Bazy danych 2013/14. Egzamin poprawkowy

Zadanie 1 (5 pkt). Relacja binarna G przechowuje graf skierowany (bez wierzchołków izolowanych). Napisz zapytanie wyznaczające wierzchołki połączone z każdym wierzchołkiem ścieżką długości 3 (tzn. zawierającą 3 krawędzie)

(a) w logice pierwszego rzędu,

(b) w algebrze relacji.

Zadanie 2 (5 pkt). Tabela *Wyniki* zawiera wyniki poszczególnych zadań na egzaminie:

```
CREATE TABLE Wyniki (
    pesel varchar(11) not null,
    zadanie integer not null check (zadanie>=1 and zadanie<=6),
    wynik integer not null check (wynik>=0 and wynik<=10),
    primary key (pesel, zadanie)
);
```

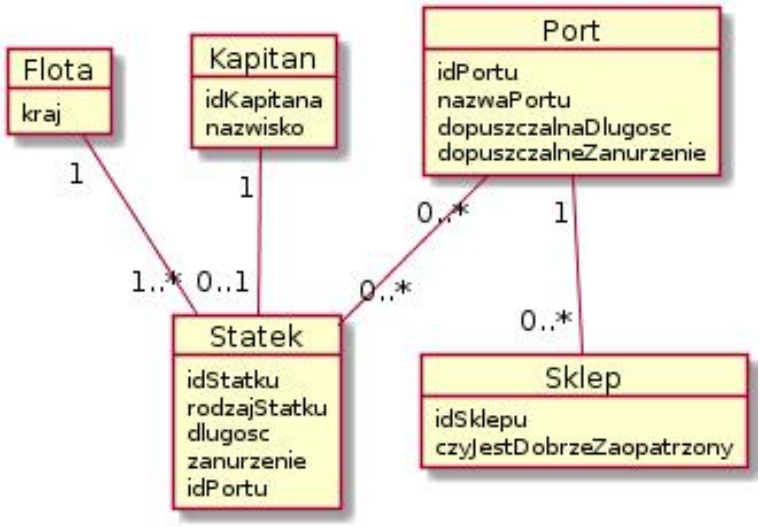
Całkowity wynik obliczany jest jako iloczyn liczby oddanych zadań i średniej arytmetycznej wyników zadań z pominięciem jednego najlepszego wyniku oraz jednego najgorszego wyniku. Jeśli student oddał mniej niż 3 zadania, to jego wynik wynosi 0 pkt. Na przykład dla wyników z zadań $3, 4, 5, 7, 7$, całkowitym wynikiem będzie $5 \cdot 5\frac{1}{3} = 26\frac{2}{3}$. Podaj zapytanie SQL obliczające listę rankingową dla egzaminu w postaci (PESEL, całkowity wynik).

Zadanie 3 (5 pkt). Dana jest tabela *Pracownicy(numer, imie, szef)*, gdzie *szef* to numer szefa. Napisz zapytanie w dowolnym z poznanych formalizmow (datalog, algebra relacji z iteracją lub SQL z rekurencją) wybierające pracowników, którzy dla dokładnie 10 osób są szefami lub podwładnymi.

Zadanie 4 (5 pkt). W kinie jest kilka sal kinowych (mają określoną liczbe numerowanych miejsc, nie uwzględniamy podziału na rzędy), w których wyświetla się filmy (tytuł, rok produkcji, reżyser, typ, czas wyświetlania, jakiś opis). Filmy wyświetla się podczas seansów. Klienci mogą sprawdzać, jakie filmy są wyświetlane w kinie przez najbliższy tydzień, a także obejrzeć, jakie seanse odbywaja się danego dnia w ciągu tygodnia. Mogą też zamawiać bilety na wybrane seanse. Takie zamówienia są obsługiwane przez kasjerów, którzy muszą je zaakceptować (zapewne po otrzymaniu płatności, ale tym się nie zajmujemy) i przydzielić klientom (być może automatycznie) odpowiednie numery miejsc.

Zaproponuj adekwatny model danych w postaci diagramu związków encji.

Zadanie 5 (5 pkt). Poniższy diagram związków encji przedstawia model danych dla flot statków.



Podsumujmy wiadomości widoczne na diagramie: flota składa się z co najmniej jednego statku, każdy statek ma kapitana, każdy statek może zacumować w pewnej liczbie portów; wiadomo też na podstawie pola idPortu, czy statek jest zacumowany i w którym porcie. Port może przyjąć dowolną liczbę statków, które spełniają wymagania odnośnie długości i zanurzenia. W każdym porcie mieści się pewna liczba sklepów, z których każdy może dobrze zaopatrzony lub źle zaopatrzony.

