4. Релационна алгебра

Релационните езиците за работа с БД се делят на няколко типа в зависимост от теоретичната си основа.

- Релационна алгебра

Пример за реализиран език от този тип е ISBL (Information System Base Language) в PRTV на IBM.

- Релационно смятане

Базира се на предикатното смятане от математическата логика. Първият език от този тип е предложен от Код и е наречен ALPHA, но никога не е реализиран. По-късно е разработен и реализиран QUEL в INGRES на Университета в Беркли.

- Език SQL (Structured Query Language)

Притежава елементи на релационното смятане и релационната алгебра.

Всички релационни езици, колкото и да са различни, имат една обща черта. Всеки оператор действа не върху един, а върху много редове и резултатът е множество редове (често е релация). Освен това всички релационни езици са еднакво мощни, т.е. всяка операция, която може да се изрази на един език може да бъде изразена и в другите, което се нарича релационна пълнота.

Ще започнем разглеждането на релационните езици с релационната алгебра, която понастоящем има основно теоретично значение. Началната релационна алгебра, предложена от Код още в първата му статия, включва осем операции. За всяка от тях важи следното: операндите са релации и резултатът е релация. Това свойство се нарича затвореност и от него следва, че резултатът от една операция може да се използва като операнд за друга, т.е. може да се пишат вложени изрази.

Според дефиницията всяка релация се състои от заглавие и тяло. Ако разглеждаме затвореността строго, трябва за всяка операция да определим как се образува заглавието и тялото на резултатната релация. Следователно необходим е набор от правила за наследяване на имената на атрибутите в резултата и специална допълна операция за преименуване на атрибути.

```
R rename A as B [, . . .]
```

Осемте операции на Код са разделени в две групи:

- традиционни операции от теорията на множествата – обединение, сечение, разлика и произведение;
- специални релационни операции – селекция, проекция, съединение и деление.

4.1. Операции от теорията на множествата

Нека `R` и `S` са релации с еднакви заглавия, което означава следното:

- Имат еднакви множества от имена на атрибути и следователно еднаква степен.
- Съответните атрибути (атрибутите с еднакви имена) са определени над едни и същи домени.

Ще казваме, че такива релации са **съвместими по тип**. Ако релациите са почти съвместими по тип, т.е. има само различия в имената на някои атрибути, то може да се приложи операцията `rename`, за да станат напълно съвместими. Изискването за съвместимост се поставя при операциите обединение (`union`), сечение (`intersect`) и разлика (`minus`), защото резултатът трябва да е релация, а не просто какво да е множество.

```
R union S
```

Заглавието на резултата е същото като на `R` и `S`. Тялото е множество от всички редове, които принадлежат на `R` или на `S`.

R intersect S

Заглавието на резултата е същото като на R и S. Тялото е множество от всички редове, които принадлежат на R и на S.

R minus S

Заглавието на резултата е същото като на R и S. Тялото е множество от всички редове, които принадлежат на R и не принадлежат на S.

R times S

Операцията произведение е всъщност разширено декартово произведение, отново заради затвореността. Тук няма изисквания за съвместимост по тип на операндите, дори се изисква имената на атрибутите в R и S да са различни, за да може резултатът да има правилно формирано заглавие. Нека схемите на операндите са: R{A1, A2, . . . , An} и S{B1, B2, . . . , Bm}. Резултатът е релация със:

- заглавие, което е теоретико-множествено обединение на заглавията на R и S, т.е. заглавието е {A1, A2,. . . , An, B1, B2,. . . , Bm}.

- тяло, което е множеството от всички редове, всеки от които е обединение (теоретико-множествено) на ред от R и ред от S, т.е редове са множества от вида
{<A1:a1>,<A2:a2>,. . .,<An:an>,<B1:b1>,<B2:b2>,. . ., <Bm:bm>}
такива, че {<A1:a1>,< A2:a2>,. . ., <An:an>} ∈ R и
{<B1:b1>,<B2:b2>,. . ., <Bm:bm>} ∈ S.

Следователно степента на резултата е сума от степените на операндите, а кардиналното число е произведение от кардиналните им числа.

Операциите union, intersect и times са **комутативни**, т.е изразите
R union S
и
S union R
са еквивалентни.

