Лабораторна робота 5
Об'єднання таблиць

Мета роботи: Навчитися виконувати об'єднання таблиць.

ЗАВДАННЯ

1. Запустити програму MS SQL Management Studio
2. Виконати завдання, перераховані в «Порядку виконання»

ЗМІСТ ЗВІТУ

Найменування і мета роботи
Завдання

ТЕОРЕТИЧНА ЧАСТИНА

Для об'єднання запитів використовується службове слово **UNION**:

1. <запит 1>
2. UNION [ALL]
3. <запит 2>

Пропозиція **UNION** призводить до появи в результуючому наборі всіх рядків кожного із запитів. При цьому, якщо визначено параметр **ALL**, то зберігаються всі дублікати вихідних рядків, в іншому випадку в результуючому наборі присутні тільки унікальні рядки. Зауважимо, що можна пов'язувати разом будь-яке число запитів. Крім того, за допомогою дужок можна задавати порядок об'єднання.

Операція об'єднання може бути виконана тільки при виконанні наступних умов:

- кількість вихідних стовпців кожного із запитів повинно бути однаковим;
- вихідні стовпчики кожного із запитів повинні бути сумісні між собою (в порядку їх слідування) за типами даних;
- в результуючому наборі використовуються імена стовпців, задані в першому запиті;
- пропозиція **ORDER BY** застосовується до результату з'єднання, тому воно може бути зазначено лише в кінці всього складеного запиту.

Знайти номери моделей і ціни ПК і портативних комп'ютерів. Сортування ціни по спадаючій:

```
SELECT model, price
FROM PC
UNION
SELECT model, price
FROM Laptop
ORDER BY price DESC;
```

Знайти тип продукції, номер моделі і ціну ПК і портативних комп'ютерів. Сортування ціни по спадаючій:

```
SELECT Product.type, PC.model, price
FROM PC INNER JOIN Product
ON PC.model = Product.model
UNION
SELECT Product.type, Laptop.model, price
FROM Laptop INNER JOIN Product
ON Laptop.model = Product.model
ORDER BY price DESC;
```

Припустимо у нас є база даних «форум» і ми хочемо дізнатися, які теми, і якими авторами були створені. Для цього найпростіше звернутися до таблиці Теми (topics).

Але, що якщо нам необхідно, щоб у відповіді на запит були ні ідентифікатори авторів, а їх імена? Вкладені запити нам не допоможуть, тому що в кінцевому підсумку вони видають дані з однієї таблиці. А нам треба отримати дані з двох таблиць (Теми і Користувачі) і об'єднати їх в одну. Запити, які дозволяють це зробити, в SQL називаються *Об'єднаннями*.

Синтаксис самого простого об'єднання наступний:

```
SELECT імена_стовбців_таблиці_1, імена_стовбцов_таблиці_2
FROM імя_таблиці_1, імя_таблиці_2;
```

Давайте створимо просте об'єднання:

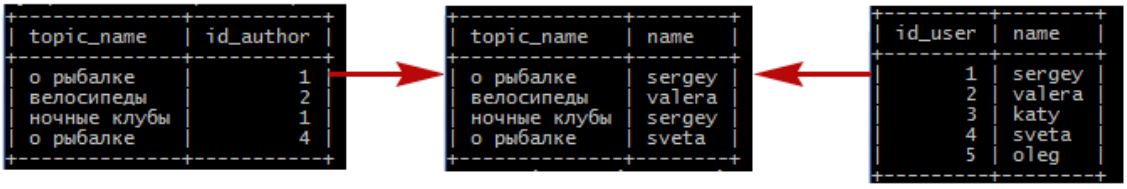
```
SELECT topic_name, name FROM topics, users;
```

Таке об'єднання науково називається декартовим, коли кожному рядку першої таблиці ставиться у відповідність кожен рядок другої таблиці. Можливо, бувають випадки, коли таке об'єднання корисно, але це явно не наш випадок.