Zaproponuj realizację tego modelu za pomocą tabel, troszcząc sie o to, by tabele były w BCNF, jeśli to możliwe bez utraty zależności funkcyjnych (uzasadnij, że są w BCNF, lub dlaczego nie mogą być w BCNF bez utraty zależności). Dla załogi jest ważne, żeby móc sformułować zapytanie, czy do danego portu można zawinąć i czy jest tam dobrze zaopatrzony sklep.

Zadanie 6 (5 pkt). Załóżmy, że tabela *Wyniki* z zadania 2 zawiera 660 wierszy, a w jednym bloku dyskowym mieści się 30 wierszy tej tabeli. Załóżmy dalej, że dysponujemy pamięcią operacyjną na 15 bloków dyskowych. Zamierzamy zrealizować następujące zapytanie:

```
select pesel, sum(wynik)
from Wyniki W1
where wynik >= (select avg(wynik) from Wyniki W2 where W1.zadanie=W2.zadanie)
  and wynik >= (select count(*) from Wyniki W3 where W1.pesel=W3.pesel)
group by pesel;
```

Podaj jak najbardziej efektywny plan wykonania tego zapytania i oszacuj koszt tego planu (liczbę operacji odczytu i zapisu bloku dyskowego). Zakładamy, że nie korzystamy z żadnych indeksów i że na początku wszystkie bufory są puste, a dane są tylko na dysku.

Rozwiązania

Zadanie 1

(a) $\forall y\ \exists z\ \exists z'\ G(x,z) \wedge G(z,z') \wedge G(z',y)$.

(b) Na przykład:

$$(\pi_1 G \cup \pi_2 G) - \pi_1\big((\pi_1 G \cup \pi_2 G) \times (\pi_1 G \cup \pi_2 G) - \pi_{1,6}\sigma_{2=3,4=5}(G \times G \times G)\big)\,.$$

Podwyrażenie $(\pi_1 G \cup \pi_2 G)$ wybiera wszystkie wierzchołki grafu.

Zadanie 2

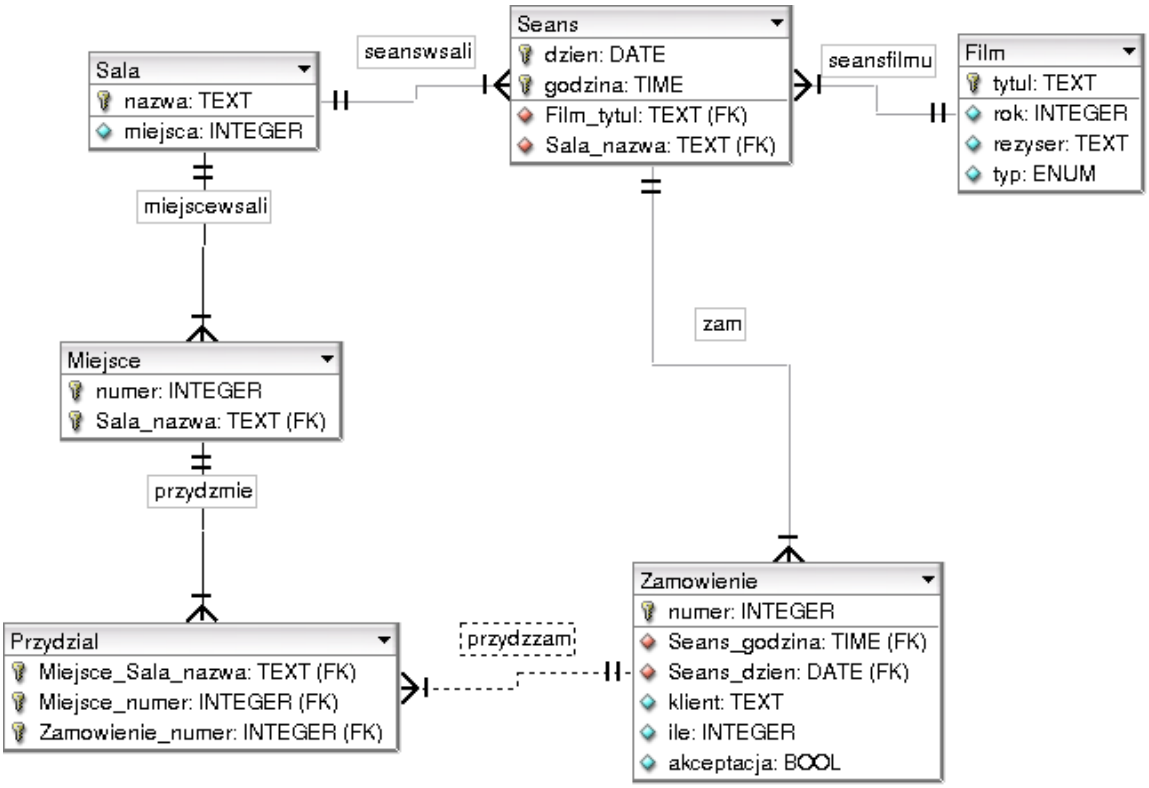
```
 SELECT pesel,
        COUNT(*)*(SUM(wynik)-MIN(wynik)-MAX(wynik))/(COUNT(*)-2) as calkowity_wynik
 FROM Wyniki
 GROUP BY pesel
 HAVING COUNT(*)>2
UNION
 SELECT pesel,0
 FROM Wyniki
 GROUP BY pesel
 HAVING COUNT(*)<=2;
```

Możliwe jest również rozwiązanie oparte na konstrukcji CASE.