Същите операции са и **асоциативни**, т.е изразите
(R union S) union T
и
R union (S union T)
са еквивалентни и следователно за удобство може да се записват без скоби.

От четирите операции само сечението не е примитивна операция, защото може да се изрази чрез разлика, т.е. в сила е следното тъждество:
R intersect S ≡ R minus (R minus S)

4.2. Специални релационни операции

Селекция (restrict, select)

Нека X и Y са атрибути на релацията R, определени над един и същи домен и за него операцията за сравнение θ ∈ {=, ≠, <, <=, >, ≥} има смисъл.

R where θ Y

Тази операция се нарича **θ-селекция**. Заглавието на резултата е същото като на R. Тялото е множество от всички редове, които принадлежат на R и за които сравнението X θ Y е истина. Вместо X или Y може да има константа, т.е.,

R where X θ literal

При операцията θ-селекция в where има просто сравнение, но може да се определи по-обща операция селекция, която вече не е примитивна.

R where условие

В where има произволен логически израз, съставен от сравнения от двата вида, логическите операции and, or, not и скоби. Такова условие, приемащо

значение истина или лъжа за всеки отделен ред се нарича restriction predicate. Основание за въвеждане на общата операция селекция ни дава свойството затвореност и следните тъждества:

```
R where c1 and c2 ≡ (R where c1) intersect (R where c2)
R where c1 or c2 ≡ (R where c1) union (R where c2)
R where not c ≡ R minus (R where c)
```

Проекция

Нека X, Y, . . . , Z са атрибути на релацията R.

```
R[X, Y, . . ., Z]
```

Заглавието на резултата е {X, Y,. . ., Z}. Тялото е множество от всички редове {<X:x>,<Y:y>,. . ., <Z:z>}, за всеки от които съществува ред в R със същите значения на атрибутите X, Y, . . . , Z.

Съединение (join)

Има няколко вида операции съединение и всички те са с два операнда. Ще започнем с **естественото съединение.**

```
R join S
```

Нека схемите на операндите са :

R{A1, A2, . . ., An, X} и S{B1, B2, . . ., Bm, X}, т.е. X е общ атрибут (единствен) за операндите и е определен над един и същи домен. Резултатът е релация със:

- заглавие, което е {A1, A2, . . ., An, X, B1, B2, . . ., Bm}
- тяло, което е множеството от всички редове от вида

{<A1:a1>,<A2:a2>,. . .,<An:an>,<X:x>,<B1:b1>,<B2:b2>,. . .,<Bm:bm>}

такива, че {<A1:a1>,<A2:a2>, . . ., <An:an>,<X:x>} ∈ R и

{<B1:b1>,<B2:b2>, . . ., <Bm:bm>,<X:x>} ∈ S

θ-съединение е по-обща операция, при която редове от две релации се съединяват (слепват) на базата на условие различно от равенство на атрибути. Нека R и S нямат общи имена на атрибути, но X ∈ R и Y ∈ S са атрибути, определени над един и същи домен, за който операцията сравнение θ има смисъл.

```
R join S where X θ Y
```

Резултатът е релация със:

- заглавие, което е същото като при операцията произведение, т.е. теоретико-множествено обединение на заглавията на R и S. Следователно степента на резултата е сума от степените на операндите.
- тяло, което е множество от всички редове, които са обединение (теоретико-множествено) на ред от R и ред от S, за които условието X θ Y е истина.

Ако операцията сравнение θ е равенство (=) имаме съединение по равенство (equijoin). Съединението е комутативна и асоциативна операция.

Съединението не е примитивна операция, защото може да се изрази чрез произведение и селекция, т.е. в сила е следното тъждество:

```
R join S where X θ Y ≡ (R times S) where X θ Y
```

Деление

```
R divide by S
```

Нека схемите на операндите са: R{X, Y} и S{Y}, т.е. Y е общ атрибут (единствен) за операндите и е определен над един и същи домен. Резултатът е релация със:

- заглавие, което е {X}

- тяло, което е множество от всички редове {<X:x>}, такива че, съществува ред {<X:x>,<Y:y>} ∈ R за всеки ред {<Y:y>} ∈ S.

