**ТЕМА № 2. ОСНОВЫ ЯЗЫКА SQL**

**ОПЕРАТОР ВЫБОРКИ ДАННЫХ**

**Цели занятия:**

1. Развивать практические умения в написании SQL-команд для комбинирования результирующих таблиц.
2. Развивать практические умения в написании SQL-команд для выборки данных из нескольких таблиц.
3. Развивать практические умения в написании SQL-команд для выборки данных с использованием вложенных запросов.

**УЧЕБНЫЕ ВОПРОСЫ**

***Задание В.1:*** Изучить БД с именем «*database0*», которая уже загружена в ваш профиль.

В вашей БД содержится две таблицы с данными: «*table*\_1» и «*table*\_2». Структура и содержимое таблиц «*table*\_1» и «*table*\_2» представлены на рисунках 1–2.

Таблица «*table*\_1» содержит информацию о студентах и сотрудниках (далее студенты), при этом столбцы имеют следующее значение: «*id*» – идентификатор студента (первичный ключ), «*fio*» – фамилия и инициалы, «*n\_group*» – номер группы, «*rang*» – должность, «*date\_b*» – дата рождения, «*salary*» – должностной оклад.

Таблица «*table*\_2» содержит информацию об итоговых оценках студентов по дисциплинам, при этом столбцы имеют следующее значение: «*id*» – идентификатор студента, «*subj*» – название дисциплины, «*mark*» – оценка. Столбцы «*id*» и «*subj*» образуют в таблице составной первичный ключ.

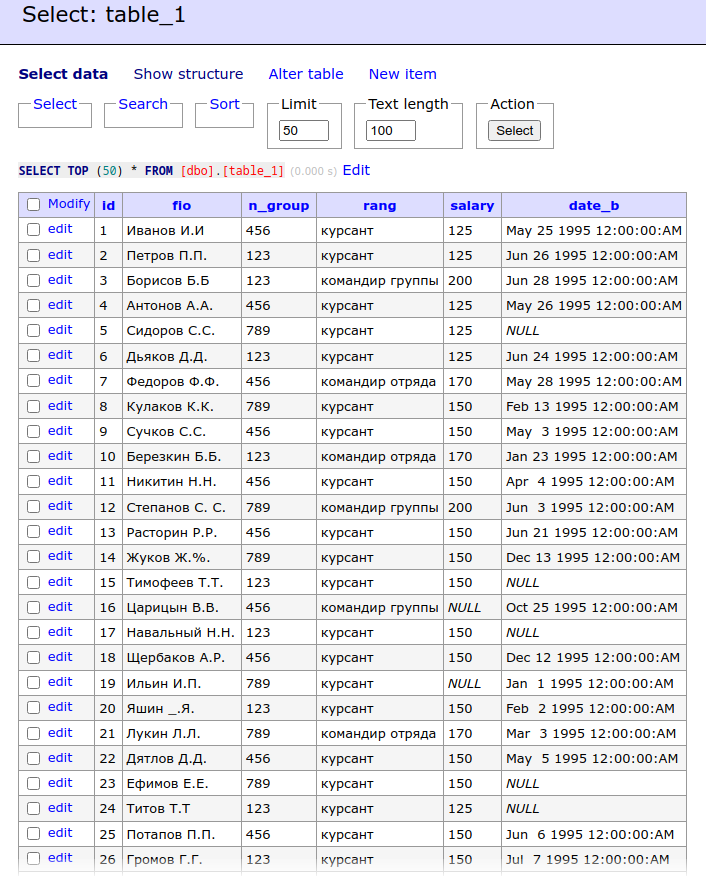


Рисунок 1 – Структура и содержимое таблицы «table\_1»

