**Лабораторная работа № 5. MySql. Встроенные функции, представления, хранимые функции**

|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. На основе базы данных **BaseMysql\_x** сформировать различные запросы с использованием *встроенных функций работы со строками*.  Используя в качестве строки свое ФИО, записанное в виде «Фамилия Имя Отчество», вывести в первый столбец фамилию с помощью функций **CONCAT**, **LEFT** и **SUBSTRING**, во второй столбец имя и отчество с помощью функции **RIGHT**. | Для работы со строками в MySQL определен ряд **встроенных функций**: CONCAT (объединяет строки), LEFT, RIGHT, SUBSTRING (вырезание определенного количества символов с левой стороны текста, с правой или с некоторой позиции строки), REPLACE (замена одной подстроки на другую), LOWER, UPPER (перевод строку в нижний или в верхний регистр), REPEATE (повтор подстроки) и др.  Например, чтобы вывести информацию одновременно о заказчике и его адресе можно использовать запрос:  **SELECT CONCAT(Наименование\_фирмы, ' ', Адрес) As Фирма FROM Заказчики;**  Для замены данных можно применить оператор:  **SELECT replace(Наименование\_фирмы, 'Луч', 'Марко') FROM Заказчики;**  Чтобы повторить текст служит команда:  **SELECT** **repeat**(**'xa'**, **6**); |
| 2.На основе таблиц базы данных **BaseMysql\_x** сформировать запросы с использованием *встроенных функций для работы с числовыми данными, датами и временем*. | Для работы с числовыми данными MySQL предоставляет ряд функций: ROUND (определяет количество знаков после запятой), CEILING, FLOOR (округление чисел в меньшую или большую сторону), RAND (генерация случайного числа) и др. К числовым функциям относятся функции возвращающие абсолютные значения, синусы и косинусы углов, квадратный корень числа и т. д.  Некоторые встроенные функции для работы с датами и временем: NOW (возвращает текущую дату и время в формате `YYYY-MM-DD hh:mm:ss`), CURDATE (возвращает значение текущего года, месяца, дня), DATE (значение даты в формате `YYYY-MM-DD`), DATEDIFF (вычисляет разницу в днях между первым и вторым параметрами), DATE\_ADD (добавляет одну дату к другой), EXTRACT (извлекает часть даты) и др.  Вывести товары с ценой, превышающей 100, с двумя знаками после запятой:  **SELECT Наименование, round(Цена, 2) As 'Цена товара' FROM Товары**  **where Цена 100;**  Пусть надо найти заказы, доставка которых должна была осуществиться 5 дней тому назад:  **SELECT \* FROM Заказы WHERE DATEDIFF(CURDATE(), Дата\_поставки) = 5;**  Определить заказы, доставка которых запланирована на десятый месяц года:  **SELECT \* FROM Заказы WHERE EXTRACT(MONTH FROM Дата\_поставки) = 10;**  Найти товары и заказчиков, у которых дата поставки больше текущей:  **SELECT Наименование\_товара, Заказчик FROM Заказы**  **WHERE Дата\_поставки curdate();** |
| 3.Для базы данных **BaseMysql\_x** разработать *представления*, которые упрощают запись запросов. | **Представление** (View) – это объект базы данных, представляющий собой *поименованный* SELECT-запрос. Представления дают возможность пользователям и приложениям получать данные без предоставления им непосредственного доступа к таблицам.  Например, пусть разработано представление, в котором определяются наименования заказанных товаров, цены их продаж, заказчики и даты поставки:  **CREATE** **View** **Zakaz** **As**  **SELECT Наименование\_товара 'Товары',**  **Цена\_продажи,**  **Заказчик**,  **Дата\_поставки FROM Заказы;**    Тогда запрос, в котором выводится информация о товарах, заказанных конкретным заказчиком, можно записать кратко:  **SELECT \* from Zakaz** **where Заказчик = 'Луч';** |
| 4.Разработать *хранимую функцию* для базы данных **BaseMysql\_x**. Функция должна принимать один или несколько параметров и возвращать один результат. | **Хранимая функция** – это объект БД, представляющий собой поименованный код T-SQL. Чтобы добавить хранимую функцию в базу данных MySQL надо использовать оператор CREATE FUNCTION.  Хранимая функция может возвращать только одно значение.  Например, пусть хранимая функция **CenaZakazy** выводит текст, в котором определяется уровень цены товара.  **DELIMITER //**  **CREATE FUNCTION CenaZakazy( cena int )**  **RETURNS varchar(20)**  **DETERMINISTIC**  **BEGIN**  **DECLARE cost varchar(20);**  **Case**  **When cena < 50 Then set cost = 'цена < 50';**  **When cena between 50 and 100 Then set cost ='цена от 50 до 100';**  **else set cost = 'цена 100';**  **END CASE;**  **RETURN cost;**  **END; //**  **DELIMITER ;**  Здесь характеристика DETERMINISTIC указывает, что функция будет возвращать одни и те же результаты для одних и тех же входных параметров при каждом вызове функции.  Первый оператор DELIMITER изменяет разделитель на двойной прямой слэш (//), а второй оператор DELIMITER меняет разделитель обратно на точку с запятой. Это позволяет передать на сервер все определение функции как единый оператор.  Информацию об уровне цен товаров, заказанных конкретным заказчиком, можно найти с помощью запроса:  **SELECT Заказы.Наименование\_товара, CenaZakazy(Заказы.Цена\_продажи)**  **FROM Заказы where Заказы.Заказчик='Луч';**  Пусть требуется подсчитать количество товаров, заказанных фирмой. Хранимая функция для подсчета может иметь следующий вид:  **DELIMITER //**  **CREATE FUNCTION CountZakazy(f varchar(20))**  **RETURNS int**  **DETERMINISTIC**  **BEGIN**  **SELECT count(Номер\_заказа) FROM Заказы z Inner Join Заказчики zk**  **On zk.Наименование\_фирмы = z.Заказчик**  **where zk.Наименование\_фирмы = f INTO c;**  **RETURN c;**  **END; //**  **DELIMITER ;**  Определить количество заказов для всех фирм можно с помощью запроса:  **select Наименование\_фирмы,**  **CountZakazy(Наименование\_фирмы) from Заказчики;** |

