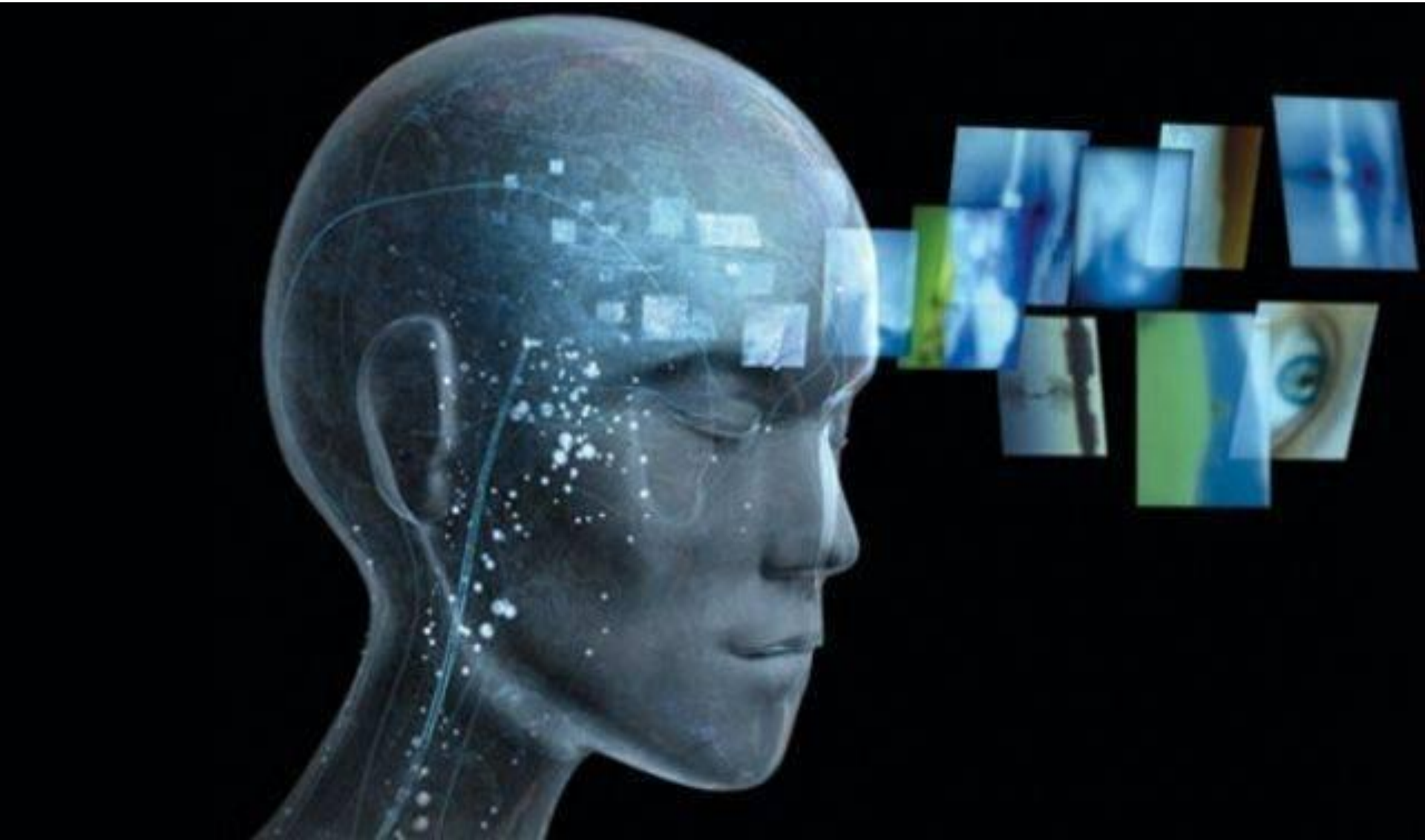
Изменение данных через представления



Ограничения применения оператора SELECT для создания представлений

- Одиночное представление должно основываться на одиночном запросе, поэтому **UNION** и **UNION ALL** **в представлениях не разрешаются.**
- Предложение **ORDER BY** **также никогда не используется** в определении представлений. Представление является реляционной таблицей-отношением, поэтому его строки по определению являются неупорядоченными.

