Создание и использование представлений (VIEW)

Нашу учебную базу данных со списками задач можно использовать в практических целях, например, таким образом. Предположим, мы решили, что должны каждый день выполнять хотя бы по одной задаче из каждой категории. Для этого мы можем ежедневно распечатывать список невыполненных задач, сгруппировав их по категориям и названиям списка.

В этом нам поможет запрос, который мы уже составляли раньше:

```
select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
and Done=0
order by Category, Tasklist
```

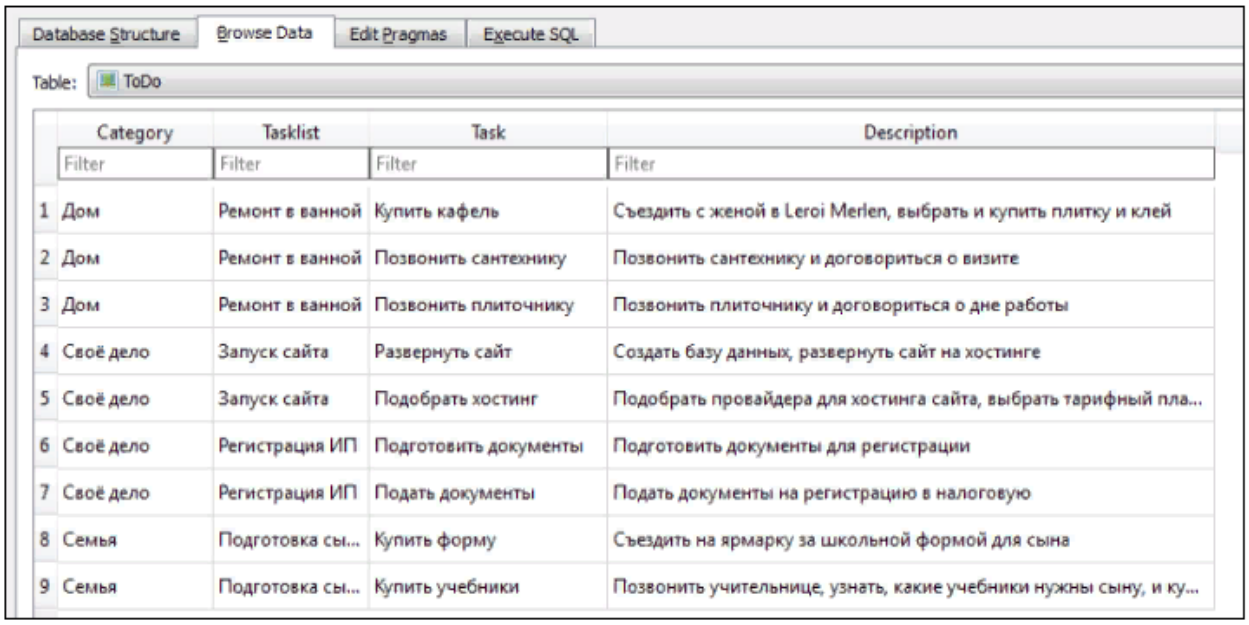
В запросе участвуют все три таблицы, связанные двумя условиями, а результат сортируется по двум полям. Вводить этот запрос каждый день в базу данных довольно утомительно, не правда ли? Хочется как-то упростить эту задачу.

Можно, конечно, сохранить этот запрос в отдельном текстовом файле, и ежедневно копировать его в программу для выполнения. Но SQL предоставляет гораздо более удобный способ: вы можете сохранить запрос прямо в базе данных в виде *представления* (это довольно неуклюжий перевод ключевого слова VIEW, но он прижился и используется повсеместно).

Выполните в учебной базе данных следующую команду:

```
create view ToDo as
select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
and Done=0
order by Category, Tasklist
```

Теперь перейдите на вкладку BrowseData и найдите в выпадающем списке элемент ToDo:



