**Лабораторные работы по курсу**

**Базы данных**

**Лабораторная работа 3**

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

**Москва, 2024**

Оглавление

[1. Теоретическая часть 3](#_Toc157970428)

[1.1. Соединение таблиц 3](#_Toc157970429)

[1.1.1. Неявное соединение таблиц 3](#_Toc157970430)

[1.1.2. Соединение с помощью JOIN 4](#_Toc157970431)

[1.1.3. Объединение, разность, пересечение таблиц 6](#_Toc157970432)

[1.2. Подзапросы 7](#_Toc157970433)

[1.2.1. Некоррелированный запрос. 7](#_Toc157970434)

[1.2.2. Коррелированный запросы 8](#_Toc157970435)

[1.3. Общие табличные выражения 9](#_Toc157970436)

[1.4. Представления 9](#_Toc157970437)

[2. Практическая часть 10](#_Toc157970438)

[2.1. Задание 1. 10](#_Toc157970439)

[2.2. Задание 2. 10](#_Toc157970440)

[2.3. Задание 3. 10](#_Toc157970441)

[Контрольные вопросы 14](#_Toc157970442)

[Список использованной литературы 15](#_Toc157970443)

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.

Рассмотрим для примера таблицы *Structural\_units, Employments, Professors* из учебной базы данных*.* Для упрощения сократим число строк в таблицах, оставив следующие значения.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Обратите внимание, что к третьему структурному подразделению не прикреплен ни один преподаватель и преподаватель по фамилии Быкова не прикреплена ни к одному подразделению.

Предположим, что нам необходимо вывести код всех преподавателей и название подразделения, в котором они трудоустроены. Для этого необходимо объединить таблицы *Structural\_units* и *Employment.*

Выполним следующий запрос:

**SELECT abbreviated\_title, professor\_id**

**FROM structural\_units, employments**

abbreviated\_title | professor\_id

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

МПСУ | 810001

СПИНТех | 810001

ИЦОД | 810001

МПСУ | 820001

СПИНТех | 820001

ИЦОД | 820001

(6 rows)

В результате запроса будет выведены значения из двух таблиц, где каждому значению из первой будут соответствовать все значения из второй таблицы. Таким образом, будет выведено (число строк в первой таблице) \* (число строк во второй таблице). В данном примере будет выведено 6 строк.

В итоговой таблице будут присутствовать все преподаватели и все подразделения независимо от реального трудоустройства. Очевидно, что в этом нет никакого смысла.

Для того, чтобы выведенная информация связывала фамилию преподавателя и только то структурное подразделение, на котором он трудоустроен необходимо добавить условие отбора. Для вывода фамилии преподавателя вместо его кода используем таблицу *Professors.*

Выберем только те строки, где значения номера подразделения в таблицах *Structural\_units* и *Employments*, а также значения номера преподавателя в таблицах *Employments* и *Professors* совпадают.

Итоговый запрос будет выглядеть следующим образом:

**SELECT structural\_units.abbreviated\_title, professors.last\_name**

**FROM structural\_units, employments, professors**

**WHERE structural\_units.structural\_unit\_id = employments.structural\_unit\_id**

**AND employments.professor\_id = professors.professor\_id**

abbreviated\_title | last\_name

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

МПСУ | Широков

СПИНТех | Воронов

(2 rows)

Этот результат несет реальную смысловую нагрузку. Как можно заметить из результата запроса, преподаватель Быкова и подразделение ИЦОД в итоговую выборку не попали, т. к. для них указанное условие не было выполнено.

Обратите внимание, что для того, чтобы обращаться к атрибуту конкретной таблицы, необходимо указывать название таблицы, отделив его от имени атрибута точкой (название\_таблицы.имя\_атрибута).

Для сокращения возможно использовать более короткую запись. Для этого после названия таблицы в выражении FROM нужно указать для неё псевдоним. Пример ниже аналогичен запросу, рассмотренному выше, однако выглядит более компактным.

