

# Мова реляційних запитів SQL

## Роль та історія розвитку SQL.

**SQL** (*Structured query language* — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом — це є лише стандарт. Не будучи мовою програмування в тому розумінні, як C або Pascal, SQL може формувати інтерактивні запити або, будучи вбудованою в прикладні програми, виступати в якості інструкцій для керування даними. Стандарт SQL, крім того, містить функції для визначення зміни, перевірки і захисту даних.

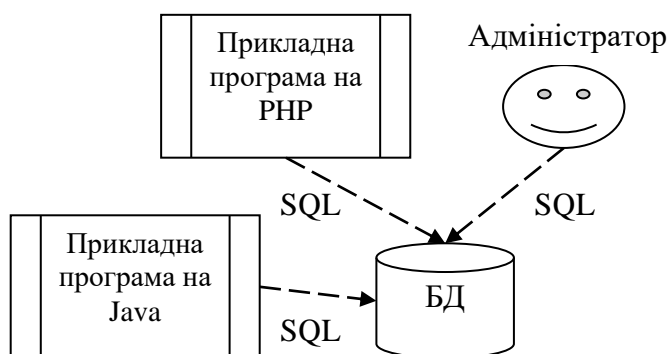


Рисунок 1 - Звернення до бази в реляційній СУБД

## Простий приклад

Простий запит для виведення списку із атрибутами Name, Address, Class із таблиці School у певній базі даних має такий вигляд:

```
SELECT Name, Address, Class
FROM School;
```

## Історія

На початку 1970-х років в одній з дослідницьких лабораторій компанії [IBM](#) була розроблена експериментальна реляційна [СУБД](#) System R, для якої потім була створена спеціальна мова SEQUEL, що дозволяла відносно просто управляти даними в цій СУБД. Аббревіатура SEQUEL розшифровувалася як англ. *Structured English QUery Language* — «структурована англійська мова запитів». Пізніше мова SEQUEL була перейменована в SQL. В 1986 році перший стандарт мови SQL був прийнятий [ANSI](#) (American National Standards Institute). [ANSI](#) — громадська організація, на відміну від міждержавної організації ISO (хоч ANSI є членом ISO), і відповідні стандарти приймаються промисловістю добровільно. Офіційною вимовою стало [ˌes kjuːˈel] — ес-кью-ел.

Метою розробки було створення простої непроцедурної мови, якою зміг би скористатися будь-який користувач, що навіть не має навиків програмування.

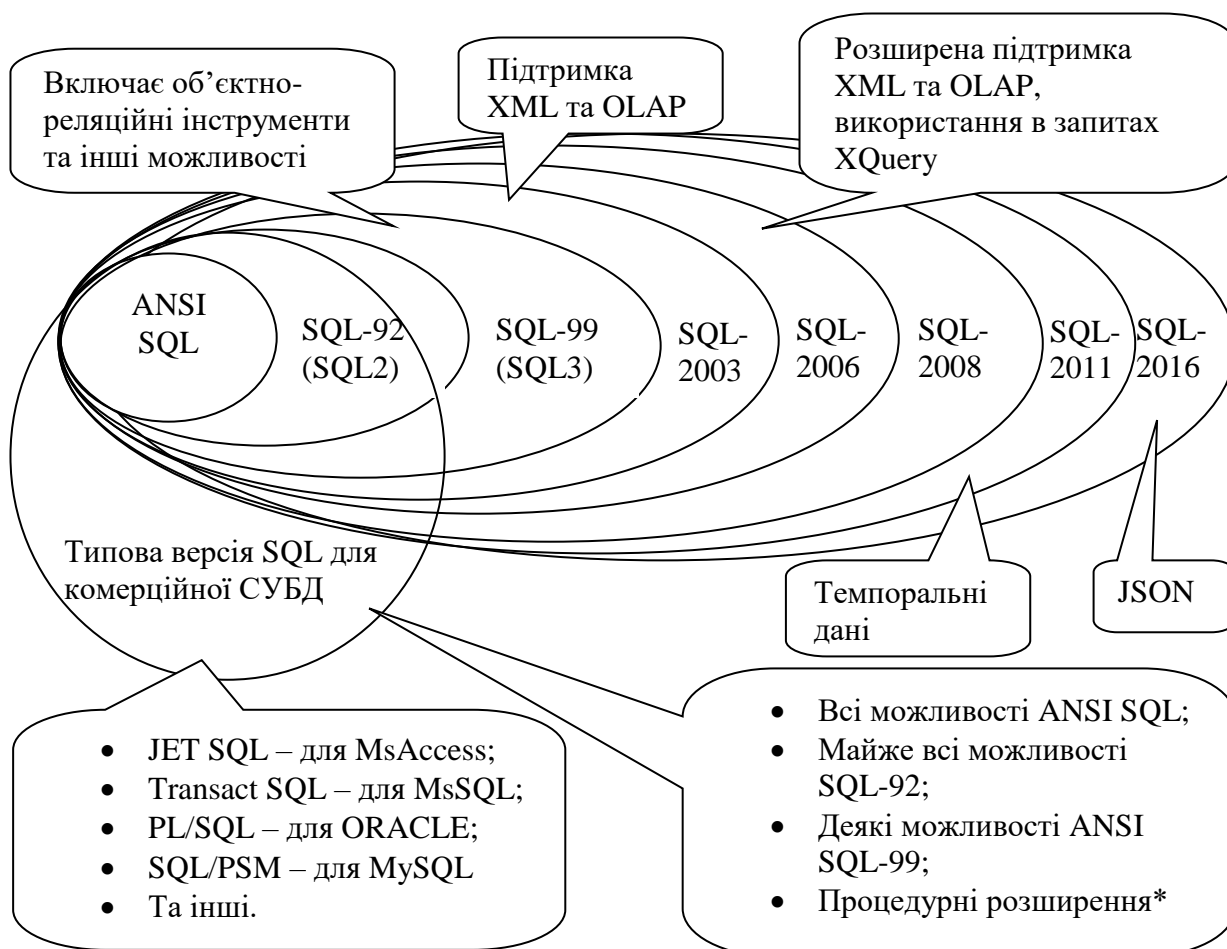
Першими СУБД, що підтримують нову мову *SQL*, стали в 1979 році [Oracle V2](#) для машин [VAX](#) від компанії [Relational Software Inc.](#) (що згодом стала компанією [Oracle](#)) і [System/38](#) від [IBM](#), заснована на [System/R](#).

