**Лабораторные работы по курсу**

**Базы данных**

**Лабораторная работа 3**

**«Использование объединяющих и вложенных запросов языка SQL»**

**Москва, 2023**

Оглавление

[1. Теоретическая часть 3](#_Toc138792104)

[1.1. Соединение таблиц 3](#_Toc138792105)

[1.1.1. Неявное соединение таблиц 3](#_Toc138792106)

[1.1.2. Соединение с помощью JOIN 4](#_Toc138792107)

[1.1.3. Объединение, разность, пересечение таблиц 5](#_Toc138792108)

[1.1.4. Пример соединения и объединения таблиц 6](#_Toc138792109)

[1.2. Подзапросы 8](#_Toc138792110)

[1.2.1. Некоррелированный запрос. 8](#_Toc138792111)

[1.2.2. Коррелированный запросы 9](#_Toc138792112)

[2. Практическая часть 10](#_Toc138792113)

[2.1. Задание 1. 10](#_Toc138792114)

[2.2. Задание 2. 10](#_Toc138792115)

[2.3. Задание 3. 10](#_Toc138792116)

[2.4. Задание 4. 10](#_Toc138792117)

[2.5. Задание 5. 10](#_Toc138792118)

[Список литературы 10](#_Toc138792119)

1. Теоретическая часть

В предыдущей лабораторной работе были рассмотрены примеры запросов на выборку данных из одной таблицы. Однако, существует большое число задач, когда требуется проанализировать информацию из нескольких таблиц. Для этого существуют операции соединения.

* 1. Соединение таблиц
     1. Неявное соединение таблиц

Самым простым способом является неявное соединение таблиц, когда таблицы объединяются перекрестно. Другими словами, каждой строка одной таблицы будет совмещаться с каждой строкой второй таблицы. В данном случае мы получаем прямое (декартово) произведение двух таблиц.

На языке SQL для неявного соединения необходимо указать требуемые таблицы через запятую после оператора FROM.

Предположим, что нам необходимо вывести ФИО всех преподавателей и полное название подразделения, в котором они трудоустроены. Для этого соединим таблицы Структурное подразделение и Трудоустройство. Для этого возможно выполнить следующий запрос:

SELECT abbreviated\_title, professor\_id FROM structural\_unit, employment;

abbreviated\_title | professor\_id

-------------------+--------------

Институт МПСУ | 81001

Институт МПСУ | 81002

Институт МПСУ | 81003

Институт МПСУ | 81004

Институт МПСУ | 81005

Институт МПСУ | 81006

Институт МПСУ | 81007

Институт МПСУ | 81008

В результате запроса будет выведены значения из двух таблиц, где каждому значению из второй будут соответствовать все значения из первой таблицы. Таким образом, будет выведено (число строк в первой таблице) \* (число строк во второй таблице). В данном примере будет выведено 24 строки.

Аналогично добавим к данным двум таблицам ещё таблицу professor.

SELECT abbreviated\_title, professor\_id, surname FROM structural\_unit, employment, professor;

abbreviated\_title | surname | wage\_rate

-------------------+---------+-----------

Институт МПСУ | Широков | 0.25

Институт СПИНТех | Широков | 0.25

| Широков | 0.25

ВМ-1 | Широков | 0.25

| Широков | 0.25

ЛПО | Широков | 0.25

ПМТ | Широков | 0.25

ЦД | Широков | 0.25

Очевидно, что такой результат запроса нас не устроит. Для того, чтобы отобрать только преподавателей, прикрепленных к месту работы, необходимо добавить условие отбора.

Выберем только те строки, где значения номера подразделения в таблицах structural\_unit и employment и значения номера преподавателя в таблицах employment и professor совпадают.

