Bazy danych 2013/14. Egzamin

Zadanie 1 (5 pkt). Baza danych przechowuje w relacji binarnej G graf skierowany.

(a) Napisz formułę logiki pierwszego rzędu $\varphi(x, y)$ bez kwantyfikatorów $\forall$, $\exists$, która definiuje zapytanie równoważne wyrażeniu $\pi_{1,2}\sigma_{2=3,1=4}\big((G - \sigma_{1=2}G) \times (G - \sigma_{1=2}G)\big)$.

(b) Napisz wyrażenie algebry relacji E, które definiuje zapytanie równoważne formule $\psi(x, y)$ danej jako $\forall z\big((G(x, z) \wedge G(z, y)) \implies (z = x \vee z = y)\big)$.

Zadanie 2 (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), które zwróci imiona pracowników, którzy mają co najwyżej pięciu niebezpośrednich podwładnych.

Zadanie 3 (5 pkt). W bazie danych przechowujemy informacje w tabelach:

- *Połączenia(telefon, klient, data, czasTrwania, koszt, taryfa)*
- *Taryfy(nrTaryfy, nazwaTaryfy)*
- *Klienci(nrKlienta, nazwaKlienta, typKlienta)*

W tej chwili w bazie znajdują się zapisy o 3 250 358 połączeniach, 1 543 klientach i 13 taryfach, przy czym 89% połączeń wykonali klienci typu C15. Na podstawie tych danych będzie przygotowywany raport oparty na poniższym zapytaniu:

```
select nazwa_klienta, sum(czasTrwania)
from Połączenia p, Taryfy t, Klienci k
where p.klient=k.nrKlienta
  and p.taryfa = t.nrTaryfy
  and t.nazwaTaryfy like 'TANI%'
  and k.typKlienta in ('B1', 'B15', 'I10', 'BB')
  and p.data between '15.06.2009' and '15.07.2009'
group by nazwa_klienta;
```

Należy wskazać najlepszy plan realizacji tego zapytania. Można w tym celu zaproponować wspierające je struktury pomocnicze. Oszacować koszt realizacji tego planu w operacjach wejścia/wyjścia na blokach dyskowych zakładając, że w jednym bloku mieści się 100 połączeń lub 50 klientów lub 200 taryf, a w pamięci operacyjnej mieści się 100 bloków.

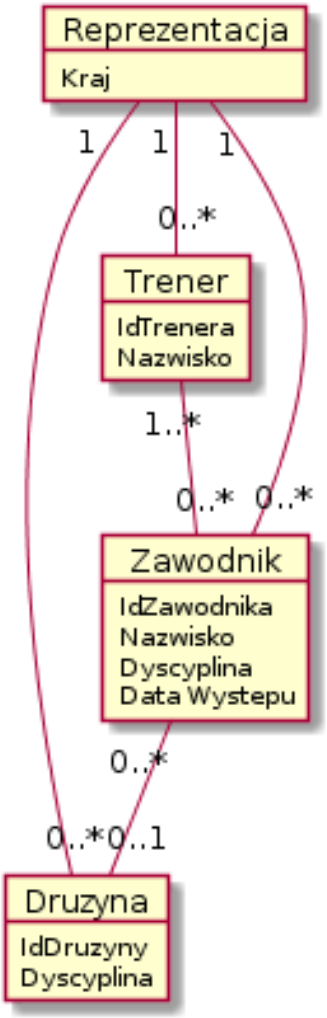
Zadanie 4 (10 pkt). Diagram przedstawia model danych dla Olimpiady Zimowej.

Podsumujmy wiadomości widoczne na diagramie: każdy zawodnik należy do co najwyżej jednej drużyny (drużyny występują tylko w odniesieniu do sportów drużynowych, takich jak hokej; w innych sportach, takich jak łyżwiarstwo figurowe, przyjmujemy że zawodnik przypisany jest jedynie do reprezentacji). Każdy zawodnik może mieć jednego lub więcej trenerów, natomiast trener może trenować 0 lub więcej zawodników. Dla uproszczenia przyjmujemy, że każdy zawodnik występuje w dokładnie jednej dyscyplinie.