Удаление представлений

- DROP VIEW <*имя представления*>

- *Например:*

DROP VIEW bonus;

Изменение значений в представлениях

- Запрос на обновление представления NEW_STUDENT

```
UPDATE NEW_STUDENT
SET CITY = ‘Москва ’
WHERE STUDENT_ID = 1004;
```

- эквивалентен выполнению команды UPDATE над базовой таблицей STUDENT.
- из-за того, что обычно в представлении отображаются данные из базовой таблицы в преобразованном или усеченном виде, применение команд модификации к таблицам-представлениям имеет некоторые особенности

Обновляемость представлений

- Если команды модификации могут выполняться в представлении, то представление является обновляемым (модифицируемым). Критерии обновляемости:

1. Представление строится на основе одной и только одной базовой таблицы.
2. Представление должно содержать первичный ключ базовой таблицы.
3. Представление не должно иметь никаких полей, которые представляют собой агрегирующие функции.
4. Представление не должно содержать DISTINCT в своем определении.
5. Представление не должно использовать GROUP BY или HAVING в своем определении.
6. Представление не должно использовать подзапросы.
7. Представление может быть использовано в другом представлении, но это представление должно быть также модифицируемыми.
8. Представление не должно использовать в качестве полей вывода константы или выражения значений.

Примеры обновляемых и не обновляемых представлений

Пример 1

```
CREATE VIEW DATEEXAM (EXAM_DATE, QUANTITY)
AS SELECT EXAM_DATE, COUNT (*)
FROM EXAM_MARKS
GROUP BY EXAM_DATE;
```

Данное представление является не обновляемым из-за присутствия в нем агрегирующей функции и GROUP BY.

Пример 2

```
CREATE VIEW LCUSTT
AS SELECT *
FROM UNIVERSITY
WHERE CITY = ‘Москва ’;
```

Это – обновляемое представление.

Примеры обновляемых и не обновляемых представлений

Пример 3

```
CREATE VIEW SSTUD (SURNAME1, NUMB, KUR)
AS SELECT SURNAME, STUDENT_ID, KURS*2
FROM STUDENT
WHERE CITY = ‘Москва ’;
```

Это представление – не модифицируемое из-за наличия выражения “**KURS*2**”.

Пример 4

```
CREATE VIEW STUD3
AS SELECT *
FROM STUDENT
WHERE STUDENT_ID IN
( SELECT MARK
FROM EXAM_MARKS
WHERE EXAM_DATE = ‘10/02/1999’);
```

Представление не модифицируется из-за присутствия в нем подзапроса

Примеры обновляемых и не обновляемых представлений

Пример 5

```
CREATE VIEW SOMEMARK
AS SELECT STUDENT_ID, SUBJ_ID, MARK
FROM EXAM_MARKS
WHERE EXAM_DATE IN ('10/02/1999',
   '10/06/1999');
```

Это – обновляемое представление.

Задание 1

Какие из представлений являются обновляемыми?

a) **CREATE VIEW DAILYEXAM AS**
SELECT DISTINCT STUDENT_ID, SUBJ_ID, MARK, EXAM_DATE
FROM EXAM_MARKS;

b) **CREATE VIEW CUSTALS AS**
SELECT SUBJECT.SUBJ_ID, SUM (MARK) AS MARK1
FROM SUBJECT, EXAM_MARKS
WHERE SUBJECT.SUBJ_ID = EXAM_MARKS.SUBJ_ID
GROUP BY SUBJECT.SUBJ_ID;

c) **CREATE VIEW THIRDEXAM**
AS SELECT *
FROM DAILYEXAM
WHERE EXAM_DATE = '10/02/1999';

d) **CREATE VIEW NULLCITIES**
AS SELECT STUDENT_ID, SURNAME, CITY
FROM STUDENT
WHERE CITY IS NULL
OR SURNAME BETWEEN 'А' AND 'Д';

Поддержка целостности данных: Внешние и родительские ключи

Поддержка целостности данных: Внешние и родительские ключи

- Поле, которое ссылается на другое поле, называется внешним ключом
- поле, на которое ссылается другое поле, называется родительским ключом .

- Так что поле UNIV_ID таблицы STUDENT – это внешний ключ (оно ссылается на поле другой таблицы), а поле UNIV_ID таблицы UNIVERSITY, на которое ссылается этот внешний ключ – это родительский ключ .

- На практике внешний ключ необязательно может состоять только из одного поля

Смысл внешнего и родительского ключей

- Каждое значение во внешнем ключе непосредственно привязано к конкретному значению в другом поле (родительском ключе).
- Значения родительского ключа должны быть уникальными
- Значения внешнего ключа не обязательно должны быть уникальными
- Строки, содержащие одинаковые значения внешнего ключа должны обязательно ссылаться на конкретное, присутствующее в данный момент в таблице, значение родительского ключа.
- ни в одной строке таблицы не должно быть значений внешнего ключа, для которых в текущий момент отсутствуют соответствующие значения родительского ключа.
- Если указанные требования выполняются в конкретный момент существования базы данных, то говорят, что данные находятся в согласованном состоянии, а сама база находится в состоянии ссылочной целостности .