Итоговый запрос будет выглядеть следующим образом:

SELECT professor.surname, professor.name, professor.patronymic ,structural\_unit.abbreviated\_title

FROM structural\_unit, employment, professor

WHERE structural\_unit.structural\_unit\_id = employment.structural\_unit\_number AND employment.professor\_id = professor.professor\_id

surname | name | patronymic | abbreviated\_title

---------+---------+------------+-------------------

Широков | Василий | Иванович | Институт МПСУ

Семенов | Андрей | Сергеевич | Институт МПСУ

Филатов | Илья | Иванович | Институт МПСУ

Ермаков | Виктор | Денисович | Институт МПСУ

Марков | Виктор | Дмитриевич | Институт МПСУ

Воронов | Николай | Михайлович | Институт МПСУ

Хахалин | Василий | Тихонович | Институт МПСУ

Козин | Николай | Дмитриевич | Институт МПСУ

Для того, чтобы обращаться к атрибуту конкретной таблицы, необходимо указывать название таблицы, отделив его от названия атрибута точкой.

Обратим внимание на то, что используя полное имя атрибута (название\_таблицы.имя\_атрибута) сильно удлиняется код запроса. Для сокращения возможно использовать более короткую запись

* + 1. Соединение с помощью JOIN

Соединённая таблица ­­­­­­— это таблица, полученная из двух других таблиц в соответствии с правилами соединения. Общий синтаксис описания соединённой таблицы:

*T1* *тип\_соединения* *T2* [ *условие\_соединения* ]

Кроме неявного соединения таблиц в языке SQL существует альтернативная форма записи операций соединение таблиц с помощью ключевого слова JOIN. Объединение происходит по столбцу, который есть в каждой из таблиц.

По типу соединения операторы JOIN подразделяются на внутренние и внешние — INNER JOIN и OUTER JOIN.

Внутреннее соединение - INNER JOIN используется для отбора строк из двух таблиц, в которых совпадают значения поля, по которому происходит объединение.

Например, объединим таблицы «Студент» и «Результат освоения дисциплины». Каждая из этих таблиц содержит поле студенческого билета. Произведем по этому полю соединение.

SELECT surname, name, mark

FROM Student

INNER JOIN Field\_comprehension

ON Field\_comprehension.student\_id = Student.student\_id;

В результате выполнения запроса на экран будет выведена информация об оценках всех студентов института.

Изображение выглядит как стол

Автоматически созданное описание

Внешнее соединение - OUTER JOIN можно разделить на правое (RIGHT), левое (LEFT) и полное (FULL).

Операция левого внешнего соединения возвращает кроме обычного результата соединения, строки из левого операнда, для которых не нашлось парного значения строки в правом операнде. Вместо значений атрибутов правого операнда будет указано неопределенное значение NULL.

Аналогично, операция правого внешнего соединения возвращает все строки, для которых не нашлось пары в пером операнде.

Полное внешнее соединение включает в себя все пересекающиеся строки и все непарные строки из обеих таблиц.

Аналогично неявному соединению, возможно произвести декартово произведение таблиц с помощью оператора JOIN. Для этого существует ключевое слово CROSS.

С помощью операций соединения возможно соединять несколько таблиц. Рассмотрим это на примере более сложного запроса. Необходимо вывести всех преподавателей и структурное подразделение, к которым они прикреплены.

Данные о преподавателях хранятся в таблице professor. Данные об структурных подразделениях – в таблице structural\_unit. Общих столбцов у данных таблиц нет, поэтому необходимо соединить их вместе с промежуточной таблицей – employment.

SELECT professor.surname, professor.name, professor.patronymic, structural\_unit.full\_title

FROM professor

INNER JOIN employment ON

employment.professor\_id = professor.professor\_id

INNER JOIN structural\_unit ON

structural\_unit.structural\_unit\_id = employment.structural\_unit\_number

surname | name | patronymic | full\_title

--------------+------------+---------------+-------------------------------------------------------------------

Александрова | Александра | Александровна | Кафедра высшей математики № 1

Аркадина | Ирина | Николаевна | Кафедра высшей математики № 1

Баранова | Виктория | | Кафедра высшей математики № 1

Березина | Наталия | Владимировна | Институт высокотехнологичного права, социальных и гуманитарных наук

