**Лекція 10. Елементи мови SQL (продовження)**

**Приклад 18**. З’єднання таблиць за довільною умовою. Розглянемо таблиці постачальників та деталей, яким надано деякий статую:

|  |  |  |
| --- | --- | --- |
| **PNUM** | **PNAME** | **PSTATUS** |
| 1 | Іван | 4 |
| 2 | Петро | 1 |
| 3 | Микола | 2 |

**Таблиця 1 Відношення P (Постачальники)**

|  |  |  |
| --- | --- | --- |
| **DNUM** | **DNAME** | **DSTATUS** |
| 1 | Болт | 3 |
| 2 | Гайка | 2 |
| 3 | Гвинт | 1 |

**Таблиця 2 Відношення D (Деталі)**

Відповідь на питання "які постачальники мають право постачати які деталі?" дає такий запит:

SELECT

P.PNUM,

P.PNAME, P.PSTATUS, D.DNUM, D.DNAME, D.DSTATUS

FROM P, D

WHERE P.PSTATUS >= D.DSTATUS;

В результаті отримуємо таблицю:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PNUM** | **PNAME** | **PSTATUS** | **DNUM** | **DNAME** | **DSTATUS** |
| 1 | Іванов | 4 | 1 | Болт | 3 |
| 1 | Іванов | 4 | 2 | Гайка | 2 |
| 1 | Іванов | 4 | 3 | Гвинт | 1 |
| 2 | Петро | 1 | 3 | Гвинт | 1 |
| 3 | Микола | 2 | 2 | Гайка | 2 |
| 3 | Микола | 2 | 3 | Гвинт | 1 |

**Застосування імен кореляції (аліасів, псевдонімів)**

Іноді потрібно виконувати запити, в яких таблиця з’єднується сама з собою, або одна таблиця з’єднується двічі з іншою таблицею. При цьому використовуються ***імена кореляції*** (***аліаси***, ***псевдонім***), які дозволяють розрізняти з’єднувані копії таблиць. Імена кореляції вводяться у розділі FROM і йдуть через пропуск після імені таблиці. Імена кореляції повинні використовуватися в якості префіксу перед іменем стовбця і відділяються від імені стовбця крапкою. Якщо в запиті вказуються одні й ті ж самі поля з різних екземплярів однієї таблиці, вони повинні бути перейменовані для запобіганню неоднозначності в найменуваннях колонок результуючої таблиці. Визначення імені кореляції діє лише під час виконання запиту.

**Приклад19**. Відібрати усі пари постачальників таким чином, щоб перший постачальник в парі мав статус, більший за статус другого постачальника:

SELECT

P1.PNAME AS PNAME1, P1.PSTATUS AS PSTATUS1, P2.PNAME AS PNAME2, P2.PSTATUS AS PSTATUS2

FROM

P P1, P P2

WHERE P1.PSTATUS1 > P2.PSTATUS2;

В результаті отримуємо таблицю:

|  |  |  |  |
| --- | --- | --- | --- |
| **PNAME1** | **PSTATUS1** | **PNAME2** | **PSTATUS2** |
| Іван | 4 | Петро | 1 |
| Іван | 4 | Микола | 2 |
| Микола | 2 | Петро | 1 |

**Приклад20**. Розглянемо ситуацію, коли деякі постачальники (контрагенти) можуть виступати як постачальники деталей, так й покупці. Таблиці, що зберігають дані можуть мети такий вигляд:

|  |  |
| --- | --- |
| **Номер контрагента**  **NUM** | **Найменування контрагента**  **NAME** |
| 1 | Іван |
| 2 | Петро |
| 3 | Микола |

**Таблиця 3 Відношення CONTRAGENTS**

|  |  |
| --- | --- |
| **Номер деталі**  **DNUM** | **Найменування деталі**  **DNAME** |
| 1 | Болт |
| 2 | Гайка |
| 3 | Винт |

**Таблиця 4 Відношення DETAILS (Деталі)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер поставщика**  **PNUM** | **Номер получателя**  **CNUM** | **Номер детали**  **DNUM** | **Поставляемое количество**  **VOLUME** |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 1 | 100 |
| 1 | 3 | 2 | 200 |
| 1 | 3 | 3 | 300 |
| 2 | 3 | 1 | 150 |
| 2 | 3 | 2 | 250 |
| 3 | 1 | 1 | 1000 |

**Таблиця 5 Відношення CD (Поставки)**

В таблиці CD (поставки) поля PNUM і CNUM є зовнішніми ключами, що посилаються на потенційний ключ NUM в таблиці CONTRAGENTS.