SELECT su.abbreviated\_title, p.last\_name

FROM structural\_units su, employments e, professors p

WHERE su.structural\_unit\_id = e.structural\_unit\_id

AND e.professor\_id = p.professor\_id

* + 1. Соединение с помощью JOIN

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. В результате запроса формируется соединенная таблица. Соединённая таблица ­­­­­­— это таблица, полученная из двух других таблиц (*T1, T2)* в соответствии с условием соединения. Общий синтаксис описания соединённой таблицы:

T1 тип\_соединения T2 [ условие\_соединения ]

По типу соединения операторы JOIN подразделяются на внутренние и внешние — INNER JOIN и OUTER JOIN, а также CROSS JOIN.

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

Формат:

SELECT столбцы FROM таблица1

[INNER] JOIN таблица2    ON условие1

[[INNER] JOIN таблица3 ON условие2]

Составим запрос, аналогичный примеру выше, в котором необходимо было вывести фамилии трудоустроенных преподавателей.

SELECT su.abbreviated\_title, p.last\_name

FROM structural\_units su

INNER JOIN employments e ON su.structural\_unit\_id = e.structural\_unit\_id

INNER JOIN professors p ON e.professor\_id = p.professor\_id

abbreviated\_title | last\_name

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

МПСУ | Широков

СПИНТех | Воронов

(2 rows)

Результат выполнения запроса совпал с примером выше.

**Внешнее соединение** - **OUTER JOIN** позволяет включить в вывод все строки из одной или обоих таблиц. можно разделить на правое (RIGHT), левое (LEFT) и полное (FULL).

Формат:

SELECT столбцы FROM таблица1

 { LEFT | RIGHT | FULL }  [OUTER] JOIN таблица2  ON условие1

   [  { LEFT | RIGHT | FULL }  [OUTER] JOIN таблица3  ON условие2 ]…

Операция левого внешнего соединения **(LEFT)** возвращает все строки левой (первой) таблицы, включая те, для которых не нашлось парного значения в правой (второй) таблице. Вместо ненайденных значений атрибутов правой таблицы будет указано неопределенное значение NULL. Аналогично, операция правого внешнего соединения **(RIGHT)** возвращает все строки правой (второй) таблицы, включая те, для которых не нашлось пары в левой. Обратите внимание, операции правого и левого соединения не коммутативны: A LEFT JOIN B ≠ B LEFT JOIN A. Вывод A LEFT JOIN B совпадает с B RIGHT JOIN A.

Полное внешнее соединение **(FULL)** включает в себя все пересекающиеся строки и все непарные строки из обеих таблиц.

Рассмотрим примеры:

Выберем из таблиц *Structural\_units* и *Employments* значения номеров телефонов всех структурных подразделений и ставку трудоустроенных профессоров.

SELECT abbreviated\_title, phone\_number, professor\_id, wage\_rate

FROM structural\_units su

LEFT JOIN employments e ON su.structural\_unit\_id = e.structural\_unit\_id;

abbreviated\_title | phone\_number | professor\_id | wage\_rate

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

МПСУ | 25-13 | 810001 | 0.25

СПИНТех | 23-45 | 820001 | 0.35

ИЦОД | 66-36 | null | null

(3 rows)

В данном примере была взята вся информация из первой (левой) таблицы и к ней присоединены значения, удовлетворяющие указанному условию. В противном случае проставлено значение *null*.

Рассмотрим второй пример. Выберем фамилии всех преподавателей и ставки тех из них, кто трудоустроен.

SELECT wage\_rate, last\_name

FROM employments e

RIGHT JOIN professors p ON e.professor\_id = p.professor\_id

wage\_rate | last\_name

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

0.25 | Широков

0.35 | Воронов

null | Быкова

(3 rows)

В текущем примере к таблице трудоустройства была присоединена таблица, содержащая преподавателей с использованием RIGHT JOIN. Таким образом была получена вся вторая (правая) таблица и оставлены только удовлетворяющие условию значения из левой. Аналогично в пустые ячейки проставлено значение *null*.

**Декартово произведение - CROSS JOIN**

Формат:

SELECT столбцы FROM таблица1

CROSS JOIN таблица2

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

SELECT abbreviated\_title, professor\_id

FROM structural\_units

CROSS JOIN employments

abbreviated\_title | professor\_id

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

МПСУ | 810001

СПИНТех | 810001

ИЦОД | 810001

МПСУ | 820001

СПИНТех | 820001

ИЦОД | 820001

(6 rows)

Ключевые слова OUTER и INNER являются устаревшим и команды LEFT OUTER JOIN, RIGHT OUTER JOIN, INNER JOIN возможно сокращать до LEFT JOIN, RIGHT JOIN и просто JOIN.

