Wykład 4

Algebraiczne podstawy implementacji strukturalnego języka zapytań (SQL) w systemach baz danych Oracle – zapytania w języku algebry relacyjnych baz danych i ich odpowiedniki w SQL

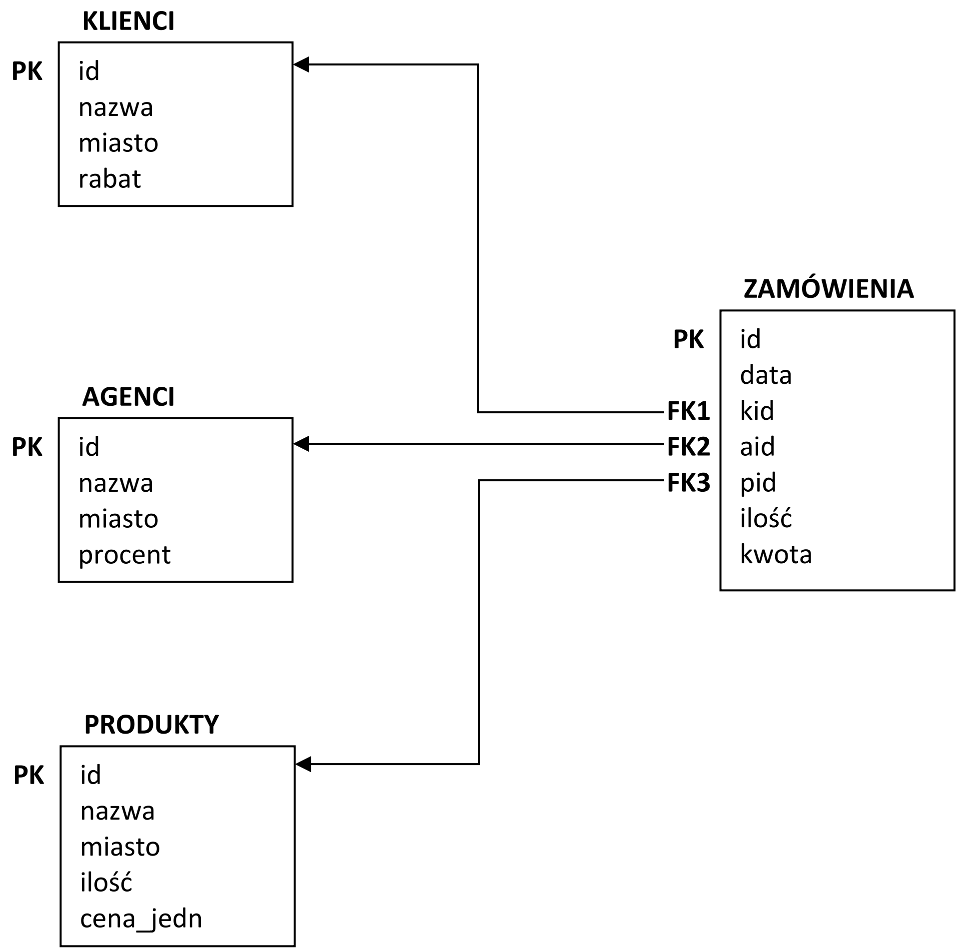
Rozpatrzymy bardzo uproszczoną bazę danych o schemacie

S = {KLIENCI, AGENCI, PRODUKTY, ZAMÓWIENIA},

której relacje mają następujące nagłówki:

N(KLIENCI)	=	{id,nazwa,miasto, rabat}
N(AGENCI)	=	{id.nazwa,miasto, procent}
N(PRODUKTY)	=	{id,nazwa,miasto, ilość, cena_jedn}
N(ZAMÓWIENIA)	=	{id,data, kid, aid, pid, ilość, kwota}

Założymy dla tej bazy następujący diagram związków encji:



W celu uproszczenia zapisu rozwiązań algebraicznych rezerwujemy i nadajemy poniższe aliasy relacjom rozważanej bazy danych:

KLIENCI = K

AGENCI = A

PRODUKTY = P

ZAMÓWIENIA = Z

Zadania z rozwiązaniami

Zad. 1. Znaleźć wszystkie identyfikatory i nazwy klientów.

