**Лекція №8. Оператор Select.**

Основний SQL-вираз для вибірки даних. Оператор **SELECT\_FROM**

Основний SQL-вираз для вибірки даних, має вигляд:

***SELECT*** списокСтовпців ***FROM*** списокТаблиць;

Оператори SELECT (вибрати) і FROM (з) в SQL-виразі, що визначають вибірку даних, є обов’язковими, тобто жоден з них не можна пропустити. В результаті виконання цього запиту створюється віртуальна таблиця, що містить зазначені стовпці і всі записи вихідної таблиці.

Оператор SELECT здійснює проекцію відношення, зазначеного у виразі FROM, на задану множину атрибутів (стовпців), зазначеного у виразі SELECT. Так, наприклад, якщо вихідна таблиця R містить стовпці A1, А2, ..., An (іншими словами, таблиця представляє деяке відношення R (A1, A2, ..., An) над атрибутами А1, А2, ..., An ), то оператор: SELECT Al, A2, …, Ak FROM R; реалізує проекцію R [A1, A2, …, Ak] цього відношення на атрибути А1, А2, …, Ak (k = 1, 2, ..., k).

У виразі FROM вказується список імен таблиць бази даних, з яких потрібно вибрати дані. У простому випадку списокТаблиць містить лише одне ім’я таблиці. Якщо ж таблиць декілька, то їх імена у списку розділяються комами. Якщо у виразі FROM вказано більше однієї таблиці, то результатна таблиця виходить з декартового добутку, перерахованих в списку таблиць. Список стовпців – це перелік імен стовпців, розділених комою, як вони визначені в таблиці, зазначеної у виразі FROM.

У теорії множин, дека́ртів добу́ток (прями́й добу́ток) двох множин X та Y — це **множина усіх можливих впорядкованих пар, у яких перший компонент належить множині X, а другий — множині Y**.

Простим прикладом оператора SQL може служити:

SELECT AUID FROM Authors;

Зрозуміло, можна вказати всі або тільки деякі стовпці. Якщо ви хочете отримати всі стовпці таблиці, то замість списку стовпців достатньо вказати символ (\*). Якщо у виразі FROM зазначено кілька таблиць, то у виразі SELECT імена стовпців повинні містити префікси, що вказують, до якої саме таблиці вони відносяться. Префікс відокремлюється від імені стовпця крапкою.

Оператор SELECT, що відображає всі стовпчики таблиці Authors, виглядає в такий спосіб:

SELECT \* FROM Authors;

Якщо нам потрібно вибрати тільки декілька стовпців:

SELECT Name, AUID FROM Authors;

Коли в запиті використовується більше однієї таблиці, хорошою практикою програмування є написання перед іменем стовпчика ім’я таблиці і розміщення між ними точки (.). Доти, поки ім’я стовпчика залишається унікальним серед імен інших стовпчиків, що беруть участь у запиті, SQL не вимагає від вас обов’язкової вказівки імені таблиці. Проте, використання конструкції “таблиця.Стовпчик” є доброю навичкою.

Заголовки стовпців в результатній таблиці можна перевизначити, призначивши для них так звані псевдоніми. Для цього в списку стовпців після відповідного стовпця слід написати вираз виду: ***AS*** заголовок\_стовпця. Наприклад:

SELECT Name As NameAuthors, DateofBirth As Birthday FROM Authors;

Псевдоніми також можна задати і для кожної таблиці після ключового слова FROM. Для цього достатньо вказати псевдонім через пробіл відразу після імені відповідної таблиці. Псевдоніми таблиць, коротші, ніж їхні імена, їх зручно використовувати в складних запитах. Наприклад:

SELECT Т1. Name, Т2.Title FROM Authors Т1, Titles Т2;

**Оператори для уточнення запиту. Порядок виконання операторів SQL-виразу**

Основний SQL-вираз для вибірки даних, має вигляд:

***SELECT*** списокСтовпців ***FROM*** списокТаблиць;

Такий запит повертає таблицю, отриману з зазначеної в операторі FROM (або з декартового добутку зазначених таблиць, якщо їх декілька), шляхом виділення в ній тільки тих стовпців, які визначені в операторі SELECT.

Для уточнення запиту на вибірку даних служить ряд додаткових операторів:

- WHERE (де) – вказує записи, які повинні увійти в результатну таблицю (фільтр записів);

- GROUP BY (групувати по) – групує записи по значеннях певних стовпців;

- HAVING (що мають, за умови) – вказує групи записів, які мають увійти до результатної таблиці (фільтр груп);

- ORDER BY (сортувати по) – сортує (впорядковує) записи.

Ці оператори не є обов’язковими. Їх можна зовсім не використовувати, або використовувати лише деякі з них, або всі одразу. Якщо застосовуються декілька операторів, то в SQL-виразі вони виконуються у певному порядку.

Запит даних з таблиці із використанням всіх перерахованих операторів уточнення запиту має наступний вигляд:

***SELECT*** списокСтовпців ***FROM*** Ім’яТаблиці

***WHERE*** УмоваПошуку

***GROUP BY*** стовпецьГрупування

***HAVING*** УмоваПошуку

***ORDER BY*** УмоваСортування;

Порядок подання операторів в SQL-виразі не співпадає з порядком їх виконання. Отже, перераховані оператори SQL-виразу виконуються в наступному порядку, передаючи один одному результат у вигляді таблиці:

- FROM – вибирає таблицю з бази даних; якщо зазначено декілька таблиць, то виконується їх декартовий добуток і результуюча таблиця передається для обробки наступному оператору;

- WHERE – з таблиці вибираються записи, що відповідають умові пошуку, і відкидаються всі інші;

- GROUP BY – створюються групи записів, відібраних за допомогою оператора WHERE (якщо він присутній в SQL-виразі); кожна група відповідає якому-небудь значенню стовпця групування. Стовпець групування може бути будь-яким стовпцем таблиці, заданої в операторі FROM, а не тільки тим, що вказаний в SELECT;

- HAVING – обробляє кожну із створених груп записів, залишаючи тільки ті з них, які задовольняють умові пошуку; цей оператор використовується тільки разом з оператором GROUP BY;

- SELECT – вибирає з таблиці, отриманої в результаті використання перерахованих операторів, тільки вказані стовпці;

- ORDER BY – сортує записи таблиці. При цьому в умові сортування можна звертатися лише до тих стовпців, які вказані в операторі SELECT.

