Сложни заявки. Специални оператори

I. Оператор EXISTS

Оператора генерира булева логическа стойност „истина“ или „лъжа“. Той може да се прилага самостоятелно в графата where или да се комбинира с други оператори от рода: OR, AND и NOT.

При използването на оператора EXISTS в качеството на аргумент той се оценява като истинен ако генерира някакви изходни данни, в противен случай ще бъде оценен като лъжа.

Нека разгледаме следния пример с използването на оператор EXISTS. Нашата задача е да извлечем от таблица customers данни, но само при условие че един или повече купувачи се намират в София. Заявката би изглеждала така:

```
SELECT IDcustomers, name, phone
FROM Customers
WHERE EXISTS
    (SELECT *
    FROM Customers
    WHERE address LIKE '%Sofia%')
```

Тази заявка е сложна поради това че в нея има вложена заявка (тази в скобите). Резултатът от нея следния: Външната заявка ще върне всички редове от таблицата само ако вътрешната заявка изведе някакъв резултат. Заявката НЕ е аналогична на следната:

```
SELECT IDcustomers, name, phone
FROM Customers
WHERE address LIKE '%Sofia%'
```

Въпроси :

1. Изпълнете двете заявки и анализирайте резултатите
2. Какъв резултат ще върне следната заявка и защо:

```
SELECT IDcustomers, name, phone
FROM Customers
WHERE EXISTS
    (SELECT * FROM Customers
    WHERE address LIKE '%Kiev%')
```

II. Свързани подзаявки и оператора EXISTS

При изпълнение на свързани подзаявки операторът EXISTS се изпълнява поотделно за всеки ред от таблицата към която съществува референция във външната заявка. Вие може да използвате оператора като предикат, генериращ различни отговори за всеки ред от таблицата към която има референция във основната заявка. В този случай заявката от вътрешната информация се запазва дори да не се извежда.

Да разгледаме следния пример селектиращ информацията за клиентите които поне веднъж са направили поръчка.

```
SELECT IDcustomers, name
FROM Customers a
WHERE EXISTS
    (SELECT *
    FROM Orders b
    WHERE a.IDcustomers=b.IDcustomers)
```

Подобна информация от базата данни би могла да бъде получена и от друга заявка. Но тук целта е да се демонстрира как може да се получи достъп то външната (а) до вътрешната (б) заявка и обратно. Тествайте заявката и анализирайте резултата.

III. Изразът NOT EXISTS

Както споменахме по рано операторът може да бъде комбиниран и с булеви оператори. Най често той бива комбиниран с оператора NOT. Разгледайте долната заявка изпълнете я и анализирайте резултата. Каква е разликата със първата заявка от това упражнение.

```
SELECT IDcustomers, name, phone
FROM Customers
WHERE NOT EXISTS
    (SELECT *
    FROM Customers
    WHERE address LIKE '%Kiev%')
```

IV. Операторите ANY и SOME

Операторите ANY и SOME се изпълняват абсолютно еднакво и са взаимозаменяеми. Разликата в терминологията отразява опит за ориентиране към интуитивните представи на потребителя.

Нека разгледаме следния пример. За целта трябва да създадем поле city в таблица customers, съдържащо града в който се намира клиента (полето адрес е предвидено за пълния адрес на клиента). Също така нека да създадем и таблица warehouse която съдържа информация за складовете на компанията и в тази таблица също имаме поле city оказващо в кой град се намира даден склад. Преди на продължим със следващата част от упражнението трябва да попълним информация в новата таблица и поле city от таблица customers.

Сега нека изпълним следната заявка: Да селектираме всички складове намиращи се от същите градове от които са и клиентите.

```
SELECT *
FROM Warehouse
WHERE city = ANY
      (SELECT city
      FROM Customers)
```

Тази заявка работи така: Операторът ANY взема всички стойности от полето city от таблица customers, получени в подзаявката, и оценява резултата като истина ако някоя от стойностите съвпада със стойността в полето city от текущия ред на външната заявка.

Задължително условие полетата между които се прави сравнение в двете заявки (вътрешната и основната) трябва да са от един и същи тип.

V. Оператор ALL

Оператор ALL приема стойност истина, ако всяка стойност, селектирана в процеса на изпълнение на подзаявката, удовлетворява условието, зададено в предиката на външната заявка.

Ако трябва в състава на изходните данни да бъдат включени само онези купувачи, чиято отстъпка е по голяма от отстъпката на всеки един купувач от София, тогава можем да използваме следната заявка:

```
SELECT *
FROM Customers
WHERE sale>ALL
      (SELECT sale
      FROM Customers
      WHERE address LIKE '%Sofia%')
```

Анализирайте заявката. Създайте поле sale в таблица customers оказващо каква отстъпка има всеки от купувачите и тествайте заявката.

VI. Обединяване на множество заявки в една (UNION)

Обединенията се изпълняват с оператор UNION. При него всички заявки се изпълняват независимо само че техните резултати се обединяват.

Да разгледаме следния пример:

```
SELECT IDcustomers, name
    FROM Customers
    WHERE address LIKE '%Sofia%'
UNION
    SELECT IDCustomers, IDorders
    FROM Orders;
```

Изпълнете заявката. Анализирайте я.

Трябва да знаете че:

- Точка и запетая да се поставя единствено след последната заявка. Така SQL ще разбере че това е последната заявка от обединението;
- Броят на колоните във всяка заявка трябва да бъде еднакъв иначе резултатите няма могат да бъдат обединени и ще получите грешка
- Някои програмни продукти не именуват колоните на таблицата с резултатите. MYSQL Workbench обаче в качеството на имена на изходната таблица използва имената на колоните селектирани в първата заявка.

Необходимо условие което не е задължително за всичките програмни продукти използващи SQL заявки е: Колоните влизащи в състава на изходните данни трябва да бъдат съвместими за обединение. Това означава че във всяка от заявките може да бъде указан еднакъв брой колони в следния порядък: първи, втори, трети и т.н., при което първите колони във всяка заявка трябва да бъдат от еднакъв тип и размер. Същото важи и за вторите колоните от всяка заявка, третите колони от всяка заявка и т.н. Както се вижда обаче от резултата на предходната заявка не всички СУБД (включително и MYSQL) се подържат към този стандарт и позволяват гъвкави обединения.

ЗАДАЧИ ЗА САМОКОНТРОЛ :

1. Прочете теоретичната част. Следвайте стъпките ако е необходимо да направите корекции върху вашата база данни и тествайте примерите. Анализирайте резултатите.
2. Като използвате структурата на оператор ANY пробвайте да направите заявка която да намира всички клиенти които някога са направили поръчка. Пробвайте да редактиране тази заявка, така че да използвате оператор IN вместо ANY
3. Покажете всички поръчки на Max.
4. Покажете всички поръчки на клиенти от Бургас.
5. Напишете заявка която да връща общата стойност на поръчките направени от всички клиенти в Бургас.
6. Напишете заявка която да връща всички поръчки, които да надвишават средната стойност на сумата от поръчките на клиент с име Max.