Щоб результуюча таблиця виглядала так, як ми хотіли, необхідно вказати умову об'єднання. Ми пов'язуємо наші таблиці за ідентифікатором автора, це і буде нашою умовою. Тобто ми вкажемо в запиті, що необхідно виводити тільки ті рядки, в яких значення поля id_author таблиці topics збігаються зі значеннями поля id_user таблиці users:

```
SELECT topic_name, name
FROM topics, users
WHERE topics.id_author = users.id_user;
```

На схемі буде зрозуміліше:



Тобто ми в запиті зробили таку умову: якщо в обох таблицях є однакові ідентифікатори, то рядки з цим ідентифікатором необхідно об'єднати в одну результуючу рядок.

Зверніть увагу на дві речі:

- Якщо в одній з поєднуваних таблиць є рядок з ідентифікатором, якого немає в інший об'єднаній таблиці, то в результуючій таблиці рядки з таким ідентифікатором не буде. У нашому прикладі є користувач Oleg (id=5), але він не створював теми, тому в результаті запиту його немає.
- При вказівці умови назву стовпця пишеться після назви таблиці, в якій цей стовпець знаходиться (через точку). Це зроблено, щоб уникнути плутанини, адже стовпці в різних таблицях можуть мати однакові назви, і MI SQL може не зрозуміти, про які конкретно стовбці йдеться річ.

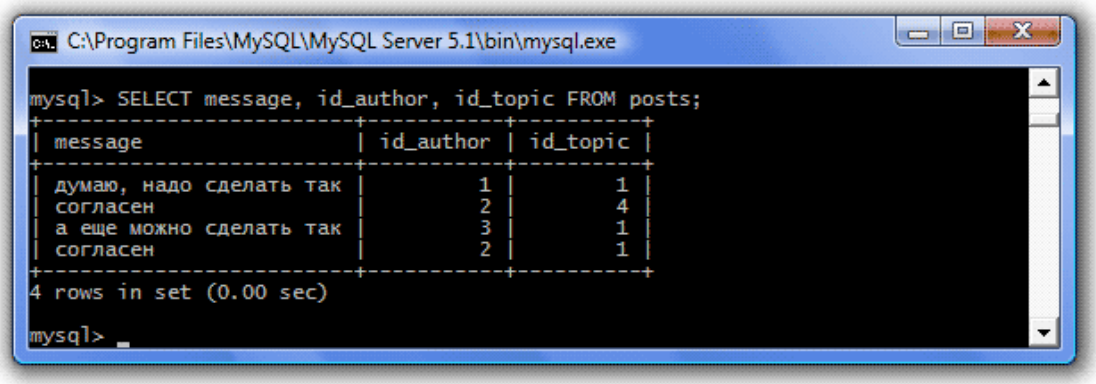
Коректний синтаксис об'єднання з умовою виглядає так:

```
SELECT імя_таблiци_1.імя_стовбца 1 _таблiци_1,
    імя_таблiци_1.імя_стовбца 2 _таблiци_1,
    імя_таблiци_2.імя_стовбца 1 _таблiци_2,
    імя_таблiци_2.імя_стовбца 2 _таблiци_2
FROM
    імя_таблiци_1, імя_таблiци_2
WHERE
    імя_таблiци_1.імя_ стовпчика _по_которому_об'єднаем =
    імя_таблiци_2.імя_ стовпчика _по_которому_об'єднаем;
```

Якщо ім'я стовпця унікально, то назву таблиці можна опустити (як ми робили в прикладі).

Як ви розумієте, об'єднання дають можливість вибирати будь-яку інформацію з будь-яких таблиць, причому об'єднуються таблиць може бути і три, і чотири, та й умова для об'єднання може бути не одне.