Відповідь на питання "хто і кому та що і в якій кількості поставляє" дається таким запитом:

SELECT

P.NAME AS PNAME,

C.NAME AS CNAME, DETAILS.DNAME, CD.VOLUME

FROM

CONTRAGENTS P,

CONTRAGENTS C, DETAILS,

CD WHERE

P.NUM = CD.PNUM AND

C.NUM = CD.CNUM AND D.DNUM = CD.DNUM;

В результаті отримуємо таблицю:

|  |  |  |  |
| --- | --- | --- | --- |
| **Найменування постачальника PNAME** | **Найменуванння отримувача CNAME** | **Найменування деталі DNAME** | **Кількість, що постачається VOLUME** |
| Іван | Петро | Болт | 100 |
| Іван | Микола | Гайка | 200 |
| Іван | Микола | Гвинт | 300 |
| Петро | Микола | Болт | 150 |
| Петро | Микола | Гайка | 250 |
| Микола | Іван | Болт | 1000 |

Зауваження. Цей же запит можна виразити в інший спосіб, наприклад, так:

SELECT

P.NAME AS PNAME, C.NAME AS CNAME, DETAILS.DNAME, CD.VOLUME

FROM

CONTRAGENTS P,

CONTRAGENTS C,

DETAILS NATURAL JOIN CD

WHERE

P.NUM = CD.PNUM AND

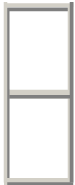
C.NUM = CD.CNUM;

**Застосування агрегатних функцій в запитах**

**Приклад21**. Отримати загальну кількість постачальників (ключове слово ***COUNT***):

SELECT COUNT(\*) AS N FROM P;

В результаті отримуємо таблицю з одним стовбцем і одним рядком, що містить кількість рядків з таблиці P:



**N**

3

**Приклад22**. Отримати загальну, максимальну, мінімальну і середню кількість деталей, що постачаються (ключові слова ***SUM***, ***MAX***, ***MIN***, ***AVG***):

SELECT

SUM(PD.VOLUME) AS SM,

MAX(PD.VOLUME) AS MX,

MIN(PD.VOLUME) AS MN,

AVG(PD.VOLUME) AS AV

FROM PD;

В результаті отримуємо таблицю з одним рядком:

|  |  |  |  |
| --- | --- | --- | --- |
| **SM** | **MX** | **MN** | **AV** |
| 2000 | 1000 | 100 | 333.33333333 |

**Застосування агрегатних функцій з групуванням**

**Пример23.** Для кожної деталі отримати сумарну кількість поставок (ключове слово ***GROUP BY***…):

SELECT

PD.DNUM,

SUM(PD.VOLUME) AS SM

GROUP BY PD.DNUM;

Цей запит буде виконуватися так. Спочатку рядки вхідної таблиці будуть згруповані таким чином, щоб у кожну групу попали рядки з однаковими значеннями DNUM. Потім всередині кожної групи буде підсумоване поле VOLUME. Від кожної групи в вихідну таблицю буде включено один рядок:

|  |  |
| --- | --- |
| **DNUM** | **SM** |
| 1 | 1250 |

|  |  |
| --- | --- |
| 2 | 450 |
| 3 | 300 |

Зауваження. В переліку полів, що відбираються, за допомогою оператора SELECT, який містить розділ GROUP BY можна включати *тільки* агрегатні функції та поля, я*кі входять в умову групування*. Наступний запит містить синтаксичну помилку:

SELECT

PD.PNUM,

PD.DNUM,

SUM(PD.VOLUME) AS SM

GROUP BY PD.DNUM;

Причина помилки в тому, що в перелік полів включене поле PNUM, яке *не входить* в розділ GROUP BY. Дійсно, в кожну отриману групу рядків може входити декілька рядків з *різними* значеннями поля PNUM. З кожної групи рядків буде сформовано по одному стоговому рядку. При цьому нема однозначної відповіді на питання, яке значення вибрати для поля PNUM в ітоговому рядку.

Зауваження. Окремі діалекти SQL не вважають це за помилку. Запит буде виконано, але передбачити, які значення будуть внесені в поле PNUM в результуючу таблицю, неможливо.

**Приклад24**. Отримати номери деталей, сумарна кількість яких перевищує 400 (ключове слово ***HAVING***…):

Зауваження. Умова, що сумарна кількість повинна бути більша за 400 не може бути сформульована в розділі WHERE, тому що в цьому розділі неможна застосовувати агрегатні функції. Умови, що використовують агрегатні функції мають бути розміщені в спеціальному розділі HAVING:

SELECT

PD.DNUM,

SUM(PD.VOLUME) AS SM GROUP BY PD.DNUM

HAVING SUM(PD.VOLUME) > 400;

В результаті отримуємо таблицю:

|  |  |
| --- | --- |
| **DNUM** | **SM** |
| 1 | 1250 |
| 2 | 450 |

Зауваження. В одному запиті можуть зустрічатися як умови відбору рядків в розділі WHERE, так й умови відбору груп в розділі HAVING. Умови відбору груп неможна перенести з розділу HAVING в розділ WHERE. Аналогічно й умови відбору рядків неможна перенести з розділу WHERE в розділ HAVING, за виключення умов, що включають поля з переліку групування GROUP BY.