Database Structure | Browse Data | Edit Pragmas | Execute SQL

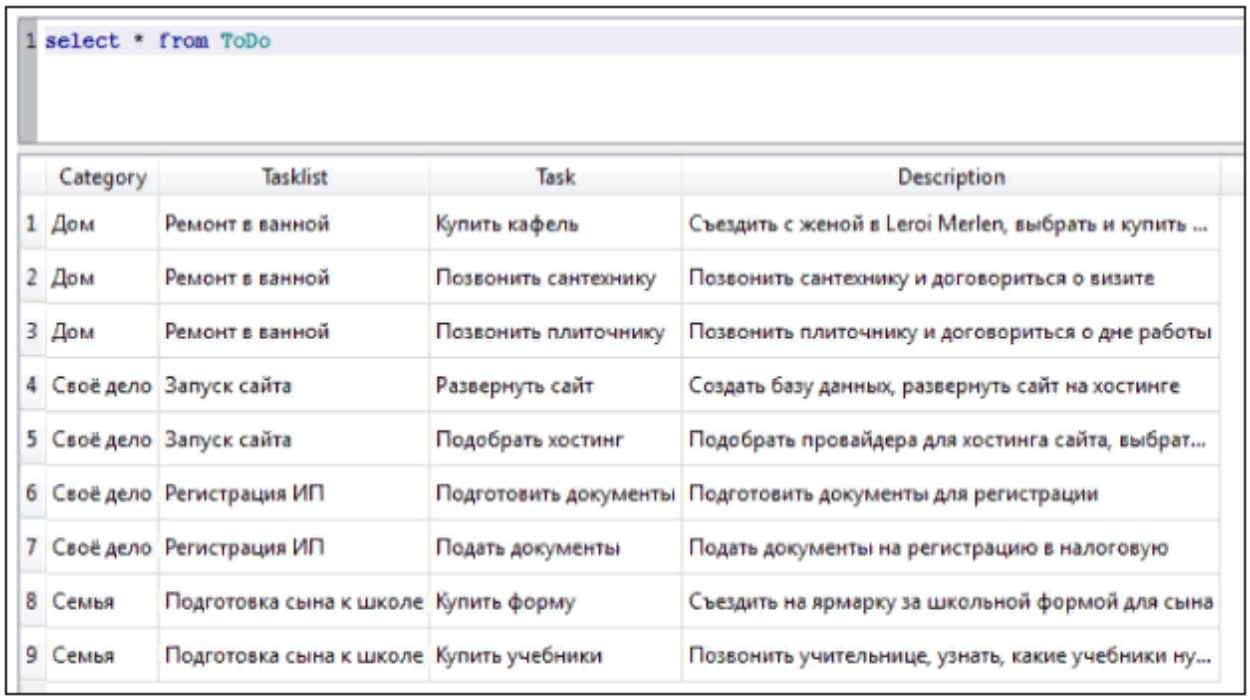
Table: ToDo

	Category	Tasklist	Task	Description
	Filter	Filter	Filter	Filter
1	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
2	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы
4	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
5	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрать тарифный пла...
6	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
7	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налоговую
8	Семья	Подготовка сы...	Купить форму	Съездить на ярмарку за школьной формой для сына
9	Семья	Подготовка сы...	Купить учебники	Позвонить учительнице, узнать, какие учебники нужны сыну, и ку...

Что это, новая таблица? Нет, это не таблица, а созданное нами представление. Но оно действительно может использоваться как таблица в запросах SELECT.

Выполните, для начала, такой запрос, и посмотрите на результат.

```
select * from ToDo
```



```
1 select * from ToDo
```

	Category	Tasklist	Task	Description
1	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...
2	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы
4	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
5	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...
6	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
7	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налоговую
8	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой для сына
9	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...

Создание и удаление представлений

Мы получили тот же результат, который вернул исходный запрос, обращающийся к трём таблицам. Но, конечно, вводить новый запрос гораздо удобнее.

Давайте разберёмся, каким же образом мы создали это представление. Команду CREATE VIEW, которую мы использовали, можно в обобщённом виде представить следующим образом:

```
create view {имя представления} as {запрос select}
```

Здесь мы видим уже знакомое нам по команде CREATE TABLE ключевое слово CREATE, а также два новых ключевых слова: VIEW и AS. Ключевое слово VIEW указывает на тип создаваемого объекта, а AS служит связкой между именем создаваемого представления и запросом SELECT, из которого оно создаётся.

Вы, наверное, уже догадались, какой командой можно уничтожить созданное представление. Для этого используется ключевое слово DROP. В обобщённом виде команда выглядит так:

```
drop view {имя представления}
```

Она очень похожа на уже знакомую нам команду DROP TABLE. Но, в отличие от DROP TABLE, эта команда не уничтожает никаких данных, так как представление само по себе не содержит записей.

Можно сказать, что содержимое представления создаётся заново каждый раз, когда к нему обращаются. Если мы поменяем данные в одной таблиц, участвующей в представлении, то изменится и результат запроса к представлению. Вы можете в этом убедиться, выполнив следующий запрос, который помечает задачу «Купить кафель» как выполненную:

```
update Tasks set Done=1 where Task='Купить кафель'
```

Если теперь снова выполнить запрос, выбирающий все «записи» из представления ToDo, то вы увидите, что эта задача исчезла из списка:

```
1 select * from ToDo
```

	Category	Tasklist	Task	Description
1	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
2	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы
3	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
4	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...
5	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
6	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налоговую
7	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой для сына
8	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...

