Тема лекції 4:

Групування та сортування даних. Використання підзапитів

- Групування записів за значеннями одного або декількох стовпців
- Упорядкування результатної таблиці
- Використання агрегатних функцій
- Використання підзапитів
- Використанням кванторів
- Підзапити і агрегатні функції
- Зв'язані (корельовані) підзапити
- Використання предиката EXISTS

Фрази GROUP BY та HAVING

- GROUP BY служить для групування записів за значеннями одного або декількох стовпців
- Якщо в SQL-виразі використовується оператор WHERE, який задає фільтр записів, то оператор GROUP BY знаходиться і виконується після нього.
- Для визначення, які записи повинні увійти в групи, служить оператор HAVING, який використовується разом з GROUP BY
- Якщо оператор HAVING не використовується, то групуванню підлягають усі записи, відфільтровані оператором WHERE
- Якщо WHERE не використовується, то групуються усі записи таблиці

Приклад 1. Використання GROUP BY

- Вивести інформацію про усіх замовників за їх місцем роботи:

```
SELECT city,cname,rating
    FROM Customers
    GROUP BY city;
```

Приклад 2. Використання GROUP BY і HAVING

- Вивести усі операції купівлі-продажу за датою, в яких платіж перевищує 1 000:

```
SELECT onum,odate,amt
   FROM Orders
   GROUP BY odate, amt
   HAVING amt>1000;
```

Фраза ORDER BY

- застосовується для упорядкування (сортування) записів
- записується і виконується вкінці запиту
- сортує записи усієї таблиці або окремих її груп, у випадку застосування оператора GROUP BY
- Після імені стовпця групування можна вказувати ключове слово, яке задає режим сортування: ASC – за зростанням (за замовчуванням) і DESC – за спаданням
- усі стовпці, які впорядковуються, повинні вказуватись у виразі SELECT

Приклад 3. Використання ORDER BY

- Впорядкуємо вивід таблиці Orders в порядку спадання номерів замовників:

```
SELECT *  FROM Orders
  ORDER BY cnum DESC;
```

Приклад 4. Використання ORDER BY

- У наступному запиті вивід таблиці Orders впорядковується за спаданням замовників. Потім для кожного замовника сортується його платіж операції купівлі-продажу:

```
SELECT *  FROM Orders
  ORDER BY cnum DESC, amt;
```

Агрегатні (статистичні) функції в SQL

- **COUNT**(параметр) – обчислює кількість записів, вказаних у параметрі.
  - Якщо необхідно отримати кількість усіх записів, то в якості параметра вказується *
  - Якщо в якості параметра вказано ім’я стовпця, то функція поверне кількість записів, в яких цей стовпець має неNULL значення
  - Щоб знати, скільки різних значень має стовпець, перед його іменем вказується ключове слово DISTINCT

Приклади 5-6. Використання COUNT

- Обчислимо кількість записів у таблиці Customers:

```
SELECT COUNT(*)  FROM Customers;
```

Виведеться число 7.

- Підрахуємо кількість продавців, які провели операції на даний час:

```
SELECT COUNT(DISTINCT snum)
    FROM Orders;
```

Виведеться число 5.

Агрегатні (статистичні) функції в SQL

- **SUM**(параметр) – обчислює суму значень стовпця, вказаного як параметр
- **AVG**(параметр) – обчислює середнє арифметичне значень стовпця, вказаного в параметрі
- Параметр може представляти собою і вираз, який містить ім’я стовпця; тоді використання DISTINCT не дозволяється
- З функціями SUM та AVG можуть використовуватись лише числові поля

Приклад 7. Використання SUM

- Знайти суму усіх операцій купівлі-продажу:

```
SELECT SUM(amt)  FROM Orders;
```

- Або знайти суму усіх операцій купівлі-продажу в доларах для продавця 1001:

```
SELECT SUM(amt/8.05)  FROM Orders
  WHERE snum=1001;
```

Приклад 8. Використання AVG

- Знаходження середнього арифметичного усіх операцій :

```
SELECT AVG(amt)  FROM Orders; \
```

- Або в доларах для замовника 2004:

```
SELECT AVG(amt/8.05)  FROM Orders
  WHERE cnum=2004;
```

Агрегатні (статистичні) функції в SQL

- **MAX**(параметр) – обчислює найбільше з усіх вибраних значень стовпця
- **MIN**(параметр) – обчислює найменше з усіх вибраних значень стовпця
- З функціями COUNT, MAX і MIN можуть використовуватись і числові, і символьні поля.