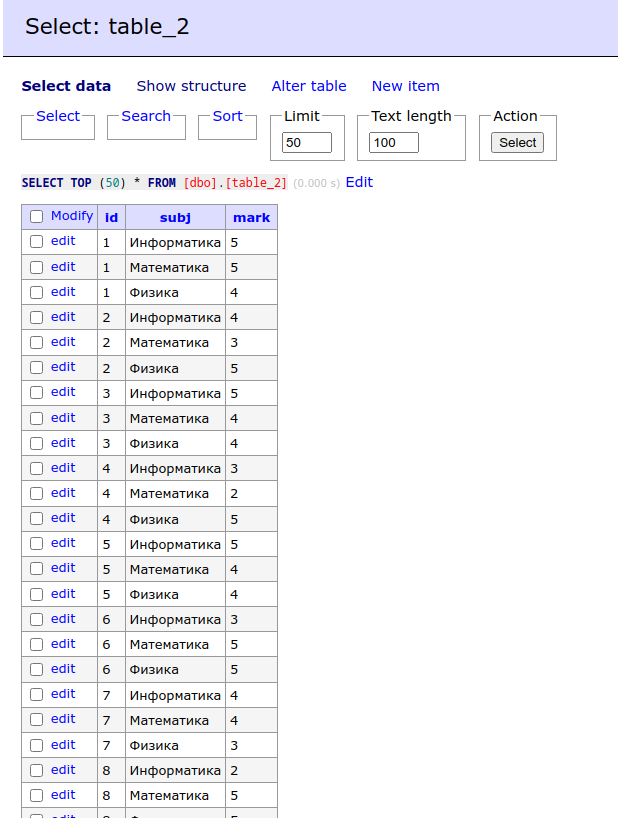


Рисунок 2 – Структура и содержимое таблицы «table\_2»

**1. КОМБИНИРОВАНИЕ РЕЗУЛЬТИРУЮЩИХ ТАБЛИЦ**

Речь пойдет об операторах UNION (объединение), INTERSECT (пересечение) и EXCEPT (разность), предназначенных для преобразования результатов двух команд SELECT в единый результат.

**Оператор UNION** комбинирует результаты двух запросов, созданных с использованием предложения SELECT, в один результат, который объединяет строки, созданные двумя запросами. Выражение UNION удаляет из результата повторяющиеся строки; выражение UNION ALL сохраняет повторы.

**Оператор INTERSECT** преобразует результаты двух запросов в один результат, который включает все общие строки в результатах выполнения двух запросов.

**Оператор EXCEPT** преобразует результаты двух запросов в один результат, который включает только те строки из первого запроса, которых нет во втором запросе.

***Операторы UNION, INTERSECT и EXCEPT имеют ряд ограничений:***

– списки столбцов команды SELECT в двух запросах должны включать одинаковое число столбцов (названий столбцов, арифметических выражений, функций и т.д.);

– соответствующие столбцы в двух запросах должны быть заданы в одинаковом порядке;

– предложение ORDER BY может использоваться только в последнем запросе команды UNION;

– можно задавать предложения GROUP BY и HAVING только в отдельных запросах; их нельзя использовать для изменения конечного результата.

***Задание 1.1:*** Комбинирование строк с помощью оператора UNION:

**а)** выбрать всю информацию из таблицы *table*\_1 о студентах 123 и 456 групп, выбрать всю информацию из таблицы *table*\_1 о студентах 123 группы, объединить результаты запросов без повторения строк;

***Решение:***

SELECT \* FROM table\_1 WHERE n\_group IN (123, 456) UNION SELECT \* FROM table\_1 WHERE n\_group=123

**б)** по заданию 1.1 (а) объединить результаты запросов с повторением строк;

**в)** по заданию 1.1 (б) провести сортировку по номеру группы в порядке возрастания, а в группах по фамилиям в алфавитном порядке.

***Задание 1.2:*** Поиск общих строк с помощью команды INTERSECT:

**а)** выбрать всю информацию из таблицы *table*\_1 о студентах 123 и 456 групп, выбрать всю информацию из таблицы *table*\_1 о студентах 123 группы, найти общие строки в результатах запросов;

***Решение:***

SELECT \* FROM table\_1 WHERE n\_group IN (123, 456) INTERSECT SELECT \* FROM table\_1 WHERE n\_group=123

**б)** по заданию 1.2 (а) провести сортировку по фамилиям в алфавитном порядке.

***Задание 1.3:*** Поиск различных строк с помощью команды EXCEPT:

**а)** выбрать всю информацию из таблицы *table*\_1 о студентах 123 группы, выбрать всю информацию из таблицы *table*\_1 о студентах 123 и 456 групп, найти различные строки в результате первого запроса;

***Решение:***

SELECT \* FROM table\_1 WHERE n\_group IN (123, 456) EXCEPT SELECT \* FROM table\_1 WHERE n\_group=123

**б)** по заданию 1.3 (а) провести сортировку по фамилиям в алфавитном порядке.

Проанализировать полученные результаты. Добиться правильного написания SQL-команд для выборки требуемых данных из таблицы *table*\_1. В случае возникновения ошибки при выполнении SQL-команды проанализировать выданное СУБД сообщение об ошибке и внести необходимые коррективы в ее код.