В 1983 р. ISO (Міжнародна організація зі стандартизації) і ANSI почали розробку стандарту мови *SQL*. Перший офіційний стандарт мови *SQL* був прийнятий [ANSI](#) в 1986 і [ISO](#) в 1987 (так званий SQL-86) і дещо уточнений в 1989 році.

Сьогодні мова *SQL* реалізована у всіх комерційних реляційних СУБД і у багатьох СУБД, які споконвічно ґрунтувалися не на реляційному підході. Всі компанії-виробники реляційних СУБД проголошують відповідність своєї реалізації стандарту *SQL*. У поточних версіях Oracle, Informix, Sybase й Microsoft SQL Server підтримуються досить потужні діалекти *SQL*, хоча реалізація в багатьох конструкціях дуже відрізняється.

## Стандарти і діалекти SQL

Історія версій стандартів-ревізій *SQL*:



\*З 1996р. внесені в стандарт ANSI, а невдовзі ISO

Подробиці на <https://en.wikipedia.org/wiki/SQL>

Рисунок 1-Співвідношення діалектів SQL

## Типи даних в SQL

Набір типів даних у діалекті SQL для комерційної СУБД може значно відрізнятися від стандарту, розширюючи його. Порівняймо стандарт з діалектами SQL від Microsoft та Oracle.

Таблиця 1. Співвідношення типів даних у стандарті SQL та діалектах SQL від Microsoft

	SQL99	Transact SQL	Oracle 10i
Рядки символів	CHAR(n) VARCHAR(n)	He-Unicode: CHAR – до 8000 VARCHAR – до 8000 TEXT – до 2ГБ Unicode: NCHAR – до 4000 NVARCHAR – до 4000 NTEXT – до 1ГБ	CHAR – до 2000 байт VARCHAR2 Змінної довжини, до 4000 байт NCHAR – Unicode, до 2000 байт NVARCHAR2 – Unicode Змінної довжини, до 4000 байт
Рядки бітів	BIT(n) BIT VARYING(n)	BINARY – до 8000 VARBINARY – до 8000 IMAGE – до 2ГБ	-
Логічний тип - 1 байт	BOOLEAN	BIT	-
Цілі числа	SHORTINT INT	TINYINT – 1 байт (0-255) SMALLINT – 2 байти (-32,768 – 32767) INT – 4 байти BIGINT – 8 байт	INT, INTEGER (38 десяткових знаків), and SMALLINT
Унікальне ціле	---	IDENTITY	-
Дійсні числа з плав.кою	FLOAT=REAL DOUBLE PRECISION	REAL – 4 байта FLOAT – 8 байт	BINARY_FLOAT – 5 байт BINARY_DOUBLE – 9 байт DOUBLE PRECISION and FLOAT – 38 десяткових цифр REAL – 18 десяткових цифр
Дійсні числа з фікс.кою	DECIMAL(n,d)	DECIMAL=NUMERIC (+-10**38) SMALLMONEY (+-200тис.) MONEY (+-900трлн)	NUMBER – 38 десяткових цифр DECIMAL – 38 десяткових цифр
Дата	DATE	SMALLDATETIME (1900-2079 pp.) DATE (від 0001-01-01 до 9999-12-31) DATETIME	DATE
Час	TIME	---	---
Глобально унікальний ідентифікатор		UNIQUEIDENTIFIER – 16 байт GUID	ROWID (ім'я файлу даних, № блоку і рядка, Ід. об'єкту БД) UROWID (до 4000 байт)
Поточні дата-час	---	TIMESTAMP	TIMESTAMP

## Прості запити на мові SQL

На цій стадії виходимо з того, що відношення створені. Розглянемо запит до відношення `Movie(title, year, length, inColor, studioName, producerC#)`,

що передбачає вибір даних про всі фільми, зняті на студії „Disney” в 1990 році. На мові SQL запит виглядає так:

```
SELECT *
FROM Movie
WHERE studioName = 'Disney' AND year = 1990;
```

Ця структура відповідає операції селекції (вибору) реляційної алгебри

$$\sigma_c(R).$$

Відношення R  
Умова C

Речення FROM  
Речення WHERE

Порада. Аналізувати і писати запити краще в порядку речень:

- FROM,
- WHERE,
- SELECT.

## Проекція в SQL

При необхідності ми можемо видалити з запиту деякі атрибути, замість „\*” перелічивши явно ті атрибути, що залишаються.

Таким чином, виразу реляційної алгебри:

$$\pi_L(\sigma_C(R))$$

відповідає простий запит

```
SELECT L FROM R WHERE C,
```

де L – список виразів, R – відношення, C – умова.

### *Перейменування атрибутів*

Іноколи треба отримати відношення, в якому назви атрибутів відрізняються від назв атрибутів відношення, названого у реченні FROM. Тоді супроводжуємо ім'я атрибуту службовим словом AS та *псевдонімом* (alias), який стане заголовком відповідного стовпчика результуючої таблиці. Службове слово AS не є обов'язковим; можна лише розділити ім'я та псевдонім пробілом.