Ограничение FOREIGN KEY (внешнего ключа)

- **FOREIGN KEY –** ограничение допустимых значений поля множеством значений родительского ключа , ссылка на который указывается при описании данного ограничения **FOREIGN KEY.**
- **FOREIGN KEY –** отклонение (блокировка) ввода значений внешнего ключа, отсутствующих в таблице с родительским ключом
- ограничение воздействует на возможность из менять или удалять значения родительского ключа

Внешний ключ как ограничение таблицы

- FOREIGN KEY <*список столбцов*>
- REFERENCES <*родительская таблица*> [<*родительский ключ*>];

- Создадим таблицу **STUDENT1** с полем **UNIV_ID**, определенным в качестве внешнего ключа , ссылающегося на таблицу **UNIVERSITY**:

```
CREATE TABLE STUDENT1
( STUDENT_ID INTEGER PRIMARY KEY,
SURNAME CHAR (25),
NAME CHAR (10),
STIPEND INTEGER,
KURS INTEGER,
CITY CHAR (15),
BIRTHDAY DATE,
UNIV_ID decimal(10,0),
CONSTRAINT UNIV_FOR_KEY1 FOREIGN KEY (UNIV_ID) REFERENCES
    UNIVERSITY (UNIV_ID));
```

Поле обязано иметь тот же тип, что и родительский ключ
Посмотреть типы полей таблицы в workbench можно с помощью Table Inspector

Если не писать CONSTRAINT

```
CREATE TABLE STUDENT2(
STUDENT_ID INTEGER PRIMARY KEY,
SURNAME CHAR (25),NAME CHAR (10),
STIPEND INTEGER,
KURS INTEGER,CITY CHAR (15),
BIRTHDAY DATE,UNIV_ID decimal(10,0),
FOREIGN KEY (UNIV_ID) REFERENCES UNIVERSITY (UNIV_ID));
```

- Имя ключу задается автоматически. Эта форма эквивалентна следующему ограничению таблицы STUDENT:

```
CONSTRAINT UNIV_FOR_KEY2 FOREIGN KEY (UNIV_ID)
REFERENCES UNIVERSITY (UNIV_ID).
```

Добавление ограничения в существующую таблицу

ALTER TABLE *<имя таблицы>*
ADD CONSTRAINT *<имя ограничения>*
FOREIGN KEY (*<список столбцов внешнего ключа>*)
REFERENCES *<имя родительской таблицы>*
[(*< список столбцов родительского ключа>*)];

- Например, команда

```
ALTER TABLE STUDENT1
ADD CONSTRAINT STUD_UNIV_FOR_KEY
FOREIGN KEY (UNIV_ID)
REFERENCES UNIVERSITY (UNIV_ID);
```

- добавляет ограничение внешнего ключа для таблицы STUDENT1.

