Додаток до Лекції 3

LIKE

Предикат LIKE порівнює рядок, зазначений у першому виразі, для обчислення значення рядка, званого значенням, що перевіряється, із зразком, який визначений у другому виразі для обчислення значення рядка. У зразку дозволяється використовувати два трафаретні символи:

символ підкреслення (_), який можна застосовувати замість будь-якого одиничного символу в значенні, що перевіряється;

символ відсотка (%) замінює послідовність будь-яких символів (число символів у послідовності може бути від 0 і більше) у значенні, що перевіряється.

[ ] - одиночний символ із набору символів (наприклад, [zxy]) або діапазону ([a-z]), вказаних у квадратних дужках. При цьому можна перерахувати відразу кілька діапазонів (наприклад, [0-9a-z]);

^ - який у поєднанні з квадратними дужками виключає з пошукового зразка символи набору або діапазону.

Якщо значення, що перевіряється, відповідає зразку з урахуванням трафаретних символів, то значення предикату дорівнює TRUE.

Нижче наведено кілька прикладів написання зразків.

'abc%' - Будь-які рядки, які починаються з літер «abc»

'abc_' - Рядки довжиною 4 символи, причому першими символами рядка повинні бути «abc»

'%z' - Будь-яка послідовність символів, яка обов'язково закінчується символом "z"

'%Rostov%' Будь-яка послідовність символів, що містить слово «Rostov» у будь-якій позиції рядка

'% % %' -Текст, що містить не менше 2-х пробілів, наприклад, "World Wide Web"

Приклад: *У таблиці Product знайти моделі, які складаються лише з цифр або лише з латинських літер (A-Z, без урахування регістру).*

Вивести: номер моделі, тип моделі

```
select model, type from Product
where model LIKE '[0-9][0-9][0-9][0-9]' or
model LIKE '[a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z]'
```

EXCEPT

У стандарті мови SQL є пропозиції оператора SELECT для виконання операцій перетину та різниці результатів запитів-операндів. Цими пропозиціями є INTERSECT [ALL] (перетин) та EXCEPT [ALL] (різниця), які працюють аналогічно до пропозиції UNION. У результуючий набір потрапляють лише ті рядки, які є в обох запитах (INTERSECT) або ті рядки першого запиту, які відсутні в другому (EXCEPT). При цьому обидва запити, що беруть участь в операції, повинні мати однакову кількість стовпців, і відповідні стовпці повинні мати однакові (або ненаведені) типи даних. Імена стовпців результуючого набору формуються із заголовків першого запиту.

Якщо не використовується ключове слово ALL (за умовчанням мається на увазі DISTINCT), то при виконанні операції автоматично усуваються дублікати рядків.

Приклад*: Знайдіть виробника, що випускає ПК, але не ноутбуки*

```
select maker from product
where type = 'pc'
EXCEPT
select maker from product
where type = 'laptop'
```

Знайдіть виробників, які виробляють як ПК, так і ноутбуки. Вивести: Maker

```
SELECT maker
FROM Product INNER JOIN PC
ON Product.model = PC.model
INTERSECT
SELECT maker
FROM Product INNER JOIN Laptop
```

```
ON Product.model = laptop.model
```

У реченні FROM допускається вказівка кількох таблиць. Просте перерахування таблиць через кому практично не використовується, оскільки воно відповідає реляційній операції, яка називається декартовим добутком. Тобто в результуючому наборі кожен рядок з однієї таблиці поєднуватиметься з кожним рядком з іншого

Тому перерахування таблиць, як правило, використовується спільно з умовою з'єднання рядків з різних таблиць, що вказуються в пропозиції WHERE. Для наведених вище таблиць такою умовою може бути збіг значень, скажімо, у стовпцях a і c:

```
SELECT * FROM A, B
WHERE a = c;
```

тобто з'єднуються ті рядки таблиць, які у зазначених стовпцях перебувають рівні значення.