Rozwiązanie algebraiczne

K[id,nazwa]

Alternatywna notacja: $\pi_{id,nazwa}(K)$

Rozwiązanie w SQL

select id, nazwa from KLIENCI;

Zad. 2. Znaleźć wszystkie dane magazynowe dotyczące produktów o nazwie *monitor*

Rozwiązanie algebraiczne

P gdzie nazwa = ‘monitor’

Alternatywna notacja: $\sigma_{nazwa='monitor'}(P)$

Rozwiązanie w SQL

```
select * from PRODUKTY
where nazwa = 'monitor';
```

Zad. 3. Znaleźć wszystkie identyfikatory klientów, którzy nie składali żadnego zamówienia poprzez agenta o identyfikatorze *a03*. Rozważyć dwa przypadki:

a) klientów, którzy jednak cokolwiek zamawiali,
b) oprócz informacji wybranych w podpunkcie (a) uwzględnić również klientów, którzy nic nie zamówili.

a)

Pozornie narzucające się, ale błędne rozwiązanie algebraiczne

(Z gdzie aid ≠ ‘a03’)[kid]

Dowód niepoprawności tego rozwiązania:

Z

id	data	kid	aid	pid	ilość	kwota
z01	01.10.2007	k01	a03	p03	10	100
z02	02.10.2007	k01	a01	p04	50	500
z03	05.10.2007	k02	a02	p06	200	1000
z05	01.10.2007	k03	a03	p01	1	10

Z gdzie aid ≠ ‘a03’

id	data	kid	aid	pid	ilość	kwota
z02	02.10.2007	k01	a01	p04	50	500
z03	05.10.2007	k02	a02	p06	200	1000

(Z gdzie aid ≠ ‘a03’)[kid]

kid
k01
k02

A jak widać z powyższej instancji relacji Z klient o identyfikatorze *k01* złożył zamówienie *z01* także poprzez agenta *a03*!

Poprawne rozwiązanie algebraiczne

Znajdujemy klientów, którzy złożyli przynajmniej jedno zamówienie przez agenta o identyfikatorze *a03*:

T = (Z gdzie aid = ‘a03’)[kid]

A następnie znajdujemy tych klientów, którzy nie zostali wybrani do relacji T:

Z[kid] – T

Rozwiązanie w SQL

```
select kid from ZAMÓWIENIA
minus
select kid from ZAMÓWIENIA
where aid = ‘a03’;
```

b)

Rozwiązanie algebraiczne

T = (Z gdzie aid = ‘a03’)[kid]

K[id] – T

Rozwiązanie w SQL

```
select id from KLIENCI
minus
select kid from ZAMÓWIENIA
where aid = ‘a03’;
```

Zad. 4. Zna1eźć identyfikatory tych klientów, którzy składali zamówienia wyłącznie przez agenta o identyfikatorze *a03*.

Pozornie narzucające się, ale błędne rozwiązanie algebraiczne

(Z gdzie aid = ‘a03’)[kid]

Dowód niepoprawności tego rozwiązania:

Z

id	data	kid	aid	pid	ilość	kwota
z01	01.10.2007	k01	a03	p03	10	100
z02	02.10.2007	k01	a01	p04	50	500
z03	05.10.2007	k02	a02	p06	200	1000
z05	01.10.2007	k03	a03	p01	1	10

Z gdzie aid = ‘a03’

id	data	kid	aid	pid	ilość	kwota
z01	01.10.2007	k01	a03	p03	10	100
z05	01.10.2007	k03	a03	p01	1	10

(Z gdzie aid =‘a03’)[kid]

kid
k01
k03

A jak widać z powyższej instancji relacji Z klient o identyfikatorze *k01* złożył zamówienie *z02* także poprzez agenta *a01*!

Poprawne rozwiązanie algebraiczne

Znajdujemy klientów, którzy złożyli przynajmniej jedno zamówienie przez innego agenta niż *a03*:

T = (Z gdzie aid ≠ ‘a03’)[kid]

Następnie znajdujemy tych klientów, którzy nie zostali wybrani do relacji T:

Z[kid] – T

Rozwiązanie w SQL

```
select kid from ZAMÓWIENIA
minus
select kid from ZAMÓWIENIA
where aid <> 'a03';
```

Zad. 5. Wypisać listę produktów nigdy nie zamawianych przez klientów z Katowic za pośrednictwem agenta z Gliwic

Rozwiązanie algebraiczne

K1 = (K gdzie miasto = 'Katowice')[id]
K1.id = K1.kid

A1 = (A gdzie miasto = 'Gliwice')[id]
A1.id = A1.aid

Wprowadzamy aliasy atrybutów: K1.id = K1.kid oraz A1.id = A1.aid, aby móc zastosować złączenie naturalne.