**Оператор WHERE**

За допомогою оператора WHERE Ви можете контролювати вміст результуючого набору оператора SELECT\_FROM. При цьому, оператор WHERE ви можете використовувати в одному із таких двох випадків:

- використовувати WHERE для обмеження вмісту результуючого набору;

- використовувати WHERE для зв’язування двох або декількох таблиць у єдиний результуючий набір.

**Використання WHERE для обмеження результуючого набору.** Умови пошуку в операторі WHERE (де) є логічними виразами, тобто приймають одне з двох можливих значень – true або false.

При складанні логічних виразів використовуються спеціальні ключові слова і символи операцій порівняння, які називають предикатами. Вирази з оператором WHERE використовуються не тільки при вибірці даних (тобто з оператором SELECT), але і при вставці, модифікації і видаленні записів. Нижче наведено список всіх предикатів:

- предикати порівняння: (=), (<), (>), (<>), (<=), (> =);

- BETWEEN;

- IN, NOT IN;

- LIKE, NOT LIKE;

- IS NULL;

- ALL, SOME, ANY;

- EXISTS;

- UNIQUE;

- DISTINCT;

- OVERLAPS;

- MATCH;

- SIMILAR.

У виразі WHERE ви можете використовувати декілька логічних операторів AND і OR

**Предикати порівняння: (=), (<), (>), (<>), (<=), (> =).**

SELECT Name, Region FROM Publishers WHERE Region = ‘Vol’

Приведений вище оператор SQL повертає в результуючому наборі *підмножину* даних. Це означає, що результуючий набір містить не всі рядки таблиці Publishers, а тільки ті з них, що задовольняють критерію виразу WHERE.

**BETWEEN.** Предикат BETWEEN (між) дозволяє задати вираз перевірки входження якого-небудь значення в діапазон, який визначається граничними значеннями. Наприклад:

SELECT PubID, Name, Region, City FROM Publishers WHERE PubID BETWEEN 10 AND 15;

Тут ключове слово AND являє собою логічний вираз (логічне "І"). Граничні значення (у прикладі це 10 і 15) входять в діапазон. Причому перше граничне значення повинно бути не більше другого.

Еквівалентним наведеному є вираз з предикатами порівняння:

WHERE PubID > = 10 AND PubID <= 15;

Крім даних числового типу, у виразах з BEETWEEN можна використовувати дані наступних типів: символьні, бітові, дати-часу. Так наприклад, щоб вибрати записи, в яких імена авторів знаходяться в діапазоні від А до Н, можна використовувати такий вираз:

SELECT Name FROM Avtors WHERE Name BETWEEN ‘A’ AND ‘Н’;

**IN і NOT IN.** Предикати IN (в) і NOT IN (не в) використовуються для перевірки входження якого-небудь значення в заданий список значень. Наприклад, якщо вам потрібний список усіх видавців у областях Волинській, Львівській і Київській, то у виразі WHERE ви можете використовувати ключове слово IN, за яким у квадратних дужках повинні випливати необхідні значення, розділені комами.

SELECT PubID, Name, City, Region FROM Publishers WHERE Region IN (‘Vol’, ‘Lviv’, ‘Kyiv’)

Якщо потрібно отримати дані про всіх видавців крім тих що проживають в областях Волинській, Львівській й Київській, то можна використовувати предикат NOT IN:

SELECT PubID, Name, City, Region FROM Publishers WHERE Region NOT IN (‘Vol’, ‘Lviv’, ‘Kyiv’)

**LIKE і NOT LIKE*.*** Предикати LIKE (схожий) і NOT LIKE (не схожий) використовуються для перевірки часткової відповідності символьних рядків. Наприклад, вам потрібно одержати всі рядки, у яких поле Region містить букву “і” у будь-якій позиції.

Критерій часткової відповідності задається за допомогою двох символів-масок: значка відсотка (%) і підкреслення (\_). Знак відсотка означає будь-який набір символів, в тому числі і порожній, а символ підкреслення – будь-який одиночний символ. Тоді ви можете використовувати такий оператор SQL SELECT\_FROM.

SELECT PubID, Name, City, Region FROM Publishers WHERE Region LIKE (‘%і%’)

Якщо вам потрібно вибрати записи про видавців, що проживають в Луцьку, на вулиці Потапова, а стовпець Адреса містить назву вулиці, номер будинку і номер квартири, то для цього підійде наступний вираз:

SELECT PubID, Name, Address FROM Publishers WHERE Address LIKE ‘%Potapova%’ AND City=‘Lutsk’;

Якщо ви хочете вибрати всіх видавців, що проживають в Луцьку, крім тих, що проживають на вулиці Потапова, то скористайтеся таким виразом:

SELECT PubID, Name, Address FROM Publishers WHERE Address NOT LIKE ‘%Potapova%’ AND City=‘Lutsk’;

Можливо, вам буде потрібно вибрати записи, що містять символи відсотка та/або підкреслення. Тоді необхідно, щоб такі символи сприймалися інтерпретатором SQL не як символи-маски. Щоб знак відсотка або підкреслення сприймався буквально, перед ним необхідно вказати спеціальний керуючий символ. Цей символ можна визначити довільно, лише б він не зустрічався в якості елемента даних. У наступному прикладі показано, як це можна зробити:

… WHERE percent LIKE ‘20 #% ‘ESCAPE’ # ‘;

Тут за ключовим словом ESCAPE вказується символ, який використовується в якості керуючого. Таким же способом можна відключити і сам керуючий символ.

У Microsoft Access в якості символів-масок використовується знак зірочка (\*), що означає будь-який набір символів і знак запитання (?) – будь-який одиночний символ.