Приклад: *Знайдіть пари моделей PC, що мають однакову швидкість і RAM. Кожна пара вказується лише один раз, тобто (i,j), але не (j,i), порядок виведення: модель з більшим номером, модель з меншим номером, швидкість і RAM.*

```
SELECT DISTINCT A.model AS model_1, B.model AS model_2, A.speed, A.ram
FROM PC AS A, PC B
WHERE A.speed = B.speed AND
A.ram = B.ram AND
A.model > B.model
```

Загальні табличні вирази (CTE)

Загальні табличні вирази (CTE) відіграє роль уявлення , яке створюється в рамках одного запиту і не зберігається як об'єкт схеми. Загальні табличні вирази дозволяють суттєво зменшити об`єм коду, якщо багаторазово доводиться звертатися до тих самих похідних таблицям .

Перерахуйте номери моделей будь-яких типів, що мають саму високу ціну по всій наявної в базі даних продукції.

```
WITH Mod_MaxPrice AS (
select model , price from PC
where price = ( select MAX( price ) from PC)
UNION
select model , price from Laptop
where price = ( select MAX( price ) from Laptop)
UNION
select model , price from Printer
where price = ( select MAX( price ) from Printer ))

select model from Mod_MaxPrice
where price = ( select MAX( price ) from Mod_MaxPrice )
```

Перетворення типів і оператор CAST

У реалізаціях мови SQL може бути виконане неявне переобрання типів. Так, наприклад, у SQL Server і Sybase ASE.Transact -SQL при порівнянні або комбінуванні значень типів smallint і int, дані типу smallint неявно перетворюються до типу int. Детально про явне та неявне перетворення типів у SQL Server можна прочитати в BOL.

Вивести середню ціну ноутбуків із попереднім текстом «середня ціна =».

Спроба виконати запит

```
SELECT N'Середня ціна =' + AVG( price )
FROM Laptop ;
```

приведе до повідомлення про помилку:

(Не допускається неявне перетворення типу char до типу money)

Це повідомлення означає, що система може виконати неявне перетворення типу char до типу money . У подібних ситуаціях може допомогти явне перетворення типів. При цьому можна скористатися функцією CONVERT. Ця функція не стандартизована, тому з метою переносимості рекомендується використовувати стандартне вираз CAST. З нього і почнемо.

Отже, якщо переписати наш запит у вигляді

```
SELECT N'Середня ціна =' + CAST(AVG( price ) AS CHAR(15))
FROM Laptop ;
```

в результаті отримаємо те, що було потрібно:

```
Середня ціна = 1003.33
```

Ми використовували вираз явного перетворення типів CAST для приведення середнього значення ціни до рядкового подання.

Умовна логіка, оператор CASE

SQL, подібно до багатьох мов програмування, дозволяє писати умовну логіку, щоб в залежності від набору умов повертати одне з безлічі можливих значень.

Реалізація такого запиту за допомогою CASE може виглядати так:

```
CASE
 WHEN умова_1 THEN повертається_значення_1
 WHEN умова_2 THEN повертається_значення_2
 WHEN умова_n THEN повертається_значення_n
 ELSE значення_за_замовчуванням
END
```

Якщо умова_1 повертає справжнє значення, то вираз CASE поверне значення_1, інакше буде зроблена перевірка на умову_2 і т.д. Якщо жодна із запропонованих умов не буде виконана, то повернеться NULL або значення_за замовчуванням, якщо була використана конструкція ELSE.

Для кожного виробника, у якого присутні моделі хоча б в одній з таблиць PC, Laptop або Printer, визначити максимальну ціну на його продукцію.

Вивести: ім'я виробника , якщо серед цін на продукцію даного виробника є NULL, то виводити для цього виробника «Немає ціни», інакше максимальну ціну

```
WITH max_price AS
(
SELECT maker, Product.model,
CASE
 WHEN price IS NULL
 THEN 'NULL'
 ELSE CAST(price AS CHAR(20))
 END price
 FROM Product LEFT JOIN  PC
```

```
ON Product.model = PC.model
WHERE product.type = 'pc'
UNION ALL

SELECT maker, Product.model,
CASE
 WHEN price IS NULL
 THEN 'NULL'
 ELSE CAST(price AS CHAR(20))
 END price
 FROM Product LEFT JOIN  Printer
ON Product.model = Printer.model
WHERE product.type = 'printer'
UNION ALL

SELECT maker, Product.model,
CASE
 WHEN price IS NULL
 THEN 'NULL'
 ELSE CAST(price AS CHAR(20))
 END price
 FROM Product LEFT JOIN  Laptop
ON Product.model = Laptop.model
WHERE product.type = 'Laptop'
)
select maker, MAX(price) AS max_price
 from max_price
 GROUP BY maker
```

	maker	max_price
1	A	NULL
2	B	1200.00
3	C	970.00
4	D	400.00
5	E	NULL

Для кожного типу продукції та кожного виробника з таблиці Product з точністю до двох десяткових знаків знайти відсоткове відношення числа моделей даного типу даного виробника до загального кількості моделей цього виробника.