Фраза GROUP BY та агрегатні функції в SQL

- Вираз GROUP BY дозволяє визначати підмножину значень в деякому стовпці і застосовувати агрегатну функцію до цієї підмножини.
- Тоді необхідно об’єднати стовпці та агрегатні функції у фразі SELECT.

Приклад 9. Використання GROUP BY з агрегатною функцією

- Знайти найбільший платіж, отриманий кожним продавцем:

```
SELECT snum, MAX(amt)
  FROM Orders
  GROUP BY snum;
```

- Результатна таблиця буде така:

```
1001      767.19
1002     1713.23
1003       75.75
1004     1309.95
1007     1098.16
```

Приклад 10. Використання GROUP BY з декількома полями

- Знайти найбільший платіж, який отримав кожен продавець за кожен день:

```
SELECT snum,odate,MAX(amt)
  FROM Orders
  GROUP BY snum,odate;
```

- Вивід буде таким:

1001	03-09-2013	767.19
1001	05-09-2013	4723.00
1001	06-09-2013	9891.88
1002	03-09-2013	5160.45
1002	04-09-2013	75.75
1002	06-09-2013	1309.95
1003	04-09-2013	1713.23
1004	03-09-2013	1900.10
1007	03-09-2013	1098.16

Фрази GROUP BY, HAVING та агрегатні функції в SQL

- Приклад 30. Необхідно дізнатись максимальний платіж, отриманий кожним продавцем за кожен день, значення якого більше 3 000.00.
- В даному випадку **не можна використовувати агрегатну функцію у виразі WHERE**, оскільки предикати оцінюються в термінах одиничного рядка, а агрегатні функції оцінюються в термінах груп рядків.
- Це означає, що не можна написати в команді SELECT з попереднього прикладу таку фразу:

WHERE MAX(amt)>3000.00

- Це буде відхиленням від строгої інтерпретації стандарту. В таких випадках слід використовувати речення HAVING.

Приклад 11. Використання GROUP BY, HAVING та агрегатної функції

- Отже, в нашому прикладі, правильною буде наступна інструкція:

```
SELECT snum, odate, MAX(amt)
  FROM Orders
  GROUP BY snum, odate
  HAVING MAX(amt)>3000.00;
```

- Вивід буде таким:

```
1001    05-09-2013    4723.00
1001    06-09-2013    9891.88
1002    03-09-2013    5160.45
```

Приклад 12. Використання GROUP BY, HAVING та агрегатної функції

- Аргументи у фразі HAVING повинні мати одне значення на групу виводу.
- Наприклад наступна інструкція є неправильною і буде заборонена:

```
SELECT snum, MAX(amt)   FROM Orders
  GROUP BY snum
  HAVING odate='03-09-2013';
```

- Поле odate не може викликатись оператором HAVING, тому що воно може мати (і дійсно має) ***більше, ніж одне значення на групу виводу***.
- Щоб уникнути такої ситуації, **оператор HAVING повинне використовувати лише агрегатні функції і поля, вибрані оператором GROUP BY.**

Приклад 13. Використання GROUP BY та агрегатної функції

- Правильний спосіб написання попереднього запиту буде такий:

```
SELECT snum, MAX(amt)
  FROM Orders
  WHERE odate='03-09-2013'
  GROUP BY snum;
```

- В результаті виведуться максимальні значення суми кожного продавця на 3 жовтня

Приклад 14. Використання GROUP BY, HAVING та агрегатної функції

- Розглянемо приклад використання в операторі HAVING полів, вибраних в GROUP BY.
- Нехай необхідно вивести найбільші платежі продавців Serres (1002) і Rifkin (1007):

```
SELECT snum, MAX(amt)
  FROM Orders
  GROUP BY snum
  HAVING snum IN (1002,1007);
```

Використання агрегатної функції від агрегату

- В строгій інтерпретації стандарту SQL ***не можна*** використовувати агрегат агрегату.
- Припустимо, що необхідно з'ясувати в який день була найбільша сума платежів. Якщо запишеться інструкція:

```
SELECT odate, MAX(SUM(amt))
  FROM Orders
  GROUP BY odate;
```

то вона буде відхилена.

- Деякі реалізації не роблять даного обмеження, оскільки деколи вкладені агрегати бувають корисними.

Запити всередині інших запитів

- В SQL існує можливість вставляти один запит в інший. Як правило, внутрішній запит генерує значення, яке перевіряється у предикаті зовнішнього запиту.
- При використанні підзапитів у предикатах, які використовують операції порівняння, необхідно, щоб результат підзапиту видавав **лише один рядок**. В іншому випадку команда не виконається.
- Якщо в результаті підзапиту не буде ніяких значень, то інструкція виконається, але не видасть ніяких результатів. Предикат, в якому розміщений такий підзапит, є невідомий і має такий ефект як невірний, тому команда не має результатів.