Нехай треба видати фільми студії Disney, випущені в 1990 році, змінивши назви стовпчиків з title і length на name і duration.

```
SELECT title AS name, length AS duration
FROM Movie
WHERE studioName = 'Disney' AND year = 1990;
```

Результат:

<i>name</i>	<i>duration</i>
Pretty Woman	119
...	...

### *Обчислювані вирази*

На місці імен атрибутів можуть стояти обчислювані вирази. Тобто простий запит може виконувати операцію розширеної проекції.

*Приклад 1.* Нехай треба отримати результат як у попередньому прикладі, але тривалість фільму треба виразити не в хвилинах, а в годинах.

```
SELECT title AS name, length*0.016667 AS lengthInHours
```

Результат:

<i>name</i>	<i>lengthInHours</i>
Pretty Woman	1.98334
...	...

Виведемо ще одиницю виміру:

```
SELECT title AS name, length*0.016667 AS length, 'hrs.' AS inHours
FROM Movie
WHERE studioName = 'Disney' AND year = 1990;
```

Результат:

<i>name</i>	<i>length</i>	<i>inHours</i>
Pretty Woman	1.98334	hrs.
...	...	...

## Вибір в SQL

Як бачимо, умовні вирази в SQL створюються на основі шести загальноприйнятих операторів порівняння:

- = дорівнює
- <> не дорівнює
- < менше
- > більше
- <= менше або дорівнює
- >= більше або дорівнює

*Чутливість до регістру.* Компілятор SQL розрізняє букви верхнього і нижнього регістрів лише якщо вони набрані у складі рядка, поміщеного в лапки. У SQL-виразі можна писати From у будь-якому регістрі, але в символічній константі 'From' <> 'FROM'.

*Приклад 2.* Наступний запит видає всі чорно-білі фільми, випущені після 1970 року.

```
SELECT title
FROM Movie
WHERE year > 1970 AND NOT inColor;
```

Пряме звертання до InColor можливе, якщо цей атрибут логічного (булевого) типу.

Наступний запит видає назви фільмів, знятих студією MGM та або випущені після 1970 року, або мають тривалість менше 90 хвилин.

```
SELECT title
FROM Movie
WHERE (year > 1970 OR length < 90) AND studioName = 'MGM';
```

Пріоритет операцій, як в інших мовах програмування:

1. NOT,
2. AND,
3. OR.

Тому дужки суттєві, оскільки без них вираз виконувався би інакше.

## Порівняння рядків

Два рядки вважаються рівними, якщо вони складаються з однакових символів, які слідуєть в одному порядку. Рядок з кількох символів може зберігатись у різних типах даних (як-от CHAR і VARCHAR), але однакові по змісту рядки будуть рівні між собою.

Операторами порівняння  $>$ ,  $<$ ,  $>=$ ,  $<=$  рядки порівнюються в лексикографічному (алфавітному) порядку. Алгоритм порівняння рядків  $a_1a_2...a_n$  та  $b_1b_2...b_m$  зрозумілий:

Якщо  $a_1a_2...a_n$  "менше"  $b_1b_2...b_m$ , то

- або  $a_1 < b_1$ ,
- або  $a_1 = b_1$  та  $a_2 < b_2$ ,
- або  $a_1 = b_1$  та  $a_2 = b_2$ , та  $a_3 < b_3$ ,
- і т.д.

Умова  $a_1a_2...a_n < b_1b_2...b_m$  виконується також якщо  $n < m$  та  $a_1a_2...a_n = b_1b_2...b_n$ , тобто перший рядок є префіксом другого.

Наприклад,

'fodder' < 'foo'

'bar' < 'bargain'

'aaa' < 'AAA' ??? У Jet SQL певного порядку не спостерігається.

Asc("A")=97, Asc("a")=65

Asc("F")=70, Asc("f")=102

На сервері Ms SQL Server при сортуванні порядок українського алфавіту підтримується вірно. У MySQL, наприклад, українські букви *і*, *є* йдуть до *а*.

*Порівняння рядків зі зразком*

Вираз порівняння виглядає так:

s LIKE p ,

Де s – рядок, що тестується, а p – зразок, тобто рядок, який може містити необов'язкові символи-шаблони:

Таблиця 2. Символи-шаблони в SQL

Символи шаблону ANSI* та Oracle 10i	Символи шаблону Jet SQL	Символи шаблону Transact SQL	Символи в рядку, що тестується
%	% та *	%	Нульова або більша кількість будь-яких символів
_	_ та ?	_	Один довільний символ
	#		Будь-яка одиночна цифра (0-9)
	[список_символів]	[список_символів]	Одиночний символ, який входить у список
	[!список_символів]	[^список_символів]	Одиночний символ, який не входить у список
	[символ1-символ2]	[символ1-символ2]	Одиночний символ, який належить до вказаного діапазону**

\* Символи шаблону ANSI доступні для використання лише при використанні ядра Microsoft® Jet версії 15.X і програми Microsoft OLE DB Provider для Jet. В іншому разі вони сприймаються як звичайні символи.

\*\*Наприклад: [А-Я] – одна з заглавних букв; [а-яА-Я0-9] – будь-яка буква або цифра.

*Приклад 3.* Нехай ми намагаємося пригадати назву фільму, у якого назва починається зі слова „Star” та наступне слово має 4 букви. Для видачі усіх таких назв напишемо запит:

```
SELECT title
FROM Movie
WHERE title LIKE 'Star ____';
```

У запит будуть уключені фільми, такі, як Star Wars та Star Trek.