**2. МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ**

Любой запрос, который выбирает данные из более чем одной таблицы, выполняет какое-либо соединение. При этом для придания большей компактности и стройности инструкции SELECT рекомендуется создавать псевдонимы таблиц с использованием предложения AS. Для этого в предложении FROM или JOIN необходимо написать:

table [AS] alias

Здесь *table* – название таблицы, а *alias* – псевдоним таблицы.

Таблицы всегда объединяются построчно при выполнении условий соединения, определенных в запросе. Те строки, которые не соответствуют заданным условиям, могут быть как включены в соединение, так и исключены из него, в зависимости от типа соединения.

Условия соединения накладываются на один из столбцов каждой таблицы, участвующей в соединении. Эти столбцы называются связанными. Значения связанных столбцов обычно проверяют на равенство (=), но можно их сравнивать, применяя другие операторы сравнения (<>, <, <=, >, >=). Связанные столбцы не обязательно должны иметь одно и то же имя.

**2.1. ЕСТЕСТВЕННОЕ СОЕДИНЕНИЕ**

Для создания естественного соединения предусмотрены два способа:

**1. С использованием предложения JOIN:**

SELECT columns

FROM table1 join\_type table2

ON join\_conditions

[WHERE search\_condition]

[GROUP BY grouping\_conditions]

[HAVING search\_condition]

[ORDER BY sort\_columns];

Здесь *columns* – это несколько разделенных запятыми выражений или названий столбцов из таблиц *table*1 или *table*2;

– *table*1 или *table*2 – это названия связываемых таблиц;

– *join\_type* задает тип соединения (естественное – JOIN, декартово произведение – CROSS JOIN, левое внешнее – LEFT OUTER JOIN, правое внешнее – RIGHT OUTER JOIN, полное внешнее – FULL OUTER JOIN);

– *join\_conditions* задает одно или несколько условий соединения.

**2. С использованием предложения WHERE.**

SELECT columns

FROM table1, table2

WHERE join\_conditions

[GROUP BY grouping\_conditions]

[HAVING search\_condition]

[ORDER BY sort\_columns];

Здесь предложение WHERE может включать еще и условия поиска (не имеющие прямого отношения к соединению таблиц), необходимые для фильтрации строк.

***Задание 2.1:*** Создание естественного соединения с помощью предложений JOIN и WHERE:

**а)** выбрать номера группы, должности, фамилии студентов, название дисциплин и итоговые оценки студентов по дисциплинам из таблиц *table*\_1 и *table*\_2 с использованием предложения WHERE;

***Решение:***

SELECT table\_1.n\_group, table\_1.rang, table\_1.fio, table\_2.subj, table\_2.mark FROM table\_1, table\_2 WHERE table\_1.id = table\_2.id

**б)** по заданию 2.1 (а) создать для таблиц «*table*\_1» и «*table*\_2» псевдонимы «*t*1» и «*t*2»;

***Решение:***

SELECT t1.n\_group, t1.rang, t1.fio, t2.subj, t2.mark FROM table\_1 t1, table\_2 t2 WHERE t1.id = t2.id

**в)** выполнить задание 2.1 (б) с использованием предложения JOIN;

***Решение:***

SELECT t1.n\_group, t1.rang, t1.fio, t2.subj, t2.mark FROM table\_1 t1 JOIN table\_2 t2 ON t1.id = t2.id

**г)** выбрать номера групп, должности, фамилии студентов 123 группы, названия дисциплин и итоговые оценки студентов по дисциплине «Физика» из таблиц *table*\_1 и *table*\_2 с использованием предложений JOIN и WHERE;

**д)** выбрать фамилии студентов и их средний балл по всем итоговым оценкам из таблиц *table*\_1 и *table*\_2 с использованием предложений JOIN и WHERE;

**е)** по заданию 2.1 (д) результат представить только о тех студентах, у которых средний балл не ниже 4, и отсортировать результат по среднему баллу в порядке убывания.

