

Лекція 6. Основні операції реляційної алгебри

Над відношеннями необхідно вміти виконувати операції, що забезпечують побудову найбільш ефективних і максимально коректних запитів до БД із метою визначення області вибірки даних, області відновлення даних (тобто даних для вставки, зміни чи видалення), правил і вимог безпеки, стійкості і цілісності даних при множинному доступі до них тощо.

Теорія реляційної моделі є сьогодні найповніше проробленою, тому що в ній математично чітко визначені такі операції. Ці операції отримали назву операцій *реляційної алгебри* [4].

У 1972 р. *Kodl* визначив 8 операцій реляційної алгебри, розподілених на 2 групи:

1. *Традиційні операції* над множинами, модифіковані з урахуванням того, що операндами є відношення: декартовий добуток, об'єднання, перетинання, різниця.

2. *Спеціальні реляційні операції*: вибірка, проєкція, з'єднання і розподіл.

П'ять з цих восьми операцій — декартовий добуток, проекція, об'єднання, різниця і вибірка — є *базовими* і складають *мінімальний набір*. Це означає, що через них можуть бути визначені будь-які інші, у тому числі і решта з трьох операцій реляційної алгебри. Ці три операції були введені тому, що вони настільки часто використовуються, що мало сенс забезпечити їхню безпосередню підтримку.

Відзначимо властивість, без якої набір математичних операцій не є алгеброю. Це **властивість замкнутості**, яка відповідає третьому принципу концепції типу і стверджує, що результат будь-якої операції має той самий тип, що й операнд(и) цієї операції. Відповідно, *реляційна властивість замкнутості* стверджує, що **результат кожної операції над відношенням** (тобто реляційної операції) **також є відношенням**.

З цієї властивості випливає висновок, що **результат однієї реляційної операції може використовуватися в якості вхідних даних для іншої**, і існує можливість формувати *вкладені вирази*. Отже:

1. *Декартовий добуток*, визначений раніше, трохи модифікується, тому що завдяки властивості замкнутості, результат операції повинен містити кортежі, а не упорядковані пари кортежів. Тому, **результатом декартового добутку чи просто добутку відношень R і S буде відношення, що складається з множини кортежів, утворених конкатенацією (зчепленням) кожного кортежу відношення R з кожним кортежем відношення S :**

$$R \times S = \{r.s \mid r \in R, s \in S\}$$

Приклад

Diagram illustrating the Cartesian product of two sets R and S .

Set R (6 rows, 3 columns):

A	B	C
X	1	a
Y	2	a
Z	3	a
W	4	b
W	5	b
W	6	b

Set S (6 rows, 2 columns):

D	E
10	f
5	e
8	d
4	c
6	b
3	a

The resulting set $R \times S$ (6 rows, 5 columns) is formed by interleaving the columns of R and S :

A	B	C	D	E
x	1	a	10	f
x	1	a	5	e
...
w	4	b	4	c
w	4	b	6	b
...
w	6	b	3	a

Зверніть увагу, що схема результуючого відношення також утворена конкатенацією схем співмножників. Тому основною вимогою цієї операції є відсутність у схемах вихідних відношень атрибутів з однаковими *іменами*.

Дана операція не дуже важлива на практиці, тому що не дає ніякої додаткової інформації порівняно з початковою. Однак вона є однією з базових і лежить в основі дуже широко використовуваної операції з'єднання. В *SQL* добуток не має відповідних команд.

2. *Проекція*. Нехай r — кортеж з відношення R ; $r[M]$ — частина цього кортежу, що містить тільки значення атрибутів, які входять до підмножини M схеми відношення. Тоді **проекцією R на M буде відношення, що складається з кортежів значень тих атрибутів, які входять до множини M :** $R[M] = \{r[M] \mid r \in R\}$

На практиці це означає видалення з відношення R атрибутів, що не входять у множину M , з наступним виключенням з отриманого відношення однакових кортежів. Приклад

R[C]	C	R[A,C]	A	C
	a		x	a
	b		y	a
			z	a
			w	b
			w	b
			w	b

Ця операція також не має прямого аналога в *SQL*, однак її можна виконати, наприклад, за допомогою найпростішого варіанта команди SELECT:

```
SELECT [DISTINCT] C FROM R;
SELECT [DISTINCT] A,C FROM R;
```

Ця команда за замовчуванням не видаляє кортежі, що дублюються. Для цього необхідно після SELECT самостійно додати оператор DISTINCT.

