Egzamin z baz danych
3 lutego 2012
Zadania i omówienie rozwiązań

Szymon Acedański Zbigniew Jurkiewicz
Paweł Kucharczyk Filip Murlak Oskar Skibski
Krzysztof Stencel

Instytut Informatyki
Uniwersytet Warszawski

Zadanie 1: Teoretyczne języki zapytań

Rozważmy relację:

```
Osoba(id,imię,nazwisko,płeć,id_osoby) // id jest kluczem, id > 0
```

Kolumna `id_osoby` zawiera identyfikator partnera danej osoby, przy czym liczba zero oznacza brak partnera. Rozważmy zapytanie:

```
SELECT o1.imię, o2.imię
  FROM Osoba o1 OUTER JOIN Osoba o2 ON (o1.id_osoby = o2.id)
```

Zapisz to zapytanie w algebrze relacji lub w języku logiki pierwszego rzędu. Wartości `NULL` reprezentuj stałą napisową `'NULL'`.

Rozwiązanie (Szymon Acedański)

$$\begin{aligned} & \pi_{o1.imie,o2.imie}(\rho_{o1}(Osoba) \bowtie_{o1.id\_osoby=o2.id} \rho_{o2}(Osoba)) \\ \cup\ & \{\langle NULL_{o1.imie}\rangle\} \times \pi_{o2.imie}(\sigma_{o2.id\_osoby=0}(\rho_{o2}(Osoba))) \\ \cup\ & \pi_{o1.imie}(\sigma_{o1.id\_osoby=0}(\rho_{o1}(Osoba))) \times \{\langle NULL_{o2.imie}\rangle\} \end{aligned}$$

$$\rho_\alpha = \rho_{\alpha.id/id,\alpha.imie/imie,\alpha.nazwisko/nazwisko,\alpha.plec/plec,\alpha.id\_osoby/is\_osoby}$$

Zadanie 2: Projektowanie baz danych

Firma komunikacji autobusowej chce gromadzić dane o:

- Posiadanym taborze autobusowym, m.in. typ, numer rejestracyjny, liczba miejsc.
- Rozkładzie jazdy, m.in. linia, skąd, dokąd, czas odjazdu, czas przyjazdu, dni kursowania.

- Kierowcach, m.in. nazwisko, imię, uprawnienia, adres, stawka za godzinę.
- Rezerwacji biletów na poszczególne kursy.

Baza ma służyć do rozliczania kierowców (tj. obliczania ich wynagrodzenia) i do optymalnego organizowania kursów autobusów. Kursy odbywają się zgodnie z rozkładem, są do nich dobierani kierowcy oraz autobusy zgodnie z rezerwacjami biletów. Jeśli rezerwacji jest więcej niż miejsc w autobusie, to wyjeżdża więcej niż jeden pojazd zgodnie z zapotrzebowaniem i możliwościami. Zaprojektuj dla takiej firmy bazę danych w postaci normalnej Boyce'a-Codda. Narysuj diagram związków-encji i podaj polecenia SQL tworzące taka bazę.

Rozwiązanie (Zbigniew Jurkiewicz)

Oto bazowe rozwiązanie. Możliwe jest rozmaite rozszerzanie tego. Większość sensownych rozszerzeń i modyfikacji akceptowałem, tj. oceniałem pozytywnie.

```
CREATE TABLE Kierowca (
  prawojazdy CHAR(9) PRIMARY KEY,
  imie VARCHAR(20) NOT NULL,
  nazwisko VARCHAR(20) NOT NULL,
  ...
);

CREATE TABLE Uprawnienia (
  kierowca CHAR(9) REFERENCES Kierowca,
  uprawnienie CHAR(2),
  PRIMARY KEY(kierowca, uprawnienie)
);

CREATE TABLE Autobus (
  nrrej CHAR(7) PRIMARY KEY,
  typ CHAR(2),
  liczbamiejsc INTEGER NOT NULL CHECK (liczbamiejsc > 0)
  ...
);

CREATE TABLE Rozklad (
  linia INTEGER PRIMARY KEY,
  skad VARCHAR(20),
  dokad VARCHAR(20),
  odjazd TIME,
  przyjazd TIME
);

CREATE TABLE DniLinii (
  linia INTEGER REFERENCES Rozklad,
  dzien INTEGER NT NULL CHECK (dzien BETWEEN 1 AND 7)
  PRIMARY KEY(linia, dzien)
);
```

```
CREATE TABLE Kurs (
  id INTEGER PRIMARY KEY,
  linia INTEGER REFERENCES Rozklad,
  data DATE,
  ilerezerwacji INTEGER NOT NULL
                        DEFAULT 0 CHECK (ilerezerwacji >= 0),
  UNIQUE(linia, data)
);

CREATE TABLE PRZEJAZD (
  kurs INTEGER REFERENCES Kurs,
  bus CHAR(7) REFERENCES Autobus,
  kierowca CHAR(9) REFERENCES Kierowca,
  PRIMARY KEY(kurs, bus, kierowca),
  UNIQUE(kurs,bus),
  UNIQUE(kurs,kierowca)
);
```