*Приклад 4.* Нехай треба відібрати всі англомовні назви, які містять частку 's присвійного відмінку. Слід виконати запит:

```
SELECT title
FROM Movie
WHERE title LIKE '% 's%';
```

Оскільки символ „'” обмежує рядки, він не може представляти себе безпосередньо. Але існує домовленість: два апострофа підряд трактуються як апостроф як такий, а не як символ обмеження рядка. Тому підрядок 's представляє шуканий рядок 's. Будуть знайдені такі назви, як „Logan's Run” або “Alice's Restaurant”.

## Значення дати і часу в SQL

Хоча дата припускає написання в різних форматах, у стандарті SQL константа дати задається певним чином:

DATE '<рік 4 знаки>.<місяць 2 знаки>.<день 2 знаки>'

Наприклад, DATE '2004-11-21' – це 21 листопада 2004 року.

Схожа схема використовується для позначення часу:

TIME '<Години по 24-годинній шкалі>.<Хвилини>.<Секунди при необхідності з десятковими знаками після крапки>'

Приклади:

TIME '10:05:03.1415' – час, до якого студенти звільняють аудиторію після лекції.

Можливо також задати зміщення відносно часу нульового меридіану (GMT – Greenwich Mean Time):

TIME '12:00:00-8:00' – полудень на Тихоокеанському узбережжі США і Канади.

TIME '12:00:00+0:00' – полудень в Лондоні.

TIME '12:00:00+2:00' – полудень в Києві, Тель-Авіві і Преторії.

Для отримання дати-часу як єдиного значення використовується формат:

TIMESTAMP '<дата> <час>'.  
'

Наприклад,

TIMESTAMP '2010-12-16 10:05:00' – дата-час завершення першої пари.

Різні діалекти SQL можуть мати свої формати дати. Ось дата написання цієї лекції (5 грудня 2010р.):

#12/5/2010# в форматі Jet SQL (припустимий будь-який рік від 100 до 9999),

'20101205' в форматі Transact SQL.

Дати і час у будь-якому разі порівнюються у хронологічному порядку.

## Значення NULL і операції з ним

Спеціальне значення NULL означає одне з трьох:

- Значення невідоме,
- Або значення в конкретній ситуації не має сенсу,
- Або значення не підлягає розголосу.

Уявімо відношення з даними кіноакторів

MovieStar(name, address, spouse) (spouse – чоловік або дружина)

Для неодружених артистів у полі spouse має стояти NULL, тому що ніяке інше значення не має сенсу.

Якщо у відношенні Studio(Name, Address, PresidentName) у студії тимчасово нема президента, то атрибут PresidentName цього відношення має бути заповненим NULL.

При вставці кортежів у відношення замість незаданого значення атрибуту присвоюється значення по замовчанню, яке може бути NULL.

При тестуванні виразів у реченні WHERE слід бути готовим до того, що деякі з компонент, що тестуються, будуть дорівнювати NULL. Існують 2 правила обчислень з NULL:

1. Якщо NULL є аргументом бінарного арифметичного оператора поруч з будь-яким іншим оператором, включаючи NULL, як-от + або - , результатом також буде NULL.
2. При співставленні (> або =) NULL з будь-яким іншим значенням, включаючи NULL,
  - a. У стандарті SQL отримуємо значення UNKNOWN. Ms SQL також передбачає значення UNKNOWN.
  - b. У Jet SQL отримуємо значення NULL.

Таким чином, NULL не є константа, безпосередньо використовувати його в виразах не можна. Наприклад,  
 $0 * \text{NULL}$  повертає NULL,  
 $\text{NULL} - \text{NULL}$  повертає NULL.

Для перевірки, чи дорівнює x значенню NULL, передбачена спеціальна конструкція  $x \text{ IS NULL}$  або  $x \text{ IS NOT NULL}$ .

### Значення UNKNOWN і операції з ним

Якщо розглядати

- FALSE як 0,
- TRUE як 1,
- UNKNOWN як  $\frac{1}{2}$ ,

то можна сформулювати 3 правила трьохзначної логіки:

1. Результатом операції  $x \text{ AND } y$  є мінімальне з значень x, y. Зокрема,  $\text{FALSE AND UNKNOWN} = \text{FALSE}$ .
2. Результатом операції  $x \text{ OR } y$  є максимальне з значень x, y. Зокрема,  $\text{TRUE OR UNKNOWN} = \text{TRUE}$ .
3. Операція заперечення NOT x дає результат 1-x, зокрема  $\text{NOT UNKNOWN} = \text{UNKNOWN}$ .

Таким чином, має місце таблиця трьохзначної логіки

*Таблиця 3. Операції трьохзначної логіки*



$x$	$y$	$x \text{ AND } y$	$x \text{ OR } y$	$\text{NOT } x$
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	UNKNOWN	FALSE	UNKNOWN	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

При тестуванні речення WHERE кортежі, які повертають FALSE або UNKNOWN, відкидаються.

*Приклад 5.* Розглянемо запит до БД фільмів

```
SELECT *
FROM Movie
WHERE length <= 120 OR length > 120;
```

На перший погляд, у підсумкове відношення мають бути включені всі кортежі. Але якщо length містить значення NULL, такі кортежі не попадають у підсумкове відношення, оскільки UNKNOWN OR UNKNOWN = UNKNOWN.

В Oracle PL-SQL функції UNKNOWN виконує NULL.