**IS NULL.** Предикат IS NULL використовується для виявлення записів, в яких той чи інший стовпець не має значення. Наприклад, для отримання записів про видавців, для яких не вказана адреса, можна використовувати наступний вираз:

SELECT PubID, Name, Address FROM Publishers WHERE Address IS NULL;

Для отримання записів, в яких стовпець Адреса містить визначені значення (тобто відмінні від NULL), можна використовувати аналогічний вираз, але з логічним оператором NOT (не):

SELECT PubID, Name, Address FROM Publishers WHERE Address IS NOT NULL;

Не слід використовувати предикати порівняння з NULL, такі як

Адреса = NULL.

**Логічні оператори**. Логічні вирази в операторах WHERE можуть бути складними, тобто складатися з двох і більше простих виразів, з’єднаних між собою логічними операторами AND і/або OR. Оператор AND виконує роль логічного оператора І, а оператор OR – оператора АБО. Логічний оператор NOT застосовується до одного виразу (можливо і до складного), розташованому праворуч від нього. Цей оператор міняє значення виразу на протилежне.

Наприклад, якщо вам потрібний список усіх видавців у областях Волинській, Львівській і Київській.

SELECT PubID, Name, City, Region FROM Publishers WHERE Region= ‘Vol’ OR Region= ‘Lviv’ OR Region= ‘Kyiv’

Зверніть увагу, що тут використовується логічний оператор OR (АБО), а не AND (І), оскільки нам потрібні видавці, які проживають або в ‘Vol’, або в ‘Lviv’ або в ‘Kyiv’. Якби замість оператора OR ми використали AND, то отримали б порожню таблицю, так як у вихідній таблиці немає жодного запису, в якій один і той же стовпець мав би різні значення.

Наступний SQL-вираз еквівалентний розглянутому раніше. Він заснований на використанні оператора IN :

SELECT PubID, Name, City, Region FROM Publishers WHERE Region IN (‘Vol’, ‘Lviv’, ‘Kyiv’);

Якщо потрібно отримати дані про всіх видавців, які не проживають ні в ‘Vol’, ні в ‘Lviv’, ні в ‘Kyiv’, то можна використати такий SQL-вираз:

SELECT PubID, Name, City, Region FROM Publishers WHERE NOT (Region= ‘Vol’ OR Region= ‘Lviv’ OR Region= ‘Kyiv’ );

Цей вираз еквівалентний наступним двом:

SELECT PubID, Name, City, Region FROM Publishers WHERE Region<> ‘Vol’ OR Region<>‘Lviv’AND Region<> ‘Kyiv’;

і:

SELECT PubID, Name, City, Region FROM Publishers WHERE Region NOT IN (‘Vol’, ‘Lviv’, ‘Kyiv’);

**Використання WHERE для зв’язування двох або декількох таблиць у єдиний результуючий набір**. У виразі WHERE можна порівнювати стовпчики, що належать різним таблицям. При цьому ви можете задати критерій, що буде зв’язувати дві або декілька таблиць у єдиний результуючий набір. Така форма виразу WHERE має такий синтаксис:

***SELECT*** table1.column, table2.column ***FROM*** tablel, table2

***WHERE*** tablel.column = table2.column

Тут table1 і table2 – різні таблиці однієї бази даних, а column – стовпчик, що міститься в обох таблицях.

SELECT Titles.Title, Publishers.Name FROM Publishers, Titles WHERE Publishers.PubID = Titles.PubID;

Наведений вище запит SQL створює результуючий набір, що містить назву книги й ім’я її видавця. Це досягається за допомогою виразу WHERE, що вказує SQL вибрати з обох таблиць тільки ті рядки, у яких поля PubID збігаються. Майте на увазі, що все це робиться без яких-небудь програмних процедур, спеціального індексування або команд сортування. Всю роботу бере на себе SQL.

Порівнювані у виразі WHERE стовпчики (Publishers. PubID і Titles. PubID) не включені у вираз SELECT нашого оператора. Для того, щоб використовувати стовпчик у виразі WHERE, необов’язково включати його в вираз SELECT, оскільки цей стовпчик вже існує у фізичній таблиці.

Об’єднання таблиць за допомогою виразу WHERE завжди повертає неоновлюваний результуючий набір. Ви не зможете змінювати дані в колонках представлення, отриманого таким способом. Для того, щоб зв’язати таблиці й одночасно зберегти можливість відновлення фізичних таблиць, що лежать в основі об’єднаного представлення, ви повинні використовувати вираз JOIN, який ми розглянемо пізніше.

В одному операторі SELECT\_FROM ви можете об’єднувати версії вирази WHERE що зв’язують і обмежують:

SELECT Titles.Title, Publishers.Name FROM Titles, Publishers WHERE Titles.PubID = Publishers.PubID AND Publishers PubID BETWEEN 5 AND 10;

Наведений вище оператор SQL вибирає тільки ті записи, для яких значення колонок PubID збігаються і знаходяться між 5 і 10.

За допомогою виразу WHERE можна зв’язувати більш двох таблиць. При цьому стовпчик, що зв’язує таблиці tablel і table2, не обов’язково повинний збігатися зі стовпчиком, що зв’язує таблиці table2 і table3. Наприклад:

SELECT Titles.PubID, Titles.Title, Publishers.Name, Authors.Name FROM Titles, Publishers, Authors WHERE Titles.PubID = Publishers.PubID AND Titles.AUID = Authors.AUID

У цьому прикладі таблиці Publishers і Titles пов’язані за допомогою стовпчика PubID, а таблиці Titles і Authors – за допомогою стовпчика AUID. Після установки зв’язку обрані стовпчики відображаються в результуючому наборі.

**Агрегатні функції мови SQL**

Стандарти мови SQL визначають базовий набір функцій, що повинен бути присутнім у всіх SQL-сумісних системах. Ці функції називаються агрегатними і використовуються для швидких обчислень над збереженими в стовпчиках числовими даними.

- COUNT – повертає число рядків і використовується для визначення загального числа рядків у результуючому наборі (це єдина агрегатна функція, що можна застосовувати для нечисловых стовпчиків);

- SUM – повертає суму всіх значень у стовпчику;

- AVG – повертає середнє для всіх значень у стовпчику;

- МАХ – повертає найбільше значення в стовпчику;

- MIN – повертає найменше значення в стовпчику.

