Лабораторна робота 7. Реалізація операцій реляційної алгебри з’єднання та об’єднання

Мета:

Ознайомити студентів з теоретичними основами операції *з'єднання* в реляційній алгебрі та її реалізацією через оператор JOIN в мові SQL, а також реалізацію операції об’єднання. Студенти повинні навчитися використовувати різні типи з'єднань (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN) для об’єднання даних з кількох таблиць у базі даних, розуміти відмінності між цими типами та їхні особливості в контексті PostgreSQL, а також навчитися застосовувати оператор UNION. Також метою є набуття навичок аналізу результатів виконання запитів з різними типами з'єднань і застосування їх на практиці для вирішення прикладних завдань.

Завдання:

1. Реалізація операції реляційної алгебри декартовий добуток:

Виконати запит на вибірку даних з двох таблиць використовуючи перехресне з’єднання.

2. Вибірка з використанням внутрішнього з’єднання:

Виконати запити на вибірку даних з двох або більше таблиць, використовуючи оператор INNER JOIN.

3. Вибірка з використанням зовнішнього з’єднання:

Виконати запити на вибірку даних з двох або більше таблиць, які реалізують:

– ліве з’єднання (оператор LEFT OUTER JOIN);

– праве з’єднання (оператор RIGHT OUTER JOIN);

– повне з’єднання (оператор FULL OUTER JOIN).

Обов’язково застосувати псевдоніми таблиць.

4. Вибірка з використанням натурального з’єднання:

Виконати запит на вибірку даних з двох або більше таблиць, використовуючи оператор NATURAL JOIN.

5. Вибірка з використанням з’єднання таблиці самої з собою:

Виконати запит на вибірку даних з будь-якої таблиці, який продемонструє порівняння рядків цієї таблиці.

6. Реалізація операції реляційної алгебри об’єднання:

Виконати запит на об’єднання результатів запитів з двох таблиць за допомогою оператора UNION.

Результат:

Студенти повинні подати SQL-скрипти, що відображають запити на вибірку з декількох таблиць бази даних з використанням операторів JOIN, UNION згідно завдання та предметної області, їх опис, а також звіт з результатами тестування.

Теоретичні відомості до виконання лабораторної роботи

Реалізація операції реляційної алгебри з’єднання

До цього всі наші запити зверталися лише до однієї таблиці. Однак запити можуть також звертатися відразу до кількох таблиць або звертатися до однієї і тієї ж таблиці так, що одночасно будуть оброблятися різні набори рядків. Запити, що звертаються до різних таблиць (або кількох екземплярів однієї таблиці), реалізують операцію з'єднання. Для цього в SQL використовується оператор JOIN.

Тобто, JOIN — це операція реляційної алгебри, яка дозволяє поєднувати рядки з двох таблиць на основі спільних атрибутів (стовпців). З'єднання є фундаментальною операцією в SQL для отримання даних із кількох таблиць.

Типи з'єднань

Перехресне з’єднання CROSS JOIN.

Повертає декартовий добуток таблиць, тобто всі можливі комбінації рядків, а набір її стовпців поєднує у собі стовпці першої таблиці з наступними стовпцями другої таблиці.

Синтаксис:

```
SELECT * FROM table1 CROSS JOIN table2;
```

Кількість рядків в результуючій таблиці буде дорівнювати добутку кількості рядків першої та другої таблиць.

Наприклад, якщо одна таблиця має 5 рядків, інша 10, то результат буде складатися з 50 рядків. Схематично його часто зображують як велику кількість перехресть між двома групами елементів.

Такий вид JOIN застосовується в онлайн-магазинах для виведення всіх можливих пар за вибраними характеристиками одягу (колір та розмір або інші параметри).

В СУБД PostgreSQL допустимий синтаксис реалізації операції декартовий добуток без явного використання CROSS JOIN:

```
SELECT * FROM table1, table2;
```

Але, якщо буде виконуватись з’єднання більше ніж двох таблиць різними типами з’єднань, то такий варіант спрацює некоректно через порядок виконання операцій.