## **Запити до декількох відношень**

### **Декартів добуток і з'єднання відношень у SQL**

Нехай необхідно визначити ім'я продюсера фільму "Star Wars" з двох відношень бази фільмів:

```
Movie(title, year, length, inColor, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

producerC# і cert# - сертифікаційний № продюсера, семантика цих атрибутів однакова.

В SQL ми виконуємо пошук за один етап, адресуючи відношення Movie та MovieExec одночасно:

```
SELECT name
FROM Movie, MovieExec
WHERE title = 'Star Wars' AND producerC# = cert#;
```

У разі відсутності речення WHERE ми отримуємо Декартів добуток відношень.

Пари Декартового добутку відбираються в результат, якщо компонент title відношення Movies містить значення 'Star Wars', а компонент producerC# - число, що співпадає з cert# у MovieExec. Останньому відповідає значення атрибуту name 'George Lukas'. Результат запиту – один рядок 'George Lukas'.

## Розрізнення одноіменних атрибутів кількох відношень

Якщо з'єднуються відношення з одноіменними атрибутами, для таких атрибутів використовуються „повні” імена атрибутів, у яких є префікс – ім'я відношення, після нього крапка. Приклад: нехай є 2 відношення

```
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, netWorth)
```

Нехай треба знайти інформацію про акторів та керівників індустрії кіно, які мають однакові адреси. Для цього складемо запит:

```
SELECT MovieStar.name, MovieExec.name
FROM MovieStar, MovieExec
WHERE MovieStar.address = MovieExec.address;
```

Його результат:

<i>MovieStar.name</i>	<i>MovieExec.name</i>
Jane Fonda	Ted Turner
...	...

Конструкцію „повного” імені атрибута можна застосовувати і в разі якщо нема співпадання імен відношень.

## Змінні кортежів та псевдоніми відношень

Інколи виникає необхідність з'єднання кількох кортежів одного відношення. SQL дозволяє надати кожному екземпляру імені відношення R у реченні FROM унікального *псевдоніму* (alias), або *змінної кортежу* (tuple variable). Ім'я відношення та псевдонім розділяються необов'язковим службовим словом AS.

Якщо псевдонім визначений у реченні FROM, він може бути використаний замість імені відношення в інших реченнях запиту, як-от SELECT, WHERE, для звернення до (в т.ч. одноіменних) атрибутів.

*Приклад 6.* Нехай треба відшукати інформацію про двох акторів, що мають одну адресу.

```
SELECT Star1.name, Star2.name
FROM MovieStar Star1, MovieStar Star2
WHERE Star1.address = Star2.address
      AND Star1.name < Star2.name;
```

Речення FROM містить визначення двох псевдонімів, Star1 та Star2.

Без умови Star1.name < Star2.name, псевдоніми Star1 та Star2 могли би вказувати на той самий кортеж.

Якщо замість оператора < (або >) застосувати <>, це тягне відбір узаємно „зворотніх” пар, як-от

<i>Star1.name</i>	<i>Star2.name</i>
Alec Baldwin	Kim Basinger
Kim Basinger	Alec Baldwin
...	...

## Об'єднання запитів

Нехай треба відібрати назви і дати випуску фільмів, які згадуються в будь-якому з двох відношень БД фільмів.

```
Movie(title, year, length, inColor, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
```

```
SELECT title, year FROM Movie
UNION
(SELECT movieTitle AS title, movieYear AS year FROM StarsIn)
```

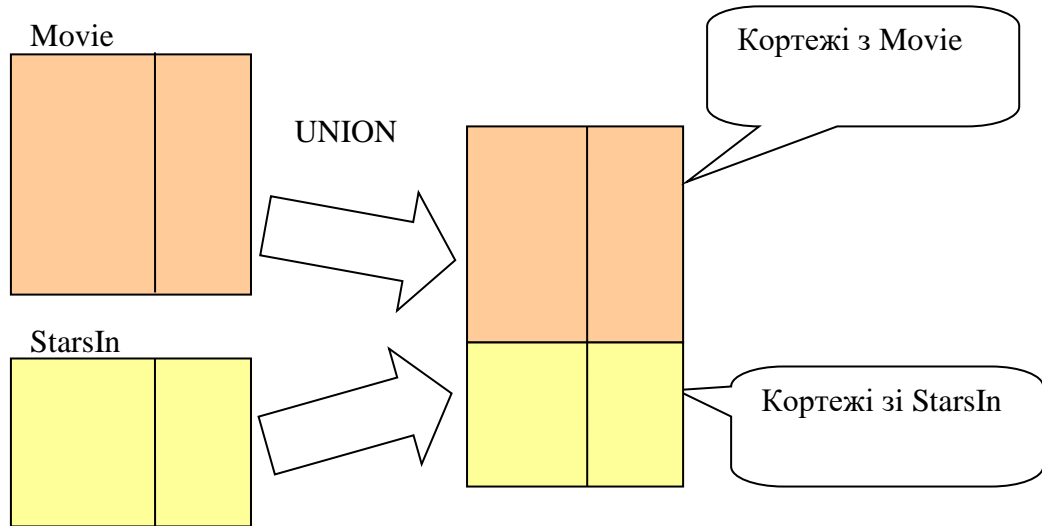


Рисунок 3. Об'єднання запитів

При об'єднанні ми просто дописуємо друге відношення в кінець першого. Оскільки імена атрибутів не співпадають, вони перейменовуються примусово. Взагалі схеми об'єднаних відношень мають співпадати. Але в Transact SQL за основу береться перше відношення, і треба лише, щоби кількість і порядок полів співпадали та їхні типи були сумісні.

UNION залишає лише один з однакових кортежів (створює множину кортежів), UNION ALL залишає дублікати (створює мультимножину).

Кількість UNION в одному реченні не обмежується, але деє біля 7 штук можуть викликати аварійне повідомлення (особливо в Access): „Надто складний запит”.

На жаль, оператори EXCEPT та INTERSECT не вдалось задіяти в Transact SQL, хоча вони згадуються в переліку ключових слів.

## Мова SQL. Підзапити та вирази з'єднання

### Підзапити

Мова SQL дозволяє використовувати один запит як допоміжний для обчислень результатів іншого. Запит, який є частиною більш „крупного” запиту, називають *підзапитом* (subquery). **Взаємна вкладеність запитів не обмежується.** Крім UNION, існують наступні способи використання підзапитів:

1. Підзапит повертає скалярне значення, яке порівнюється з іншим значенням у реченні WHERE.
2. Підзапит повертає відношення, яке використовується в реченні WHERE саме як відношення.
3. Підзапит оперує відношеннями в реченні FROM.

## Підзапити для обчислення скалярних значень

Якщо запит повертає скалярне значення, він може бути використаний як підзапит, поміщений у круглі дужки і розташований там, де можна задати константу або ім'я атрибуту.

*Приклад 7.* Нехай як в одному з попередніх прикладів знову необхідно визначити ім'я продюсера фільму “Star Wars” з двох відношень бази фільмів:

```
Movie(title, year, length, inColor, studioName, producerC#)
MovieExec(name, address, cert#, netWorth)
```

Продюсер і фільм пов'язані сертифікаційним номером. Нехай підзапит знайде сертифікаційний № продюсера фільму “Star Wars”, а „основний” запит – його ім'я.

```
1)  SELECT name
2)  FROM MovieExec
3)  WHERE cert# =
4)      (SELECT producerC#
5)      FROM Movie
6)      WHERE title = 'Star Wars'
      );
```

Можна вважати, що підзапит у рядках 4-6 поверне відношення з одним атрибутом і одним кортежем, як-от ‘12345’. Якщо підзапит повернув би кілька значень або не повернув би нічого, **виникла би помилка часу виконання (runtime error)**.

Після виконання підзапиту виконується „основний” запит, приблизно такий:

```
SELECT name
FROM MovieExec
WHERE cert# = 12345;
```

Його результатом має бути рядок ‘George Lucas’.

## Умови рівня відношення

Нехай підзапит R повертає множину кортежів з одним атрибутом (для оператора EXISTS це неважливо). Існують наступні оператори, які можна застосувати до відношення R.

*Таблиця 4. Оператори умов рівня відношення*

№	Умова	Дорівнює True, якщо і лише якщо	Еквівалентний вираз
1	EXISTS R	R не пусто	
2	s IN R	s дорівнює одному з значень у R	
3	s NOT IN R	s не дорівнює жодному зі значень у R	
4	s > ALL R	s більше будь-якого значення в R	
5	В попередньому виразі оператори порівняння можуть бути: =, >=, <, <=, <>	Відповідно змінюється зміст умови	$s <> ALL R \equiv s NOT IN R$
6	s > ANY R	s більше бодай одного значення в R	
7	В попередньому виразі оператори порівняння можуть бути: =, >=, <, <=, <>	Відповідно змінюється зміст умови	$s = ANY R \equiv s IN R$

Крім того, до всіх цих операторів може застосовуватись NOT.

Наприклад: нехай по Цеху механізації треба видати дорожні листи, по яких працювали трактори (нехай у тракторів це слово є в назві). Для MsSQL і Oracle:

```
SELECT Dpd, Npd FROM vPd
WHERE MachId IN (SELECT Id FROM CIMach WHERE Name LIKE "Трактор%")
```

### Умови рівня кортежу

Нагадаємо, що кортеж у SQL – це список скалярних значень у круглих дужках, як-от (123, 'foo') – компонентами кортежу є константи,

(name, address, netWorth) - компонентами кортежу є атрибути,

(name, address, 10000000) – мішаний кортеж у якому і константи, і атрибути.

Якщо кортеж *t* має компоненти, що відповідають атрибутам відношення *R*, розташованими в однаковому порядку, припустимо співставляти *t* та *R* за допомогою операторів EXISTS, IN, ALL, ANY, розглянутих вище. Наприклад:

*t* IN *R*                      у *R* існує хоча б один кортеж, що співпадає з *t*

*t* <> ANY *R*                у *R* існує хоча б один кортеж, відмінний від *t*

*Приклад 8.* Пошук імен продюсерів кінофільмів, у яких знімався конкретний актор.

Реляційна схема БД наступна:

Movie(title, year, length, inColor, studioName, producerC#)

StarsIn(movieTitle, movieYear, starName)

MovieExec(name, address, cert#, netWorth),

а ось запит, який містить 2 вкладені підзапити:

1) SELECT name		
2) FROM MovieExec		Імена продюсерів
3) WHERE cert# IN		
4) (SELECT producerC#		
5) FROM Movie		Номери продюсерів
6) WHERE (title, year) IN		
7) (SELECT movieTitle, movieYear		
8) FROM StarsIn		Фільми актора
9) WHERE starName = 'Harrison Ford'		Актор
)		
);		

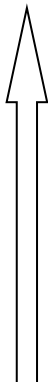


Рисунок 4. Запит з укладеними підзапитами для визначення продюсерів, які співпрацювали з актором

Запит, який містить ієрархію вкладених запитів, слід аналізувати „зсередини назовні”.

- Рядки 7-9. Підзапит відбирає з відношення StarsIn ті кортежі, які містять значення 'Harrison Ford', повертає значення компонентів movieTitle та movieYear.
- Рядки 4-6. Результат – відношення з одним атрибутом, що містить набір сертифікаційних номерів продюсерів фільмів за участі Гарісона Форда.

<i>title</i>	<i>year</i>
Star Wars	1977
Raiders of the Lost Ark	1981
The Fugitive	1993
...	...

Рисунок 3. Екземпляр відношення, що повертає “середній” підзапит прикладу 2.

- „Основний” запит, рядки 1-3. З відношення MovieExec відбираються кортежі, значення компоненту cert# яких містяться в „середньому” запиті. В результуюче відношення відбирається компонента name.

Запит з підзапитами з цього прикладу припускає представлення у вигляді „плаского” запиту „select-from-where”, у реченні FROM якого перелічені всі відношення, згадані в запиті прикладу 2 і його підзапитах. Зв’язки IN замінюються умовами рівності, які розміщуються в реченні WHERE.