Такий оператор SQL демонструє всі п’ять агрегатних функцій.

SELECT COUNT(Units) AS UnitCount, AVG(Units) AS UnitAvg, SUM(Units) AS UnitSum, MIN(Units) AS UnitMin, MAX(Units) AS UnitMax FROM BookSales;

У одному операторі SELECT\_FROM можна одночасно використовувати вираз WHERE і функції, що агрегують. Такий запит показує, як за допомогою виразу WHERE можна обмежити кількість рядків, що беруть участь у розрахунках із використанням функцій, що агрегують.

SELECT COUNT(Units) AS UnitCount, AVG(Units) AS UnitAvg, SUM(Units) AS UnitSum, MIN(Units) AS UnitMin, MAX(Units) AS UnitMax FROM BookSales WHERE Qtr =1;

**Оператор GROUP BY**

Оператор GROUP BY (групувати по) використовується для групування записів за значеннями одного або декількох стовпців. Якщо в SQL-виразі використовується оператор WHERE, то оператор GROUP BY знаходиться і виконується після нього. Якщо WHERE не використовується, то групуються всі записи вихідної таблиці.

Припустимо, що на основі таблиці Publishers треба згрупувати дані по регіонах. Для цього можна скористатися наступним SQL-виразом:

SELECT Name, Company, Region FROM Publishers GROUP BY Region;

При цьому записи з однаковими назвами регіонів розташовані поруч один з одним (в одній групі).

Наприклад, ви хочете створити набір даних, що містить назву книги і сумарна кількість її проданих примірників. Це можна зробити за допомогою такого запиту:

SELECT Title, SUM(Units) AS UnitsSold FROM BookSales, Titles Where Titles.ISBN= BookSales.ISBN GROUP BY Titles.ISBN;

Вираз GROUP BY потребує, щоб усі числові стовпчики в списку полів вибірки оператора SELECT були аргументами якої-небудь агрегатної функції мови SQL (SUM, AVG, MIN, MAX, COUNT). Крім того, у цьому випадку в списку полів вибірки оператора SELECT не можна використовувати зірочку (\*).

Оператор GROUP BY збирає записи в групи і впорядковує (сортує) групи за алфавітом (точніше, за ASCII-кодами символів).

**Оператор HAVING**

Оператор HAVING зазвичай використовується разом з оператором групування GROUP BY і задає фільтр записів в групах. Оператор HAVING діє точно так само, як і оператор WHERE, із тієї лише різницею, що дія оператора HAVING поширюється на результуючі стовпчики, створені виразом GROUP BY, а не на стовпчики фізичної таблиці.

Нехай, ви хочете одержати список назв усіх книг, проданих за рік у кількості більш 100 примірників.

SELECT Title, SUM(Units) AS UnitsSold FROM BookSales, Titles Where Titles.ISBN=BookSales.ISBN GROUP BY Titles.ISBN HAVING SUM(Units)>100;

Стовпчики, використані у виразі HAVING, не обов’язково повинні бути присутнім у списку полів вибірки оператора SELECT. Структура виразу HAVING підпорядковується тим же правилам, що і структура виразу WHERE. У ньому ви можете використовувати логічні оператори AND, OR і NOT. Такий запит SQL повертає суми продажів по всіх книгах, проданих у кількості більш 100 примірників, найменування яких містять у другій позиції букву “а”.

SELECT Title, SUM(Sales) AS SalesAmt FROM BookSales, Titles Where Titles.ISBN=BookSales.ISBN GROUP BY Titles.ISBN HAVING SUM(Units)>100 AND Title Like ‘\_a%’;

Якщо в SQL-виразі оператора GROUP BY немає, то оператор HAVING застосовується до всіх записів, що повертається оператором WHERE. Якщо ж відсутній і WHERE, TO HAVING діє на всі записи таблиці.

**Оператор ORDER BY**

Оператор SELECT\_FROM повертає записи в результуючий набір у тому порядку, у якому вони зустрічаються у фізичній таблиці даних. Але якщо ви захочете показати їх у відсортованому виді, то для цього ви можете скористатися оператором ORDER BY.

Цей оператор сортує рядки всієї таблиці або окремих її груп (у випадку використання оператора GROUP BY). Якщо у виразі запиту оператора GROUP BY немає, то оператор ORDER BY розглядає всі записи таблиці як одну групу.

Додавання ключових слів ASC або DESC після імені поля у виразі ORDER BY вказує на зростаючий або спадаючий порядок сортування відповідного стовпчика. Якщо порядок не заданий, припускається сортування по зростанню. Тобто значення ASC прийнято за замовчуванням, тому якщо необхідне сортування, наприклад, в алфавітному порядку, то спеціально вказувати порядок не вимагається;

У такому прикладі показано, як відобразити записи таблиці Authors, відсортовані по полю Name в спадаючому порядку.

SELECT \* FROM Authors ORDER BY Name DESC;

У виразі ORDER BY може бути зазначено більше одного поля. SQL створить результуючий набір, що відбиває сукупний порядок сортування, зазначений у виразі ORDER BY.

Наприклад:

SELECT Region, City FROM Publishers ORDER BY Region DESC, City ASC;

В даному запиті ми сполучили в результуючому наборі можливість зміни порядку рядків із можливістю зміни порядку стовпчиків.

**Вираз DISTINCT**

Відразу за оператором SELECT до списку стовпців можна використовувати ключові слова ALL (всі) і DISTINCT (відрізняються), які вказують, які записи подавати в результуючій таблиці. Якщо ці ключові слова не використовуються, будуть вибратися всі записи, що також відповідає використанню ключового слова ALL. У випадку використання DISTINCT в результуючій таблиці подаються тільки унікальні записи. При цьому якщо у вихідній таблиці знаходяться декілька ідентичних записів, то з них вибирається тільки перший.

Наприклад, ви хочете одержати список усіх покупців, що зробили хоча б одне замовлення. Проблема полягає в тому, що деякі клієнти зробили декілька замовлень, а ви не хочете бачити ті самі імена в результуючому наборі. Щоб виключити дублікати, можна використовувати ключове слово DISTINCT.