$Z \bowtie K1$ – dane o zamówieniach złożonych przez klientów z Katowic

$(Z \bowtie K1) \bowtie A1$ – dane o zamówieniach złożonych przez klientów z Katowic za pośrednictwem agentów z Gliwic

$((Z \bowtie K1) \bowtie A1)[pid]$ – identyfikatory produktów zamawianych przez klientów z Katowic za pośrednictwem agentów z Gliwic

$W = P[id] - ((Z \bowtie K1) \bowtie A1)[pid]$
ewentualnie: $W = Z[pid] - ((Z \bowtie K1) \bowtie A1)[pid]$

Jedno z możliwych rozwiązań w SQL

```
select id from PRODUKTY
minus
select Z.pid from ZAMÓWIENIA Z, KLIENCI K, AGENCI A
where Z.kid = K.id and Z.aid = A.id
        and K.miasto = 'Katowice' and A.miasto = 'Gliwice';
```

Formalnie powyższe rozwiązanie w SQL odpowiada następującemu równoważnemu rozwiązaniu algebraicznemu:

P[id] – (((Z × K) × A) gdzie Z.kid = K.id i Z.aid = A.id
i K.miasto = ‘Katowice’ i A.miasto = ‘Gliwice’)[pid]

lub

S = (Z × (K gdzie miasto = ‘Katowice’) × (A gdzie A.miasto = ‘Gliwice’))
gdzie Z.kid = K.id i Z.aid = A.id

W = P[id] – S[pid]

Wracając do poprzednio zastosowanych aliasów relacji:

K1 = K gdzie miasto = ‘Katowice’
A1 = A gdzie A.miasto = ‘Gliwice’

możemy napisać:

S = (Z × K1 × A1) gdzie Z.kid = K1.id i Z.aid = A1.id

Natomiast uwzględniając aliasy atrybutów:
K1.id = K1.kid oraz A1.id = A1.aid:

S = (Z × K1 × A1) gdzie Z.kid = K1.kid i Z.aid = A1.aid

czyli S = Z ⊳⊲ K1 ⊳⊲ A1

Ostatnią zależność można pokazać w sposób ogólny. Zrobimy to na przykładzie pojedynczego złączenia.

<u>Zad. 6a</u>. Wyrazić **naturalne równozłączenie wewnętrzne** dwóch relacji *R* i *S* o nagłówkach N(*R*)={*X1*,*X2*,...,*Xn*,*Z1*,*Z2*,...,*Zk*} i N(*S*)={*Z1*,*Z2*,...,*Zk*,*Y1*,*Y2*,...,*Ym*}, gdzie $n,k,m \geq 0$, za pomocą operatorów produktu, selekcji i projekcji.

<u>Rozwiązanie</u>

$R \rhd\lhd S$
= ((R × S) gdzie R.Z1 = S.Z1 i R.Z2 = S.Z2 i … i R.Zk = S.Zk) [X1,X2, … , Xn, R.Z1,R.Z2, … , R.Zk, Y1,Y2, … , Ym]

Alternatywna notacja:

$R \rhd\lhd S$
$= \pi_{X1,X2,\ldots,Xn,R.Z1,R.Z2,\ldots,R.Zk,Y1,Y2,\ldots,Ym}(\sigma_{R.Z1 = S.Z1 \wedge R.Z2 = S.Z2 \wedge \ldots \wedge R.Zk = S.Zk}(R \times S))$

<u>Rozwiązanie w SQL</u>

$R \rhd\lhd S$:

```
select * from R natural join S;
```

$\pi_{X1,X2,\ldots,Xn,R.Z1,R.Z2,\ldots,R.Zk,Y1,Y2,\ldots,Ym}(\sigma_{R.Z1 = S.Z1 \wedge R.Z2 = S.Z2 \wedge \ldots \wedge R.Zk = S.Zk}(R \times S))$:

```
select X1, X2 , … , Xn , R.Z1, R.Z2 , … , R.Zk , Y1, Y2, …. Ym
from R, S
where R.Z1 = S.Z1 and R.Z2 = S.Z2 and … and R.Zk = S.Zk;
```

<u>Pytania</u>:
Co będzie wynikiem złączenia naturalnego relacji *R* i *S*:

i) gdy relacje *R* i *S* nie będą posiadały współdzielonych atrybutów,

ii) gdy relacje *R* i *S* będą kompatybilne?

Zad. 6b. Pokazać, że **naturalne równozłączenie zewnętrzne** relacji R i S o nagłówkach podanych w zad. 6a może być wyrażone za pomocą operatorów: unii, selekcji, projekcji oraz produktu

Rozwiązanie

Wprowadźmy dwie relacje pomocnicze: R_{NULL} i S_{NULL} o nagłówkach odpowiednio: $N(R_{NULL}) = N(R)$ i $N(S_{NULL}) = N(S)$, z których każda ma tylko 1 krotkę o wartościach NULL dla każdego atrybutu.