Быкова | Елена | Николаевна | Кафедра высшей математики № 1

Вернадский | Владимир | Иванович | Институт цифрового дизайна

Витгенштейн | Людвиг | | Институт лингвистического и педагогического образования

Воронов | Николай | Михайлович | Институт микроприборов и систем управления имени Л.Н. Преснухина

* + 1. Объединение, разность, пересечение таблиц

Кроме соединения таблиц, когда в результате операции атрибуты одной таблицы будут добавлены к атрибутам другой существуют операции ***объединения***. В данном случае число атрибутов не изменяется, но в итоговой таблице будут содержаться значения из нескольких таблиц. При объединении таблиц необходимо соблюдать условие, что тип данных каждого столбца первой таблицы должен совпадать с типом данных соответствующего столбца во второй таблице. Имена столбцов в объединяемых таблицах не обязательно должны быть одинаковыми.

В языке SQL для объединения таблиц используется оператор UNION.

*запрос1* UNION [ALL] *запрос2*

Т.к. оператор UNION объединяет строки из двух таблиц результатов, в объединенной таблице могут содержаться повторяющиеся строки. По умолчанию оператор UNION в процессе своего выполнения удаляет повторяющиеся строки.

Кроме операции объединения существуют операции ***пересечения*** (INTERSECT) и ***разности*** (EXCEPT). Операция пересечения оставляет только общие строки из двух таблиц, а операция разности удаляет из первой таблицы значения, содержащиеся во второй. Данные три операции в языке SQL тесно связаны с логическими операциями булевой алгебры.

|  |  |  |
| --- | --- | --- |
| ***Операция*** | ***SQL операция*** | ***Логическая операция*** |
| Объединение | UNION | A **OR** B |
| Пересечение | INTERSECT | A **AND** B |
| Разность | EXCEPT | A **AND NOT** B |

* + 1. Пример соединения и объединения таблиц

Для закрепления материала рассмотрим еще один пример. Предположим, что имеются две таблицы, содержащие игроков двух команд. Два игрока входят в составы обеих из них. Применим к ним операции соединения.

Изображение выглядит как стол

Автоматически созданное описание

|  |  |  |  |
| --- | --- | --- | --- |
| SQL запрос | Возвращенные значения | Комментарий | |
| SELECT К1.Фамилия, К2.Фамилия  FROM К1, К2;  Или  SELECT К1.Фамилия, К2.Фамилия  FROM К1  CROSS JOIN К2 |  | Декартово произведение двух таблиц.  Каждой игроку из первой команды сопоставляется каждый игрок из второй команды. | |
| SELECT К1.Фамилия, К2.Фамилия  FROM К1  FULL OUTER JOIN К2 ON  К1.Фамилия = К2.Фамилия |  | Отбор всех игроков из двух команд с возможностью совпадений записей в левой и правой таблицах. Если таковых нет, на пустой стороне вставляется NULL. | |
| SELECT К1.Фамилия, К2.Фамилия  FROM К1  LEFT OUTER JOIN К2 ON  К1.Фамилия = К2.Фамилия |  | Отбираются все игроки из первой команды и игроки из второй, которые также играют за первую команду. | |
| SELECT К1.Фамилия, К2.Фамилия  FROM К1  RIGHT OUTER JOIN К2 ON  К1.Фамилия = К2.Фамилия |  | Отбираются все игроки из второй команды и игроки из первой, которые также играют за вторую команду. | |
| SELECT К1.Фамилия, К2.Фамилия  FROM К1  INNER JOIN К2 ON К1.Фамилия = К2.Фамилия |  | Отбираются участники, играющие за обе команды | |
| SELECT Фамилия FROM К1  UNION  SELECT Фамилия FROM К2 |  | Объединение всех участников команд без повторяющихся значений. | |
| SELECT Фамилия FROM К1  INTERSECT  SELECT Фамилия FROM К2 |  | Объединение всех участников команд. | |
| SELECT Фамилия FROM К1  EXCEPT  SELECT Фамилия FROM К2 |  | | Выбор игроков, играющих за первую команду, но не играющих за вторую. | |
| SELECT Фамилия FROM К2  EXCEPT  SELECT Фамилия FROM К1 |  | Выбор игроков, играющих за вторую команду, но не играющих за первую. | |

* 1. Подзапросы

Результатом выполнения запроса является набор кортежей, оформленный в виде таблицы. Данную таблицу возможно вывести на экран для просмотра или использовать в других запросах. В таком случае, запрос, используемый внутри другого запроса называют вложенным запросом или подзапросом.