SELECT DISTINCT Name FROM Buyers, BookSales WHERE Buyers. BuyerID – BookSalesID. BuyerID;

Якщо список полів в операторі SELECT містить у собі більше одного стовпчика, то усі вони використовуються для оцінки унікальності рядка.

SELECT DISTINCT Surname, Name, Lastname FROM Buyers, BookSales WHERE Buyers. BuyerID – BookSalesID. BuyerID;

Зверніть увагу, що перший запит для кожного значення поля Name таблиці даних повертає по одному запису. Другий запит повертає для кожного значення поля Name декілька записів, оскільки усі вони відрізняються один від одного значеннями полів Surname та Lastname.

У Microsoft Access крім ключових слів ALL та DISTINCT після SELECT можна використовувати ключове слово ТОР з додатковими параметрами, яке використовуються для обмеження числа записів у результуючому наборі. Припустимо, що ви хочете одержати список п’ятьох кращих бестселерів із таблиці продажів. У цьому випадку ви можете скористатися виразом TOP n, що повертає перші n записів. Якщо у вас є два записи з однаковими значеннями, то SQL поверне обидва записи. Якби в нашому прикладі з книгами п’ятий і шостий записи були однакові, то в результуючий набір потрапить шість записів, а не п’ять.

Застосовуючи вираз TOP n, ви зобов’язані використовувати вираз ORDER BY, гарантуючи тим самим, що результуючий набір буде відсортований. У противному випадку ви одержите довільний набір, тому що SQL спочатку виконує вираз ORDER BY, а потім уже вибирає запитані вами перші n записів. Зрозуміло, що без виразу ORDER BY ви не одержите очікуваного результату. Якщо в запиті є присутнім і оператор WHERE, то SQL виконує ці оператори в такому порядку: WHERE, ORDER BY, TOP n. Як бачите, відсутність у цьому випадку виразу ORDER BY по всій можливості призведе до появи сміття в результуючому наборі.

SELECT ТОР 5 \* FROM BookSales ORDER BY Sales DESC;

Результуючий набір містить перші n записів вибірки, відсортованої по спаданню або по зростанню в залежності від наявності ключових слів DESC або ASC.

Вираз TOP n PERCENT повертає не n, a n відсотків перших записів.

SELECT TOP 5 PERCENT \* FROM BookSales ORDER BY Sales;

**СКЛАДНІ ЗАПИТИ В SQL**

У SQL складні запити є комбінацією простих SQL-запитів. Кожен простий запит в якості відповіді повертає набір записів (таблицю), а комбінація простих запитів повертає результат тих чи інших операцій над відповідями на прості запити.

У SQL складні запити отримуються з інших запитів наступними способами:

- застосуванням до SQL-запитів операторів об’єднання і з’єднання наборів записів, що повертаються запитами. Ці оператори називають теоретико-множинними або реляційними.

- вкладенням SQL-виразу запиту в SQL-вираз іншого запиту. Перший з них називають підзапитом, а другий – зовнішнім або основним запитом;

**Теоретико-множинні операції**

Над наборами записів, що містяться в таблицях бази даних і/або повертаються запитами, можна здійснювати теоретико-множинні операції, такі як декартовий добуток, об’єднання, перетин і віднімання.

**Декартовий добуток наборів записів**.

***SELECT*** списокСтовпців ***FROM*** Tl, T2, ... Тn;

повертає набір записів, отриманий в результаті декартового добутку наборів записів з таблиць Tl, T2, ... Тn. Таблиці, зазначені в операторі FROM, можуть бути як таблицями бази даних, так і віртуальними таблицями.

Іноді потрібно отримати декартовий добуток таблиці самої на себе. У цьому випадку необхідно застосувати різні псевдоніми для цієї таблиці, наприклад:

SELECT списокСтовпців FROM Mytab Tl, Mytab T2;

Спроба виконати запит:

SELECT списокСтовпців FROM Mytab, Mytab;

призведе до помилки.

У списку стовпців слід використовувати повні імена стовпців, використовуючи псевдоніми таблиць, або символ (\*), якщо потрібно отримати всі стовпці.

Наприклад, потрібно знайти прізвища авторів, що мають однакові дні народження. При з’єднанні таблиці з її ж копією вводяться псевдоніми (аліаси) таблиці. Запит для пошуку прізвищ авторів, що мають однакові дні народження, виглядає таким чином:

SELECT FIRST. Name, SECOND.Name

FROM Authors FIRST, Authors SECOND

WHERE FIRST.Date\_of\_birth = SECOND. Date\_of\_birth

У цьому запиті введені два псевдоніма для однієї таблиці Authors, що дозволяє коректно задати вираз, що зв’язує дві копії таблиці. Щоб виключити повторення рядків у виведеному результаті запиту через повторне порівняння однієї і тієї ж пари авторів, необхідно задати порядок проходження для двох значень так, щоб одне значення було менше, від іншого, що робить предикат асиметричним.

SELECT FIRST. Name, SECOND.Name

FROM Authors FIRST, Authors SECOND

WHERE FIRST.Date\_of\_birth = SECOND. Date\_of\_birth

AND FIRST. Name < SECOND.Name

Для декартового добутку в SQL також допустимий синтаксис з ключовими словами CROSS JOIN (перехресне з’єднання):

***SELECT*** списокСтовпців ***FROM*** Mytab Tl ***CROSS JOIN*** Mytab T2;

Розглянуті вирази працюють в повнофункціональних базах даних. У Microsoft Access для отримання декартового добутку можливе використання виразу (SELECT списокСтовпців FROM Tl, T2, ... Тn), тільки якщо всі таблиці в списку мають різні імена. Якщо потрібно декартовий добуток таблиці самої на себе, то у виразі (SELECT списокСтовпців FROM Mytab Tl, Mytab T2) Access автоматично додасть ключове слово AS перед кожним псевдонімом. Спроба використання ключових слів CROSS JOIN В Access призведе до помилки.

