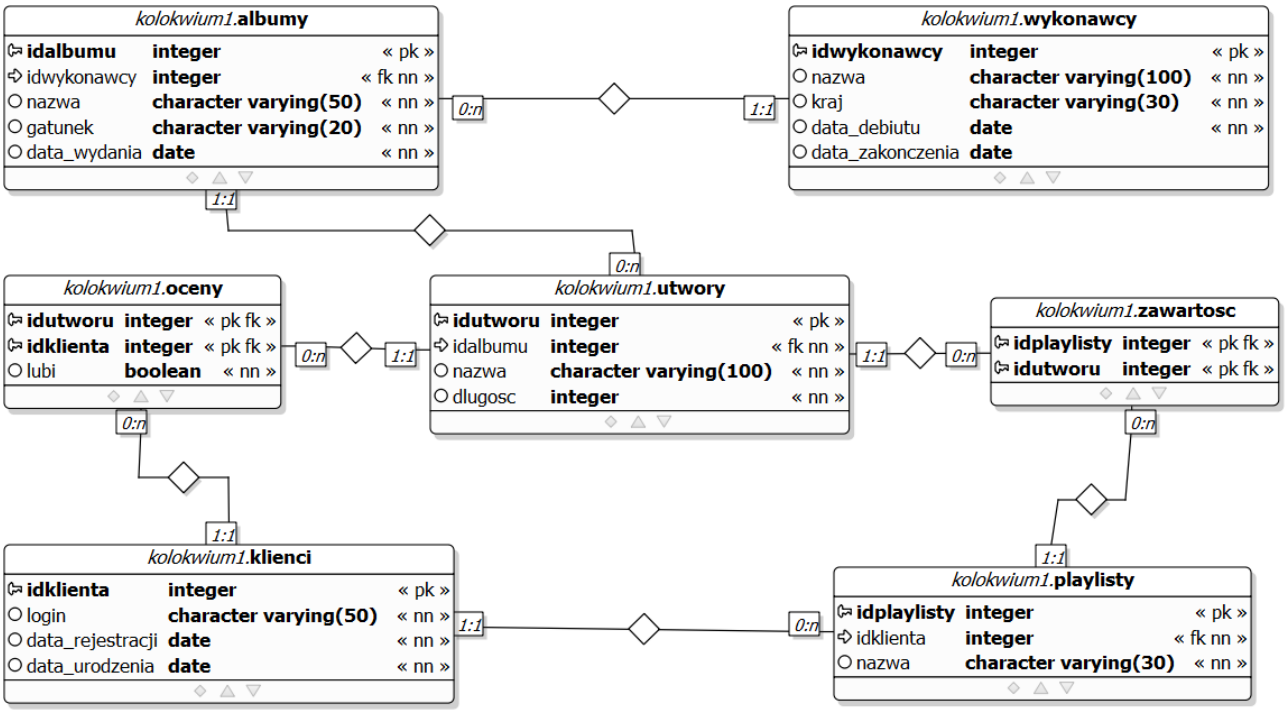
   - Każda **dieta** jest podzielona na poszczególne **dni**. Każdy dzień reprezentuje menu jednej diety w danym terminie.
   - W skład każdego dnia wchodzi 5 **dań**, przy czym każde z nich może powtarzać się w różne dni w różnych dietach.
   - Każde danie może mieć (lub nie mieć) informację o **składnikach** odżywczych.
   - Każde danie należy do **kategorii** (np. wege, mięsne, itd...). Hierarchia kategorii może mieć strukturę drzewa (np. jedna kategoria może być podkategorią innej).

2. *(5p.)* Dana jest relacja R o schemacie H = {A,B,C,X,Y,Z} oraz zbiór zależności funkcyjnych
   $F = \{\{A, B\} \to \{C, Y\}, \{A, X\} \to Y, Y \to X, Y \to \{B, C\}, A \to Z\}$
   Sprawdź czy dekompozycja:
   H_1 = {A,B,C,Y}, H_2 = {A,B,X}, H_3 = {A,Z} jest dekompozycją bezstratną.