Существуют два типа подзапросов:

**Некоррелированный подзапрос** – оператор SELECT вложенный в другой запрос SQL, не связанный с внешним запросом (он может быть выполнен отдельно от него).

**Коррелированный подзапрос** – оператор SELECT вложенный в другой запрос SQL, и ссылающийся на один или несколько столбцов внешнего запроса.

Рассмотрим типы запросов на примерах.

* + 1. Некоррелированный запрос.

Выведем всех преподавателей, чей оклад больше среднего. Для этого создадим подзапрос, вычисляющий среднюю зарплату всех преподавателей вуза. Далее, используя скалярный результат этого подзапроса найдем все большие значения.

SELECT professor.surname, professor.name ,professor.сurrent\_position

FROM professor

WHERE professor.salary::numeric >

(

SELECT AVG(professor.salary::numeric)

FROM professor

)

surname | name | сurrent\_position

-------------+------------+------------------

Широков | Василий | доцент

Семенов | Андрей | профессор

Филатов | Илья | доцент

Воронов | Николай | доцент

Орлов | Даниил | профессор

Овсянников | Даниил | доцент

Воронов | Артём | доцент

Михайлова | Анастасия | профессор

Выведем всех студентов, которые имеют хотя бы одну оценку 2 за один из экзаменов. Для этого создадим подзапрос, выбирающих всех двоечников и используя их номера студенческого билета найдем их фамилию и имя.

SELECT surname, name

FROM student

WHERE student.student\_id IN(

SELECT field\_comprehension.student\_id

FROM field\_comprehension

WHERE field\_comprehension.mark = 2

)

surname | name

-----------+------------

Алехин | Андрей

Коровина | Мария

Кузьмин | Мирослав

Гаврилов | Александр

Самойлов | Артём

Новиков | Кирилл

Маркин | Даниэль

Беляков | Иван

* + 1. Коррелированный запросы

Создадим аналогичный запрос с поиском всех студентов, имеющих хотя бы одну оценку 2, но с помощью коррелированного запроса. Обратите внимание, что во вложенном запросе происходит обращение к таблице «Студент», не указанной после ключевого слова FROM данного подзапроса. Если мы попытаемся запустить отдельно от основного данный подзапрос, то в результате будет получена ошибка.

SELECT surname, name

FROM student

WHERE 2 = ANY

(

SELECT field\_comprehension.mark

FROM field\_comprehension

WHERE field\_comprehension.student\_id = student.student\_id

)

surname | name

-----------+------------

Алехин | Андрей

Коровина | Мария

Кузьмин | Мирослав

Гаврилов | Александр

Самойлов | Артём

Новиков | Кирилл

Маркин | Даниэль

Беляков | Иван

Создадим запрос, выводящий средний балл каждого из студентов. Для этого создадим подзапрос, вычисляющий средний балл и ссылающийся на атрибут "Студент"."Номер студенческого билета" из внешнего запроса.

SELECT surname, name, (

SELECT CAST(AVG(mark) AS NUMERIC(2,1))

FROM Field\_comprehension

WHERE Field\_comprehension.student\_id = Student.student\_id

) AS "Средняя оценка"

FROM Student

ORDER BY "Средняя оценка" DESC;

surname | name | Средняя оценка

-----------+---------+----------------

Князева | Есения | 4.4

Хомяков | Илья | 4.3

Андреева | Ева | 4.3

Григорьев | Даниил | 4.3

Селезнев | Максим | 4.3

Казаков | Николай | 4.3

Ершова | Валерия | 4.3

Морозов | Илья | 4.2

1. Практическая часть

Напишите SQL запросы к учебной базе данных в соответствии с вариантом.