Представления в запросах SELECT

В запросах SELECT представление ничем не отличается от таблицы. К нему можно применять запросы с условиями отбора, например:

```
select Task, Description from ToDo where Category='Своё дело'
```

```
select Task, Description from ToDo where Category='Своё дело'
```

	Task	Description
1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
2	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...
3	Подготовить документы	Подготовить документы для регистрации
4	Подать документы	Подать документы на регистрацию в налоговую

Представление «ничего не знает» о тех полях таблиц, которые в него не включены. Например, в наше представление ToDo не включены номера задач, и следующий запрос не будет выполнен:

```
select Task_ID from ToDo
```

База данных вернёт нам ошибку:

```
no such column: Task_ID: select Task_ID from ToDo
```

Но, как мы уже говорили, в запросах SELECT представление ничем не отличается от таблицы, и, следовательно, может участвовать в сложных запросах, обращающихся к нескольким таблицам. А значит, если нам это зачем-то нужно, мы можем узнать номера задач, вошедших в список ToDo, с помощью такого запроса:

```
select Task_ID, ToDo.Task
from Tasks, ToDo
where Tasks.Task=ToDo.Task
```

	Task_ID	ToDo.Task
1	4	Позвонить сантехнику
2	8	Позвонить плиточнику
3	1	Развернуть сайт
4	7	Подобрать хостинг
5	9	Подготовить документы
6	11	Подать документы
7	5	Купить форму
8	6	Купить учебники

Обратите внимание: нам пришлось использовать в списке полей полное имя поля ToDo.Task, потому что поле с таким именем есть и в представлении ToDo, и в таблице Tasks. Если бы мы этого не сделали, то база данных вернула бы нам ошибку

```
ambiguous column name: Task: select Task_ID, Task
from Tasks, ToDo
where Tasks.Task=ToDo.Task
```

Использование представлений

Мы создали наше первое представление для того, чтобы не вводить каждый раз руками длинный запрос SELECT. Конечно, эта проблема актуальна только в том случае, если все запросы SQL вводятся вручную.

На заре развития баз данных так и было: пользователи вводили запросы к базе данных вручную через консоль, и одной из целей представлений в языке SQL действительно было повышение скорости ввода запросов и минимизация ошибок при вводе.

Но представлениям нашлось широкое применение и после того, как на смену консолям пришли графические интерфейсы, а запросы к базам данных стали посылать не люди, а программы. Мы рассмотрим только один пример, которым использование представлений, конечно, далеко не исчерпывается.

Предположим, что нам так понравилось использование нашей учебной базы данных со списками задач, что мы научили свою жену (мужа, друга, подругу и т. п.) пользоваться ей, а именно: ежедневно распечатывать список невыполненных задач. Иными словами, у нашей базы данных появился ещё один пользователь.

Но в какой-то момент нам стала тесной примитивная структура базы данных. Мы решили добавить к ней новые возможности: приоритеты задач, плановые даты их выполнения, состояния «срочная задача» и «отложенная задача» и т. д. Мы ставим эксперименты с базой, добавляя к ней новые таблицы и поля. Но при этом важнейшей функцией для второго пользователся остаётся вывод списка невыполненных задач, который даёт представление ToDo. Этот пользователь умеет выполнять только запрос SELECT * FROM ToDo для получения списка задач на сегодняшний день.

Представление ToDo позволит нам скрыть от второго пользователя особенности устройства нашей базы данных. Мы можем создавать новые таблицы, добавлять и удалять поля, но если мы при этом соответствующим образом изменяем запрос, формирующий представление ToDo, то с точки зрения другого пользователя ничего не изменится: его запрос работает так, как будто в базе данных ничего не менялось.

Мы в очень упрощённом виде описали способ решения проблемы, которая постоянно возникает при выпуске новых версий программ. Чтобы добавить в программу новые возможности, надо изменять структуру базы данных, но при этом старые функции должны работать так, как будто в базе данных ничего не изменилось. Поэтому сокрытие структуры базы данных с помощью представлений используется очень широко.

Создание и использование представлений — практика

Составьте запрос, создающий представление с именем Shops, содержащее перечень магазинов, в которых была совершена хотя бы одна покупка.

Составьте запрос, создающий представление с именем Big_Spendings, содержащее следующую информацию по покупках на сумму не менее 2000 руб:

- название магазина,
- дата покупки,

- сумма покупки.

Имена соответствующих полей должны быть такими же, как в таблице Spendings.

Составьте запрос, создающий представление с именем Shop_Categories, содержащее названия магазинов (поле Shop) и наименования категорий, к которым относятся покупки, сделанные в этих магазинах (поле Category).

4. Составьте запрос, создающий представление с именем Goods_Details, содержащее следующую информацию обо всех покупках отслеживаемых товаров:

- наименование товара (Goods_Name),
- дата покупки (Spending_Date),
- количество единиц (Quantity),
- потраченная на этот товар сумма (Cost).