Приклад. Припустимо існує база даних з наступними даними:

Students (Таблиця студентів)

student_id	student_name
1	Іван Петренко
2	Марія Іванова
3	Олена Коваленко

Supervisors (Таблиця викладачів)

supervisor_id	supervisor_name
101	проф. Іванов
102	доц. Сидоренко
103	доц. Шевченко

Theses (Теми кваліфікаційних робіт)

thesis_id	student_id	supervisor_id	thesis_title
1001	1	101	Оптимізація баз даних
1002	2	103	Алгоритми машинного навчання
1003	NULL	102	Квантові обчислення

Необхідно знайти всі можливі варіації керування викладачів студентами, тобто підготувати можливі варіанти пар виконання кваліфікаційних робіт.

```
SELECT student_name, supervisor_name
FROM Students CROSS JOIN Supervisors;
```

Результат:

student_name	supervisor_name
Іван Петренко	проф. Іванов
Іван Петренко	доц. Сидоренко
Іван Петренко	доц. Шевченко
Марія Іванова	проф. Іванов
Марія Іванова	доц. Сидоренко
Марія Іванова	доц. Шевченко
Олена Коваленко	проф. Іванов
Олена Коваленко	доц. Сидоренко
Олена Коваленко	доц. Шевченко

Внутрішнє з’єднання INNER JOIN.

Цей режим об'єднання результатів пошуку в базах даних SQL включається автоматично. Для кожного рядка першої таблиці в результуючій таблиці міститься рядок для кожного рядка другої таблиці, що задовольняє умові з'єднання. Тобто, повертає тільки ті рядки, для яких є відповідність в обох таблицях (рис. 1).

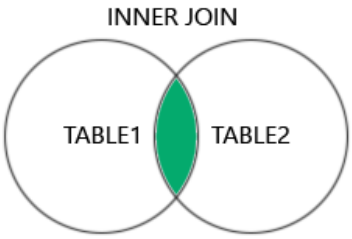

Рисунок 1 – Схематична робота оператора INNER JOIN

Синтаксис:

```
SELECT * FROM table1 INNER JOIN table2 ON table1.column = table2.column;
```

Слово «INNER» необов’язкове. Якщо вказати JOIN без типу, то спрацює саме INNER JOIN. Умова з'єднання вказується в частині ON та визначає найбільш загальну форму умови з'єднання: вказуються вирази логічного типу, подібні до тих, що використовуються в частині WHERE. Пара рядків першої та другої таблиць відповідають один одному, якщо вираз ON повертає для них true.

В СУБД PostgreSQL допустимий наступний синтаксис:

```
SELECT * FROM table1, table2 WHERE table1.column = table2.column;
```

Цей синтаксис з'явився до синтаксису JOIN/ON, прийнятого SQL-92. Таблиці просто перераховуються у реченні FROM, а вираз порівняння додається у пропозицію WHERE. Але він є застарілим і не рекомендується для використання через складнощі з читабельністю та підтримкою великих запитів.

Приклад. Необхідно вивести інформацію про студентів, їх теми кваліфікаційних робіт та керівників цих робіт.

```
SELECT s.student_name, t.thesis_title, sup.supervisor_name
FROM Students s INNER JOIN Theses t ON s.student_id = t.student_id
INNER JOIN Supervisors sup ON t.supervisor_id = sup.supervisor_id;
```

Результат:

student_name	**thesis_title**	**supervisor_name**
Іван Петренко	Оптимізація баз даних	проф. Іванов
Марія Іванова	Алгоритми машинного навчання	доц. Шевченко

В даному прикладі представлено відразу декілька нових моментів: це з’єднання трьох таблиць та псевдоніми таблиць.

Дійсно, в одному запиті можна виконувати з’єднання і більше, ніж двох таблиць. Вхідні таблиці можуть бути результатами іншого з’єднання. Для однозначного визначення порядку з’єднань пропозиції JOIN можна укладати у дужки. Якщо дужки відсутні, пропозиції JOIN обробляються зліва направо. У складних запитах можна комбінувати різні типи JOIN в одному запиті.

Псевдоніми таблиць та стовпців

Таблицям у запиті можна дати тимчасове ім’я, за яким до них можна буде звертатися в рамках запиту. Таке ім'я називається псевдонімом таблиці або аліасом.

Визначити псевдонім таблиці можна використавши наступний синтаксис

… FROM табличне_посилання [AS] псевдонім

Ключове слово AS є необов'язковим. Псевдонімом може бути будь-який ідентифікатор.

Псевдоніми часто застосовуються для призначення коротких ідентифікаторів довгим іменам таблиць з метою покращення читабельності запитів. Псевдонім стає новим ім'ям таблиці у межах поточного запиту, тобто після призначення псевдоніма використовувати вихідне ім’я таблиці у цьому запиті не можна.

Хоча переважно псевдоніми використовуються для зручності, вони бувають необхідні. Наприклад коли таблиця з’єднується сама із собою або якщо в таблицях, що з’єднуються, зустрічаються однакові імена атрибутів. У цьому випадку серверу потрібно конкретизувати з якої таблиці саме потрібно взяти той чи інший атрибут. У разі неоднозначності визначення псевдонімів можна використовувати дужки.