[В начало практикума](#_Содержание)

## Лабораторная работа № 6. MySql. Хранимые процедуры, индексы, курсоры

|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. Разработать *хранимые процедуры* для базы данных **BaseMysql\_x** и продемонстрировать их работу. | **Хранимая процедура** ̶ это набор инструкций SQL, которые могут храниться на сервере. Процедура может вернуть одно значение, несколько значений, результирующий набор.  Пример процедуры подсчета количества всех заказов.  **DELIMITER //**  **CREATE PROCEDURE CountZakazy()**  **BEGIN**  **DECLARE K INT;**  **set k = (SELECT COUNT(Номер\_заказа) FROM Заказы);**  **SELECT k;**  **END//**  **DELIMITER ;**  Вызов процедуры: **call CountZakazy;**  В хранимой процедуре, которая осуществляет ввод данных в таблицу Товары используются три входных параметра:    **DELIMITER //**  **create procedure TovaryInsert (t varchar(20), cn real, kl int)**  **begin**  **INSERT into Товары (Наименование, Цена, Количество) values (t, cn, kl);**  **end//**  **DELIMITER ;**  Вызов процедуры: **call TovaryInsert ('Тумба', 20, 30);**  В следующей хранимой процедуре используется один выходной параметр:  **delimiter //**  **CREATE PROCEDURE simpleproc (OUT param1 INT)**  **BEGIN**  **SELECT COUNT(\*) INTO param1 FROM Заказчики;**  **END;//**  **delimiter ;**  **CALL simpleproc(@a);**  **SELECT @a;**  Символ **@** перед переменной означает глобальную переменную. |
| 2. Разработать *индексы* для быстрого поиска информации в базе данных и исследовать их действия.  3. Изучить использование команды EXPLAIN для оптимизации запросов и получения информации об индексах. | **Индекс** – это объект базы данных, позволяющий **ускорить поиск** в определенной таблице, так как при этом данные организуются в виде сбалансированного бинарного дерева поиска.  Для исследования индекса можно создать временную таблицу с именем, например, **EX**:  **CREATE DATABASE temporary;**  **USE temporary;**  **CREATE temporary table EX**  **( Ckey int auto\_increment,**  **Number int,**  **Words varchar(100),**  **PRIMARY KEY (Ckey)**  **);**  Заполнить эту таблицу можно с помощью процедуры и оператора ее вызова:  **DELIMITER //**  **create PROCEDURE zapolnenie()**  **BEGIN**  **DECLARE i INT DEFAULT 0;**  **WHILE i < 1000 DO**  **INSERT EX(Number, Words) values(floor(30000\*RAND()), repeat('строка ', 10));**  **set i = i + 1;**  **END WHILE;**  **END;//**  **DELIMITER ;**  **CALL zapolnenie();**  Чтобы проверить, какие индексы используются в таблице, надо использовать команду:  **SHOW INDEXES from EX;**  Время выполнения запроса для подсчета количества строк в таблице определяется с помощью скрипта:  **set profiling=1; #подключение профилирования (анализа производительности)**  **SELECT count(\*) from EX;**  **show profiles;**  Результат – время выполнения запроса **0.01576825** сек. После создания индекса, например, по столбцу **Number**, время выполнения значительно уменьшится.  **CREATE index EXPLRE on EX(Number asc);** |
| 4. Разработать *курсоры* для базы данных **BaseMysql\_x** и продемонстрировать их работу. | **Курсор** является программной конструкцией, которая дает возможность пользователю обрабатывать строки результирующего набора запись за записью. В MySQL курсоры могут использоваться только в хранимых процедурах.  Пусть требуется определить количество заказов конкретной фирмы и вывести результат в определенном виде: **Фирма …, количество заказов - …**  **DELIMITER //**  **CREATE PROCEDURE ProcCurcor(zk varchar(20))**  **BEGIN**  **DECLARE done int DEFAULT false;**  **DECLARE c int default 0;**  **DECLARE cur1 CURSOR FOR**  **SELECT Заказчик from Заказы where Заказчик=zk;**  **DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = true;**  **OPEN cur1;**  **read\_loop: LOOP**  **FETCH cur1 INTO zk;**  **IF done THEN LEAVE read\_loop; END IF;**  **set c = c + 1; #определение количества заказов в цикле LOOP**  **END LOOP;**  **select (CONCAT('Фирма ', zk, ', ', 'количество заказов', ' - ', c));**  **CLOSE cur1;**  **END;//**  **DELIMITER ;**  Курсор объявляется в операторе DECLARE. Переменная **done** и оператор If используются для определения конца обрабатываемых данных.  Для выполнения курсора нужно выполнить команду:  **call ProcCurcor('Луч');** |

## 5. Создать хранимую процедуру, которая принимает входные параметры и возвращает выходные, а также использует курсор для базы данных BaseMysql\_x.

[В начало практикума](#_Содержание)

## 