Об’єднання наборів записів (UNION). Нерідко потрібно об’єднати записи двох або більше таблиць зі схожими структурами в одну таблицю. Інакше кажучи, до набору записів, що повертається одним запитом, потрібно додати записи, що повертаються іншим запитом. Для цього є оператор UNION (об’єднання):

Зaпит1 UNION Запит2;

При цьому в результатній таблиці залишаються тільки записи, які відрізняються. Щоб зберегти в ній всі записи, після оператора UNION слід написати ключове слово ALL.

Оператор UNION можна застосовувати тільки до таблиць, які задовольнять наступним умовам сумісності:

- кількості стовпців поєднуваних таблиць повинні бути рівні;

- дані у відповідних стовпцях поєднуваних таблиць повинні мати сумісні типи.

При цьому, імена відповідних стовпців і їх розміри можуть бути різними. Важливо, щоб кількості стовпців були рівні, а їх типи були сумісні. Щоб об’єднати набори записів з несумісними за типом даних стовпцями, слід застосувати функцію перетворення типу даних CAST.

Наприклад, якщо у вас є таблиця видавців і таблиця покупців, то можна сформувати список усіх видавців і покупців, що мешкають у Волинськівй області. Можна спочатку написати оператор SQL для вибірки рядків із таблиці видавців, а потім за допомогою другого оператора SQL вибрати рядки з таблиці покупців. Після чого об’єднати обидва цих оператори в одну фразу мови SQL, скориставшись ключовим словом UNION, – і ви одержите єдиний результуючий набір, що містить результати обох запитів.

Такий оператор SQL створює результуючий набір, що містить усіх видавців і покупців, що мешкають у Волинській області.

SELECT Name, City, Region, Zip FROM Publishers WHERE Region=‘Vol’

UNION

SELECT Name, City, Region, Zip FROM Buyers WHERE Region=‘ Vol’;

При формуванні результуючого набору оператор UNION використовує імена стовпчиків першого запиту і при необхідності перетворить тип даних, обраних у відповідності з другим запитом.

Оператор UNION можна також використовувати для об’єднання запитів до однієї і тої ж таблиці даних. Такий оператор SQL повертає в одному результуючому наборі лідерів і аутсайдерів продажів:

SELECT SUM(Sales) AS TotSales, Title FROM BookSales

GROUP BY Title HAVING SUM(Sales)>4000

UNION

SELECT SUM(Sales) AS TotSales, Title FROM BookSales

GROUP BY Title HAVING SUM(Sales)<1000

ORDER BY TotSales;

Коли потрібно об’єднати записи двох таблиць, що мають однойменні стовпці з сумісними типами, можна виконати оператор **UNION CORRESPONDING** (об’єднання відповідних):

SELECT \* FROM Таблица1

UNION CORRESPONDING (списокСтовпців)

SELECT \* FROM Таблиця 2;

Після ключового слова CORRESPONDING МОЖНА вказати стовпці, наявні одночасно в обох таблицях. Якщо цей список стовпців не зазначений, то передбачається список всіх імен.

Переріз наборів записів (**INTERSECT**). Переріз двох наборів записів здійснюється за допомогою оператора INTERSECT (переріз), що повертає таблицю, записи в якій містяться одночасно в двох наборах:

Запит1 INTERSECT Запит2;

При цьому в результатній таблиці залишаються тільки запису, які відрізняються.

Щоб зберегти у ній повторювані записи, після оператора INTERSECT слід написати ключове слово ALL.

Як і в операторі UNION, В INTERSECT можна використовувати ключове слово CORRESPONDING. У цьому випадку вихідні набори даних не обов’язково повинні бути сумісними для об’єднання, але відповідні стовпці повинні мати однакові тип і довжину.

Якщо в операторі INTERSECT використовується ключове слово CORRESPONDING, то після нього в круглих дужках можна вказати імена стовпців, значення яких повинні перевірятися на рівність при виконанні операції перетину.

Віднімання наборів записів (**EXCEPT**). Для отримання записів, що містяться в одному наборі і відсутні в іншому, служить оператор EXCEPT (за винятком):

Запит1 ЕХЕРТ Запит2;

За допомогою цього оператора з першого набору видаляються записи, що входять у другий набір. Так само, як і в операторах UNION і INTERSECT, в операторі EXCEPT можна використовувати ключове слово CORRESPONDING.

**Операції з’єднання**

Операції з’єднання наборів записів повертають таблиці, записи в яких виходять шляхом деякої комбінації записів з’єднуваних таблиць. Для цього використовується оператор JOIN (поєднати).

Існують кілька різновидів з’єднання, яким відповідають певні ключові слова, що додаються до слова JOIN. Так, наприклад, декартовий добуток є операцією перехресного з’єднання. У SQL-виразі для позначення цієї операції використовується оператор **CROSS JOIN**. Втім, декартовий добуток можна отримати і без використання цих ключових слів. В основі будь-якого з’єднання наборів записів лежить операція їх декартового добутку.

**Природне з’єднання (NATURAL JOIN).** Розглянемо суть операції природного з’єднання на типовому прикладі. Нехай нам потрібно створити результуючий набір, що містить назву книги й ім’я її видавця. Зрозуміло, що в отриманому декартовому добутку нас можуть цікавити не всі записи, а тільки ті, в яких ідентичні стовпці мають однакову значення. Така таблиця і буде природним з’єднанням таблиць.

Вона отримається за допомогою наступного запиту

SELECT Titles.Title, Publishers.Name FROM Publishers, Titles WHERE Publishers.PubID = Titles.PubID;

Даний запит можна переписати, використовуючи псевдоніми:

SELECT T1.Title, Т2.Name FROM Titles Т1, Publishers Т2 WHERE Т1.PubID = Т2.PubID;

Еквівалентний запит з оператором NATURAL JOIN виглядає наступним чином:

SELECT T1.Title, Т2.Name FROM Titles Т1 NATURAL JOIN Publishers Т2;

У Microsoft Access оператор NATURAL JOIN не підтримується. Замість нього використовується INNER JOIN (внутрішнє з’єднання) і ключове слово ON (при), за яким слідує умова відбору записів.

SELECT T1.Title, Т2.Name FROM Titles Т1 INNER JOIN Publishers Т2 ON Т1.PubID = Т2.PubID;

INNER JOIN також можна застосовувати і в повнофункціональних базах даних.