```
SELECT name
FROM MovieExec, Movie, StarsIn
WHERE cert# = producerC# AND
      title = movieTitle AND
      year = movieYear AND
      starName = 'Harrison Ford';
```

Рисунок 5. Запит без підзапитів для рішення задачі прикладу 2.

Але у останнього рішення є особливість: якщо продюсер кілька разів співпрацював з актором, у відношенні-результаті буде кілька кортежів для цього продюсера.

### Корельовані підзапити

У простих випадках підзапит виконується один раз, повертає константу, і результат використовується у „зовнішніх” запитах. У більш складних випадках підзапит має виконуватись багатократно, по одному разу для кожного значення, отриманого підзапитом ззовні. Такі підзапити називають *корельовані* (correlated subquery).

*Приклад 9.* Нехай треба знайти кінофільми, назви яких повторюються два рази і більше, а роки випуску різні (оскільки атрибути title, year становлять ключ). БД представлена схемою

Movie(title, year, length, inColor, studioName, producerC#).

Повний текст запиту:

```
1) SELECT title
2) FROM Movie Old
3) WHERE year < ANY
4)     (SELECT year
5)     FROM Movie
6)     WHERE title = Old.title
);
```

Рисунок 6. Запит з корельованим підзапитом

- Підзапит (рядки 4-6) шукає значення року випуску фільму з заданою назвою, хоча ця назва невідома (підзапит схожий на функцію з параметром).
- По мірі переглядання кортежів Movie у процесі обробки зовнішнього запиту (рядки 1-3) кожний з них надає значення Old.title для використання у вкладеному запиті з метою обчислення умови WHERE (рядки 3-6).

Умова в рядку 3 виконується, допоки зміст компонента year змінної кортежу old не дорівнює максимальному з років фільмів з одною назвою. Тому при виконанні „основного” запиту буде повернуто копії певної назви, на одиницю менше загальної кількості фільмів з такою назвою: назва, використана двічі, буде повернута один раз; тричі – два рази і т.д.

У підзапиті є атрибути двох видів:

- Або атрибут належить до одної зі змінних кортежу підзапиту (title),
- Або атрибут належить до запиту вищого рівня (old.title). Для звернення до нього введений псевдонім відношення (змінна кортежу) old.

## Підзапити в реченнях FROM

У цьому випадку текст підзапиту у круглих дужках поміщається у список імен відношень речення FROM. Оскільки підсумкове відношення підзапиту не має імені, треба надати йому псевдонім. Тоді можна посилатись на його атрибути як на атрибути інших відношень зі списку FROM.

*Приклад 10.* Повернімося до прикладу 2: з якими продюсерами співпрацював актор ‘Harrison Ford’?

Реляційна схема БД наступна:

```
Movie(title, year, length, inColor, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
MovieExec(name, address, cert#, netWorth),
```

Нехай є запит, який повертає сертифікаційні номери всіх цих продюсерів. Пов’язавши це відношення з відношенням MovieExec, визначимо імена продюсерів.

```
1)  SELECT name
2)  FROM MovieExec, (SELECT producerC#
3)                               FROM Movie, StarsIn
4)                               WHERE title = movieTitle AND
5)                               year = MovieYear AND
6)                               starName = 'Harrison Ford'
7)                               ) Prod
8)  WHERE cert# = Prod.producerC#;
```

Рисунок 7. Запит з підзапитом у реченні FROM

Підзапит повертає відношення Prod з сертифікаційними номерами продюсерів, які співробітничали з актором ‘Harrison Ford’. Умова рівності сертифікаційних номерів (рядок 8) з’єднує відношення MovieExec та Prod. Увесь запит повертає множину значень атрибуту Name відношення MovieExec, яким відповідають значення producerC# відношення Prod.

## Вирази з'єднання в SQL

SQL дозволяє використовувати різні варіанти оператора з'єднання (join) двох відношень для створення нових відношень:

Український термін	Англійський термін	Відповідник у реляційній алгебрі
Декартів добуток	Cartesian product	$R \times S$
Натуральне з'єднання	natural join	$R \bowtie S$
Тета-з'єднання	theta-join	$R \bowtie_C S$
Зовнішнє з'єднання	outerjoin	$R \Join S$

Оператори можуть використовуватись як у самостійних запитах, так і в підзапитах.

### Декартів добуток - Перехресне з'єднання

Це найпростіша форма з'єднання. Наприклад, для отримання декартового добутку двох відношень

```
Movie(title, year, length, inColor, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
```

можна написати

```
Movie CROSS JOIN StarsIn;
```

і результатом буде відношення з дев'ятьма атрибутами – усі можливі пари кортежів

Movie та StarsIn. З тим же результатом можна написати

```
Movie, StarsIn
```

### Тета-з'єднання

Перехресне з'єднання в чистому вигляді використовується рідко. Ширше застосовується тета-з'єднання

```
R JOIN S ON C
```

Сенс цієї конструкції: обчислення декартового добутку  $R \times S$  та після цього застосування відбору по критерію C, заданому після ON.

*Приклад 11.* Нехай треба з'єднати відношення

```
Movie(title, year, length, inColor, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
```

при умові, що злиттю підлягають лише кортежі, які посилаються на один і той же фільм.

Тобто вони повинні мати однакові значення компонентів

- Title = movieTitle
- Та year = movieYear.

Речення FROM запиту має вигляд:

```
Movie JOIN StarsIN ON
```

```
title = movieTitle AND year = movieYear;
```

Результат – також відношення з дев'ятьма атрибутами з початковими іменами. Кожний кортеж Movie з'єднується з деякими кортежами StarsIn, але лише з тими, де співпадають значення назви і року виробництва. В підсумку 2 стовпця будуть зайвими, оскільки містять однакові значення. Для уникнення цього слід вираз з'єднання помістити у речення FROM, а у речення SELECT – бажані атрибути.