**2.2. ВНЕШНИЕ СОЕДИНЕНИЯ**

Во внешнем соединении считываются все строки хотя бы из одной таблицы. При этом порядок следования таблиц для внешнего соединения, имеет значение. Рабочие таблицы внешнего соединения называются левой и правой таблицами. Внешние соединения бывают трех видов:

**1. Левое внешнее объединение** (LEFT OUTER JOIN). Результат исполнения левого внешнего соединения включает все строки из левой таблицы, заданные в предложении LEFT OUTER JOIN, а не только строки, которые совпадают для связанных столбцов. Если для строки в левой таблице нет соответствия в правой таблице, то в результате строка будет содержать NULL для всех столбцов в списке SELECT, которые были считаны из правой таблицы. Создание левого внешнего соединения:

SELECT columns

FROM left\_table

LEFT [OUTER] JOIN right\_table

ON join\_conditions

Здесь *columns* – это несколько разделенных запятыми выражений или названий столбцов из *left\_table* или *right\_table*;

– *left\_table* или *right\_table* – это названия связываемых таблиц;

– *join\_conditions* задает одно или несколько условий соединения.

Ключевое слово OUTER является опциональным (т.е. можно опустить).

**2. Правое внешнее соединение** (RIGHT OUTER JOIN) является прямой противоположностью левому. Считываются все строки из правой таблицы. Если для строки из правой таблицы нет соответствия в левой таблице, то в результате строка будет содержать NULL для всех столбцов в списке SELECT, которые были считаны из левой таблицы. Создание правого внешнего соединения:

SELECT columns

FROM left\_table

RIGHT [OUTER] JOIN right\_table

ON join\_conditions

**3. Полное внешнее соединение** (FULL OUTER JOIN) является комбинацией левого и правого внешних соединений. Считывает все строки как из левой, так и из правой таблиц. Если для строки нет соответствия в другой таблице, столбцы в списке SELECT другой таблицы будут содержать NULL. Если в таблицах есть соответствие, то строка будет содержать значения из них. Создание полного внешнего соединения:

SELECT columns

FROM left\_table

FULL [OUTER] JOIN right\_table

ON join\_conditions

***Задание 2.2:*** Создание внешних соединений с помощью предложения JOIN:

**а)** выбрать номера групп, должности, фамилии студентов из таблицы *table*\_1, а также названия дисциплин и итоговые оценки студентов по дисциплинам из таблицы *table*\_2 с использованием левого внешнего соединения;

***Решение:***

SELECT t1.n\_group, t1.rang, t1.fio, t2.subj, t2.mark FROM table\_1 t1 LEFT JOIN table\_2 t2 ON t1.id = t2.id

**б)** выбрать номера групп, должности, фамилии студентов из таблицы *table*\_1, а также названия дисциплин и итоговые оценки студентов по дисциплинам из таблицы *table*\_2 с использованием правого внешнего соединения;

***Решение:***

SELECT t1.n\_group, t1.rang, t1.fio, t2.subj, t2.mark FROM table\_1 t1 RIGHT JOIN table\_2 t2 ON t1.id = t2.id

**в)** выбрать номера групп, должности, фамилии студентов из таблицы *table*\_1, а также названия дисциплин и итоговые оценки студентов по дисциплинам из таблицы *table*\_2 с использованием полного внешнего соединения;

***Решение:***

SELECT t1.n\_group, t1.rang, t1.fio, t2.subj, t2.mark FROM table\_1 t1 FULL JOIN table\_2 t2 ON t1.id = t2.id

Проанализировать полученные результаты. Добиться правильного написания SQL-команд для выборки требуемых данных из таблиц *table*\_1 и *table*\_2. В случае возникновения ошибки при выполнении SQL-команды проанализировать выданное СУБД сообщение об ошибке и внести необходимые коррективы в ее код.

**3. ВЛОЖЕННЫЕ ЗАПРОСЫ**

Рассмотрим использование законченных операторов SELECT, внедренных в тело другого оператора SELECT. Внешний (второй) оператор SELECT использует результат выполнения внутреннего (первого) оператора для определения содержания окончательного результата всей операции. Внутренние запросы могут находиться в конструкциях WHERE и HAVING внешнего оператора SELECT – в этом случае они получают название подзапросов, или вложенных запросов. Кроме того, внутренние операторы SELECT могут использоваться в операторах INSERT, UPDATE и DELETE.

**Виды подзапросов:**

1. Скалярный подзапрос возвращает значение, выбираемое из пересечения одного столбца с одной строкой, т.е. единственное значение.