Для прикладу давайте створимо запит, який покаже нам всі повідомлення, до яких тем вони відносяться і авторів цих повідомлень. Звичайно, вся ця інформація зберігається в таблиці Повідомлення (posts):



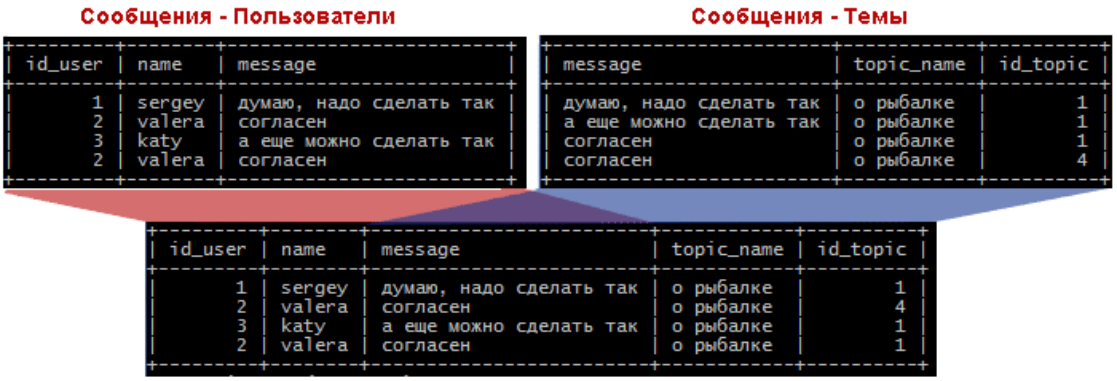
Але щоб замість ідентифікаторів відображалися імена і назви, нам доведеться зробити об'єднання трьох таблиць:

```
SELECT posts.message, topics.topic_name, users.name
FROM posts, topics, users
WHERE posts.id_author = users.id_user
AND posts.id_topic = topics.id_topic;
```

```
mysql> SELECT posts.message, topics.topic_name, users.name
    -> FROM posts, topics, users
    -> WHERE posts.id_author=users.id_user AND posts.id_topic=topics.id_topic;
+--------------------------+--------------+--------+
| message                  | topic_name   | name   |
+--------------------------+--------------+--------+
| согласен с автором       | о рыбалке    | sergey |
| можно сделать по другому | ночные клубы | valera |
| всем привет              | о рыбалке    | katy   |
| у меня другое мнение     | о рыбалке    | sergey |
+--------------------------+--------------+--------+
4 rows in set (0.09 sec)

mysql>
```

Тобто ми об'єднали таблиці Повідомлення і Користувачі умовою posts.id_author = users.id_user, а таблиці Повідомлення і Теми – умовою posts.id_topic = topics.id_topic



id_user	name	message
1	sergey	думаю, надо сделать так
2	valera	согласен
3	katy	а еще можно сделать так
2	valera	согласен

message	topic_name	id_topic
думаю, надо сделать так	о рыбалке	1
а еще можно сделать так	о рыбалке	1
согласен	о рыбалке	1
согласен	о рыбалке	4

id_user	name	message	topic_name	id_topic
1	sergey	думаю, надо сделать так	о рыбалке	1
2	valera	согласен	о рыбалке	4
3	katy	а еще можно сделать так	о рыбалке	1
2	valera	согласен	о рыбалке	1

Об'єднання, які ми розглядали, називаються *Внутрішніми об'єднаннями*. Такі об'єднання пов'язують рядки однієї таблиці з рядками іншої таблиці (а може ще й третій таблиці). Але бувають ситуації, коли необхідно, щоб в результат були включені рядки, які не мають пов'язаних. Наприклад, коли ми створювали запит, які теми і якими авторами були створені, користувач Oleg в результуючу таблицю не потрапив, тому що тим не створював, а тому і пов'язаної рядки в об'єднаній таблиці не мав.

Тому, якщо нам буде потрібно скласти дещо інший запит - вивести всіх користувачів і теми, які вони створювали, якщо такі є - то нам доведеться скористатися *Зовнішнім об'єднанням*, що дозволяє виводити всі рядки однієї таблиці і наявні пов'язані з ними рядки з іншої таблиці. Для цього ми трохи змінимо запит:

```
SELECT users.name, topics.topic_name
FROM users LEFT JOIN topics
ON users.id_user = topics.id_author;
```

І отримаємо бажаний результат - всі користувачі і теми, ними створені. Якщо користувач не створював тему, то в відповідному стовпці стоїть значення NULL.

