Заняття 6. Формування багатотабличного SQL-запиту

Зв'язані (корельовані) підзапити

- Основною ознакою зв'язаного (корельованого) підзапиту є те, що він не може бути виконаним самостійно, без зв'язку з основним запитом.
- Формально це реалізується тим, що підзапит посилається на таблицю, яка вказується в основній частині запиту.

- Типовий абстрактний зв'язаний підзапит:

  SELECT A FROM T1 WHERE T1.B =

  (SELECT T2.B FROM T2 WHERE T2.C = T1.C)

  - Цей запит використовує дві таблиці T1 і T2, в яких є стовпці з однаковими іменами B, C, і однаковими типами.

Виконання запиту з корельованим підзапитом

- Спочатку береться увесь перший запис з таблиці T1. Цей запис називається поточним. Значення стовпців для цього запису є доступними і можуть використовуватись у підзапиті.
- Після цього виконується підзапит, який повертає список значень стовпця B таблиці T2 у тих записах, в яких значення стовпця C рівне значенню стовпця C з таблиці T1. Припускаємо, що підзапит повертає єдине значення (оскільки в операторі WHERE основного запиту операція =).
- Тепер виконується оператор WHERE основного запиту. Якщо значення стовпця B поточного запису таблиці T1 рівне значенню, яке вибрав підзапит, то цей запис виділяється зовнішнім запитом і поміщається в результатну таблицю. Якщо умова оператора WHERE основного запиту не виконується, то вибраний запис ігнорується.
- Після цього відбувається перехід на наступний запис таблиці T1. Аналогічно все виконується для кожного запису таблиці T1.

Зв’язані (корельовані) підзапити

- Приклад 1. Знайти інформацію про усіх замовників, що здійснювали замовлення 3 жовтня:

```
SELECT * FROM Customers
WHERE '2009-10-03' IN
(SELECT odate FROM Orders
WHERE Customers.cnum = Orders.cnum);
```

- Приклад 2. (Зв'язані підзапити у фразі HAVING). Просумувати платежі за кожен день, виводячи дати, де сума платежів була б на 2000 більша від максимального платежу за цей день.

```
SELECT odate, SUM(amt)
FROM Orders o1
GROUP BY odate
HAVING SUM(amt) >
(SELECT 2000.00 + MAX(amt)
FROM Orders o2
WHERE o1.odate = o2.odate);
```

Використання предиката EXISTS

- Приклад 3. (Перевірка на непорожній результат). Отримати відомості про замовників, які зробили хоча б одну покупку.

```
SELECT * FROM Customers
WHERE EXISTS
(SELECT DISTINCT onum FROM Orders
WHERE Customers.cnum = Orders.cnum);
```

Внутрішнє з’єднання (INNER JOIN)

- **NATURAL JOIN** – природне з'єднання.
- **JOIN…ON** – з’єднання за умовою.
- **JOIN…USING** – з’єднання за іменами стовпців.

Природне з'єднання (NATURAL JOIN)

- Перевіряються на рівність усі одноіменні стовпці таблиці (з'єднання за рівністю).
- Синтаксис:

```
SELECT Таблиця1.*, Таблиця2.*
FROM Таблиця1 NATURAL JOIN Таблиця2;
```

Умовне з'єднання (JOIN…ON), з'єднання за іменами стовпців (JOIN…USING)

- JOIN…ON використовується, коли потрібно з'єднати за іншими логічними умовами, не обов'язково за рівністю.
- JOIN…USING подібне на природне з'єднання. Відмінність полягає в тому, що можна вказати, які саме стовпці повинні перевірятись.

Внутрішнє з’єднання (INNER JOIN)

- При внутрішньому з'єднанні рядки таблиць, що не співпадають виключаються.
- Приклад 4. Вивести імена продавців і замовників з однакових міст.

```
SELECT Sellers.sname, Sellers.city, Customers.cname
FROM Sellers, Customers
WHERE Sellers.city = Customers.city AND Sellers.snum = Customers.snum;

SELECT s.sname, s.city, c.cname
FROM Sellers s INNER JOIN Customers c
ON s.snum = c.snum WHERE s.city = c.city;
```

- Приклад 5. Знайти усі операції купівлі-продажу, в яких брали участь замовники, які знаходяться у інших містах, ніж продавці, які їх обслуговували.

```
SELECT Orders.onum, Orders.amt, Orders.odate, Customers.cname,
Sellers.sname
FROM Sellers, Customers, Orders
WHERE Customers.city <> Sellers.city
AND Orders.cnum = Customers.cnum
AND Orders.snum = Sellers.snum;
```

- SELECT o.onum, o.amt, o.odate, c.cname, s.sname
  FROM Customers c JOIN Orders o ON c.cnum = o.cnum
  JOIN Sellers s ON o.snum = s.snum
  WHERE c.city <> s.city;