Если столбцы, по которым производится соединение, имеют одинаковое название, то вместо записи условия с помощью ключевого слова ON возможно использовать ключевое слово USING.

Формат:

SELECT столбцы FROM таблица1

{ [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN таблица2

USING ( список столбцов соединения )

В отличии от соединения с использованием ключевого слова ON, USING удаляет в итоговой таблице повторяющийся столбец, по которому происходит соединение. Например запрос, указанный ниже, аналогичен рассмотренному примеру на внутреннее соединение.

SELECT abbreviated\_title, last\_name

FROM structural\_units

INNER JOIN employments USING(structural\_unit\_id)

INNER JOIN professors USING(professor\_id)

* + 1. Объединение, разность, пересечение таблиц

Кроме соединения таблиц, когда в результате операции атрибуты (столбцы) 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. Количество выбранных столбцов из всех таблиц также должно совпадать. Имена столбцов в объединяемых таблицах не обязательно должны быть одинаковыми.

В языке SQL для объединения таблиц используется оператор UNION.

Формат:

SELECT\_выражение1

UNION [ALL] SELECT\_выражение2

[UNION [ALL] SELECT\_выражениеN]

Исходные таблицы могут содержать идентичные строки, тогда при объединении по умолчанию повторяющиеся строки удаляются. Если необходимо чтобы общая таблица содержала все строки, включая повторяющиеся используют параметр ALL.

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 |

Рассмотрим еще один пример. Вычислим

|  |  |  |
| --- | --- | --- |
| SQL запрос | Возвращенные значения | Комментарий |
| SELECT professor\_id FROM professors  UNION  SELECT professor\_id FROM employments; | professor\_id  --------------  820001  840001  810001  (3 rows) | Объединение всех кодов преподавателя без повторяющихся значений. |
| SELECT professor\_id FROM professors  INTERSECT  SELECT professor\_id FROM employments; | professor\_id  --------------  810001  820001  (2 rows) | Вывод только трудоустроенных преподавателей |
| SELECT professor\_id FROM professors  EXCEPT  SELECT professor\_id FROM employments | professor\_id  --------------  840001  (1 row) | Вывод нетрудоустроенных преподавателей. | |

* 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. Запрос, используемый внутри другого запроса, называют вложенным запросом или подзапросом.

Существуют два типа подзапросов: Оформление?

**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 last\_name, first\_name, salary::numeric**

**FROM professors**

**WHERE salary::numeric >**

**(**

**SELECT AVG(salary::numeric)**

**FROM professors**

**)**

last\_name | first\_name | salary

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

Широков | Василий | 98000.00

Семенов | Андрей | 210000.00

Филатов | Илья | 85000.00

Воронов | Николай | 98000.00

Серебряков | Александр | 150000.00

Воронов | Артём | 98000.00

Михайлова | Анастасия | 210000.00

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

)

* + 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 *Students*, не указанной после ключевого слова FROM данного подзапроса. Если мы попытаемся запустить отдельно от основного данный подзапрос, то в результате будет получена ошибка.

**SELECT last\_name, first\_name**

**FROM students**

**WHERE 2 = ANY**

**(**

**SELECT mark**

**FROM field\_comprehensions**

**WHERE field\_comprehensions.student\_id = students.student\_id**

**)**

Создадим запрос, выводящий средний балл каждого из студентов. Для этого создадим подзапрос, вычисляющий средний балл и ссылающийся на атрибут **student\_id** из таблицы *Students*, используемой во внешнем запросе.

