**Лабораторная работа №4**

**Подзапросы**

*Цель**работы*:

приобретение практических навыков по созданию  запросов с подзапросами на языке SQL.

***Краткие теоретические сведения***

Язык Transact-SQL позволяет сравнивать значения столбца с результатом другой инструкции SELECT. Такая конструкция, где предложение WHERE инструкции SELECT содержит одну или больше вложенных инструкций SELECT, называется подзапросом (subquery). Первая инструкция SELECT подзапроса называется внешним запросом (outer query), а внутренняя инструкция (или инструкции) SELECT, используемая в сравнении, называется вложенным запросом (inner query). Первым выполняется вложенный запрос, а его результат передается внешнему запросу.

Существует два типа подзапросов:

* независимые;
* связанные.

В независимых подзапросах вложенный запрос логически выполняется ровно один раз. Связанный запрос отличается от независимого тем, что его значение зависит от переменной, получаемой от внешнего запроса. Таким образом, вложенный запрос связанного подзапроса выполняется каждый раз, когда система получает новую строку от внешнего запроса.

Независимый подзапрос может применяться со следующими операторами:

* операторами сравнения;
* оператором IN;
* операторами ANY и ALL;

**Подзапросы и операторы сравнения**

Использование оператора равенства (=) в независимом подзапросе показано в примере 4.1.

Пример 4.1. Выборка имен и фамилий сотрудников отдела Research

USE sample;

SELECT emp\_fname, emp\_lname

FROM employee

WHERE dept\_no =

(SELECT dept\_no FROM department WHERE dept\_name = 'Research');

Результат выполнения этого запроса:

|  |  |
| --- | --- |
| **emp\_fname** | **emp\_lname** |
| John | Barrimore |
| Sybill | Moser |

В примере 4.1 сначала выполняется вложенный запрос, возвращая номер отдела разработки (d1). После выполнения внутреннего запроса подзапрос в примере 4.1 можно представить следующим эквивалентным запросом:

USE sample

SELECT emp\_fname, emp\_lname

FROM employee

WHERE dept\_no = 'd1';

В подзапросах можно также использовать любые другие операторы сравнения, при условии, что вложенный запрос возвращает в результате одну строку. Это очевидно, поскольку невозможно сравнить конкретные значения столбца, возвращаемые внешним запросом, с набором значений, возвращаемым вложенным запросом.

**Подзапросы и оператор IN**

Оператор IN позволяет определить набор выражений (или констант), которые затем можно использовать в поисковом запросе. Этот оператор можно использовать в подзапросах при таких же обстоятельствах, т. е. когда вложенный запрос возвращает набор значений. Использование оператора IN в подзапросе показано в примере 4.2.

Пример 4.2. Получение всей информации о сотрудниках, чей отдел находится в Далласе

USE sample;

SELECT \* FROM employee

WHERE dept\_no IN

(SELECT dept\_no FROM department WHERE location = 'Dallas');

Результат выполнения этого запроса:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_no** | **emp\_fname** | **emp\_lname** | **dept\_no** |
|  |  |  |  |
| 25348 | Matthew | Smith | d3 |
| 10102 | Ann | Jones | d3 |
| 18316 | John | Barrimore | d1 |
| 28559 | Sybil] | Moser | dl |

Каждый вложенный запрос может содержать свои вложенные запросы. Подзапросы такого типа называются подзапросами с многоуровневым вложением. Максимальная глубина вложения (т. е. количество вложенных запросов) зависит от объема памяти, которым компонент Database Engine располагает для каждой инструкции SELECT. В случае подзапросов с многоуровневым вложением система сначала выполняет самый глубокий вложенный запрос и возвращает полученный результат запросу следующего высшего уровня, который в свою очередь возвращает свой результат запросу следующего уровня над ним и т. д. Конечный результат выдается запросом самого высшего уровня.

Запрос с несколькими уровнями вложенности показан в примере 4.3.

Пример 4.3. Выборка фамилий всех сотрудников, работающих над проектом Apollo

USE sample;

SELECT emp\_lname

FROM employee WHERE emp\_no IN

(SELECT emp\_no FROM works\_on WHERE project\_no IN

(SELECT project\_no FROM project WHERE project\_name = 'Apollo'));

Результат выполнения этого запроса:

|  |
| --- |
| **emp\_lname** |
| Jones |
| James |
| Bertoni |
| Moser |

В примере 4.3 самый глубокий вложенный запрос выбирает из таблицы project\_no значение p1. Этот результат передается следующему вышестоящему запросу, который обрабатывает столбец project\_no в таблице works\_on. Результатом этого запроса является набор табельных номеров сотрудников: (10102, 29346, 9031, 28559). Наконец, самый внешний запрос выводит фамилии сотрудников, чьи номера были выбраны предыдущим запросом.