- SELECT o.onum, o.amt, o.odate, c.cname, s.sname
  FROM Customers c JOIN Orders o ON c.cnum = o.cnum
  JOIN Sellers s ON o.snum = s.snum
  AND c.city <> s.city;

З'єднання таблиці з собою

- Приклад 6. Знайти усі пари замовників, які мають однаковий рейтинг.
- Рішення 1 (з дублюванням):

```
SELECT c1.cname, c2.cname, c1.rating
FROM Customers c1 JOIN Customers c2
ON c1.rating = c2.rating;
```

- Рішення 2 (без дублювання):

```
SELECT c1.cname, c2.cname, c1.rating
FROM Customers c1, Customers c2
WHERE c1.rating = c2.rating
AND c1.cname < c2.cname;
```

- Або

```
SELECT c1.cname, c2.cname, c1.rating
FROM Customers c1 JOIN Customers c2
ON c1.rating = c2.rating
AND c1.cname < c2.cname;
```

Перехресне з'єднання (CROSS JOIN)

- Відповідає операції розширеного декартового добутку, тобто операції з'єднання двох таблиць, при якому кожний запис першої таблиці з'єднується з кожним записом другої таблиці.
- У деяких СУБД не використовується (В MySQL використовується).

```
SELECT <список стовпців>
FROM  Таблиця1, Таблиця2;
```

- Або

```
SELECT <список стовпців>
FROM  Таблиця1 CROSS JOIN Таблиця2;
```

Зовнішні з’єднання

- **LEFT OUTER JOIN** – ліве з’єднання.
- **RIGHT OUTER JOIN** – праве з’єднання.
- **FULL JOIN** – повне з’єднання.
- **UNION JOIN** – об’єднане з’єднання.

Відмінність внутрішнього і зовнішнього з'єднання

- При внутрішньому з'єднанні відкидаються усі записи, для яких немає відповідних записів одночасно в обох таблицях.
- При зовнішньому з'єднанні такі невідповідні записи повинні залишатись.

Ліве з'єднання (LEFT OUTER JOIN)

- При лівому зовнішньому з'єднанні невідповідні записи, які є в лівій таблиці (від оператора JOIN), зберігаються в результатній таблиці, а ті, які є в правій таблиці – відкидаються.

- Приклад 7. Переглянути всіх продавців разом із їх операціями купівлі-продажу, які вони обслуговували 3 жовтня 2009 року.

```
SELECT s.sname, o.onum
FROM Sellers s LEFT JOIN Orders o
ON o.odate = '2009-10-03'
AND s.snum = o.snum;
```

sname	onum
Peel	NULL
Serres	3003
Serres	3005
Axelrod	NULL
Motika	3002
Rifkin	3006

- Альтернативний варіант LEFT JOIN:

```
SELECT s.sname, o.onum
FROM Sellers s, Orders o
WHERE o.odate = '2009-10-03'
AND s.snum = o.snum
UNION
SELECT s.sname, NULL
FROM Sellers s
WHERE s.sname NOT IN
(SELECT s.sname
FROM Sellers s, Orders o
WHERE o.odate = '2009-10-03'
AND s.snum = o.snum);
```

sname	onum
Serres	3003
Serres	3005
Motika	3002
Rifkin	3006
Peel	NULL
Axelrod	NULL

Праве з'єднання (RIGHT OUTER JOIN)

- При правому зовнішньому з'єднанні невідповідні записи, які є у правій таблиці, зберігаються в результатній таблиці, а ті, які є в лівій таблиці – відкидаються.

- Приклад 8. Переглянути всіх клієнтів разом з продавцями, які їх обслуговують, а також відобразити продавців які не мають клієнтів.

```
SELECT c.cnum, s.sname
FROM customers c RIGHT JOIN sellers s
ON c.snum = s.snum;
```

cnum	sname
2001	Peel
2006	Peel
2003	Serres
2004	Serres
2002	Axelrod
2007	Motika
2008	Rifkin
NULL	Sally

Повне з’єднання (FULL JOIN)

- Повне з'єднання є комбінацією лівого і правого з'єднань.
- Воно показує всі рядки з обох таблиць: за наявності збігів – з'єднані, в іншому випадку – з NULL-значеннями в стовпцях з іншої таблиці.
- В MySQL немає оператора FULL JOIN!
- Для використання FULL JOIN в MySQL потрібно об'єднати два однакові запити із LEFT JOIN та RIGHT JOIN за допомогою оператора UNION.