Poza tym zachodzi szereg oczywistych zależności:

- zawodnik występuje w dokładniej jednej reprezentacji,
- trener pracuje dla dokładniej jednej reprezentacji,
- drużyna należy do dokładniej jednej reprezentacji,
- zawodnik może występować wielokrotnie podczas olimpiady, w szczególności w encji Zawodnik atrybut idZawodnika wyznacza nazwisko oraz dyscplinę, ale nie wyznacza atrybutu *DataWystepu*,
- Klucze dla poszczególnych encji to odpowiednio *Kraj*, *IdTrenera*, (*IdZawodnika, DataWystepu*) i *IdDruzyny*.

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).



Rozwiązania

Zadanie 1

(a) Chyba nie da sie inaczej: $E(x,y) \wedge E(y,x) \wedge \neg(x = y)$.

(b) Na przykład:

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

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

Zadanie 2 Można na przykład tak:

```
WITH RECURSIVE sub(numer, imie, szef, szefszefow) AS (
   SELECT numer, imie, numer, numer FROM pracownicy
   UNION ALL
   SELECT pracownicy.*, szefszefow FROM pracownicy, sub
   WHERE pracownicy.szef = sub.numer
)
SELECT pracownicy.imie FROM sub RIGHT JOIN pracownicy
   ON sub.szefszefow = pracownicy.numer AND sub.szef != sub.szefszefow
GROUP BY pracownicy.numer, pracownicy.imie HAVING COUNT(szefszefow) < 5;
```

Zadanie 3 Rozwiązań możliwych do zaakceptowania jest w tym zadaniu wiele, dlatego oprócz pewnego rozwiązania wzorcowego (które jednak nie jest jedynym obowiązującym) zostanie tutaj przedstawione wiele komentarzy na temat lepszych i gorszych elementów, jakie w rozwiązaniach mogły się pojawić.

Rozwiązanie wzorcowe:

1. Dane dotyczące taryf i klientów zajmują w sumie 32 bloki. Zmieszczą się więc w pamięci – należy je tam wczytać, równocześnie dokonując selekcji klientów tych typów, które występują w zapytaniu.

2. Nie da się na podstawie danych z zadania oszacować ile bloków dla klientów pozostanie, jednak da się uzasadnić, że dane o klientach – z ew. już usuniętą informacją o typie klienta, natomiast z przygotowanym miejscem do zliczania sumy czasu rozmów nadal zmieszczą się w pamięci – informacji o połączeniach, gdzie jest m.in. nr klienta i czas rozmowy mieści się w bloku 100, klientów jest 1,5 tyś, daje to w najgorszym wypadku 15 dodatkowych bloków, czyli cały czas pozostaje ponad 50 wolnych bloków pamięci do przetwarzania połączeń.

3. Ponieważ interesować nas będzie jedynie 11% połączeń, dostęp do tabeli połączeń powinien zostać zorganizowany inaczej niż przez liniowe przejrzenie tabeli. Do wzorcowego rozwiązania przyjmijmy, że mamy indeks wewnętrzny (hash) po klientach (inne możliwe rozwiązania – patrz komentarze).

4. Złączenie będzie realizowane przez wersję algorytmu Nested Loops, gdzie z jednej strony następuje iteracja po klientach (już tylko tych odpowiednich typów), a z drugiej po połączeniach, gdzie początek interesującego nas bloku połączeń odnajdujemy za pomocą funkcji mieszającej. W trakcie przechodzenia przez połączenia dla konkretnego klienta weryfikujemy, czy dotyczą one interesującej nas taryfy oraz daty i odpowiednio poprawiamy sumę czasów połączeń dla klienta.

Koszty:

- Taryfy czytane są raz – 1 odczyt.
- Klienci czytani są do pamięci raz - 31 odczytów.
- Odczytamy około 0,11*(3 250 358/100) bloków z tabeli połączeń – około 360 odczytów.

Komentarze. Zapytanie składa się generalnie rzecz biorąc z dwóch złączeń, trzech selekcji oraz grupowania. Uwaga o 89% połączeń wykonywanych przez klientów typu C15 (czyli takich, których nie występują w zapytaniu) miała posłużyć jako wskazówka, że opłaca się zaprojektować dostęp do tablicy połączeń inaczej niż przez liniowe przeglądanie. Podobnie informacje dotyczące ilości rekordów w tabelach miały służyć jako wskazówki, że:

- nie ma żadnego znaczenia (ani sensu) robienie jakiś wymyślnych struktur dostępu do taryf – tak czy tak zajmują one jeden blok, a jeden odczyt to i tak najlepsze co się da osiągnąć;
- tabela klientów zajmuje 31 bloków – czyli zmieści się w pamięci.