NULL – jest predefiniowaną wartością „pustą”, tzn. oznacza brak wartości.

Wówczas:

$R \rhd\lhd_O S =$

((R × S) gdzie R.Z1 = S.Z1 i R.Z2 = S.Z2 i … i R.Zk = S.Zk)[X1,X2,…, Xn,R.Z1,R.Z2,…, R.Zk,Y1,Y2,…, Ym]

∪

((R × S) gdzie R.Z1 ≠ S.Z1 lub R.Z2 ≠ S.Z2 lub … lub R.Zk ≠ S.Zk) [X1,X2, … , Xn, R.Z1,R.Z2, … , R.Zk] × S_{NULL}[Y1,Y2, … , Ym]

∪

R_{NULL}[X1,X2, … , Xn] × ((R × S) gdzie R.Z1 ≠ S.Z1 lub R.Z2 ≠ S.Z2 lub … lub R.Zk ≠ S.Zk) [R.Z1,R.Z2, … , R.Zk,Y1,Y2, … , Ym]

Alternatywna notacja:

$R \rhd\lhd_O S =$

$\pi_{X1,X2,\ldots,Xn,R.Z1,R.Z2,\ldots,R.Zk,Y1,Y2,\ldots,Ym}(\sigma_{R.Z1=S.Z1 \wedge R.Z2=S.Z2 \wedge \cdots \wedge R.Zk=S.Zk}(R \times S))$

$\cup$

$\pi_{X1,X2,\ldots,Xn,R.Z1,R.Z2,\ldots,R.Zk}(\sigma_{R.Z1 \neq S.Z1 \vee R.Z2 \neq S.Z2 \vee \cdots \vee R.Zk \neq S.Zk}(R \times S)) \times \pi_{Y1,Y2,\ldots,Ym}(S_{NULL}))$

$\cup$

$\pi_{X1,X2,\ldots,Xn}(R_{NULL}) \times \pi_{R.Z1,R.Z2,\ldots,R.Zk,Y1,Y2,\ldots,Ym}(\sigma_{R.Z1 \neq S.Z1 \vee R.Z2 \neq S.Z2 \vee \cdots \vee R.Zk \neq S.Zk}(R \times S))$

Najprostsza reprezentacja lewej strony powyższej zależności w standardzie SQL:

```
select * from R natural full join S;
```

Prawa strony stanowi podstawę do starego zapisu w Oracle SQL (nadal dostępnego w nowych wersjach serwera):

```
select X1,X2, … , Xn, R.Z1,R.Z2, … , R.Zk, Y1,Y2, … , Ym from R, S
where R.Z1=S.Z1(+) and R.Z2=S.Z2(+) and … and R.Zk=S.Zk(+)
union
select X1,X2, … , Xn, R.Z1,R.Z2, … , R.Zk, Y1,Y2, … , Ym from R, S
where R.Z1(+)=S.Z1 and R.Z2(+)=S.Z2 and … and R.Zk(+)=S.Zk;
```

Jednak warto zaznaczyć, że powyższe zapytanie SQL należy interpretować jako: $(R \rhd\lhd_{LO} S) \cup (R \rhd\lhd_{RO} S)$

Zad. 7. Wypisać nazwy klientów, którzy zamawiali wszystkie produkty o cenie 50zł. Wskazówka: wykorzystać operator dzielenia.

Rozwiązanie algebraiczne

P1 = (P gdzie cena_jedn = 50)[id]

K1 = Z[kid,pid] ÷ P1

W = (K1 ⊳⊲ K)[nazwa]

Rozwiązanie w SQL

W systemach baz danych operator dzielenia zwykle nie jest implementowany. Jak pamiętamy, dzielenie nie należy do zupełnego zbioru działań algebry relacyjnych baz danych. Dlatego, aby podać rozwiązanie tego zadania w SQL musimy zastanowić się jak wyrazić dzielenie poprzez operatory zupełnego zbioru działań algebry relacyjnych baz danych.

Zad. 8. Wyrazić dzielenie relacji R przez S o nagłówkach N(*R*)={*X1*,*X2*,...,*Xn*,*Z1*,*Z2*,...,*Zk*} i N(*S*)={*Z1*,*Z2*,...,*Zk*} poprzez działania zupełnego zbioru działań algebry relacyjnych baz danych.