Взагалі хорошим стилем вважається вказувати повні імена стовпців у запиті з’єднання, щоб запит не «поламався», якщо пізніше до таблиці будуть додані стовпці з іменами, що повторюються в іншій таблиці.

Псевдоніми стовпців задаються так само, як і псевдоніми таблиць. Використовуються зазвичай для того, щоб змінити назву стовпця результату запиту або задати назву стовпця, якщо в силу певних операцій вона відсутня.

Зовнішнє з’єднання OUTER JOIN

OUTER JOIN надає інформацію не тільки з внутрішньої частини пошуку, а ще й із зовнішньої. Тобто програма шукає не тільки точкові збіги за обраними раніше критеріями, а дозволяє надати більш «вільні» результати пошуку, що включають елементи з таблиць, які хоч і збігаються з критеріями в SQL-запиті, але не повністю.

OUTER JOIN має наступні типи:

— LEFT OUTER JOIN – ліве зовнішнє з’єднання;
— RIGHT OUTER JOIN – праве зовнішнє з’єднання;
— FULL OUTER JOIN – повне зовнішнє з’єднання;

Слово «OUTER» необов’язкове для всіх типів.

Ліве зовнішнє з’єднання LEFT JOIN.

Спершу виконується внутрішнє з'єднання (INNER JOIN). Потім до результату додаються всі рядки з першої таблиці, яким не відповідають жодні рядки в другій таблиці, а замість значень стовпців другої таблиці вставляються NULL. Таким чином, у результуючій таблиці завжди буде мінімум один рядок для кожного рядка з першої таблиці.

Тобто іншими словами, ліве зовнішнє з’єднання повертає всі рядки з лівої таблиці і відповідні рядки з правої. Якщо в правій таблиці немає відповідності, повертаються NULL (рис. 2).

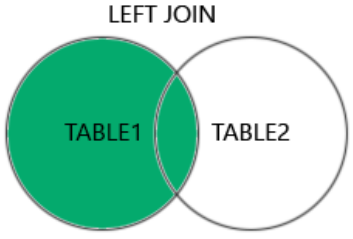

Рисунок 2 – Схематична робота оператора LEFT JOIN

Синтаксис:

SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column = table2.column;

Приклад. Необхідно вивести інформацію про студентів та їх теми кваліфікаційних робіт, а також включити студентів, які ще не мають тем кваліфікаційних робіт. Результат цього запиту зручно використовувати для навантаження студентів, які ще не обрали тему кваліфікаційної роботи.

```
SELECT s.student_name, t.thesis_title
FROM Students s LEFT JOIN Theses t ON s.student_id = t.student_id;
```

Результат:

student_name	thesis_title
Іван Петренко	Оптимізація баз даних
Марія Іванова	Алгоритми машинного навчання
Олена Коваленко	NULL

Праве зовнішнє з'єднання RIGHT OUTER JOIN.

Спершу виконується внутрішнє з'єднання (INNER JOIN). Потім до результату додаються всі рядки з другої таблиці, яким не відповідають жодні рядки в першій таблиці, а замість значень стовпців першої таблиці вставляються NULL. Це з'єднання є зворотним до лівого (LEFT JOIN): у результуючій таблиці завжди буде мінімум один рядок для кожного рядка з другої таблиці.

Тобто іншими словами, праве зовнішнє з'єднання повертає всі рядки з правої таблиці і відповідні рядки з лівої. Якщо в лівій таблиці немає відповідності, повертаються NULL (рис. 3).

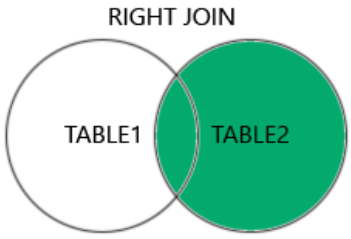

Рисунок 3 – Схематична робота оператора RIGHT JOIN

Синтаксис:

SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.column = table2.column;

Приклад. Необхідно вивести інформацію про студентів та їх теми кваліфікаційних робіт, а також теми, які ще не за ким не закріплені. Результат цього запиту зручно використовувати, коли необхідно запропонувати вільні теми робіт.

```
SELECT s.student_name, t.thesis_title
FROM Students s RIGHT JOIN Theses t
ON s.student_id = t.student_id;
```

Результат:

student_name	thesis_title
Іван Петренко	Оптимізація баз даних
Марія Іванова	Алгоритми машинного навчання
NULL	Квантові обчислення

Повне зовнішнє з’єднання FULL OUTER JOIN.