2. Строковый подзапрос возвращает значения нескольких столбцов таблицы, но в виде единственной строки.

3. Табличный подзапрос возвращает значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке.

Чаще всего подзапросы используются в предложении WHERE, которые будут принимать одну из следующих форм:

– WHERE test\_expr op (subquery);

– WHERE test\_expr [NOT] IN (subquery);

– WHERE test\_expr op ALL (subquery);

– WHERE test\_expr op ANY (subquery);

– WHERE [NOT] EXISTS (subquery).

Здесь *test\_expr* является названием столбца, выражением или запросом;

– *op* – это оператор сравнения (=, <>, <, <=, > или >=);

– *subquery* – подзапрос.

***Задание 3.1:*** Сравнение значений, возвращаемых подзапросом, с использованием операторов сравнения:

**а)** выбрать из таблицы *table*\_1 всю информацию о самом молодом студенте;

**б)** выбрать из таблиц *table*\_1 и всю информацию о студентах, у которых оклад выше среднего. Результат отсортировать по размеру оклада в порядке убывания, а при одинаковых окладах по фамилии в алфавитном порядке;

**в)** выбрать из таблиц *table*\_1 и *table*\_2 фамилии и средний балл студентов по всем дисциплинам, результат предоставить о тех студентах, у которых средний балл выше общего среднего балла (по всем записям из таблицы *table*\_2), результат отсортировать по среднему баллу в порядке убывания.

***Решение:***

***а)***

SELECT \* FROM table\_1 WHERE date\_b =

(SELECT MAX (date\_b) FROM table\_1)

***б)***

SELECT \* FROM table\_1 WHERE salary > (SELECT AVG (salary) FROM table\_1) ORDER BY salary DESC, fio

***в)***

SELECT t1.fio, AVG (t2.mark) FROM table\_1 t1, table\_2 t2 WHERE t1.id=t2.id GROUP BY t1.fio HAVING AVG (t2.mark) > (SELECT AVG (mark) FROM table\_2) ORDER BY AVG (t2.mark) DESC

***Задание 3.2:*** Проверка на вхождение во множество с использованием оператора IN:

**а)** выбрать из таблицы *table*\_1 всю информацию о студентах, имеющих неудовлетворительные оценки по дисциплинам;

**б)** выбрать из таблицы *table*\_1 всю информацию о студентах, не имеющих неудовлетворительные оценки по дисциплинам;

***Решение:***

***а)***

SELECT \* FROM table\_1 WHERE id IN

(SELECT id FROM table\_2 WHERE mark =2)

***б)***

SELECT \* FROM table\_1 WHERE id NOT IN

(SELECT id FROM table\_2 WHERE mark =2)

***Задание 3.3:*** Сравнение значений подзапроса с помощью ключевых слов ALL и ANY.

**Оператор ALL** выполняет булеву проверку подзапроса на предмет того, во всех ли строках существуют значения. Оператор ANY и его синоним SOME выполняют булеву проверку подзапроса на предмет того, существует ли хоть одно значение в анализируемых строках.

**Ключевые слова ANY и ALL** могут использоваться с подзапросами, возвращающими один столбец чисел. Если подзапросу будет предшествовать ключевое слово ALL, условие сравнения считается выполненным только в том случае, если оно выполняется для всех значений в результирующем столбце подзапроса. Если тексту подзапроса предшествует ключевое слово ANY, то условие сравнения будет считаться выполненным, если оно удовлетворяется хотя бы для какого-либо (одного или нескольких) значения в результирующем столбце подзапроса.

**а)** выбрать из таблиц *table*\_1 и *table*\_2 фамилии и средний балл студентов по всем дисциплинам, результат предоставить о тех студентах, у которых средний балл выше среднего балла по каждой из дисциплин. Результат отсортировать по среднему балу в порядке убывания;

**б)** выбрать из таблиц *table*\_1 и *table*\_2 фамилии и средний балл студентов по всем дисциплинам, результат предоставить о тех студентах, у которых средний балл ниже среднего балла хотя бы по одной из дисциплин. Результат отсортировать по среднему балу в порядке возрастания.