W ogólności można pokazać, że dzielenie można wyrazić za pomocą operatorów produktu, projekcji, selekcji i różnicy:

R ÷ S = R[X1,X2,...,Xn] – (R[X1,X2,...,Xn] × S – R)[X1,X2,...,Xn]

Zamiast dowodu krótkie wyjaśnienie:

T1 = R[X1,X2,...,Xn] × S – zestaw krotek skonstruowanych poprzez utworzenie wszystkich możliwych kombinacji podkrotek t(X1,X2,...,Xn) z relacji R i krotek relacji S

R[X1,X2,...,Xn] × S – R – z wszystkich możliwych kombinacji krotek relacji S i podkrotek t(X1,X2,...,Xn) z relacji R (czyli z krotek relacji T1) odrzucamy te, które należą także do relacji R.

T2 = (R[X1,X2,...,Xn] × S – R) [X1,X2,...,Xn] – otrzymujemy podkrotki t(X1,X2,...,Xn), które nie należą do relacji R, ale są podkrotkami t(X1,X2,...,Xn) krotek utworzonych poprzez wszystkie możliwe kombinacje krotek relacji S i podkrotek t(X1,X2,...,Xn) z relacji R, czyli znajdujemy brakujące podkrotki t(X1,X2,...,Xn) w relacji R, które są „odpowiednikami" podkrotek v(Z1,Z2,...,Zk).

Innymi słowy do relacji T2 trafia krotka t(X1,X2,...,Xn), która nie jest podkrotką relacji R, ale jest podkrotką relacji T1, której „odpowiada" pewna podkrotka v(Z1,Z2,...,Zk) w relacji T1.

W efekcie różnica:

R[X1,X2,...,Xn] – (R[X1,X2,...,Xn] × S – R)[X1,X2,...,Xn]

powoduje odrzucenie z zestawu wszystkich podkrotek t(X1,X2,...,Xn) należących do relacji R takich podkrotek t(X1,X2,...,Xn) relacji R, którym brakuje w relacji R przynajmniej jednej „odpowiadającej" podkrotki v(Z1,Z2,...,Zk), należącej do zestawu wszystkich możliwych kombinacji krotek relacji S i podkrotek t(X1,X2,...,Xn) relacji R, a to jest równoważne dzieleniu R ÷ S.

Przykład

R

X	Y
x1	y1
x1	y2
x2	y2
x3	y1
x5	y2
x4	y2
x5	y1
x6	y3

S

Y
y1
y2

R ÷ S

X
x1
x5

Zgodnie z ogólną tożsamością

$R \div S = R[X1,X2,...,Xn] - (R[X1,X2,...,Xn] \times S - R)[X1,X2,...,Xn]$

wykonujemy poszczególne działania dla naszego przykładu:

R[X]

X
x1
x2
x3
x4
x5
x6

R[X] × S

X	Y	
x1	y1	–
x1	y2	–
x2	y1	
x2	y2	–
x3	y1	–
x3	y2	
x4	y1	
x4	y2	–
x5	y1	–
x5	y2	–
x6	y1	
x6	y2	

R[X] × S – R

X	Y
x2	y1
x3	y2
x4	y1
x6	y1
x6	y2

(R[X] × S – R)[X]

X
x2
x3
x4
x6

R[X] – (R[X] × S – R)[X]   A to jest tożsame z R ÷ S

X
x1
x5

Teraz możemy podać <u>rozwiązanie zad. 7 w SQL</u>

W rozwiązaniu algebraicznym zad. 7 wyrazimy dzielenie poprzez operatory produktu, projekcji, selekcji i różnicy:

P1 = (P gdzie cena_jedn = 50)[id]

K1 = Z[kid,pid] ÷ P1 = Z[kid] – (Z[kid] × P1 – Z[kid,pid])[kid]

W = (K1 ⊳⊲ K)[nazwa]

Wówczas w SQL możemy zapisać to w następujący sposób:

```
select nazwa
from KLIENCI natural join
(select kid from ZAMÓWIENIA
minus
select kid from (select kid, id from ZAMÓWIENIA, PRODUKTY
                  where cena_jedn = 50
                 minus
                 select kid, pid from ZAMÓWIENIA)
)
```

lub

```
select nazwa from KLIENCI
where id in (select kid from ZAMÓWIENIA
             minus
             select kid from (select kid, id from ZAMÓWIENIA,
                                                  PRODUKTY
                              where cena_jedn = 50
                              minus
                              select kid, pid from ZAMÓWIENIA)
            )
```

lub

```
select nazwa
from KLIENCI, (select kid from ZAMÓWIENIA
               minus
               select kid from (select kid, id from ZAMÓWIENIA,
                                                    PRODUKTY
                                where cena_jedn = 50
                                minus
                                select kid, pid from ZAMÓWIENIA)
              )
 where id = kid;
```

Zad. 9a. Wypisać identyfikatory agentów, którzy realizowali zamówienia **na przynajmniej cały zestaw produktów**, jaki zamówił klient o identyfikatorze c04 (a być może bardziej liczny).