**SELECT last\_name, first\_name, (**

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

**FROM Field\_comprehensions**

**WHERE Field\_comprehensions.student\_id = Students.student\_id**

**) AS "Средняя оценка"**

**FROM Students**

**ORDER BY "Средняя оценка" DESC;**

last\_name | first\_name | Средняя оценка

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

Савельев | Максим | 4.3

Гаганов | Павел | 4.2

Лембке | Юлия | 4.1

Фадеева | Ксения | 4.1

Лобов | Александр | 4.1

Иртеньев | Николай | 4.1

Верховцев | Иван | 4.1

Ракитин | Михаил | 4.0

Ильин | Владимир | 4.0

* 1. Общие табличные выражения

Для упрощения сложного запроса возможно использовать конструкцию CTE (Common table expression), позволяющую разбить его на несколько частей. Эта конструкция   определяет временные таблицы, которые существуют только для одного запроса. CTE похожи на вложенные запросы, но более оптимизированы. Вложенный запрос в отличие от CTE повторяется для каждой строки, которую нашел основной запрос, что повышает ресурсоемкость и замедляет работу кода.

Формат:

WITH <название выражения> AS (

SELECT | INSERT | UPDATE |DELETE

), AS …

SELECT …

Например, следующий запрос вычисляет всех студентов, чьи оценки выше средней по всему вузу.

**WITH avg\_mark AS (**

**SELECT cast(avg(mark) AS numeric(2,1)) AS a\_mark FROM field\_comprehensions**

**)**

**SELECT s.student\_id,**

**last\_name,**

**first\_name,**

**cast(avg(mark) AS numeric(2,1)) AS "avg. mark"**

**FROM students s**

**INNER JOIN field\_comprehensions fc ON s.student\_id = fc.student\_id**

**GROUP BY s.student\_id**

**HAVING avg(mark) > (SELECT a\_mark FROM avg\_mark)**

**ORDER BY "avg. mark" DESC;**

student\_id | last\_name | first\_name | avg. mark

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

828170 | Савельев | Максим | 4.3

830114 | Гаганов | Павел | 4.2

892391 | Фадеева | Ксения | 4.1

834421 | Иртеньев | Николай | 4.1

816993 | Лембке | Юлия | 4.1

825852 | Лобов | Александр | 4.1

833207 | Верховцев | Иван | 4.1

895508 | Ракитин | Михаил | 4.0

894671 | Бессонов | Алексей | 4.0

* 1. Представления

При работе с базами данных очень часто приходится выполнять одинаковые сложные и объемные запросы. Каждый раз составлять длинный запрос оказывается весьма трудоемко. Для упрощения работы возможно сформировать из такого запроса представление, к которому далее возможно обращаться, как будто это обычная таблица какое угодно число раз. Представление состоит из строк и столбцов, которые могут формироваться из одной или нескольких таблиц. Сокращенный формат команды:

CREATE VIEW ***имя*** AS ***запрос***

Например, создадим представление на основе запроса, выводящего трудоемкость дисциплины в формате «Количество часов/ЗЕТ»

CREATE VIEW labor\_intensity AS

SELECT field\_name AS "Field name", (36\*zet::numeric)::varchar || '/' || zet as "Labor intensity"

FROM field

ORDER BY "Labor intensity"

Далее к созданному представлению возможно обратиться, как к таблице.

SELECT \* FROM labor\_intensity

Field name | Labor intensity

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

Риторика | 108/3

Основы управления проектами | 108/3

Основы рыночной экономики | 108/3

Безопасность жизнедеятельности | 108/3

Экология | 108/3

Иностранный язык | 108/3

История | 108/3

Правоведение | 108/3

1. Практическая часть

Напишите SQL запросы к учебной базе данных в соответствии с вариантом. Вариант к практической части выбирается по формуле: V = (N % 10) +1, где N – номер в списке группы, % - остаток от деления.

* 1. Задание 1.

with count\_degree AS (

SELECT DISTINCT degree from professor

) SELECT count(degree) FROM count\_degree

Рассмотрите следующий запрос. Какой результат был получен после его выполнения. Измените запрос, поставив вместе значения degree в агрегатную функцию символ «\*». Сравните результаты выполнения запросов и объясните его.

* 1. Задание 2.

|  |  |
| --- | --- |
| № варианта | № запросов |
| 1 | 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101 |
| 2 | 2, 12, 22, 32, 42, 52, 62, 72, 82, 92, 102 |
| 3 | 3, 13, 23, 33, 43, 53, 63, 73, 83, 93, 103 |
| 4 | 4, 14, 24, 34, 44, 54, 64, 74, 84, 94, 104 |
| 5 | 5, 15, 25, 35, 45, 55, 65, 75, 85, 95, 105 |
| 6 | 6, 16, 26, 36, 46, 56, 66, 76, 86, 96, 106 |
| 7 | 7, 17, 27, 37, 47, 57, 67, 77, 87, 97, 107 |
| 8 | 8, 18, 28, 38, 48, 58, 68, 78, 88, 98, 108 |
| 9 | 9, 19, 29, 39, 49, 59, 69, 79, 89, 99, 109 |
| 10 | 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110 |

* 1. Задание 3.