Приклад 15. Використання підзапиту

- Припустимо, що відомо ім’я продавця – Motika, але не знаємо значення його поля snum. Необхідно вивести усі операції купівлі-продажу, які обслуговуються цим продавцем:

```
SELECT *   FROM Orders
  WHERE snum=
      (SELECT snum  FROM Salers
         WHERE sname=’Motika’);
```

Приклад 15. Використання підзапиту (пояснення)

- Щоб проаналізувати зовнішній (основний) запит, SQL оцінює внутрішній запит (або підзапит), який розміщений у виразі WHERE. Єдиним знайденим рядком підзапиту буде snum=1004. Однак, SQL не видає такий результат, а поміщає його у предикат основного запиту як WHERE snum=1004. В результаті виведеться рядок з таблиці Orders з порядком 3002. Такий вигляд запиту є універсальний. Він буде працювати навіть тоді, коли номер Motika зміниться, а за допомогою простої заміни імені підзапит можна використовувати його для інших продавців.

Використання підзапитів

- Предикати, які включають підзапит, використовують вираз:

<скалярна форма> <оператор> <підзапит>,

- а, не

<підзапит> <оператор> <скалярний вираз>

- Стандарт ANSI забороняє вставляти для порівняння два підзапити:

<підзапит> <оператор> <підзапит>.

Підзапити з використанням кванторів

- ALL (усі) – квантор загальності

- SOME (деякий) – квантор існування
- ANY (який-небудь) – квантор існування

Використанням квантора загальності ALL (усі)

- Нехай маємо дві таблиці: T1, яка містить стовпець A, і T2, яка містить стовпець B. Тоді підзапит з квантором ALL :

```
SELECT A  FROM T1
  WHERE A  <оператор порівняння>
        ALL (SELECT B  FROM T2);
```

- Запит виводить ті значення стовпця A, для яких оператор порівняння є істинним для усіх значень стовпця B.

Використанням кванторів існування SOME (деякий), ANY (який-небудь)

- Нехай маємо дві таблиці: T1, яка містить стовпець A, і T2, яка містить стовпець B. Тоді підзапит з квантором SOME:

```
SELECT A  FROM T1
   WHERE A  <оператор порівняння>
                SOME (SELECT B  FROM T2);
```

- Запит виводить ті значення стовпця A, для яких оператор порівняння є істинним для хоча б одного зі значень стовпця B.

Приклад 16. Використання агрегатних функцій у підзапитах

- Вивести інформацію про операції з платежем вище середнього на 4 вересня:

```
SELECT *   FROM Orders
  WHERE amt>
      (SELECT AVG(amt) FROM Orders
          WHERE odate=’ 04-09-2013’);
```

- Середній платіж на 4 вересня = 894.49. Отже, виведуться усі рядки, в яких значення поля amt є більшим.

Використання агрегатних функцій у підзапитах

- Згруповані агрегатні функції за допомогою GROUP BY, видають декілька значень. Таким чином вони ***не використовуються*** у підзапитах, навіть, коли оператор GROUP BY або HAVING виводять одну групу.
- У підзапитах **необхідно використовувати** одиничну агрегатну функцію у виразі WHERE, щоб уникнути небажаних груп.

Приклади 17-18. Використання агрегатних функцій у підзапитах

- Наступний запит, який повинен знайти середнє значення комісійних продавця у Лондоні, ***не зможе*** використовуватись у підзапиті.

  SELECT AVG(comm) FROM Salers

  GROUP BY city

  HAVING city='London';

- Інший спосіб, який **може** використовуватись у підзапиті, такий:

  SELECT AVG(comm) FROM Salers

  WHERE city='London';

Підзапити, в результаті яких виходить декілька значень

- IN – використовується з підзапитами
- BETWEEN – не використовується з підзапитами
- LIKE – не використовується з підзапитами

Приклад 19. Використання підзапитів, в результаті яких виходиться декілька значень

- Вивести всю інформацію про операції купівлі-продажу для продавців у Лондоні.
- Тут використаємо з підзапитом оператор IN, оскільки команда не буде працювати з оператором порівняння:

```
SELECT *  FROM Orders
  WHERE snum IN
      (SELECT snum  FROM Salers
        WHERE city='London');
```

Приклад 20. Використання підзапитів у фразі HAVING

- Такі підзапити можуть використовувати свої агрегатні функції, якщо вони не виводять декількох значень.
- Наступна інструкція рахує кількість замовників з рейтингом, вищим середнього, у місті San Jose.

```
SELECT rating,COUNT(cnum)
  FROM Customers
  GROUP BY rating
  HAVING rating>
       (SELECT AVG(rating)
          FROM Customers
          WHERE city='San Jose';
```

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.