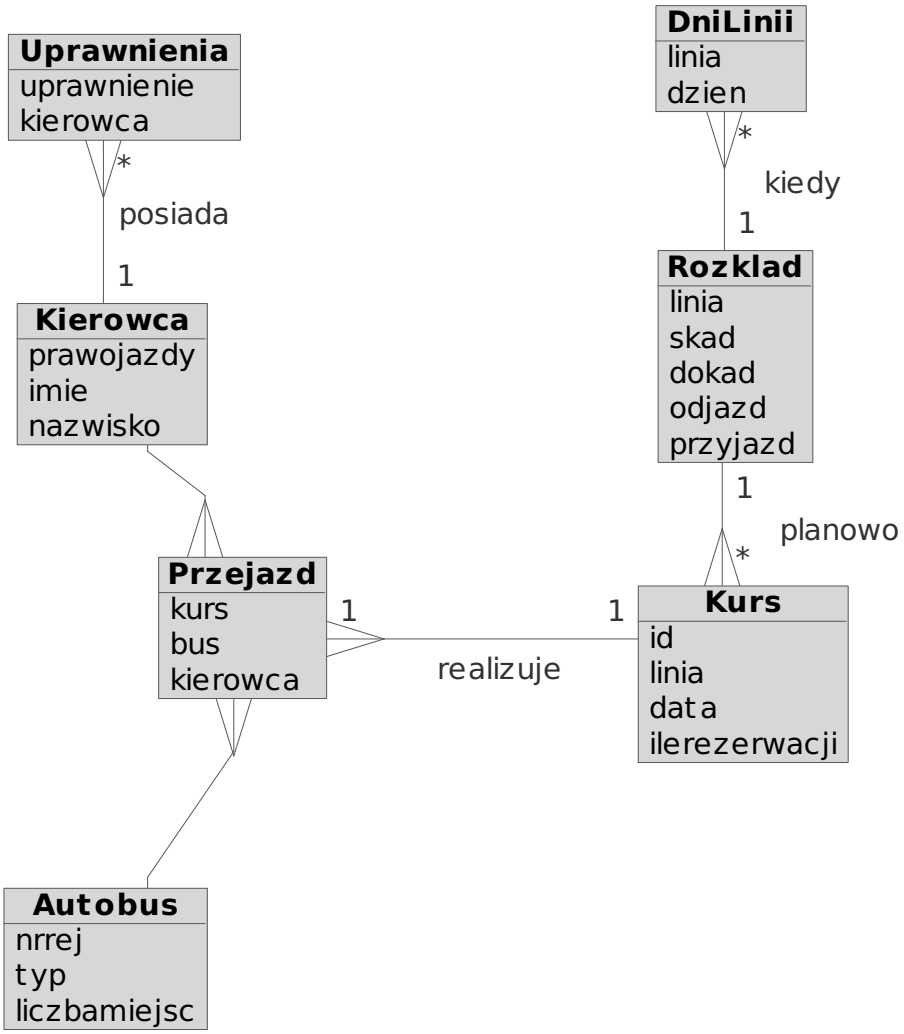

Diagram: drugie Strona 1

Zadanie 3: Zależności funkcyjne

Rozważmy schemat relacji $R(A_1, A_2, \ldots, A_n)$ i permutację $p : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$. Wiemy, ze dla każdego $k = 1, 2, \ldots, n$ zachodzi zależność funkcyjna $A_k \longrightarrow A_{p(k)}$. Te n zależności funkcyjnych stanowi reprezentację wszystkich zależności funkcyjnych zachodzących w tej tabeli. Określić liczbę kluczy tej relacji zależnie od właściwości permutacji p.

Rozwiązanie (Oskar Skibski)

Przedstawmy permutacje ρ w postaci cyklowej:

$$\rho = (x_1, \rho(x_1), \ldots, \rho^{c_1-1}(x_1))(x_2, \rho(x_2), \ldots, \rho^{c_2-1}(x_2)) \ldots (x_k, \rho(x_k), \ldots, \rho^{c_k-1}(x_k)).$$

Powyższa permutacja składa się zatem z k cykli o długościach $c_1, c_2, \ldots, c_k$. Niech $\rho^i(x_a)$ będzie dowolnym elementem. Skorzystajmy z właściwości rozkładu:

- z definicji cyklu wiemy, że dla dowolnego innego elementu z tego cyklu $\rho^j(x_a)$ zachodzi $\rho^{(c_a+j-i)}(\rho^i(x_a)) = \rho^j(x_a)$;
- z rozłączności cyklów wiemy, że dla dowolnego elementu innego cyklu $\rho^j(x_b)$ nie istnieje takie m, że $\rho^m(\rho^i(x_a)) = \rho^j(x_b)$.