**Подзапросы и операторы ANY и ALL**

Операторы ANY и ALL всегда используются в комбинации с одним из операторов сравнения. Оба оператора имеют одинаковый синтаксис:

***column\_name operator [ANY|ALL] query***

Параметр operator обозначает оператор сравнения, а параметр query — вложенный запрос.

Оператор ANY возвращает значение TRUE (истина), если результат соответствующего вложенного запроса содержит хотя бы одну строку, удовлетворяющую условию сравнения. Ключевое слово SOME является синонимом ANY. Использование оператора ANY показано в примере 4.4.

Пример 4.4. Выборка табельного номера, номера проекта и названия должности для сотрудников, которые не затратили большую часть своего времени при работе над одним из проектов

USE sample;

SELECT DISTINCT emp\_no, project\_no, job

FROM works\_on

WHERE enter\_date > ANY

(SELECT enter\_date

FROM works\_on);

Результат выполнения этого запроса:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_no** | | **project\_no** | **Job** |
| 2581 | | p3 | Analyst |
| 9031 | | p1 | Manager |
| 9031 | | p3 | Clerk |
| 10102 | | p3 | Manager |
| 18316 | p2 | | NULL |
| 25348 | p2 | | Clerk |
| 28559 | p1 | | NULL |
| 28559 | p2 | | Clerk |
| 29346 | p1 | | Clerk |
| 29346 | p2 | | NULL |

В примере 4.4 каждое значение столбца enter\_date сравнивается со всеми другими значениями этого же столбца. Для всех дат этого столбца, за исключением самой ранней, сравнение возвращает значение TRUE (истина), по крайней мере, один раз. Строка с самой ранней датой не попадает в результирующий набор, поскольку сравнение ее даты со всеми другими датами никогда не возвращает значение TRUE (истина). Иными словами, выражение enter\_date > ANY (SELECT enter\_date FROM works\_on) возвращает значение TRUE, если в таблице works\_on имеется любое количество строк (одна или больше), для которых значение столбца enter\_date меньше, чем значение enter\_date текущей строки. Этому условию удовлетворяют все значения столбца enter\_date, за исключением наиболее раннего.

Оператор ALL возвращает значение TRUE, если вложенный запрос возвращает все значения, обрабатываемого им столбца.

Настоятельно рекомендуется избегать использования операторов ANY и ALL. Любой запрос с применением этих операторов можно сформулировать лучше используя функцию EXISTS.

**Связанные подзапросы**

Подзапрос называется связанным (correlated), если любые значения вложенного запроса зависят от внешнего запроса. В примере 4.5 показано использование связанного подзапроса.

Пример 4.5. Выборка фамилий всех сотрудников, работающих над проектом p3

USE sample;

SELECT emp\_lname

FROM employee WHERE 'p3' IN

(SELECT project\_no FROM works\_on

WHERE works\_on.emp\_no = employee.emp\_no);

Результат выполнения этого запроса:

|  |
| --- |
| **emp\_lname** |
| Jones |
| Bertoni |
| Hansel |

В примере 4.5 вложенный запрос должен логически выполниться несколько раз, поскольку он содержит столбец emp\_no, который принадлежит таблице employee во внешнем запросе, и значение столбца emp\_no изменяется каждый раз, когда проверяется другая строка таблицы employee во внешнем запросе.

Проследим, как система может выполнять запрос в примере 4.5. Сначала система выбирает первую строку таблицы employee (для внешнего запроса) и сравнивает табельный номер сотрудника в этом столбце (25348) со значениями столбца works\_on.emp\_no вложенного запроса. Поскольку для этого сотрудника имеется только одно значение project\_no равное p2, вложенный запрос возвращает значение p2. Это единственное значение результирующего набора вложенного запроса не равно значению p3 внешнего запроса, условие внешнего запроса (WHERE 'p3' IN...) не удовлетворяется и, следовательно, внешний запрос не возвращает никаких строк для этого сотрудника. Далее система берет следующую строку таблицы employee и снова сравнивает номера сотрудников в обеих таблицах. Для этой строки в таблице works\_on есть две строки, для которых значение project\_no равно p1 и p3 соответственно. Следовательно, вложенный запрос возвращает результат p1 и p3. Значение одного из элементов этого результирующего набора равно константе p3, поэтому условие удовлетворяется, и отображается соответствующее значение второй строки столбца emp\_lname (Jones). Такой же обработке подвергаются все остальные строки таблицы employee, и в конечном результате возвращается набор из трех строк.