Зв’язані (корельовані) підзапити

- 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

1. Спочатку береться увесь перший запис з таблиці T1. Цей запис називається поточним. Значення стовпців для цього запису є доступними і можуть використовуватись у підзапиті.
2. 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 =).

Виконання запиту з корельованим підзапитом

3. 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.
4. Після цього відбувається перехід на наступний запис таблиці T1. Аналогічно все виконується для кожного запису таблиці T1.

Приклад 21. Використання зв’язаного підзапиту

- Знайдемо інформацію про усіх замовників на 3 вересня:

```
SELECT *    FROM Customers
  WHERE ‘03-09-2013’ IN
      (SELECT odate  FROM Orders
          WHERE
Customers.cnum=Orders.cnum );
```

Приклад 22. Використання зв’язаного підзапиту

- (Перевірка правильності ведення БД). Перевірити, чи співпадають поля snum і cnum в кожному рядку таблиці Customers, у запиті до таблиці Orders.

```
SELECT *
  FROM Orders main
  WHERE NOT snum =
      (SELECT snum
          FROM Customers
          WHERE cnum=main.cnum );
```

- Правильний результат – жодного рядка
- Даний запит сприймає поле cnum як первинний ключ таблиці Customers, який має лише одне значення у цій таблиці.

Зв’язані підзапити у фразі HAVING

- Нагадаємо, що фраза HAVING може використовувати лише агрегатні функції, вказані у виразі SELECT, або поля, що використовуються в операторі GROUP BY.
- В такому випадку підзапит буде виконуватись один раз для кожної групи, а не для кожного рядка.

Приклад 23. Використання зв’язаних підзапитів у фразі HAVING

- Просумувати платежі за кожен день, виводячи дати, де сума платежів була б на 2000 більша від максимального платежа за цей день.

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

Приклад 23. Використання зв’язаних підзапитів у фразі HAVING (пояснення)

- Підзапит обчислює максимальне значення платежів для усіх записів з однаковою датою, що й у поточної агрегатної групи основного запиту. Це виконується, як і раніше, з використанням оператора WHERE.
- Нагадаємо, що сам підзапит не повинен використовувати речення GROUP BY або HAVING.

Використання предиката EXISTS

- Виконуючи запит на вибірку даних, ми не завжди впевнені, що результат містить хоча б один рядок.
- Якщо результат запиту є порожнім, то недоцільно надалі проводити обробку даних.
- Таким чином корисно знати, чи містить результат запиту які-небудь дані.
- Для цього призначений предикат EXISTS (існування). Він набуває істинне значення лише тоді, коли результатна таблиця запиту містить хоча б дин рядок.

Використання предиката EXISTS

- Приклад 2. (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);
```

Практичні завдання до БД «Операції купівлі-продажу»

- Напишіть запит, який би вивів найбільші суми для замовників Hoffman та Giovanni.
- Напишіть запит, який би вибрав найбільшу суму операції для кожного замовника.
- Напишіть запит, який би вибрав найнижчий рейтинг замовника з кожного міста.
- Напишіть запит, який би виводів інформацію про операції купівлі-продажу, в яких брав участь замовник Giovanni, припустивши, що його номер ви не знаєте.
- Напишіть запит, який би вивів імена та рейтинги усіх замовників, які у певних операціях оплатили платіж, який менший або рівний середньому платежу усіх операцій купівлі-продажу.
- Напишіть запит, який би обчислив максимальний платіж усіх операцій купівлі-продажу для кожного продавця, в якого цей максимальний платіж більший від середнього платежа усіх операцій.
- Напишіть запит, який би вивів усю інформацію про операції купівлі-продажу, які обслуговують продавці з комісійними 12%-15%.
- Напишіть запит, який би вивів усю інформацію про операції купівлі-продажу, в яких брали участь замовники з рейтингом 200.
- Напишіть запит, який би вивів усю інформацію про операції купівлі-продажу, в яких брали участь замовники з Риму.
- Напишіть запит, який би вивів номери та імена усіх замовників з мінімальними рейтингами у їхніх містах.
- Напишіть команду, яка б вивела імена та номери кожного продавця, який брав участь лише у одній операції. Результат представте в алфавітному порядку.

Дякую за увагу