Самостоятельно разработайте 3 **осмысленных** запроса к базе данных, используя приведенные в данной лабораторной работе материалы. Вариант выбирается в соответствии с номером по списку. Из созданных запросов создайте представления.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант: | **1,2,9** | | | **3,4,10** | | | **5,6** | | | **7,8** | | |
| Номер запроса: | **1** | **2** | **3** | **1** | **2** | **3** | **1** | **2** | **3** | **1** | **2** | **3** |
| INNER JOIN | + |  |  | + |  |  | + |  |  | + |  |  |
| LEFT JOIN |  | + |  |  |  |  |  | + |  |  |  |  |
| RIGHT JOIN |  |  |  |  | + |  |  |  |  |  | + |  |
| UNION |  |  | + |  |  |  |  |  |  |  |  | + |
| EXCEPT |  |  |  |  |  | + |  |  |  |  |  |  |
| INTERSECT |  |  |  |  |  |  |  |  | + |  |  |  |

Сборник запросов к лабораторной работе

*INNER JOIN*

1. Вывести ФИО, должности, оклад, номер трудового договора всех преподавателей
2. Вывести ФИО, номера студенческих билетов, оценки по освоению дисциплин и названия дисциплин для всех студентов
3. Вывести наименование группы и наименование структурного подразделения, к которому она относится (для всех групп)
4. Вывести ФИО преподавателей, названия преподаваемых дисциплин и название групп, у которых они ведутся
5. Вывести номер трудового договора и название структурного подразделения, к которому он относится
6. Вывести ФИО студента и освоенные им дисциплины с оценками
7. Вывести ФИО преподавателей, названия преподаваемых ими дисциплин
8. Вывести названия групп и названия их структурных подразделений
9. Вывести названия групп и ФИО их руководителей (руководителей стр. подр. группы)
10. Вывести ФИО всех студентов, освоивших дисциплину Базы Данных на 5
11. Вывести все дисциплины, которые ведутся у группы ИТД-31
12. Вывести ФИО всех преподавателей и их зарплату, если зарплата больше средней по России (зарплата = оклад \* ставка)
13. Вывести полные названия структурных подразделений и названия дисциплин, преподаваемых в них
14. Вывести руководителя структурного подразделения и количество студентов в его подчинении
15. Вывести ФИО и ставку всех трудоустроенных преподавателей
16. Вывести ФИО и срок действия студенческого билета для каждого студента
17. Вывести ФИО всех студентов очной формы обучения
18. Вывести ФИО преподавателей, их ставку и стаж
19. Вывести ФИО всех студентов с оценками, освоивших дисциплину «Философия» меньше, чем на 4
20. Вывести ФИО преподавателей, работающих в МПСУ
21. Вывести названия групп, в которых есть хотя бы 1 студент, освоивший дисциплину «Экология» на 5
22. Вывести название дисциплин и стаж ведущих преподавателей
23. Вывести ФИО студента и его логин и пароль

*LEFT JOIN, RIGHT JOIN*

1. Вывести дисциплины и их руководителей (по структурному подразделению) (если есть, иначе - null)
2. Вывести фамилии преподавателей и фамилии их однофамильцев среди студентов (если есть, иначе - null)
3. Вывести ФИО всех студентов, для студентов ИБ-21 вывести ещё оценки
4. Вывести ФИО студента, название предмета, его оценку по предмету (если она есть, иначе null)
5. Вывести названия структурных подразделений и группы, которые им принадлежат (если они есть, иначе - null)
6. Вывести список преподавателей и наименование структурного подразделения, в котором они трудоустроены (если не трудоустроены - null)
7. Вывести ФИО преподавателя и название дисциплины, которую он ведёт (если нет - null)
8. Вывести ФИО всех студентов и форму их обучения
9. Вывести ФИО всех студентов и предмет, по которым у них долг (если долга нет - null), отсортировать по ФИО
10. Вывести название группы и контактный номер подразделения, к которому относится группа
11. Вывести ФИО преподавателя, стаж и максимальную сложность (ЗЕТ) среди преподаваемых им дисциплин
12. Вывести ФИО студентов-должников и дату, до которой действителен его студенческий
13. Вывести ФИО студентов и максимальную оценку среди освоенных им дисциплин
14. Вывести ФИО преподавателя, стаж его работы и номер контактного телефона его структурного подразделения
15. Вывести ФИО всех студентов, чей логин из таблицы Пользователи сожержится в их электронной почте