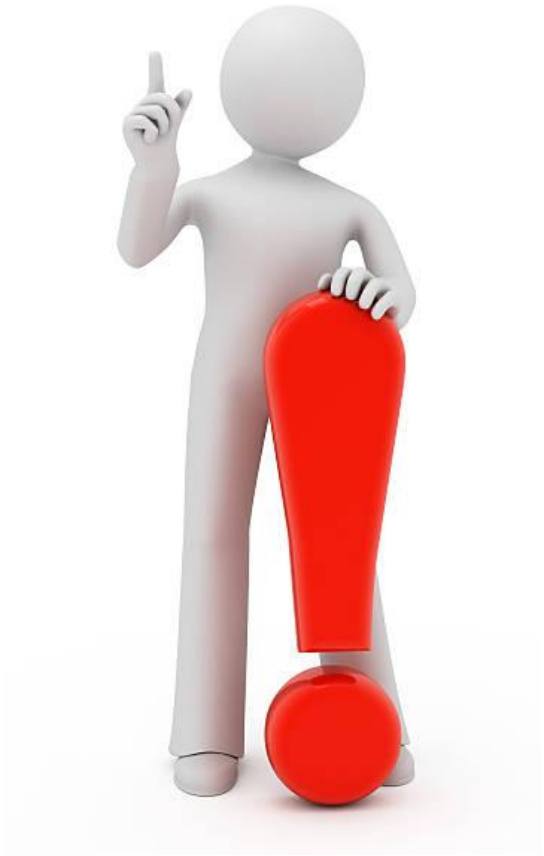
Задание 2

- Создайте таблицу с именем SUBJECT_1, с теми же полями, что в таблице SUBJECT (предмет обучения). Поле SUBJ_ID является первичным ключом.
- Создайте таблицу с именем SUBJ_LECT_1 (учебные дисциплины преподавателей), с полями LECTURER_ID (идентификатор преподавателя) и SUBJ_ID (идентификатор преподаваемой дисциплины). Первичным ключом (составным) таблицы является пара атрибутов LECTURER_ID и SUBJ_ID, кроме того, поле LECTURER_ID является внешним ключом, ссылающимся на таблицу LECTURER_1, аналогичную таблице LECTURER (преподаватель), а поле SUBJ_ID является внешним ключом, ссылающимся на таблицу SUBJECT_1, аналогичную таблице SUBJECT.

Поддержание ссылочной целостности

- Родительский ключ должен быть **уникальным** и **не содержать пустых значений (NULL)**. Следовательно, при объявлении внешнего ключа необходимо убедиться, что все поля, которые используются как родительские ключи, имеют или ограничение **PRIMARY KEY** или ограничения **UNIQUE** и **NOT NULL**.
- Внешний ключ может содержать только те значения, которые фактически представлены в родительском ключе, **или являются пустыми (NULL).** Попытка ввести другие значения в этот ключ должна быть отклонена, поэтому **объявление внешнего ключ а, как NOT NULL, не является обязательным.**

Использование первичного ключа в качестве уникального внешнего ключа



- Ссылка внешних ключей только на первичные ключ и считается хорошим стилем программирования SQL-запросов .

Действие ограничений при использовании команд модификации (delete, update)

[CONSTRAINT symbol] FOREIGN KEY (index_col_name, ...) REFERENCES table_name (index_col_name, ...)

[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT}]

[ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT}]

- **CASCADE** – если строка в родительской таблице удалена, то все эти строки автоматически удаляются также и из дочерней таблицы, значения внешнего ключа которой равны значениям ссылочного ключа в строке родительской таблицы.
- **SET NULL** – строки дочерней таблицы автоматически обновляются, поэтому столбцам во внешнем ключе также присваивается значение SQL NULL.
- **NO ACTION = RESTRICT** – если связанные записи родительской таблицы обновляются или удаляются со значениями которые уже/еще содержатся в соответствующих записях дочерней таблицы, то база данных не позволит изменять записи в родительской таблице

Действие ограничений при использовании команд модификации (delete, update)

```
CREATE TABLE NEW_EXAM_MARKS(
STUDENT_ID INTEGER NOT NULL,
SUBJ_ID INTEGER NOT NULL,
MARK INTEGER,
DATA DATE,
CONSTRAINT EXAM_PR_KEY PRIMARY KEY (STUDENT_ID, SUBJ_ID),
CONSTRAINT SUBJ_ID_FOR_KEY FOREIGN KEY (SUBJ_ID)
REFERENCES SUBJECT,
CONSTRAINT STUDENT_ID_FOR_KEY FOREIGN KEY (STUDENT_ID)
REFERENCES STUDENT  ON UPDATE CASCADE ON DELETE NO ACTION);
```

- В этом примере при попытке **изменения значения поля STUDENT_ID** таблицы STUDENT будет автоматически обеспечиваться **каскадная корректировка этих значений** в таблице EXAM_MARKS. То есть при изменении идентификатора студента STUDENT_ID в таблице STUDENT сохранятся в се ссылки на его оценки. Однако любая **попытка удаления (DELETE) записи о студенте** из таблицы STUDENT **будет отвергаться**, если в таблице EXAM_MARKS существуют записи об оценках данного студента .

Задание

- Создайте таблицу с именем SUBJ_LECT_1 как в предыдущем задании, но добавьте для всех ее внешних ключей режим обеспечения ссылочной целостности, запрещающий обновление и удаление соответствующих родительских ключей.