**Використання підзапитів (внутрішніх або вкладених запитів)**

Зручним засобом, що дозволяє формулювати запити більш просто і зрозуміло, є можливість використання підзапитів, вкладених в основний запит.

**Приклад25**. Отримати список постачальників, статус яких меншій за максимальний статус в таблиці постачальників:

SELECT \* FROM P

WHERE P.STATYS <

(SELECT MAX(P.STATUS)

FROM P);

Зауваження. Поле P.STATUS порівнюється з результатом підзапиту, тому підзапит повинен бути сформульований так, щоб повертати таблицю, яка складається *рівно з одного рядка й однієї колонки*.

Зауваження. Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати *один раз* вкладений підзапит і отримати максимальне значення статусу.
2. Просканувати таблицю постачальників P, кожного разу порівнюючи значення статусу постачальника з результатом підзапиту, і відібрати лише ті рядки, в яких статус менше за максимальний.

**Приклад26**. Застосування предикату ***IN***. Отримати список постачальників, які поставляють деталь номер 2:

SELECT \* FROM P

WHERE P.PNUM IN

(SELECT DISTINCT PD.PNUM FROM PD

WHERE PD.DNUM = 2);

Зауваження. В даному випадку вкладений підзапит може повернути таблицю, яка містить декілька рядків.

Зауваження. Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати *один раз* вкладений підзапит і отримати список номерів постачальників, що поставляють деталь номер 2.
2. Просканувати таблицю постачальників P, кожен раз перевіряючи, чи міститься

номер постачальника в результаті підзапиту.

**Приклад27**. Застосування предикатe ***EXIST***. Отримати список постачальників, що поставляють деталь номер 2:

SELECT \* FROM P WHERE EXIST

(SELECT \*

FROM PD

WHERE

PD.PNUM = P.PNUM AND

PD.DNUM = 2);

Зауваження. Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Просканувати таблицю постачальників P, *кожного разу виконуючи підзапит* з новим значенням номера постачальника, який береться з таблиці P.
2. В результат запиту включити лише ті рядки з таблиці постачальників, для яких

вкладений підзапит повернув непорожню множину рядків.

Зауваження. На відміну від двох попередніх прикладів, вкладений підзапит містить параметр (зовнішнє посилання), що передається з основного запиту - номер постачальника P.PNUM. Такі підзапити називаються ***корельованими*** (***correlated***). Зовнішнє посилання може приймати різні значення для кожного рядка-кандидата, який оцінюється за допомогою підзапиту, тому підзапит має виконуватися заново для кожного рядка, що відбирається в основному запиті. Такі підзапити характерні для предикату EXIST, але можуть бути використані й у інших підзапитах.

Зауваження. Може здаватися, що запити, що містять корельовані підзапити будуть виконуватися повільніше за запити з некорельованими підзапитами. Насправді це не так, тому що те, як користувач сформулював запит, *не визначає*, як цей запит буде виконуватися. Мова SQL є непроцедурною, а декларативною. Це означає, що користувач, який формулює запит, просто описує, я*ким має бути отриманий результат запиту*, а те, як цей результат буде отримано - за це відповідає сама СУБД.

**Приклад28**. Застосування предикату ***NOT EXIST***. Отримати список постачальників, які не поставляють деталь номер 2:

SELECT \* FROM P

WHERE NOT EXIST (SELECT \*

FROM PD WHERE

PD.PNUM = P.PNUM AND PD.DNUM = 2);

Зауваження. Як і в попередньому прикладі, тут використовується корельований підзапит. Різниця лише в тому, що в основному запиті будуть відібрані ті рядки з таблиці постачальників, для яких вкладений підзапит не видає жодного рядка.

**Приклад29**. Отримати імена постачальників, які поставляють усі деталі:

SELECT DISTINCT PNAME FROM P

WHERE NOT EXIST (SELECT \*

FROM D

WHERE NOT EXIST

(SELECT \*

FROM PD

WHERE

PD.DNUM = D.DNUM AND PD.PNUM = P.PNUM));

Зауваження. Даний запит містить два вкладених підзапити та реалізує реляційну операцію *ділення відношень*.

Самий внутрішній підзапит параметризований двома параметрами (D.DNUM, P.PNUM) і має такий сенс: відібрати усі рядки, які містять дані про поставки постачальника з номером PNUM деталі з номером DNUM. Заперечення NOT EXIST говорить про те, що даний постачальник не поставляє дану деталь. Зовнішній до нього підзапит, який сам є вкладеним і параметризованим параметром P.PNUM, має таки сенс: відібрати список деталей, які не постачаються постачальником PNUM. Заперечення NOT EXIST говорить про те, що для постачальника з номером PNUM не повинно бути деталей, які б не поставлялися цим постачальником. Це в точності означає, що у зовнішньому запиті відбираються лише постачальники, що поставляють усі деталі.