При природному з’єднанні, коли використовується оператор NATURAL JOIN, перевіряється рівність всіх однойменних стовпців з’єднуваних таблиць.

Умовне з’єднання (JOIN. .. ON). Умовне з’єднання схоже на з’єднання з умовою рівності. Відмінність полягає в тому, що в якості умови може виступати будь який вираз, що записується після ключового слова ON (при), а не WHERE. Якщо умова виконується для поточного запису декартового добутку, то він входить у результатну таблицю.

Наприклад, створити результуючий набір, що містить назву книги й ім’я її видавця, який проживає в Lutsk.

SELECT T1.Title, Т2.Name FROM Titles Т1 JOIN Publishers Т2 ON (Т1. PubID = Т2. PubID) AND (City=‘Lutsk’);

У Microsoft Access використовується оператор INNER JOIN ... ON. У повнофункціональних базах даних також допустиме ключове слово INNER

З’єднання по іменах стовпців (JOIN. .. USING). З’єднання по іменах стовпців схоже на природне з’єднання. Відмінність полягає в тому, що можна вказати, які однойменні стовпці повинні перевірятися. Нагадаємо, що в природному з’єднанні (NATURAL JOIN) перевіряються всі однойменні стовпці.

Припустимо, є дві таблиці Publishers та Buyers, які мають однойменні поля. Нехай нам потрібно вивести всі можливі комбінації прізвищ покупців і видавців, які проживають в одній і тій же області і в тому самому місті. Для цього слід певним чином з’єднати таблиці Publishers і Buyers. Природне з’єднання в цьому випадку не підійде, оскільки при ньому перевіряються всі однойменні стовпці. У випадку, якщо жоден видавець не є покупцем, і ми використаємо операцію природного з’єднання, то в результаті буде отримана порожня таблиця. Тому слід використовувати з’єднання по іменах стовпців. Це з’єднання може бути представлено так:

SELECT Publishers.Lastname Buyers.Lastname FROM Buyers JOIN Publishers USING (Region, City);

Очевидно, даний запит можна сформулювати інакше:

SELECT Publishers.Lastname Buyers.Lastname FROM Buyers, Publishers WHERE Buyers.Region= Publishers. Region AND Buyers. City = Publishers.City;

У Microsoft Access оператор JOIN ... USING не підтримується. Замість нього можна використовувати INNER JOIN ... ON. Розглянений запит в Access можна сформулювати так:

SELECT Publishers.Lastname Buyers.Lastname FROM Buyers INNER JOIN Publishers ON Buyers.Region= Publishers. Region AND Buyers. City = Publishers.City;

**Зовнішні з’єднання.** Всі з’єднання таблиць, розглянуті дотепер, є внутрішніми. У всіх прикладах замість ключового слова JOIN можна писати **INNER JOIN** (внутрішнє з’єднання). З таблиці, одержаної при внутрішньому з’єднанні, відкидаються всі записи, для яких немає відповідних записів одночасно в обох таблицях. При зовнішньому з’єднанні такі невідповідні записи зберігаються.

За допомогою спеціальних ключових слів LEFT OUTER, RIGHT OUTER, FULL та UNION, написаних перед JOIN, можна виконати відповідно ліве, праве, повне поєднання і об’єднання-з’єднання. У SQL-виразі запиту таблиця, зазначена ліворуч від оператора JOIN, називається лівою, а зазначена праворуч від нього – правою.

Ліве з’єднання (**LEFT OUTER JOIN**). При лівому зовнішньому з’єднанні невідповідні записи, наявні в лівій таблиці, зберігаються в результатній таблиці, а наявні в правій – видаляються. Порожні поля мають значення NULL.

У Microsoft Access ключове слово OUTER не допускається. Пишуть LEFT JOIN.

Еквівалентний запит, який не використовує ключові слова LEFT OUTER JOIN, виглядає досить громіздко!

Праве з’єднання (**RIGHT OUTER JOIN**). При правому зовнішньому з’єднанні невідповідні записи, наявні в правій таблиці, зберігаються в результатній таблиці, а наявні в лівій – видаляються. Ключовими словами в такому запиті є RIGHT OUTER JOIN.

У Microsoft Access RIGHT JOIN.

Повне з’єднання (**FULL JOIN**). Повне з’єднання виконує одночасно і ліве, і праве зовнішні з’єднання. Ключовими словами в запиті є FULL JOIN.

У Microsoft Access оператор FULL JOIN не підтримується.

Об’єднання-з’єднання (**UNION JOIN**). Операцію об’єднання-з’єднання ще називають об’єднання-злиття. Ключовими словами в запиті є UNION JOIN. При об’єднанні-з’єднанні створюється віртуальна таблиця, що містить всі стовпці обох вихідних таблиць. При цьому стовпці з лівої вихідної таблиці містять всі записи лівої таблиці, а в тих же записах встовпцях з правої вихідної таблиці містяться значення NULL. Аналогічно, стовпці з правої вихідної таблиці містять всі записи правої таблиці, а ці ж записи в стовпцях з лівої таблиці містять NULL. Загальна кількість записів у віртуальній таблиці дорівнює сумі кількості записів, наявних в обох вихідних таблицях.

**Підзапити**

Підзапит – це SQL-вираз, що починається з оператора SELECT, який міститься в умові оператора WHERE АБО HAVING для іншого запиту. Таким чином, підзапит – це запит на вибірку даних, вкладений в інший запит. Зовнішній запит, що містить підзапит, якщо тільки він сам не є підзапитом, не обов’язково повинен починатися з оператора SELECT. У свою чергу, підзапит може містити інший підзапит і т. д. При цьому спочатку виконується підзапит, що має найглибший рівень вкладення, і т. д. Часто, але не завжди, зовнішній запит звертається до однієї таблиці, а підзапит – до іншої. На практиці саме цей випадок найбільш цікавий.

**Прості підзапити**. Прості підзапити характеризуються тим, що вони формально ніяк не пов’язані з їх зовнішніми запитами. Це дозволяє спочатку виконати підзапит, результат якого потім використовується для виконання зовнішнього запиту.