3. *Об'єднання*. Це операція одержання **відношення, складеного з кортежів, що належать або відношенню R , або відношенню S , або обом:** $R \cup S = \{r \mid r \in R \vee r \in S\}$

Очевидно, що обидва вхідних відношення повинні мати *сумісні типи атрибутів*. Приклад

R[B,C] ∪ S	1	a
	2	a
	3	a
	4	b
	5	b
	6	b
	10	f
	4	c

З відношення S необхідно забрати 2 останні рядки.

У даному прикладі не видно, яка буде схема результуючого відношення. Тому багато мов даних, у тому числі, деякі діалекти *SQL*, зокрема, *PostgreSQL* висувають ще більш жорстку вимогу — вихідні відношення повинні мати *однакові схеми*. Тоді і результат матиме таку ж схему.

У *SQL* об'єднання виконує команда UNION:

```
SELECT B, C FROM R UNION SELECT * FROM S;
```

4. *Різниця*. Це операція одержання **відношення, що складається з кортежів відношення**

R , які не входять до відношення S :

$$R - S = \{r \mid r \in R \wedge r \notin S\}$$

Вимоги до атрибутів тут такі ж, що і для операції об'єднання. Приклад:

$$R[B,C] - S$$

1	a
2	a
4	b
5	b

Прямого аналога в *стандартному SQL* немає, однак, більшість *діалектів* мають команду MINUS або SUBTRACT:

SELECT B, C FROM R MINUS SELECT D, E FROM S;

В *PostgreSQL* цю операцію реалізує команда EXCEPT — *виключення*, яка має аналогічний формат.

5. *Вибірка (обмеження)*. Якщо проекція буде відображення значень одного атрибута на інші (одного стовпця таблиці на інші чи на самого себе), то вибірка виконує відображення кортежів, результатом якого є **відношення, що містить підмножину всіх кортежів відношення R , для яких виконується (є правдивою) деяка логічна умова:**

$$G_F(R, c) = \{r \mid r \in R \wedge F(r[M], c), c = const \vee c \in S\}$$

F – це будь-яка функція, частіше, операція порівняння. Тому порівнювані значення повинні бути визначені в тому ж самому домені, а F повинна мати сенс для цього домена. Наприклад, безглуздо порівнювати на *більше* (<) чи *менше* (>) значення атрибута „колір“, хоча деякі СУБД допускають і це (наприклад, залежно від порядку проходження при визначенні).

У *SQL* операція обмеження реалізована у вигляді оператора WHERE.

1) $G_{=}(R[A], 'z')$:

SELECT * FROM R WHERE A = 'z';

A	B	C
z	3	a

2) $G_{\neq}(R[B], S[D])$:

SELECT A, B, C FROM R, S WHERE R.B <> S.D;

A	B	C
x	1	a
y	2	a

У другому прикладі наведений ще один цікавий елемент *SQL* — *префікс імені поля* у вигляді імені таблиці, відокремленого крапкою: ' R .' і ' S .'. Префікси потрібні для уточнення імен полів, якщо вони співпадають (однакові) в різних таблицях. Якщо імена атрибутів різні, префікси можна опустити.

І, нарешті, розглянемо ще одну операцію, що хоч і не відноситься до базових, але використовується настільки часто, що має своє відображення в більшості версій *SQL*: операція *з'єднання*. Для представлення цієї операції відразу розглянемо приклад.

SELECT R.A, R.B, R.C, S.D FROM R, S WHERE R.C = S.E;

A	B	C	D
x	1	a	3
y	2	a	3
z	3	a	3
w	4	b	6
w	5	b	6
w	6	b	6

SELECT A, B, C FROM R WHERE C = E AND B = D;

A	B	C
z	3	A
w	6	B