*UNION/EXCEPT/INTERSECT*

1. Вывести ФИО всех женщин в институте (среди преподавателей и студентов)
2. Вывести ФИО всех мужчин в институте (среди преподавателей и студентов)
3. Вывести фамилии всех в ВУЗе (среди студентов и преподавателей) (каждую фамилию выводить 1 раз)
4. Вывести название всех групп и всех структурных подразделений
5. Вывести номера всех студентов и преподавателей
6. Вывести фамилии всех студентов и преподавателей-ассистентов, начинающихся на Ф
7. Вывести логины всех пользователей и почтовые адреса всех студентов
8. Вывести полные и сокращенные названия всех структурных подразделений (использовать UNION)
9. Вывести все дни рождения студентов, и все даты выдачи студенческих, отсортировать по возрастанию
10. Вывести ФИО всех преподавателей и руководителей структурных подразделений
11. Вывести названия всех дисциплин, в последней строке вывести среднюю сложность дисциплин (ЗЕТ)
12. Вывести в одном столбце: - максимальный оклад среди преподавателей; - средний оклад среди преподавателей; - минимальный оклад среди преподавателей;
13. Вывести в одном столбце: - самое популярное имя среди студентов; - фамилии всех студентов с этим именем;
14. Вывести в одном столбце: - названия всех групп; - число всех студентов;
15. Вывести в одном столбце: - email-адреса всех студентов; - число уникальных домаенных адресов (gmail.com, yandex.ru)
16. Вывести ФИ и номера студенческих билетов студентов, а также дисциплины по которым у них нет двоек
17. Вывести ФИО преподавателей, у которых нет однофамильцев среди студентов
18. Вывести номера нетрудоустроенных преподавателей (без трудового договора)
19. Вывести ФИ преподавателей, которые не являются студентами
20. Вывести ФИ студентов, которые не являются преподавателями
21. Вывести номера всех преподавателей, которые не являются преподавателями дисциплины Базы Данных (использовать EXCEPT)
22. Вывести ФИ всех студентов, у которых нет тёзок-однофамильцев среди преподавателей
23. Вывести номера студентов, у которых студенческий билет будет просрочен в следующем году
24. Вывести всех студентов, у кого в поле электронной почты указана внешняя почта (домен не ieee.ru)
25. Вывести ФИ всех студентов, кроме студентов МПСУ
26. Вывести ФИ, оклад и должность всех преподавателей, кроме преподавателей МПСУ Сортировать по убыванию оклада
27. Вывести ФИ и оклад всех преподавателей, кроме ассистентов. Сортировать по возрастанию окладов
28. Вывести логины всех пользователей, у которых слабый пароль
29. Вывести имена руководителей подразделений, у которых в подчинении меньше 30 человек (среди студентов)
30. Вывести всех студентов, кроме тех, у кого завершил действие студенческий билет
31. Вывести всех студентов ИБ-21, у которых есть хотя бы одна двойка
32. Вывести всех студентов, кто является преподавателем
33. Вывести всех студентов, кто освоил хотя бы 1 дисциплину (есть хотя бы 1 оценка)
34. Вывести номера всех структурных подразделений, дающих трудоустройство
35. Вывести имена всех тёзок среди студентов и преподавателей
36. Вывести номера всех трудоустроенных преподавателей
37. Вывести номера всех нетрудоустроенных преподавателей
38. Вывести всех тёзок-однофамильцев среди студентов и преподавателей
39. Вывести номера всех студентов, чей студ. билет будет просрочен в следующем году (использовать INTERSECT)
40. Вывести номера всех преподавателей дисциплины Базы Данных (использовать INTERSECT)
41. Вывести номера всех преподавателей, у которых есть хотя бы 1 дисциплина со сложностью 5 ЗЕТ (использовать INTERSECT)
42. Вывести номера всех студентов-должников, у которых электронная почта относится к домену ieee.ru (использовать INTERSECT)
43. Вывести номера всех студентов-должников группы ИВТ-21
44. Вывести номера всех преподавателей, работающих на полную ставку и преподающих дисциплину Философия
45. Вывести номера всех преподавателей МПСУ