**Подзапросы и функция EXISTS**

Функция EXISTS принимает вложенный запрос в качестве аргумента и возвращает значение FALSE, если вложенный запрос не возвращает строк и значение TRUE в противном случае. Рассмотрим работу этой функции на нескольких примерах, начиная с примера 4.6.

Пример 4.6. Выборка фамилий всех сотрудников, работающих над проектом p1

USE sample;

SELECT emp\_lname

FROM employee

WHERE EXISTS (SELECT \* FROM works\_on

WHERE employee.emp\_no = works\_on.emp\_no

AND project\_no = 'p1');

Результат выполнения этого запроса:

|  |
| --- |
| **emp\_lname** |
| Jones |
| James |
| Bertoni |
| Moser |

Вложенный запрос функции EXISTS почти всегда зависит от переменной из внешнего запроса. Поэтому функция EXISTS обычно определяет связанный подзапрос.

Database Engine может обрабатывать запрос в примере 4.6. в следующем порядке. Сначала внешний запрос рассматривает первую строку таблицы employee (сотрудник Smith). Далее функция EXISTS определяет, есть ли в таблице works\_on строки, чьи номера сотрудников совпадают с номером сотрудника в текущей строке во внешнем запросе и чей project\_no равен p1. Поскольку сотрудник Smith не работает над проектом p1, вложенный запрос возвращает пустой набор, вследствие чего функция EXISTS возвращает значение FALSE. Таким образом, сотрудник Smith не включается в конечный результирующий набор. Этому процессу подвергаются все строки таблицы employee, после чего выводится конечный результирующий набор.

В примере 4.7 показано использование функции NOT EXISTS.

Пример 4.7. Выборка фамилий сотрудников, чей отдел не расположен в Сиэтле

USE sample;

SELECT emp\_lname

FROM employee

WHERE NOT EXISTS (SELECT \* FROM department

WHERE employee.dept\_no = department.dept\_no

AND location = 'Seattle');

Результат выполнения этого запроса будет следующим:

|  |
| --- |
| **emp\_lname** |
| Smith |
| Jones |
| Barrimore |
| Moser |

Список выбора инструкции SELECT во внешнем запросе с функцией EXISTS не обязательно должен быть в форме SELECT \*, как в предыдущем примере. Можно использовать альтернативную форму SELECT colum\_list, где column\_list представляет список из одного или нескольких столбцов таблицы. Обе формы равнозначны, потому что функция EXIST только проверяет на наличие (или отсутствие) строк в результирующем наборе. По этой причине в данном случае правильнее использовать форму SELECT \*.

**Задание на лабораторную работу.**

Используя подзапросы получить следующую информацию из базы данных.

**Вариант 1**

 1. Список читателей, у которых на руках находятся книги.

2. Список читателей, у которых находятся книги, срок возврата которых прошел.

3. Список издательств, книги которых поступили в прошлом году.

4. Фамилию читателя с максимальным сроком задержки возврата книги.

5. Фамилии читателей, которые должны вернуть книги в ближайшие 3 дня.

**Вариант 2**

1. Список клиентов, имеющих карточки, срок действия которых не истек.

2. Список клиентов, имеющих карточки «Visa Master».

3. Список клиентов, вносивших деньги на свои счета за последние 3 дня.

4. Фамилию клиента, внесшего максимальную сумму в прошлом месяце.

5. Список клиентов, имеющих более одной карточки.

**Вариант 3**

1. Список сотрудников, занимающих должность «Программист».
2. Список сотрудников, уволенных в прошлом месяце.
3. Список отделов, в которых есть программисты.
4. Название отдела, в котором максимальное количество сотрудников.
5. Список отделов, в которые были приняты новые сотрудники за последние 3 дня.

**Вариант 4**

1. Список старост факультета ФИТР.
2. Номер группы, в которой числится максимальное количество студентов.
3. Список студентов, у которых отсутствует номер или дата приказа о зачислении.
4. Список групп, в которых числятся студенты Иванов или Петров или Сидоров.
5. Список студентов, зачисленных два года назад.