Отже, ми додали в наш запит ключове слово - *LEFT JOIN*, вказавши тим самим, що з таблиці зліва треба взяти все рядки, і поміняли ключове слово *WHERE* на *ON*. Крім ключового слова *LEFT JOIN* може бути використано ключове слово *RIGHT JOIN*. Тоді будуть вибиратися всі рядки з правої таблиці і наявні пов'язані з ними з лівої таблиці. І нарешті, можливо повне зовнішнє об'єднання, яке отримає всі рядки з обох таблиць і зв'яже між собою ті, які можуть бути пов'язані. Ключове слово для повного зовнішнього об'єднання - *FULL JOIN*.

Давайте змінимо в нашому запиті лівосторонній об'єднання на правосторонній:

```
SELECT users.name, topics.topic_name
FROM users RIGHT JOIN topics
ON users.id_user = topics.id_author;
```

Тепер у нас є всі теми (всі рядки з правої таблиці), а ось користувачі тільки ті, які теми створювали (тобто з лівої таблиці вибираються тільки ті рядки, які пов'язані з правого таблицею).

Підіб'ємо підсумок. Синтаксис для зовнішнього об'єднання наступний:

```
SELECT імя_таблиці_1.імя_стовбца, імя_таблиці_2.імя_стовбца
FROM імя_таблиці_1 ТИП ОБ'ЄДНАННЯ імя_таблиці_2
ON умова_ об’єднання;
```

Перетин і різниця

У стандарті мови SQL є пропозиції оператора **SELECT** для виконання операцій перетину і різниці результатів запитів-операндів. Цими пропозиціями є **INTERSECT [ALL]** (перетин) і **EXCEPT [ALL]** (різниця), які працюють аналогічно пропозицією **UNION**. У результуючий набір потрапляють тільки ті рядки, які присутні в обох

запитах (INTERSECT) або тільки ті рядки першого запиту, які відсутні в другому (EXCEPT). При цьому обидва запити, які беруть участь в операції, повинні мати однакове число стовпців, і відповідні стовпці повинні мати однакові типи даних. Імена стовпців результуючого набору формуються з заголовків першого запиту.

Якщо не використовується ключове слово **ALL** (за замовчуванням мається на увазі DISTINCT), то при виконанні операції автоматично усуваються дублікати рядків. Якщо вказано ALL, то кількість дубльованих рядків підпорядковується наступним правилам (n1 - число дублікатів рядків першого запиту, n2 - число дублікатів рядків другого запиту):

- **INTERSECT ALL**: min (n1, n2)
- **EXCEPT ALL**: n1 - n2, якщо n1> n2.

Знайти кораблі, які присутні як в таблиці Ships, так і в таблиці Outcomes.

```
SELECT name FROM Ships
INTERSECT
SELECT ship FROM Outcomes;
```

В реляційній алгебрі операція перетину є комутативність, оскільки вона може бути застосовна до відносин з однаковими заголовками. Ми і в SQL можемо поміняти запити місцями. Вищенаведене рішення дасть той же результат, що і наступне:

```
SELECT ship FROM Outcomes
INTERSECT
SELECT name FROM Ships;
```

за винятком заголовка. У першому випадку єдиний стовпець буде мати заголовок name, а в другому - ship.

Знайти кораблі з таблиці Outcomes, які відсутні в таблиці Ships.

Завдання легко вирішується за допомогою оператора **EXCEPT**:

```
SELECT ship FROM Outcomes
```

```
2.  EXCEPT
3.  SELECT name FROM Ships;
```

Операція різниці не є комутативність, тому якщо переставити місцями запити, то ми отримаємо рішення зовсім іншого завдання:

"Знайти кораблі з таблиці Ships, які відсутні в таблиці Outcomes".