Розглядаючи прості підзапити, слід виділити три часткових випадки:

- підзапити, що повертають єдине значення;

- підзапити, що повертають список значень з одного стовпця таблиці;

- підзапити, що повертають набір записів.

Розглянемо ці окремі випадки більш детально.

**Робота з єдиним значенням**. Припустимо, з таблиці продажем BookSales потрібно вибрати дані про книги для яких кількість проданих більше середнього значення. Це можна зробити за допомогою наступного запиту:

SELECT \* FROM BookSales WHERE Units> (SELECT АVG (Units) FROM BookSales);

У даному запиті спочатку виконується підзапит (SELECT АVG (Units) FROM BookSales). Він повертає єдине значення (а не набір записів) – середнє значення стовпця кількість проданих Units. Якщо сказати точніше, то даний підзапит повертає єдиний запис, що містить єдине поле. Далі виконується зовнішній запит, який виводить всі стовпці таблиці клієнти і записи, в яких значення стовпця Units більше значення, отриманого за допомогою підзапиту. Таким чином, спочатку виконується підзапит, а потім зовнішній запит, що використовує результат підзапиту.

Спроба виконати наступний простий запит призведе до помилки:

SELECT \* FROM BookSales WHERE Units> АVG (Units);

Вираз підзапиту обов’язково має бути укладений в круглі дужки.

**Робота зі списком значень з одного стовпця**. Підзапит, взагалі кажучи, може повертати декілька записів. Наприклад, нам потрібно вивести назви книг, які вже продавалися, тобто є відомості про продажі даних книг в таблиці BookSales.

SELECT Title FROM Titles WHERE ISBN IN (SELECT ISBN FROM BookSales);

Щоб в умові зовнішнього оператора WHERE можна було використовувати оператори порівняння, що вимагають одного значення, використовуються квантори, такі як ALL (наприклад …> ALL (підзапит) означає, що наше значення повинне бути більше всіх значень вибраних підзапитом ) і SOME (наприклад …> SOME (підзапит) означає, що наше значення повинне бути більше хоча б одного значення з значень вибраних підзапитом).

Вкладені підзапити можуть містити умови, зумовлені оператором WHERE. Зрозуміло, це один з можливих варіантів запитів, який повертає необхідні дані. Робота з набором записів У повнофункціональних базах даних підзапити можна використовувати не тільки в операторах WHERE І HAVING, але і в операторі FROM:

SELECT T.стовпець1, T.стовпець2, ... , T.стовпець n FROM (SELECT ...) T WHERE ... ;

Тут таблиці, яка повертається підзапитом в операторі FROM, присвоюється псевдонім Т, а зовнішній запит виділяє стовпці цієї таблиці і, можливо, записи відповідно до деякої умови, яка зазначена в операторі WHERE.

**Пов’язані підзапити**

Пов’язані (корельовані) підзапити з практичної точки зору найбільш цікаві. Однак для їх застосування важливо розуміти, що саме буде робити СКБД для надання відповіді. Для початку достатньо усвідомити, що при виконанні запитів, що містять пов’язані підзапити, немає такого чіткого поділу в часі виконання між підзапитом і запитом, як у випадку простих підзапитів. У випадку простих підзапитів спочатку виконується підзапит, а потім запит що його містить. У випадку пов’язаних підзапитів порядок виконання запиту зовсім інший, і його бажано розуміти, щоб уникнути непорозумінь. Основна ознака пов’язаного підзапиту полягає в тому, що він не може бути виконаний самостійно, поза всяким зв’язком з основним запитом. Формально ця ознака виявляється в виразі складного запиту наступним чином: підзапит посилається на таблицю, яка згадується в основному запиті. Отже, підзапит повинен бути виконаний в якомусь контексті з поточним станом виконання основного запиту.

Розглянемо деякий абстрактний і, в той же час, типовий запит, який містить пов’язаний підзапит:

SELECT A FROM T1

WHERE B=

(SELECT B FROM T2 WHERE C=T1.C)

Даний запит на вибірку даних (оскільки він починається з оператора SELECT) містить підзапит, сформульований в виразі, розміщеному в основному запиті після ключового слова WHERE.

Даний запит використовує дві таблиці: Т1 і Т2, в яких є стовпці з однаковими іменами В і С і однаковими типами. Підзапит, розташований у виразі після ключового слова WHERE основного запиту (SELECT B FROM T2 WHERE C=T1.C), звертається до цих же таблиць. Оскільки одна з таблиць (Т1) зустрічається як в підзапиті, так і в зовнішньому запиті, то підзапит не можна виконати самостійно, без зв’язку із зовнішнім запитом. Тому виконання запиту в цілому відбувається наступним чином:

1. Спочатку виділяється перший запис з таблиці Т1, зазначеної в операторі FROM зовнішнього запиту (весь запис таблиці Т1, а не тільки значення стовпця А). Цей запис називається поточним. Значення стовпців для цього запису доступні і можуть бути використані в підзапиті.

2. Потім виконується підзапит, який повертає список значень стовпця В таблиці Т2 для яких значення стовпця С дорівнює значенню стовпця С таблиці Т1.

3. Запит, сформульований в розглянутому прикладі, передбачає, що його підзапит повертає єдине значення (список з єдиним елементом), оскільки в операторі WHERE використовується оператор порівняння (=). Ми будемо вважати, що в цьому прикладі підзапит повертає єдине значення. Тепер виконується оператор WHERE основного запиту. Якщо значення стовпця В поточному (виділеному) записі таблиці Т1 дорівнює значенню, отриманому підзапитом, то цей запис виділяється зовнішнім запитом.

4. Оператор SELECT зовнішнього запиту виконує перевірку умови свого оператора WHERE. А саме він перевіряє, чи рівне поточне значення стовпця в таблиці Т1 значенню, одержаному підзапитом. Якщо так, то значення стовпця А поточного запису таблиці Т1 поміщається в результатну таблицю, в іншому випадку запис ігнорується. Потім відбувається перехід до наступного запису таблиці Т1. Тепер для неї виконується підзапит. Аналогічним чином все описане відбувається для кожного запису таблиці Т1.