Rozwiązanie algebraiczne

P1 = (Z gdzie kid = ‘c04’)[pid]

A1 = Z[aid,pid] ÷ P1

Rozwiązanie w SQL

Zgodnie z tożsamością: Z[aid,pid]÷P1 = Z[aid] – (Z[aid]×P1–Z[aid,pid])[aid] możemy napisać zapytanie SQL:

```
select aid from ZAMÓWIENIA
minus
select aid from (select Z1.aid, Z2.pid
                 from ZAMÓWIENIA Z1, ZAMÓWIENIA Z2
                  where Z2.kid = ‘c04’
                 minus
                 select aid, pid from ZAMÓWIENIA);
```

Zad. 9b. Wypisać identyfikatory agentów, którzy realizowali zamówienia na **wyłącznie cały zestaw produktów**, jaki zamówił klient o identyfikatorze c04.

P1 = (Z gdzie kid = ‘c04’)[pid]

P2 = Z[pid] – P1 – produkty, które nie były zamawiane przez klienta c04

A1 = Z[aid,pid] ÷ P1 – identyfikatory agentów, którzy realizowali zamówienia na przynajmniej cały zestaw produktów, które zamawiał klient c04

A2 = (Z[aid,pid] ⊳⊲ P2)[aid] – identyfikatory agentów, którzy realizowali zamówienia na produkty (przynajmniej jeden produkt) nie zamawiane przez klienta c04

W = A1 – A2

Zapytania algebraiczne: P1 = (Z gdzie kid = ‘c04’)[pid] , P2 = Z[pid] – P1 , A1 = Z[aid,pid] ÷ P1 , A2 = (Z[aid,pid] ⊳⊲ P2)[aid] , W = A1 – A2

Rozwiązanie w SQL

```
(select aid from ZAMÓWIENIA
minus
select aid from (select Z1.aid, Z2.pid
                 from ZAMÓWIENIA Z1, ZAMÓWIENIA Z2
                  where Z2.kid = ‘c04’
                 minus
                 select aid, pid from ZAMÓWIENIA)
)
minus
(
select aid from ZAMÓWIENIA
where pid in (select pid from ZAMÓWIENIA
              minus
              select pid from ZAMÓWIENIA
              where kid=‘c04’)
);
```

Alternatywne rozwiązanie w SQL

```
(select aid from ZAMÓWIENIA
minus
select aid from (select Z1.aid, Z2.pid
                 from ZAMÓWIENIA Z1, ZAMÓWIENIA Z2
                  where Z2.kid = ‘c04’
                 minus
                 select aid, pid from ZAMÓWIENIA)
)
minus
(
select aid from ZAMÓWIENIA
                natural join  (select pid from ZAMÓWIENIA
                               minus
                               select pid from ZAMÓWIENIA
                               where kid=‘c04’)
);
```

Przykład do zadań 9a i 9b

Z[kid,aid,pid]

kid	**aid**	**pid**
c04	a01	p01
c04	a02	p02
c01	a03	p01
c02	a03	p02
c03	a03	p03
c01	a04	p01
c02	a04	p02

P1

P1 = (Z gdzie kid = ‘c04’)[pid]

pid
p01
p02

A1

A1 = Z[aid,pid] ÷ P1

aid
a03
a04

P2

P2 = Z[pid] – P1

pid
p03

A2

A2 = (Z[aid,pid] ⊳⊲ P2)[aid]

aid
a03

W

W = A1 – A2

aid
a04

<u>Zad. 10</u>. Wypisać identyfikatory klientów, którzy zamówili zarówno produkty p01, jak i p07.

<u>Rozwiązanie algebraiczne</u>

Przyjęte więzy integralności (a dokładnie jednoatrybutowy klucz główny id relacji ZAMÓWIENIA) narzucają rozwiązanie tego zadania przy założeniu, że produkty p01 i p07 były zamawiane w różnych zamówieniach.

R1 = (Z gdzie pid = ‘p01’)[kid]

R2 = (Z gdzie pid = ‘p07’)[kid]

W = R1 ∩ R2

W oczywisty sposób <u>nie jest poprawne</u> rozwiązanie:
(Z gdzie pid = ‘p01’ i pid = ‘p07’)[kid] – daje ono zbiór pusty!