Об'єднане з'єднання (UNION JOIN)

- При об'єднаному з'єднанні створюється віртуальна таблиця, яка містить усі стовпці двох вихідних таблиць.
- При цьому стовпці з лівої вихідної таблиці містять усі свої записи, а в тих же записах в стовпцях з правої таблиці містяться значення NULL.
- Аналогічно, стовпці з правої таблиці містять усі свої записи, а ці ж записи з лівої таблиці містять NULL.
- Загальна кількість записів, які містяться в результатній таблиці рівна сумі кількості записів, які є в обох вихідних таблицях.
- Як правило, результат об'єднаного з'єднання розглядається в якості проміжного при виконанні більш складного запиту.
- В MySQL немає оператора UNION JOIN!

T1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4

T2

X	Y
x1	y1
x2	y2
x3	y3

Об’єднане з’єднанняT1 та T2

A	B	C	X	Y
a1	b1	c1		
a2	b2	c2		
a3	b3	c3		
a4	b4	c4		
			x1	y1
			x2	y2
			x3	y3

Теоретико-множинні операції в SQL

- **UNION** – об’єднання наборів записів.
- **INTERSECT** – перетин наборів записів.
- **EXEPT** – віднімання наборів записів.

Об'єднання наборів записів (UNION)

Запит1 UNION Запит2;

- До набору рядків, які виводяться одним запитом додаються рядки, які виводяться другим запитом.
- Об'єднуються рядки двох чи більше таблиць з подібними структурами в одну таблицю.
- При об'єднанні в результатній таблиці залишаються лише різні рядки.
- Щоб зберегти в результатній таблиці усі рядки, необхідно написати UNION ALL.
- Коли використовується об'єднання більш ніж двох запитів, можна використовувати дужки для визначення порядку запитів.

- Приклад 9. Вивести усіх продавців та замовників з Лондона:

```
SELECT snum AS num, sname AS name
  FROM Sellers
  WHERE city = 'London'
UNION
SELECT cnum AS num, cname AS name
  FROM Customers
  WHERE city = 'London'
ORDER BY name DESC;
```

num	name
1001	Peel
1004	Motika
2001	Hoffman
2006	Clemens

Правила використання оператора UNION

При об'єднанні рядків двох таблиць їх стовпці повинні бути сумісними.

- Це означає, що кожний запит повинен вказувати однакову кількість стовпців і в однаковому порядку.
- Типи полів повинні бути теж сумісні.
- Однак, імена відповідних стовпців та їх розміри можуть бути різними.
- Щоб об'єднати рядки таблиць з несумісними (по типу даних) стовпцями, необхідно застосувати функцію перетворення типу даних ***CAST()*.**

Упорядкування об’єднання наборів записів

- Використовується фраза:

  ORDER BY <ім'я стовпця>
- Можна упорядковувати об’єднання за декількома полями, одне всередині іншого, і вказати ASC або DESC для кожного.

Перетин наборів записів (INTERSECT)

Запит1 INTERSECT Запит2;

- Перетин наборів рядків повертає таблицю, рядки якої містяться одночасно у двох наборах.
- При перетині в результатній таблиці залишаються лише різні записи.
- Щоб зберегти в ній усі записи, необхідно написати INTERSECT ALL.
- В MySQL немає оператора INTERSECT!

- Приклад 10. Вивести номери усіх продавців, які обслуговували операції купівлі-продажу і 3 жовтня, і 6.
- Із використанням оператора INTERSECT:

```
SELECT snum FROM Orders
        WHERE odate = '2009-10-03';
INTERSECT ALL
SELECT snum FROM Orders
        WHERE odate = '2009-10-06';
```

- Без використання оператора INTERSECT:

```
SELECT DISTINCT snum FROM Orders
WHERE snum IN
(SELECT snum FROM Orders WHERE odate = '2009-10-03')
AND snum IN
(SELECT snum FROM Orders WHERE odate = '2009-10-06');
```

Віднімання наборів записів (EXEPT)

Запит1 EXEPT Запит2;

- Віднімання наборів рядків повертає таблицю, рядки якої містяться в одному наборі за виключенням тих, які містяться в другому наборі.
- В MySQL немає оператора EXEPT!

- Приклад 11. Вивести номери усіх продавців, які обслуговували операції купівлі-продажу 3 жовтня, але не обслуговували 6 жовтня.
- Без використанням оператора EXEPT:

```
SELECT DISTINCT snum FROM Orders
WHERE snum IN
(SELECT snum FROM Orders WHERE odate = '2009-10-03')
AND snum NOT IN
(SELECT snum FROM Orders WHERE odate = '2009-10-06');
```

Завдання 1. Написати багатотабличні запити для власної бази даних

- Написати 2 запити із корельованими підзапитами.
- Перевірити 1 пару таблиць на непорожні значення зовнішніх ключів (оператори LEFT JOIN або RIGHT JOIN).
- Написати 1 запит із аналогом INTERSECT або EXEPT.
- Написати декілька запитів для внутрішнього з'єднання таблиць (мінімум 4).