**Вариант 5**

1. Список водителей, выезжавших в рейс на прошлой неделе.
2. Список автомобилей марки «МАЗ».
3. Список автомобилей, выезжавших в рейс на автомобилях МАЗ, ВАЗ или ГАЗ.
4. Фамилию водителей, выезжавшего в рейс максимальной протяженности.
5. Список водителей, не выезжавших в рейс за последние 3 дня.

**Вариант 6**

1. Список номеров стоимостью более 50 руб.
2. Список номеров, освобождающихся на следующей неделе.
3. Список клиентов, имеющих бронь на номера.
4. Список самых дешевых номеров.
5. Список клиентов, проживающих в номерах «Люкс» на втором этаже.

**Вариант 7**

1. Список врачей – терапевтов.
2. Список пациентов, записанных на прием к терапевту на сегодня.
3. Фамилию врача, к которому записано максимальное количество пациентов на сегодня.
4. Список врачей, которых посетил пациент Иванов на прошлой неделе.
5. График приема врача Петрова на сегодня.

**Вариант 8**

1. Список товаров, входящих в «Группу1».
2. Список адресов и фамилий клиентов, которым товар должен быть доставлен сегодня.
3. Фамилию и адрес клиента, сделавшего самый дорогой заказ.
4. Список номеров заказов, в которых присутствуют товары «Товар1» или «Товар2» или «Товар3».
5. Список адресов, по которым доставлялись заказы за последние 3 дня.

**Вариант 9**

1. Список аэропортов, в которые осуществляются полеты из аэропорта Минск.
2. Список пассажиров на рейс Минск-Париж на сегодня.
3. Список рейсов, на которые приобретал билет пассажир Иванов.
4. Список номеров рейсов и цену билета в аэропорты Берлин, Париж, Амстердам из Минска.
5. Номер рейса, на который продано максимальное количество билетов на прошлой неделе.

**Вариант 10**

1. Список учеников, не имеющих оценок по математике.
2. Список предметов, которые преподает преподаватель Иванов.
3. Список учеников, получивших оценки ниже 6 на прошлой неделе.
4. Фамилию ученика, имеющего максимальное количество оценок.
5. Фамилии и оценки учеников по предметам Математика, Физика, Информатика.

**Вариант 11**

1. Список спектаклей на следующую неделю.
2. Название спектакля, на который продано максимальное количество билетов.
3. Список спектаклей, на которые не продано ни одного билета на прошлой неделе.
4. Список свободных мест на спектакль «Отелло» на завтра.
5. Список свободных мест на спектакль «Отелло» на завтра в 1, 2 и 5 рядах.

**Вариант 12**

1. Список студентов, размещенных в общежитии вчера.
2. Список студентов, размещенных в двухместных комнатах.
3. Список номеров комнат, в которых размещены студенты группы «Группа1».
4. Номер комнаты, в которой проживает студент Петров.
5. Номера групп, в которых учатся студенты, проживающие в комнате номер 99.

**Вариант 13**

1. Список дисциплин, читаемых кафедрой ПОВТ.
2. Список дисциплин, по которым в 5 семестре сдают экзамен.
3. Список дисциплин, изучаемых на 1 и 2 курсах.
4. Название дисциплины, количество часов на изучение которой максимально.
5. Список дисциплин модуля «Модуль1».

**Вариант 14**

1. Список районов города Минска.
2. Город, имеющий максимальное количество районов.
3. Список типов недвижимости в Первомайском районе г. Минска.
4. Список типов недвижимости в Минске, Бресте и Гомеле.
5. Список районов с самой дешевой недвижимостью.

**Вариант 15**

1. Список клиентов, записанных на сегодня.
2. Список клиентов, записанных на самые дорогие услуги.
3. Список клиентов, которых обслуживал сотрудник Иванов вчера.
4. Список услуг оказанных клиентам сотрудниками Иванов, Петров, Сидоров вчера.
5. Список услуг, не востребованных клиентами сегодня.

**Содержание отчета о лабораторной работе**

1. Название и цель работы.
2. Номер группы, фамилия и инициалы студента.
3. Задание на лабораторную работу.
4. Перечень запросов и результатов их выполнения.
5. Выводы.

**Контрольные вопросы:**

1. Что такое подзапрос?
2. Как используется оператор IN в подзапросе?
3. Для чего предназначены операторы ALL и ANY?
4. Для чего предназначена функция EXIST?

|  |  |
| --- | --- |
|  |  |