Спочатку виконується внутрішнє з'єднання. Потім до результату додаються всі рядки з першої таблиці, яким не відповідають жодні рядки в другій таблиці, а замість значень стовпців другої таблиці вставляються NULL. І нарешті, в результат включаються всі рядки з другої таблиці, яким не відповідають жодні рядки в першій таблиці, а замість значень стовпців перщої таблиці вставляються NULL.

Тобто іншими словами, повне зовнішнє з’єднання повертає всі рядки з обох таблиць. Рядки без відповідностей отримують NULL-значення (рис. 4).

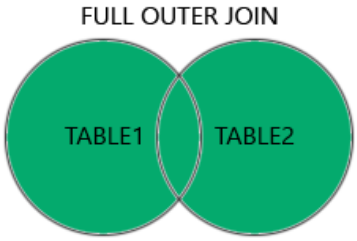

Рисунок 4 – Схематична робота оператора FULL JOIN

Синтаксис:

```
SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.column = table2.column;
```

Приклад. Необхідно вивести інформацію про всіх студентів та всі запропоновані теми кваліфікаційних робіт. Такий запит допоможе виявити студентів, які ще не обрали теми кваліфікаційних робіт, та виявити вільні теми, що зручно для остаточного розподілення тем.

```
SELECT s.student_name, t.thesis_title
FROM Students s FULL JOIN Theses t ON s.student_id = t.student_id;
```

Результат:

student_name	thesis_title
Іван Петренко	Оптимізація баз даних
Марія Іванова	Алгоритми машинного навчання
Олена Коваленко	NULL
NULL	Квантові обчислення

Умова з’єднання у пропозиції ON може також містити вирази, не пов'язані безпосередньо зі з'єднанням. Але потрібно з обережністю використовувати такі конструкції (а краще не використовувати) оскільки обмеження, розміщене в пропозиції ON, обробляється до операції з’єднання, а обмеження WHERE – після. Це не має значення при внутрішніх з'єднаннях, але важливо для зовнішніх, оскільки може видати нерелевантний результат.

PostgreSQL для всіх типів з’єднання дозволяє використовувати USING як альтернативу ON, коли об'єднання відбувається за стовпцями з однаковими іменами:

```
SELECT * FROM table1 JOIN table2 USING (common_column);
```

USING приймає список загальних імен через кому і формує умову з’єднання з рівністю цих значень. Наприклад, запис з’єднання таблиць T1 і T2 з USING (a, b) формує умову

```
… ON T1.a = T2.a AND T1.b = T2.b.
```

Більше того, при виведенні JOIN USING виключаються надмірні стовпці: обидва зіставлені стовпці виводити не потрібно, оскільки вони містять однакові значення. Тоді як JOIN ON видає всі стовпці з першої таблиці, а за ними всі стовпці з другої таблиці, JOIN USING виводить один стовпець для кожної пари (в зазначеному порядку), за ними всі стовпці, що залишилися з першої таблиці і, нарешті, всі стовпці другої таблиця, що залишилися.

Приклад. Необхідно вивести повну інформацію про всіх студентів та всі запропоновані теми кваліфікаційних робіт.

```
SELECT * FROM Students FULL OUTER JOIN Theses USING (student_id);
```

Результат:

student_id	student_name	thesis_id	supervisor_id	thesis_title
1	Іван Петренко	1001	101	Оптимізація баз даних
2	Марія Іванова	1002	103	Алгоритми машинного навчання
3	Олена Коваленко	NULL	NULL	NULL
NULL	NULL	1003	102	Квантові обчислення

Натуральне з'єднання NATURAL JOIN

PostgreSQL підтримує NATURAL JOIN, яке автоматично виконує з'єднання на основі стовпців з однаковими іменами:

Синтаксис:

```
SELECT * FROM table1 NATURAL JOIN table2;
```

NATURAL – скорочена форма USING, вона утворює список USING із усіх імен стовпців, які у обох вхідних таблицях співпадають. Як і з USING, ці стовпці опиняються у вихідній таблиці в єдиному екземплярі. Якщо стовпців з однаковими іменами немає, NATURAL JOIN діє як JOIN ... ON TRUE і видає декартовий добуток рядків.