Слід сказати, що не всі СУБД підтримують ці пропозиції в операторі SELECT. Немає підтримки INTERSECT / EXCEPT, наприклад, в **MySQL**, а в **MS SQL Server** вона з'явилася, лише починаючи з версії 2005, і то без ключового слова ALL. ALL з INTERSECT / EXCEPT також ще не реалізована в **Oracle**. Слід зазначити, що замість стандартного EXCEPT в Oracle використовується ключове слово **MINUS**.

Тому для виконання операцій перетину і різниці можуть бути використані інші засоби. Тут доречно зауважити, що один і той же результат можна отримати за допомогою різних формулювань оператора SELECT. У разі перетину і різниці можна скористатися предикатом існування **EXISTS**.

На закінчення розглянемо приклад на використання операції **INTERSECT ALL**.

Знайти виробників, які випускають не менше двох моделей ПК і не менше двох моделей принтерів.

```
1.  SELECT maker FROM (
2.  SELECT maker FROM Product WHERE type = 'PC'
3.  INTERSECT ALL
4.  SELECT maker FROM Product WHERE type = 'Printer')
5.  GROUP BY maker
6.  HAVING COUNT(*) > 1;
```

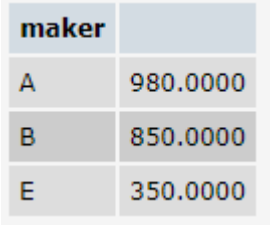
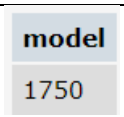
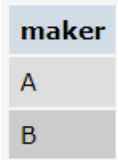
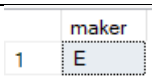
INTERSECT ALL в під запиті цього рішення залишить мінімальне число дублікатів, тобто якщо виробник випускає 2 моделі ПК і одну модель принтера (або навпаки), то він буде присутній в результуючому наборі один раз. Далі ми виконуємо угруповання по виробнику,

залишаючи тільки тих з них, хто присутній в результатах підзапиту більше одного разу.

ПОРЯДОК ВИКОНАННЯ

Виконайте нижченаведені завдання

N	Завдання	Відповідь
1	Для кожного виробника, що випускає ПК-блокноти з об'ємом жорсткого диска щонайменше 10 Гбайт, знайти швидкості таких ПК-блокнотів. Висновок: виробник, швидкість. (INNER JOIN)	maker | speed B | 750 A | 600 A | 750 A | 450
2	Знайдіть номери моделей та ціни всіх наявних у продажу продуктів (будь-якого типу) виробника B (латинська літера). (UNION)	model | price 1121 | 850.0000 1750 | 1200.0000
3	Знайдіть виробника, що випускає ПК, але не блокноти ПК. (EXCEPT)	maker E
4	Знайдіть виробників, які б виробляли як ПК зі швидкістю не менше 750 МГц, так і ПК-блокноти зі швидкістю не менше 750 МГц. Вивести: Maker/ (INTERSECT)	maker A B
5	Знайдіть виробників найдешевших кольорових принтерів. Вивести: maker, price	maker | price D | 270.0000
6	Для кожного виробника, що має моделі в таблиці Laptop, знайдіть середній розмір екрану ПК-блокнотів, що випускаються ним. Вивести: maker, середній розмір	maker | AVG_screen A | 13 B | 14 C | 12

	екрану.	
7	Знайдіть максимальну ціну ПК, що випускаються кожним виробником, який має моделі в таблиці PC. Вивести: maker, максимальна ціна.	 maker A 980.0000 B 850.0000 E 350.0000
8	Перерахуйте номери моделей будь-яких типів, що мають найвищу ціну за всією наявною в базі даних продукцією.	 model 1750
9	Знайти тих виробників ПК, всі моделі ПК, які є в таблиці PC	 maker A B
10	Знайти виробників, які випускають лише принтери або лише PC. При цьому шукані виробники PC повинні випускати щонайменше 3 моделі.	 maker 1 E