***Решение:***

***а)***

SELECT t1.fio, AVG (t2.mark) FROM table\_1 t1, table\_2 t2 WHERE t1.id=t2.id GROUP BY t1.fio HAVING AVG (t2.mark) > ALL (SELECT AVG (mark) FROM table\_2 GROUP BY subj)ORDER BY AVG (t2.mark) DESC

***б)***

SELECT t1.fio, AVG (t2.mark) FROM table\_1 t1, table\_2 t2 WHERE t1.id=t2.id GROUP BY t1.fio HAVING AVG (t2.mark) < ANY (SELECT AVG (mark) FROM table\_2 GROUP BY subj)ORDER BY AVG (t2.mark) DESC

***Задание 3.4:*** Проверка наличия выборки с использованием оператора EXISTS.

**Ключевые слова EXISTS и NOT EXISTS** используются для проверки того, вернул ли подзапрос хотя бы одну строку или нет. Ключевые слова EXISTS и NOT EXISTS предназначены для использования только совместно с подзапросами. Результат их обработки представляет собой логическое значение TRUE или FALSE. Для ключевого слова EXISTS результат равен TRUE в том и только в том случае, если в возвращаемой подзапросом результирующей таблице присутствует хотя бы одна строка. Если результирующая таблица подзапроса пуста, результатом обработки ключевого слова EXISTS будет значение FALSE. Для ключевого слова NOT EXISTS используются правила обработки, обратные по отношению к ключевому слову EXISTS. Поскольку по ключевым словам EXISTS и NOT EXISTS проверяется лишь наличие строк в результирующей таблице подзапроса, то эта таблица может содержать произвольное количество столбцов.

**а)** выбрать из таблицы *table*\_1 всю информацию о студентах, у которых есть оценки по «Физике»;

**б)** выбрать из таблицы *table*\_1 всю информацию о студентах, у которых нет оценок по «Физике»;

***Решение:***

***а)***

SELECT \* FROM table\_1 t1 WHERE EXISTS (SELECT \* FROM table\_2 t2 WHERE t1.id=t2.id AND subj = 'Физика')

***б)***

SELECT \* FROM table\_1 t1 WHERE NOT EXISTS (SELECT \* FROM table\_2 t2 WHERE t1.id=t2.id AND subj = 'Физика')

Проанализировать полученные результаты. Добиться правильного написания SQL-команд для выборки требуемых данных из таблицы *table*\_1. В случае возникновения ошибки при выполнении SQL-команды проанализировать выданное СУБД сообщение об ошибке и внести необходимые коррективы в ее код.

**ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ**

***Задание 1.*** Выбрать из таблицы *table*\_1 всю информацию о студентах, у которых оклад ***x***.

***Задание 2.*** Выбрать из таблицы *table*\_1 всю информацию о студентах, имеющих отличные оценки по « ***y*** »***.***

***Задание 3.*** Выбрать из таблицы *table*\_1 всю информацию о студентах, имеющих оценку ***z*** по дисциплинам.

Варианты по списку:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***x*** | ***y*** | ***z*** |  | ***x*** | ***y*** | ***z*** |
| **1** | Не минимальный | Физике | 2 | **16** | Не максимальный | Физике | 3 |
| **2** | Минимальный | Математике | 3 | **17** | Не минимальный | Математике | 2 |
| **3** | Максимальный | Информатике | 4 | **18** | Не максимальный | Информатике | 4 |
| **4** | Не максимальный | Физике | 5 | **19** | Минимальный | Физике | 5 |
| **5** | Не минимальный | Математике | 5 | **20** | Максимальный | Математике | 3 |
| **6** | Минимальный | Информатике | 4 | **21** | Минимальный | Информатике | 5 |
| **7** | Не максимальный | Физике | 3 | **22** | Не минимальный | Физике | 2 |
| **8** | Максимальный | Математике | 2 | **23** | Не максимальный | Математике | 4 |
| **9** | Не максимальный | Информатике | 2 | **24** | Не минимальный | Информатике | 5 |
| **10** | Максимальный | Физике | 3 | **25** | Максимальный | Физике | 5 |
| **11** | Минимальный | Математике | 5 | **26** | Минимальный | Математике | 2 |
| **12** | Не минимальный | Информатике | 4 | **27** | Не максимальный | Информатике | 5 |
| **13** | Не минимальный | Физике | 3 | **28** | Максимальный | Физике | 3 |
| **14** | Максимальный | Математике | 2 | **29** | Минимальный | Математике | 4 |
| **15** | Минимальный | Информатике | 5 | **30** | Не максимальный | Информатике | 5 |