Приклад. Необхідно вивести ім'я та прізвище студентів, назви їх тем кваліфікаційних робіт та ПІБ керівника.

```
SELECT student_name, thesis_title, supervisor_name
FROM Students NATURAL JOIN Theses NATURAL JOIN Supervisors;
```

Результат:

student_name	thesis_title	supervisor_name
Іван Петренко	Оптимізація баз даних	проф. Іванов
Марія Іванова	Алгоритми машинного навчання	доц. Шевченко

Замкнене з'єднання

З'єднання можна виконати над однією таблицею (з'єднання таблиці самої із собою). Це називається ще замкненим з'єднанням.

Приклад. Потрібно вивести імена та прізвища повних однофамільців, тобто студентів, у яких повністю співпадає ім'я та прізвище.

```
SELECT s1.student_name AS student1, s2.student_name AS student2
FROM Students s1 JOIN Students s2 ON s1.student_name = s2.student_name
WHERE s1.student_id <> s2.student_id;
```

Для демонстрації роботи цього прикладу додамо до таблиці студентів ще один кортеж.

Students (Таблиця студентів)

student_id	student_name
1	Іван Петренко
2	Марія Іванова
3	Олена Коваленко
4	***Іван Петренко***

Результат:

student1	student2
Іван Петренко	Іван Петренко

Реалізація операції реляційної алгебри об'єднання

Операція реляційної алгебри об'єднання в SQL реалізується командою UNION.

UNION по суті додає результати другого запиту до результатів першого (хоча ніякий порядок рядків, що повертаються, при цьому не гарантується).

Щоб можна було обчислити об'єднання результатів двох запитів, ці запити мають бути «сумісними для об'єднання», що означає, що вони повинні мати однакову кількість стовпців і відповідні стовпці мають бути сумісних типів даних.

Загальна структура запиту з оператором UNION

```
SELECT atr1[, atr2, ...] FROM table1[,table2 ...]
UNION [ALL]
SELECT atr1[, atr2, ...] FROM table1[,table2 ...];
```

UNION за замовчуванням прибирає повторення у результуючій таблиці, як це робить оператор DISTINCT. Для відображення із повторенням є необов'язковий параметр ALL.

Приклад. Необхідно об'єднати результати двох різних SQL-запитів, які отримують інформацію про студентів і теми кваліфікаційних робіт та студентів без закріплених тем:

```
SELECT s.student_name, t.thesis_title
FROM Students s JOIN Theses t ON s.student_id = t.student_id
UNION
SELECT s.student_name, 'Немає теми' AS thesis_title
FROM Students s LEFT JOIN Theses t ON s.student_id = t.student_id
WHERE t.thesis_id IS NULL;
```

Результат:

student_name	thesis_title
Іван Петренко	Оптимізація баз даних
Марія Іванова	Алгоритми машинного навчання
Олена Коваленко	Немає теми

Пояснення:

1) Перший запит вибирає студентів, які мають закріплені теми кваліфікаційних робіт.

2) Другий запит вибирає студентів, які не мають тем, і до них застосовується оператор LEFT JOIN.

3) UNION поєднує результати цих двох запитів, усуваючи дублікати.

Структура звіту до лабораторної роботи

Для кожного з запитів представити:

1) словесна постановка задачі, що вирішується;

2) SQL-код рішення;

3) скриншот отриманого результату.

1. Виконати запит на вибірку даних, який повертає:

1.1) декартовий добуток двох таблиць;

2. Виконати запити на вибірку даних, які повертають:

2.1) всі стовпці двох таблиць та рядки, які мають рівність між первинним та зовнішнім ключами;

2.2) деякі стовпці трьох таблиць, рядки яких відповідають тільки внутрішньому з'єднанню.

3. Виконати запити на вибірку даних з застосуванням псевдонімів таблиць, які повертають:

3.1) результат з'єднання двох таблиць з відображенням всіх рядків лівої таблиці;

3.2) результат з'єднання трьох таблиць з відображенням всіх рядків правої таблиці;

3.3) результат з'єднання двох таблиць з відображенням всіх рядків;

3.4) результат повного з'єднання двох таблиць з відображенням тільки тих рядків, в яких числове поле більше заданого значення;

4. Виконати запит на вибірку даних, який повертає:

4.1) повторити виконання пункту 3.1 з використанням натурального з'єднання.

5. Виконати запит на вибірку даних, який повертає:

5.1) рядки таблиці, які мають значення заданого стовпця вище ніж значення цього стовпця в рядку з id=5.

6. Виконати запит на вибірку даних, який повертає:

6.1) результат об'єднання відповідей двох запитів з відображенням всіх рядків. Кожний з запитів має містити умову відбору рядків.

Звіт до лабораторної роботи 7 можна здати онлайн на сайті ДО edu.op.edu.ua до початку вашого заняття.