Także <u>niepoprawne</u> jest rozwiązanie
(Z gdzie pid = ‘p01’ lub pid = ‘p07’)[kid]

<u>Rozwiązanie w SQL</u>

```
select kid from ZAMÓWIENIA where pid = ‘p01’
intersect
select kid from ZAMÓWIENIA where pid = ‘p07’;
```

Gdyby więzy integralności zezwalały na zamówienie więcej produktów w jednym zamówieniu (tzn. przy tym samym identyfikatorze zamówienia), to można byłoby szukać np. klientów, którzy w jednym zamówieniu zamawiali obydwa produkty p01 i p07. Wówczas rozwiązanie powinno być następujące:

S1 = (Z gdzie pid = ‘p01’)[kid,zid]

S2 = (Z gdzie pid = ‘p07’)[kid,zid]

W2 = (S1 ∩ S2)[kid]

<u>Rozwiązanie w SQL</u>

```
select kid from
(select kid, zid from ZAMÓWIENIA where pid = ‘p01’
intersect
select kid, zid from ZAMÓWIENIA where pid = ‘p07’);
```

Zad. 11. Wypisać identyfikatory zamówień, nazwy klientów i nazwy agentów dla zamówień z września 2007. Wykorzystać złączenie, a następnie przedstawić je w postaci projekcji odpowiedniej selekcji iloczynu kartezjańskiego.

R = (Z gdzie data >= '01.09.2007' i data<='30.09.2007')[zid,kid,aid]

K.id = K.kid, A.id = A.aid, K.nazwa=K.knazwa, A.nazwa=A.anazwa

T = R ⊳⊲ K[kid,knazwa] ⊳⊲ A[aid,anazwa]

W = T[zid,K.knazwa,A.anazwa]

Relacja T jest równoważna relacji:

T2 = (R × K[kid,nazwa] × A[aid,nazwa]
    gdzie R.kid = K.kid i R.aid = A.aid)[zid,K.nazwa,A.nazwa]

Rozwiązania w SQL za pomocą złączeń

```
select zid, knazwa, anazwa
from ZAMÓWIENIA
natural join (select id as kid, nazwa as knazwa from KLIENCI)
natural join (select id as aid, nazwa as anazwa from AGENCI)
where data >= to_date('01.09.2007', 'DD.MM.YYYY')
   and data <= to_date('30.09.2007', 'DD.MM.YYYY');
```

lub

```
select kid, K.nazwa, A.nazwa
from ZAMÓWIENIA Z
inner join KLIENCI K on Z.kid = K.id
inner join AGENCI A on Z.aid = A.id
where data >= to_date('01.09.2007', 'DD.MM.YYYY')
   and data <= to_date('30.09.2007', 'DD.MM.YYYY');
```

Rozwiązanie w SQL za pomocą iloczynu kartezjańskiego

```
select kid, KLIENCI.nazwa, AGENCI.nazwa
from ZAMÓWIENIA Z, KLIENCI K, AGENCI A
where Z.kid = K.id and Z.aid = A.id
   and Z.data >= to_date('01.09.2007', 'DD.MM.YYYY')
   and Z.data <= to_date('30.09.2007', 'DD.MM.YYYY');
```

Zad. 12. Wypisać wszystkie pary aid dla agentów działających w jednym mieście, dokonując eliminacji par agentów samych z sobą.

A = A1, A = A2

W = ((A1 × A2) gdzie A1.miasto=A2.miasto i A1.id≠A2.id)[A1.id,A2.id]

Rozwiązanie w SQL

select A1.id, A2.id from AGENCI A1, AGENCI A2
where A1.miasto = A2.miasto and A1.id ≠ A2.id;

Przykład do zad. 12

(A1 × A2)[A1.id,A1.miasto,A2.id,A2.miasto]

A1.id	**A1.miasto**	**A2.id**	**A2.miasto**	
a01	Katowice	a01	Katowice	–
a01	Katowice	a02	Katowice	
a01	Katowice	a03	Chorzów	–
a02	Katowice	a01	Katowice	
a02	Katowice	a02	Katowice	–
a02	Katowice	a03	Chorzów	–
a03	Chorzów	a01	Katowice	–
a03	Chorzów	a02	Katowice	–
a03	Chorzów	a03	Chorzów	–

<u>Zad. 13</u>. Wypisać identyfikatory i rabat klientów, mających:
a) największą zniżkę,
b) najmniejszą zniżkę.