Временные таблицы

Временные таблицы

- Используют, если возникает необходимость хранить и менять временные данные
- по возможности создаются в памяти, поэтому с ними быстрее операции *INSERT/UPDATE*(при превышении определенного объема они автоматически пишутся на диск).
- автоматически удаляются при закрытии соединения
- видны только для текущего соединения.
- можно разбить один сложный *SELECT* на несколько простых с использованием временных таблиц

Пример

```
CREATE TEMPORARY TABLE tmp2 (SELECT *
   FROM STUDENT WHERE CITY = 'Воронеж');

SELECT * FROM EXAM_MARKS WHERE
   STUDENT_ID IN (select STUDENT_ID from
   tmp2);
```

Пример2

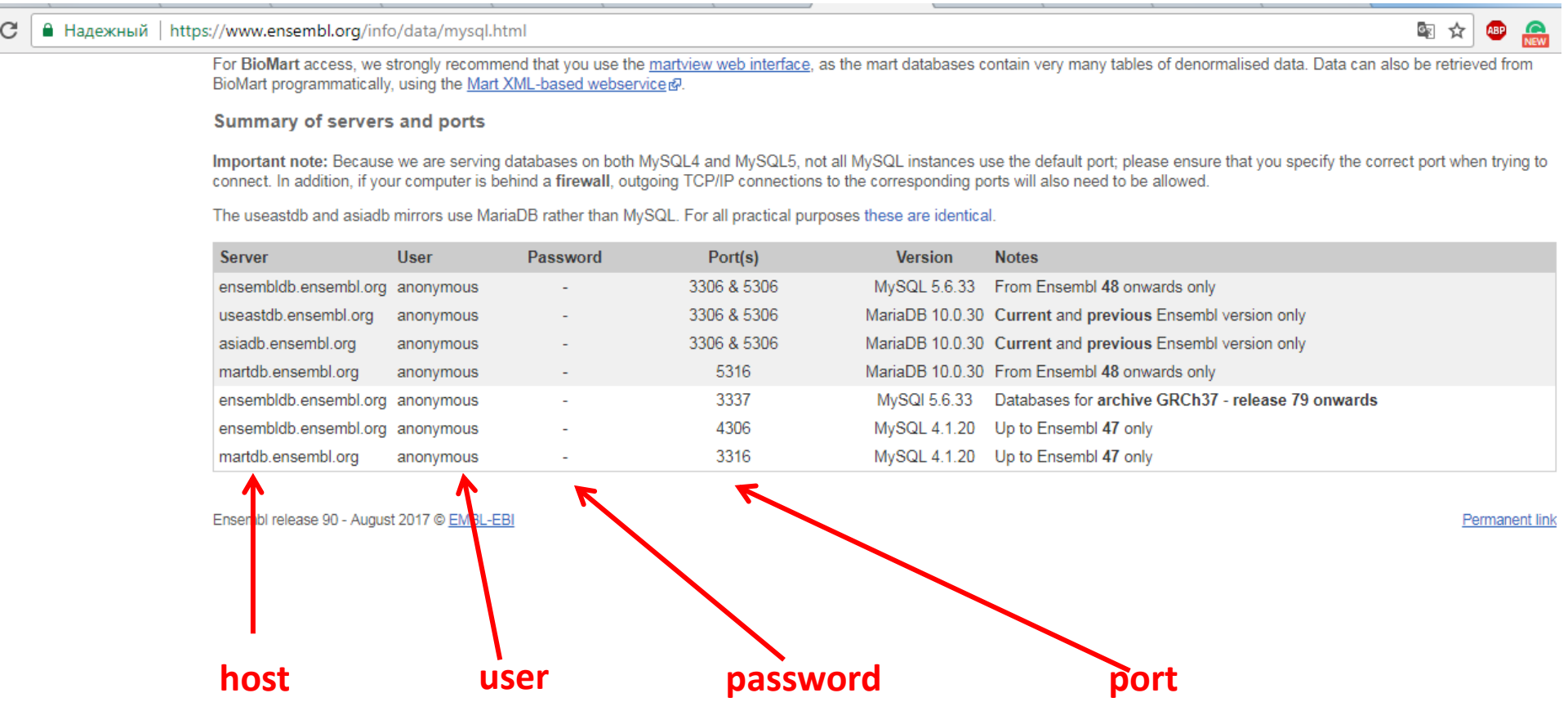
```
CREATE TEMPORARY TABLE SalesSummary (
    product_name VARCHAR(50) NOT NULL,
    total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00,
    avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00,
    total_units_sold INT UNSIGNED NOT NULL DEFAULT 0
);

INSERT INTO SalesSummary(product_name, total_sales,
    avg_unit_price, total_units_sold)VALUES('cucumber',
    100.25, 90, 2);
```

Задание

- Создайте временную таблицу для преподавателей из Кирова
- Проверьте, что в новой сессии таблицы нет

Подключение к внешней БД

Как подключиться к внешней базе?



Как подключиться к внешней базе?