3. *(5p.)* Napisz zapytania SQL tworzące w bazie tabele *klienci* i *playlisty* (patrz: załączony schemat). Zadbaj o utworzenie właściwych kluczy głównych i kluczy obcych (mogą być częścią zapytań CREATE lub stanowić odrębne zapytania typu ALTER). Dodatkowo nałóż ograniczenia na kolumny, aby nazwa playlisty miała co najmniej 5 znaków oraz kraj wykonawcy mógł przyjmować tylko jedną z wartości: 'Polska', 'Niemcy', 'Hiszpania'.

4. *(5p.)* Korzystając z operatora *any* napisz zapytanie SQL pobierające z bazy ID wszystkich playlist, dla których co najmniej jeden ze znajdujących się na nich utworów jest dłuższy niż 300 sekund oraz wszystkie ich utwory należą do gatunku 'Pop'.

5. *(5p.)* Napisz funkcję o nazwie *uzupelnij_ playliste*, która przyjmuje trzy argumenty: *idplaylisty_ od* (int), *idplaylisty_ do* (int), *polub* (boolean). Funkcja skopiuje z playlisty *idplaylisty_ od* do playlisty *idplaylisty_ do* utwory, które nie występują na tej drugiej. Jeżeli parametr *polub* jest równy TRUE to dla skopiowanych utworów funkcja doda oceny pozytywne (lubi = TRUE), wystawione przez właściciela drugiej playlisty, ale tylko jeśli jeszcze nie mają od niego ocen. Funkcja zwraca tabelę zawierającą wszystkie utwory (wiersze z tabeli *utwory*) znajdujące się na playliście *idplaylisty_ do* po operacji kopiowania.



Test II group A

You do not need to use the schema name in the answers to the tasks - it is assumed that *search_path* is set to *kolokwium1*.

1. *(5p.)* AGH is designing a new dormitory system. Draw an entity relationship diagram as described below. Bold words indicate entities that should be included in the diagram.

   - Each **dormitory** contains at least 30 **rooms**.
   - Each **student** can have (or not have) a **reservation** for a room.
   - A maximum of 3 reservations can be made for each room.
   - There are at least 20 **employees** in each dormitory. The hierarchy of employees may have a tree structure (e.g., one employee may report to another employee).

2. *(5p.)* Schema H = {A,B,C,X,Y,Z} and a set of functional dependencies are given
   $F = \{\{A, B\} \to \{C, Y\}, \{A, X\} \to Y, Y \to X, Y \to \{B, C\}, A \to Z\}$
   Check if decomposition:
   H_1 = {A,Y}, H_2 = {B,Y,Z}, H_3 = {A,C,X,Z} is losseless.

3. *(5p.)* Write SQL queries that create the *albums* and *artists* tables in the database (see attached schema). Take care to create the correct primary keys and foreign keys (these can be part of CREATE queries or be separate ALTER queries). Additionally, add constraints on the columns so that the client login has at least 5 characters and the album genre can only take one of the values: 'Rock', 'Pop', 'Metal'.

4. *(5p.)* Using the *all* and *any* operators (both), write a SQL query that retrieves the IDs of all playlists from the database for which all the songs in them are longer than 300 seconds and at least one of their songs belongs to the 'Pop' genre.

5. *(5p.)* Write a function named *fill_playlist* that takes three arguments: *playlistid_from* (int), *playlistid_to* (int), *like* (boolean). The function will copy the songs from the playlist *playlistid_from* to the playlist *playlistid_to*, that do not appear on the latter. If the parameter *like* is equal to TRUE then for the copied songs the function will add positive votes (liked = TRUE), given by the owner of the second playlist, but only if they do not yet have votes from him. Function returns a table containing all the songs (rows from table *songs*) in the playlist *playlistid_to* after the copy operation.