Zadanie 3 Tworzymy tabelę ze wszystkimi parami (szef, podwładny), sumujemy z jej permutacją i grupując po pierwszej kolumnie znajdujemy tych pracowników, którzy mają 10 wierszy.

```
WITH RECURSIVE sub(numer, szef) AS (
   SELECT numer, NULL FROM pracownicy
   UNION ALL
   SELECT sub.szef, pracownicy.numer FROM sub, pracownicy
   WHERE pracownicy.szef = sub.numer
)
SELECT *
FROM (SELECT * FROM sub) UNION (SELECT szef, numer FROM sub)
GROUP BY numer HAVING COUNT(szef) = 10;
```

Zadanie 4 W poniższym rozwiązaniu encja *Miejsce* nie jest niezbędna, ani jej związek z *Salą*. Można numer miejsca wrzucić do *Przydział*, wtedy będą większe złaczenia, ale zniknie cykl.



Zadanie 6 Decydujemy się na dodanie tabeli *PortStatek*, w której gromadzimy dane o portach i statkach, które mogą w nich zacumować. Tabela ta jest implicite zakodowana w tabelach *Port* i *Statek* i poprawne jest rozwiązanie, które pomija tabelę *PortStatek*.

```
CREATE TABLE Flota (
  kraj VARCHAR(30) PRIMARY KEY
);

CREATE TABLE Kapitan (
  idKapitana INTEGER PRIMARY KEY,
  idStatku VARCHAR(30) REFERENCES Statek(idStatku),
  nazwisko VARCHAR(30) NOT NULL
);
```

```
CREATE TABLE Statek (
  idStatku INTEGER PRIMARY KEY,
  idKapitana INTEGER REFERENCES Kapitan(idKapitana),
  kraj VARCHAR(30) REFERENCES Flota(kraj),
  rodzajStatku VARCHAR(30) NOT NULL,
  dlugosc INTEGER NOT NULL,
  zanurzenie INTEGER NOT NULL,
  idPortu INTEGER REFERENCES Port(idPortu),

);

CREATE TABLE Port (
  idPortu INTEGER PRIMARY KEY,
  nazwaPortu VARCHAR(30) NOT NULL,
  dopuszczalnaDlugosc INTEGER NOT NULL,
  dopuszczalneZanurzenie INTEGER NOT NULL,
);

CREATE TABLE PortStatek (
  idPortu INTEGER FOREIGN KEY REFERENCES Port(idPortu),
  idStatku INTEGER FOREIGN KEY REFERENCES Statek(idStatku),
  PRIMARY KEY (idPortu,idStatku)
);

CREATE TABLE Sklep (
  idSklepu INTEGER PRIMARY KEY,
  idPortu INTEGER REFERENCES Port(idPortu),
  czyJestDobrzeZaopatrzony BOOLEAN NOT NULL
);
```

Zadanie 6 Do realizacji grupowania potrzebne jest sortowanie, a ponieważ tabela zawiera 22 bloki, to trzeba do tego użyć merge sort. Zmieści się on w dwóch przebiegach, przy czym tak naprawdę trzeba będzie zwolnić w RAMie osiem bloków (22 minus 15 i minus 1 – potrzebny do scalania).

Z definicji tabeli wynika, że może być co najwyżej 6 zadań. Dlatego można średnie dla zadań policzyć równolegle z pierwszą fazą sortowania – potrzebne będzie do tego miejsce na 12 liczb (6 na sumy wyników zadań i 6 na ilości poszczególnych zadań). Można przyjąć, że to są zmienne, można też przeznaczyć na to jeden blok z buforów – trzeba wtedy zarezerwować jeden blok i odpowiednio poprawić wyliczenia zapisów i odczytów na dysk).

Drugi z warunków w klauzuli where oraz samo prezentowanie wyników można realizować podczas fazy scalania – znów ponieważ jest maksymalnie 6 zadań, więc jest gwarancja, że dla jednego numeru PESEL grupowanie (i obliczenie count(*)) będzie można zrobić w

oparciu o jeden bufor.

Tak więc koszt wynosi:

- I faza merge sort – 22 odczyty, 8 zapisów.
- II faza merge sort – 8 odczytów (reszta z pierwszej fazy może zostać w buforach).