## **Лабораторная работа № 7 Работа с представлениями**

**Теоретические сведения**

Рассмотрим следующие вопросы:

* создание представлений с помощью оператора *CREATE VIEW*;
* удаление представлений с помощью оператора *DROP VIEW*.

Основная структурная единица реляционных БД – таблицы, но язык SQL предоставляет еще один способ организации данных. Представление – это запрос на выборку, которому присваивается уникальное имя и который можно сохранять или удалять из БД как хранимую процедуру. Представления позволяют увидеть результаты сохраненного запроса так, как будто это полноценная таблица. MySQL, встретив в запросе ссылку на представление, ищет его определение в БД. Пользовательский запрос с участием представления преобразуется в эквивалентный запрос к исходным таблицам. Если определение представления простое, то каждая строка представления формируется «на лету». Если определение сложное, MySQL материализует представление в виде временной таблицы. Клиент, обращаясь к представлению, будет видеть только столбцы результирующей таблицы. Не имеет значения, сколько столбцов в исходной таблице и является ли запрос, лежащий в основе представления, одно- или многотабличным. Клиенту можно запретить обращаться к исходным таблицам, но снабдить привилегиями обращения к представлениям. На одном наборе таблиц можно создать гибкие системы доступа.

Преимущества представлений:

* *безопасность* – каждый пользователь имеет доступ к небольшому числу представлений, содержащих только ту информацию, которую ему позволе­но знать;
* *простота запросов* – с помощью представления можно извлечь данные из не­скольких таблиц и представить их как одну таблицу (запрос ко многим таблицам заменяется однотабличным запросом к представлению);
* *простота структуры* – представления позволяют создать для каждого пользовате­ля собственную структуру БД (отображаются данные, которые ему нужны);
* *защита от изменений* – таблицы и их структура могут постоянно изменяться и переименовываться; представления позволяют создавать виртуальные таблицы со старыми именами и структурой, позволяя избежать модификации приложений.

Недостатки представлений:

* *производительность* – представления создают видимость существования таблицы, и MySQL приходится преобразовывать запрос к представлению в запрос к исходным таблицам; если представление отображает многотабличный запрос, то простой запрос к представлению становится сложным объединением;
* *ограничение на обновление* – когда пользователь пытается обновить строки представления, MySQL необходимо обновить строки в исходных таблицах; это возможно только для простых представлений, сложные представления доступны только для выборки.

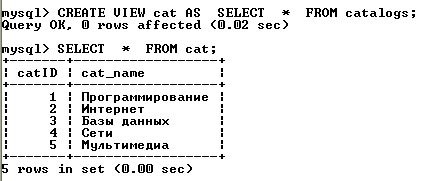
Поэтому не стоит везде применять представления вместо исходных таблиц.

**Создание представлений**. Осуществляется при помощи оператора

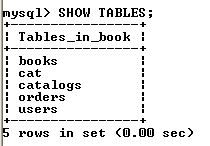
*CRFATE [OR REPLACE] [ALGORITHM = {UNDEFINED / MERGE / TEMPTABLE}] VIEW view\_name [(column\_list)] AS select\_statement*

*[WITH [CASCADED / LOCAL] CHECK OPTION];*

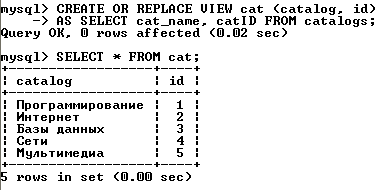
Оператор создает представление *view\_name* со столбцами, перечисленными в *column\_list*, на основании запроса *select\_statement*. Рассмотрим создание представления *cat*, которое дублирует таблицу *catalogs* базы данных *book*:



Представление рассматривается как полноценная таблица и может быть просмотрено в списке таблиц БД при помощи оператора *show tables*:



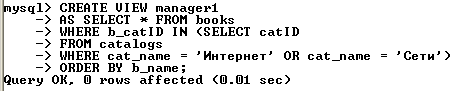
При создании представления можно явно указать список столбцов, изменить их названия и порядок следо­вания, например:

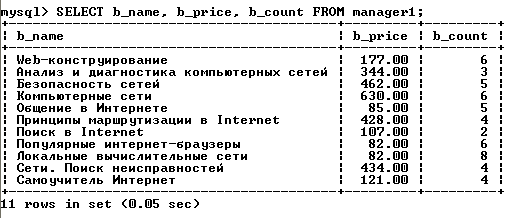


Представления, не содержащие дополнитель­ных столбцов, называются *вертикальными представлениями.* Они применяются для ограничения доступа пользователей к столбцам таблицы. Пример вертикального пред­ставления см. ниже.

Кроме вертикальных представлений используются *горизонтальные представ­ления,* которые делают види­мыми только те строки, с которыми работают пользователи. Например, чтобы в электронном магазине каждый менеджер видел только те товарные позиции, за которые отвечает, можно создать представления для менеджеров. Учетные записи менеджеров следует лишить привилегии доступа к таблице и разрешить просматривать только свои представления.

Создадим представление *manager1* для менеджера, работающего с каталогами «Интернет» и «Сети»:



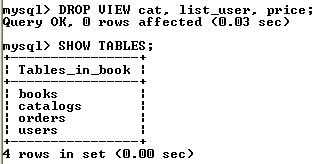


Наиболее удобно использовать представления для формирования сгруппированных таблиц. При работе с такими таблицами MySQL самостоятельно формирует временную таблицу – см. пункт «Пример выполнения работы» (пример 2).

**Удаление представлений.** Выполняется с помощью оператора:

*DROP VIEW [IF EXISTS] view\_name [, view\_name] … ;*

Оператор позволяет уничтожить одно или несколько представлений, например:



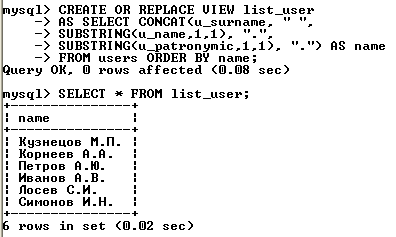
**Практическая работа**

При выполнении лабораторной работы необходимо:

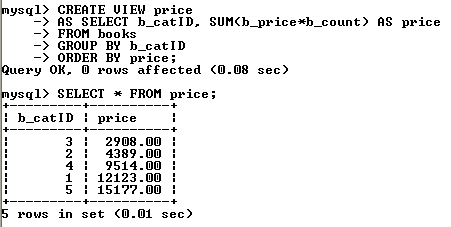
* для заданной предметной области создать два представления в БД;
* сформировать запрос к одному из представлений;
* составить отчет по лабораторной работе.

**Пример выполнения работы**

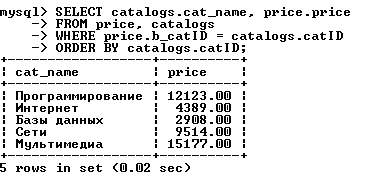
* + 1. Создадим вертикальное представление *list\_user*, которое будет отобра­жать фамилию и инициалы покупателей, скрывая другие поля.



2. Создадим представление *price* с общей стоимостью книг в каждом каталоге.



Сформируем запрос к таблице *catalogs* и представлению *price*.



Сформируем запрос к представлению *price* (получение минимального и максимального значений стоимости книг в каталогах и общей стоимости всех книг).