<u>Rozwiązanie algebraiczne zad. 13a</u>

K = K1

K = K2

Ta = ((K1 × K2) gdzie K1.rabat >= K2.rabat)[K1.id,K2.id]

Wa = ((Ta÷K2[K2.id]) ⊳⊲ K1) [K1.id, K1.rabat]

<u>Rozwiązanie algebraiczne zad. 13b</u>

K = K1

K = K2

Tb = ((K1 × K2) gdzie K1.rabat <= K2.rabat)[K1.id,K2.id]

Wb = ((Tb÷K2[K2.id]) ⊳⊲ K1) [K1.id, K1.rabat]

<u>Przykład do zad. 13a</u>

K

id	nazwa	miasto	rabat
k01	Jan Nowak	Gliwice	20
k02	Maria Gajdek	Szczecin	30
k03	Anna Kowalska	Warszawa	10
k04	Paweł Sowa	Katowice	30

K = K1, K = K2

K1 × K2

K1.id	K1.nazwa	K1.miasto	K1.rabat	K2.id	K2.nazwa	K2.miasto	K2.rabat
k01	Jan Nowak	Gliwice	20	k01	Jan Nowak	Gliwice	20
k01	Jan Nowak	Gliwice	20	k02	Maria Gajdek	Szczecin	30
k01	Jan Nowak	Gliwice	20	k03	Anna Kowalska	Warszawa	10
k01	Jan Nowak	Gliwice	20	k04	Paweł Sowa	Katowice	30
k02	Maria Gajdek	Szczecin	30	k01	Jan Nowak	Gliwice	20
k02	Maria Gajdek	Szczecin	30	k02	Maria Gajdek	Szczecin	30
k02	Maria Gajdek	Szczecin	30	k03	Anna Kowalska	Warszawa	10
k02	Maria Gajdek	Szczecin	30	k04	Paweł Sowa	Katowice	30
k03	Anna Kowalska	Warszawa	10	k01	Jan Nowak	Gliwice	20
k03	Anna Kowalska	Warszawa	10	k02	Maria Gajdek	Szczecin	30
k03	Anna Kowalska	Warszawa	10	k03	Anna Kowalska	Warszawa	10
k03	Anna Kowalska	Warszawa	10	k04	Paweł Sowa	Katowice	30
k04	Paweł Sowa	Katowice	30	k01	Jan Nowak	Gliwice	20
k04	Paweł Sowa	Katowice	30	k02	Maria Gajdek	Szczecin	30
k04	Paweł Sowa	Katowice	30	k03	Anna Kowalska	Warszawa	10
k04	Paweł Sowa	Katowice	30	k04	Paweł Sowa	Katowice	30

Na żółto zostały zaznaczone krotki, spełniające kryterium K1.rabat >= K2.rabat .

(K1 × K2) gdzie K1.rabat >= K2.rabat

K1.id	K1.nazwa	K1.miasto	K1.rabat	K2.id	K2.nazwa	K2.miasto	K2.rabat
k01	Jan Nowak	Gliwice	20	k01	Jan Nowak	Gliwice	20
k01	Jan Nowak	Gliwice	20	k03	Anna Kowalska	Warszawa	10
k02	Maria Gajdek	Szczecin	30	k01	Jan Nowak	Gliwice	20
k02	Maria Gajdek	Szczecin	30	k02	Maria Gajdek	Szczecin	30
k02	Maria Gajdek	Szczecin	30	k03	Anna Kowalska	Warszawa	10
k02	Maria Gajdek	Szczecin	30	k04	Paweł Sowa	Katowice	30
k03	Anna Kowalska	Warszawa	10	k03	Anna Kowalska	Warszawa	10
k04	Paweł Sowa	Katowice	30	k01	Jan Nowak	Gliwice	20
k04	Paweł Sowa	Katowice	30	k02	Maria Gajdek	Szczecin	30
k04	Paweł Sowa	Katowice	30	k03	Anna Kowalska	Warszawa	10
k04	Paweł Sowa	Katowice	30	k04	Paweł Sowa	Katowice	30

Ta = ((K1 × K2) gdzie K1.rabat >= K2.rabat)[K1.id,K2.id]

K1.id	K2.id
k01	k01
k01	k03
k02	k01
k02	k02
k02	k03
k02	k04
k03	k03
k04	k01
k04	k02
k04	k03
k04	k04

K2[K2.id]

K2.id
k01
k02
k03
k04

Ta÷K2[K2.id]

K1.id
k02
k04

K1

K1.id	K1.nazwa	K1.miasto	K1.rabat
k01	Jan Nowak	Gliwice	20
k02	Maria Gajdek	Szczecin	30
k03	Anna Kowalska	Warszawa	10
k04	Paweł Sowa	Katowice	30

(Ta÷K2[K2.id]) ⊳⊲ K1

K1.id	K1.nazwa	K1.miasto	K1.rabat
k02	Maria Gajdek	Szczecin	30
k04	Paweł Sowa	Katowice	30

Wa = ((Ta÷K2[K2.id]) ⊳⊲ K1) [K1.id, K1.rabat]

K1.id	K1.rabat
k02	30
k04	30