* 1. Задание 1.

|  |  |
| --- | --- |
| № варианта |  |
| 1 | Выведите фамилии и имена всех людей, кто причастен к вузу. |
| 2 | Выведите список имен студентов и преподавателей, которые совпадают. |
| 3 | Выведите имена студентов, которые **не** совпадают с именами преподавателей. |
| 4 | Сделайте проверку, все ли преподаватели трудоустроены. |
| 5 | Выведите номер студенческого билета студента, который получил 2 по любым дисциплинам. |
| 6 | Вывести ФИО преподавателей, которые работают на четверть ставки. |

* 1. Задание 2.

|  |  |
| --- | --- |
| № варианта |  |
| 1 | Выведите информацию о том, к каким институтам относятся группы. |
| 2 | Выведите ФИО преподавателей, их предметы и в каких группах они ведут. |
| 3 | Выполните запрос, выводящий группу, ее институт/кафедру, руководителя. |
| 4 | Выведите Фамилии и Имена студентов, кто получил 5 по Базам данных. |
| 5 | Вывести ФИО преподавателей, у которых зарплата больше 20 000. |
| 6 | Отсортировать результат предыдущего задания по алфавиту и исключить повторения после выполнения запроса. |

* 1. Задание 3.

|  |  |
| --- | --- |
| № варианта |  |
| 1 | Выведите все номера преподавателей и проверьте, есть ли совпадения с номерами студентов. В случае совпадения-вывести номер в столбцах студента и преподавателя. |
| 2 | Сделайте запрос на проверку трудоустройства преподавателей. |
| 3 | Выведите весь список студентов и проверьте оценки по всем дисциплинам только у ИБ-21. |
| 4 | Выведите студента, его группу и преподавателя, у которого фамилия = фамилия студента. |
| 5 | Выведите список принадлежности учебных групп к структурным подразделениям. |
| 6 | Выведите почты студентов и ФИО, у которых пароль начинается с *2002.* |

* 1. Задание 4.

|  |  |
| --- | --- |
| № варианта |  |
| 1 | Выведите самое часто повторяемое имя у студентов. |
| 2 | Выведите полные названия структурных подразделений, который обучают очно. |
| 3 | Выведите номера студенческих билетов и когда истечет их срок у студентов группы ИБ-21. |
| 4 | Выведите фамилию и имя всех студентов, кто учится заочно. |
| 5 | Выведите преподавателей, среднюю оценку их студентов, которая выше средней по институту. |
| 6 | Сгруппировать предыдущее задание по группам. Посчитать, сколько человек из каждый группы с таким паролем и вывести. |

* 1. Задание 5.

Самостоятельно разработайте 5 **осмысленных** запросов к базе данных, используя приведенные в данной лабораторной работе материалы. Вариант выбирается в соответствии с номером по списку.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Вариант 1,5 | | | | | Вариант 2,6 | | | | | Вариант 3 | | | | | Вариант 4 | | | | |
| Ключевое слово | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| INNER JOIN | + | + |  |  |  | + | + |  |  |  | + | + |  |  |  | + | + |  |  |  |
| LEFT OUTER JOIN |  |  | + |  |  |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |
| RIGHT OUTER JOIN |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  | + |  |  |
| UNION |  |  |  | + |  |  |  |  | + |  |  |  |  | + |  |  |  |  | + |  |
| EXCEPT |  |  |  |  | + |  |  |  |  | + |  |  |  |  |  |  |  |  |  |  |
| INTERSECT |  |  |  |  |  |  |  |  |  |  |  |  |  |  | + |  |  |  |  | + |

# Список литературы

|  |  |
| --- | --- |
| [1] | Документация к PostgreSQL 15.1, 2022. |
| [2] | «Исходный код СУБД postgres,» [В Интернете]. Available: https://github.com/postgres/postgres. [Дата обращения: 30 01 2023]. |
| [3] | Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, p. 662 . |
| [4] | Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, p. 582. |
| [5] | Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, p. 336. |