Przekładając powyższe uwagi na język zależności funkcyjnych otrzymujemy, że do domknięcia dowolnego atrybutu względem zbioru zależności funkcyjnych należą wszystkie atrybuty tworzące z nim cykl i nic poza tym:

$$(A_{\rho^i(x_a)})^+ = \{A_{x_a}, A_{\rho(x_a)}, \ldots, A_{\rho^{c_a-1}(x_a)}\}.$$

Do dowolnego nadklucza należeć musi zatem co najmniej jeden element z każdego cyklu (inaczej nie uzyskamy elementów tego cyklu), a aby nadklucz ten był kluczem nie może się w nim znajdować więcej niż jeden taki atrybut (są one bowiem równoważne). Liczba kluczy jest zatem równa $\prod_{i=1}^{k} c_i$ – na tyle sposobów możemy wybrać po jednym elemencie z każdego cyklu.

Zadanie 4: Wykonywanie zapytań

Dla następującego schematu tabel (nigdzie nie ma wartości NULL):

```
Emp(id, name, hiredate)        // id jest kluczem
Order(id, date, discount, eid) // id jest kluczem,
                               // eid to klucz obcy do Emp
Detail(oid, pos, product, amount, price)
                               // (oid, pos) jest kluczem,
                               // oid to klucz obcy do Order
```

Tabele te zawierają odpowiednio 10^4, 10^6 i 10^8 krotek. Na kluczach głównych są pozakładane pogrupowane B+drzewa. Innych indeksów nie ma. Określić możliwie najlepszy plan wykonania zapytania:

```
SELECT e.id, sum(d.amount * d.price)
  FROM Emp e, Order o, Detail d
  WHERE e.id = o.eid AND o.id = d.oid
  GROUP BY e.id;
```

Wzór rozwiązania (Paweł Kucharczyk)

Po pierwsze należy zauważyć, że nie ma potrzeby korzystania z tabeli `Emp`. `Order.eid` jest kluczem obcym do `Emp.id`. Złączenie w zapytaniu jest złączeniem wewnętrznym i jako takie nie uwzględni pracowników, którzy nie mają żadnego zamówienia.

Dzięki obecności pogrupowanych B+drzew na `Detail` i `Order` wystarczy w celu obliczenia zapytania przejść każdą z tabel (według tych indeksów) raz.

1. Przy przechodzeniu przez `Detail` obliczamy sumę `amount * price` - wartość zamówienia.
2. Przejście przez tabelę `Order` jest potrzebne, aby odczytać `Order.eid`.
3. `Order.eid` jest potrzebne, aby sumę obliczoną w pkt. 1 dodać do innych zamówień danego pracownika - w tym celu zakładamy mapę/pohashowaną tabelę.
4. Taka tabela będzie zawierała 10^4 liczb, więc raczej nie ma niebezpieczeństwa, że nie zmieści się w RAMie. Po przejściu przez całą tabelę `Detail` tabela z pkt. 3 zawiera wynik zapytania.

Uwagi:

- rozwiązania gdzie wynik z pkt. 3 był zbierany nie w RAMie tylko jako tabela pośrednia na dysku też były akceptowane.
- za wykonanie złączenia z tabelą `Emp` były obcinane punkty (ale nie dużo).
- rozpoczęcie złączeń od tabeli `Emp` było traktowane jako niedobry (nieoptymalny) plan wykonania zapytania. Ewentualne punkty przydzielane w takiej sytuacji zależały od tego czy opisany plan w ogóle działał

Zadanie 5: SQL

Dla schematu tabel jak w zadaniu 4 napisać zapytanie SQL znajdujące identyfikator i nazwisko każdego takiego pracownika, który

- obsłużył co najmniej 10 zamówień ORAZ
- w każdym jego zamówieniu najtańsza pozycja tego zamówienia (tj. iloczyn `amount * price`) jest nie mniejsza niż 50% najdroższej pozycji.

Rozwiązanie (Filip Murlak)

```
SELECT Emp.id, Emp.name
FROM Emp, Order
WHERE Emp.id = Order.eid AND Emp.id NOT IN
     (SELECT o.eid
     FROM Order o, Detail d
     WHERE o.id = d.oid
     GROUP BY o.eid, o.id
     HAVING 0.5 * MAX(d.amount * d.price) >
                         2 * MIN(d.amount * d.price))
GROUP BY Emp.id, Emp.name
HAVING COUNT(Order.id) >= 10;
```