```
SELECT title, year, length, inColor, studioName,
       producerC#, starName
```

```
FROM Movie JOIN StarsIN ON
```

```
title = movieTitle AND year = movieYear;
```



## Натуральне з'єднання

Нехай треба розібратись, хто з акторів є одночасно продюсерами або президентами студій.

Треба проаналізувати відношення

```
MovieStar(name, address, gender, birthdate)
```

```
MovieExec(name, address, cert#, netWorth).
```

Відношення з'єднуються по атрибутах з однаковими іменами. Результат матиме вигляд:

```
R(name, address, gender, birthDate, cert#, netWorth).
```

Для його отримання треба виконати запит

```
SELECT * FROM MovieStar NATURAL JOIN MovieExec
```

На жаль, ні в Jet SQL, ні у Transact SQL нема цього оператора, хоча в PL/SQL є. Реалізація такого запиту існуючими засобами Transact SQL:

```
SELECT MovieStar.name AS name, MovieStar.address AS address,
gender, birthDate, cert#, netWorth
FROM MovieStar INNER JOIN MovieExec
ON MovieStar.name=MovieExec.name AND
MovieStar.address=MovieExec.address
```

## Зовнішнє тета-з'єднання

Оператори зовнішнього з'єднання, які ми розглядали як операції реляційної алгебри, призначені для поповнення підсумкового відношення інформацією *висячих кортежів* (dangling tuples). При з'єднанні деякі кортежі одного з відношень R залишаються „висячими”, якщо умова відповідності в реченні ON не виконується для жодного кортежу з другого відношення S.

Розглянемо операцію *зовнішнього тета-з'єднання* (theta-outerjoin). Нехай знову (приклад 5) треба з'єднати відношення

```
Movie(title, year, length, inColor, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
```

при умові співпадання назви і року випуску фільму.

Запит на повне зовнішнє з'єднання:

```
SELECT title, year, starName
FROM Movie FULL OUTER JOIN StarsIn ON title=movieTitle AND
year=movieYear
```

У результат будуть уключені:

- Кортежі Movie для фільмів за участі акторів, згаданих у StarsIn;
- Висячі кортежі Movie для фільмів, відсутніх у StarsIn;
- Висячі кортежі StarsIn для акторів, що не знімалися у фільмах, перелічених у Movie.

Запит на ліве зовнішнє з'єднання:

```
SELECT title, year, starName
FROM Movie LEFT OUTER JOIN StarsIn ON title=movieTitle AND
year=movieYear
```

У результат будуть уключені:

- Кортежі Movie для фільмів за участі акторів, згаданих у StarsIn;
- Висячі кортежі Movie для фільмів, відсутніх у StarsIn.

- ~~Висячі кортежі StarsIn для акторів, що не знімалися у фільмах, перелічених у Movie.~~

Запит на праве зовнішнє з'єднання:

SELECT title, year, starName

FROM Movie RIGHT OUTER JOIN StarsIn ON title=movieTitle AND year=movieYear

У результат будуть уключені:

- Кортежі Movie для фільмів за участі акторів, згаданих у StarsIn;
- ~~Висячі кортежі Movie для фільмів, відсутніх у StarsIn;~~
- Висячі кортежі StarsIn для акторів, що не знімалися у фільмах, перелічених у Movie.

### З'єднання у діалектах Transact SQL та PL/SQL

На жаль, у цих діалектах багато чого відсутнє відносно SQL92.

Таблиця 5. З'єднання у TransactSQL (MsSQL Server), PL/SQL (ORACLE)

<i>Transact SQL</i>	<i>PL/SQL</i>
<del>NATURAL JOIN</del>	NATURAL JOIN
JOIN = INNER JOIN	JOIN = INNER JOIN
INNER JOIN	INNER JOIN
CROSS JOIN WHERE = INNER JOIN ON	CROSS JOIN WHERE = INNER JOIN ON
LEFT JOIN = LEFT OUTER JOIN	LEFT JOIN = LEFT OUTER JOIN Синтаксис декартового добутку та (+) на стороні NULL у реченні WHERE
<del>OUTER JOIN</del>	<del>OUTER JOIN</del>
RIGHT JOIN = RIGHT OUTER JOIN	RIGHT JOIN = RIGHT OUTER JOIN Синтаксис декартового добутку та (+) на стороні NULL у реченні WHERE
FULL JOIN = FULL OUTER JOIN	? Працює як INNER JOIN

Приклад правого зовнішнього з'єднання для Oracle – варіант синтаксису з «+» (хоча «+» вказується для лівого відношення):

```
select      "LOCATIONS"."CITY", "COUNTRIES"."COUNTRY_NAME"

from        "LOCATIONS", "COUNTRIES"

where       "LOCATIONS"."COUNTRY_ID" (+) ="COUNTRIES"."COUNTRY_ID"
```

В результат увійдуть також країни без локацій.

I JOIN, i (+) належать до стандарту SQL92.