Вивести: maker , type , відсоткове відношення числа моделей даного типу до загального числа моделей виробника

```
WITH coll_mod AS (
select maker , type , sum_mod from
( select maker , CAST(COUNT( type ) as DECIMAL (5,4)) as sum_mod from Product
GROUP BY maker ) as tabl1
CROSS JOIN
( select type from Product
GROUP BY type ) as tabl2
)
select coll_mod.maker , coll_mod.type ,
CASE
WHEN
( col_mod / sum_mod ) IS NULL
THEN '.00'
ELSE CAST(( col_mod / sum_mod ) as DECIMAL (5,4))
END
from coll_mod LEFT JOIN
( select maker , type , CAST(COUNT( type ) as DECIMAL (5,4)) as col_mod from Product
GROUP BY type , maker ) as tabl3
ON coll_mod.maker = tabl3.maker
AND coll_mod.type = tabl3.type
ORDER BY maker
```

Функція ROW_NUMBER

Функція ROW_NUMBER нумерує рядки, що повертаються запитом. З її допомогою можна, можливо виконати більше складне упорядкування рядків у звіті , ніж те , що дає пропозиція ORDER BY у рамках Стандарту SQL-92.

Використовуючи функцію ROW_NUMBER можна :

- поставити нумерацію, яка буде відрізнятись від порядку сортування рядків результуючого набору;
- створити " ненаскрізну " нумерацію, тобто виділити групи з спільного безліч рядків і пронумерувати їх окремо для кожної групи;
- використовувати одномірно кілька способів нумерації, оскільки фактично нумерація не залежить від сортування рядків запиту.

Пропозиція OVER, з якою використовується функція ROW_NUMBER визначає порядок нумерації рядків. При цьому використовується додаткове пропозиція ORDER BY, яка не має відношення до порядку виведення рядків запиту. Якщо ви подивіться на результат, то зауважте, що порядок рядків у результуючому наборі та порядок нумерації не збігаються.

Конструкція PARTITION BY задає групи рядків, для яких виконується незалежна нумерація. Група визначається рівністю значень у списку стовпців, перерахованих у цій конструкції, у рядків, що становлять групу .

Пронумерувати унікальні пари {maker, type} з Product, упорядкувавши їх наступним чином:

- *ім'я виробника (maker) за зростанням;*
- *тип продукту (type);*
- *якщо якийсь виробник випускає кілька типів продукції, то виводити його ім'я тільки в першій рядку;*
- *інші рядки для ЦЬОГО виробника повинні містити порожній рядок символів ('').*

```
SELECT num1,
CASE
WHEN num1 > 1 THEN ''
ELSE maker
```

```
END AS maker ,
 type

 from
( select row_number () over ( partition BY maker ORDER BY maker ) num1,
maker , type from Product
GROUP BY maker , type ) as tabl1
```

Конструкція TOP N WITH TIES

Конструкція TOP N WITH TIES дозволяє відібрати з відсортованого набору перших N рядків.

Вивести всі рядки з таблиці Product, крім трьох рядків із найменшими номерами моделей та трьох рядків із найбільшими номерами моделей.

```
select * from Product
WHERE model NOT IN
( select TOP 3 WITH TIES model from product
ORDER BY model )
AND
model NOT IN
( select TOP 3 WITH TIES model from product
ORDER BY model DESC)
```

maker	model	type
E	1260	PC
A	1276	Printer
D	1288	Printer
A	1298	Laptop
C	1321	Laptop
A	1401	Printer
A	1408	Printer
D	1433	Printer
E	1434	Printer
B	1750	Laptop