Tabela połączeń ma o kilka rzędów wielkości więcej danych niż pozostałe, tak więc organizacja dostępu do niej jest najbardziej „wrażliwą" częścią zadania. Oprócz tego co zaproponowano w rozwiązaniu wzorcowym, możliwe jest zastosowanie wielu innych pomysłów. Można zrobić indeks typu B+ drzewo

- po klientach
- po klientach i datach
- ew. po klientach, datach i czasach rozmów – to po to, aby nie wszystkie dane do obliczeń były w indeksie i nie trzeba było robić ekstra odwołania do tabeli.

W pierwszych dwóch wypadkach mamy jeszcze wybór między indeksem wewnętrznym (czyli jakąś formą klastra indeksowego) lub klasycznym indeksem zewnętrznym. Wszystkie te rozwiązania niosą ze sobą trudność oszacowania kosztu nawigacji po indeksie, z zadaniu nie ma informacji o rozmiarach kolumn, więc z góry jesteśmy skazani na szacunki, a nie dokładne obliczenia. Pomysł dodania do indeksu również daty nie jest zły, ale z informacji przekazanych w zadaniu nie wynika, czy na tym coś zyskamy – wiemy, że wybierzemy co

najwyżej 11% zawartości tabeli połączeń ze względu na typy klientów, natomiast jak rozłożone są daty połączeń i czy to jakoś dodatkowo ograniczy wolumen przeglądanych wierszy nie wiadomo.

Indeksy bitmapowe. Jako ciekawostkę (znajomość tego nie była wymagana, więc i nie oczekiwano takich pomysłów w rozwiązaniach) można dodać ewentualność zorganizowania na połączeniach map bitowych dotyczących taryf i/lub klientów. Taka mapa bitowa (patrz np. indeksy bitmapowe w Oracle) może być użyta do zakodowania taryf lub klientów występujących w złączeniach i w ten sposób, po wstępnym wyliczeniu np. zbioru nr taryf interesujących w zapytaniu można już zrezygnować ze realizacji złączenia tabel na rzecz wyszukiwania w mapie bitowej (na połączeniach) pasujących rekordów. Operacje na mapach bitowych to szybko realizowane operacje OR lub AND (logiczne na bitach). Można też łącznie uwzględniać kilka przeszukiwanych map bitowych. Rozwiązania takie są obecnie często stosowane w hurtowniach danych.

Zadanie 4 Decydujemy się na dodanie tabeli *TrenerZawodnik*, w której gromadzimy dane, kto kogo trenuje. Encję *Zawodnik* rozbijamy na dwie tabele, *Zawodnik* oraz *Wystep*, w ten sposob gwarantując postać BCNF. Oczywiście nie powoduje to utraty zależności.

```
CREATE TABLE Reprezentacja (
  idreprezentacji INTEGER PRIMARY KEY,
  kraj VARCHAR(30) NOT NULL
);

CREATE TABLE Druzyna (
  iddruzyny INTEGER PRIMARY KEY,
  idreprezentacji INTEGER NOT NULL,
  dyscyplina VARCHAR(30) NOT NULL
);

CREATE TABLE Trener (
  idtrenera INTEGER PRIMARY KEY,
  idreprezentacji INTEGER NOT NULL,
  iddruzyny INTEGER,
  nazwisko VARCHAR(30) NOT NULL
);

CREATE TABLE Zawodnik (
  idzawodnika INTEGER PRIMARY KEY,
  dyscyplina VARCHAR(30) NOT NULL,
  iddruzyny INTEGER
);
```

```
CREATE TABLE TrenerZawodnik (
  idzawodnika INTEGER,
  idtrenera INTEGER,
  PRIMARY KEY (idzawodnika,idtrenera)
);

CREATE TABLE Wystep (
  idwystepu INTEGER PRIMARY KEY,
  idzawodnika INTEGER NOT NULL,
  godzina DATE,
  UNIQUE (idzawodnika,godzina)
);
```

Do przyjęcia jest wydzielenie związku *(Druzyna, Zawodnik)* jako oddzielnej tabeli; podobnie wprowadzenie dwóch podtypów zawodników (drużynowych i niedrużynowych).