Делението също не е примитивна операция, защото може да се изрази чрез проекция, произведение и разлика.

R divide by S ≡ R[X] minus((R[X] times S) minus R)[X]

Примери за релационните операции.

R

A	B
1	aa
2	aa
2	bb
3	bb

S

A	B
1	cc
2	aa

T

C	D
1	cc
2	aa

P

B
aa
bb

R union S

A	B
1	aa
2	aa
2	bb
3	bb
1	cc

R intersect S

A	B
2	aa

R minus S

A	B
1	aa
2	bb
3	bb

R times T

A	B	C	D
1	aa	1	cc
1	aa	2	aa
2	aa	1	cc
2	aa	2	aa
2	bb	1	cc
2	bb	2	aa
3	bb	1	cc
3	bb	2	aa

R where B='aa'

A	B
1	aa
2	aa

R[B]

B
aa
bb

R divide by P

A
2

R join T where A=C

A	B	C	D
1	aa	1	cc
2	aa	2	aa
2	bb	2	aa

R join T where A>C

A	B	C	D
2	aa	1	cc
2	bb	1	cc
3	bb	1	cc
3	bb	2	aa

Примери на релационни изрази, които извличат от БД-служители следните данни:

♦ Пълните данни за служителите, които работят в отдел 1 и получават заплата по-малка от 400.

```
emp where dno=1 and sal < 400
```

♦ Номерата, имената и длъжностите на всички служители.
```
emp[eno, ename, job]
```

♦ Номерата, имената и длъжностите на служителите, които работят в отдел 1 и получават заплата по-малка от 400.
```
(emp where dno=1 and sal < 400)[eno, ename, job]
```

♦ Номерата на отделите, в които работят служители със заплата по-малка от 500.
```
(emp where sal < 500)[dno]
```

♦ Номерата на отделите, в които няма служители.
```
dep[dno] minus emp[dno]
```

♦ Номер, име на служител и име на отдела, в който работи за всички служители (за служителите със заплата по-малка от 400).
```
(emp join dep)[eno, ename, dname]
((emp join dep) where sal <400)[eno, ename, dname]
((emp where sal <400) join dep)[eno, ename, dname]
```

♦ Номерата и имената на отделите, в които няма служители.
```
((dep[dno] minus emp[dno]) join dep)[dno, dname]
```

♦Номер, име на служител, длъжност и наименование на проекта, в който участва служителя за служителите от отдел 1.
```
(((emp where dno=1) join emp_pro) join project)[eno, ename, job, pname]
```

♦ Номерата (и имената и длъжностите) на служителите, които участват във всички проекти.
```
emp_pro[eno, pno] divide by project[pno]
((emp_pro[eno, pno] divide by project[pno]) join emp)[eno, ename, job]
```

♦ Имената на служителите, които не участват в проект с номер 100.
```
((emp[eno] minus (emp_pro where pno=100)[eno]) join emp)[ename]
```

Релационо присвояване

Релационни изрази, включващи тези осем операции, ни дават инструмент за извличане на данни от БД. Те обаче не са достатъчни за да се изразят всички функции над БД. Минималното, което е необходимо още е операцията присвояване.
```
R := S
```

`S` може да е произволен релационен израз, като `R` и `S` трябва да са съвместими по тип. Тази операция е основата за реализиране на функциите по обновяване на БД, а именно добавяне, изтриване и изменение на редове.

Добавяне на редове към релация може да се изрази посредством операцията обединение. Напр. добавяне на нов служител в релацията emp:
```
emp:=emp union {<eno:200>,<ename:'Ivanov'>,<sal:300>,<job:'ac'>,<dno:2>}
```

Изтриване на редове от релация може да се изрази посредством операцията разлика. Напр. изтриване на отдел 6 от релацията dep:
```
dep:= dep minus {<dno:6>, <dname:'xxx'>, <budget:2000>, <mgr:null>}
```

Използването на `union` и `minus` вместо специални оператори insert, delete и update не е много удобно. Затова в една релационна СУБД, използваща релационната алгебра като език, трябва да има по-удобни, явни оператори за обновяване. Например, операторът за добавяне на редове може да изглежда така:

`insert` *`релационен-израз`* `into R`