В*ложенные запросы*

1. Вывести ФИ и возраст всех студентов, обучающихся на очной или очно-заочной форме
2. Вывести ФИ студентов и их среднюю оценку по дисциплине Базы Данных, если она больше 3
3. Вывести номера всех трудовых договоров, ставка по которым выше средней
4. Вывести ФИ студентов, у которых оценка хотя бы по одной из освоенных дисциплин выше, чем средняя по ВУЗу
5. Вывести всех ФИ всех студентов группы ИБ-21, у которых хотя бы одна оценка выше средней оценки по группе
6. Вывести список всех студентов с просроченным студенческим билетом
7. Вывести всех отличников группы ИВТ-21
8. Вывести ФИ всех студентов, которым студенческий билет был выдан в текущем году
9. Вывести средний возраст студентов каждого структурного подразделения
10. Вывести названия всех структурных подразделений, обучающих только очно
11. Вывести ФИ всех студентов, обучающихся на заочной форме обучения
12. Вывести ФИ всех студентов, чей средний балл выше среднего балла в ИТД-21
13. Вывести ФИ и должности всех преподавателей, чей оклад меньше среднего
14. Вывести ФИ и должности всех преподавателей, у которых стаж работы больше среднего и зарплата выше средней
15. Вывести всех ФИ всех преподавателей, у которых максимальная сложность среди преподаваемых дисциплин больше средней
16. Вывести ФИ всех преподавателей, чей номер трудового договора начинается с определённой/ых цифры/цифр
17. Вывести ФИ преподавателей с максимальным стажем по каждой дисциплине (вывести также стаж и название дисциплины)
18. Вывести ФИ всех преподавателей и указать больше ли их оклад, чем 0.5
19. Вывести ФИ произвольного преподавателя и число двоечников по его дисциплине
20. Вывести ФИ и средний балл каждого студента произвольной группы
21. Вывести ФИ и номера всех студентов, чей студенческий действителен на текущую дату
22. Вывести ФИ преподавателя с минимальным стажем по дисциплине Базы Данных
23. Вывести ФИ и дату окончания действия студ. билетов студентов определённой группы
24. Вывести ФИ всех студентов, у которых есть хотя бы одна не освоенная дисциплина
25. Вывести ФИ всех студентов и их форму обучения
26. Вывести ФИ всех трудоустроенных преподавателей института МПСУ
27. Вывести фамилию преподавателя и среднюю оценку за освоение его дисциплины
28. Вывести ФИ всех отличников института
29. Вывести ФИ преподавателей (или преподавателя) с максимальным окладом
30. Вывести дисциплины с ЗЕТ выше среднего
31. Вывести руководителей группы ИТД
32. Вывести ФИ и номера студ. билетов всех студентов, чей средний балл по освоенным дисциплинам ниже 3
33. Вывести ФИО и дату рождения самого молодого студента
34. Выведите всех студентов, средняя оценка которых больше 3, в порядке возрастания оценки.
35. Выведите 10 лучших студентов 3-го курса института МПСУ (имеющих самую высокую среднюю оценку), окончивших учебу без троек и двоек. Отсортируйте ее по убыванию среднего балла. Таблица должна содержать ФИО, номер студенческого билета и средний балл.

Контрольные вопросы

1. Существует ли отличие между использованием ключевых слов INNER JOIN или перечисления таблиц через запятую в предложении FROM?
2. Для чего предназначено соединение LEFT JOIN?
3. В чем отличие коррелированного от некоррелированного подзапроса?
4. Назовите отличие соединения от объединения
5. В каких случаях используется ключевое слово INNER и OUTER?

Список использованной 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. |