**Курсовая работа**

**по теме СУБД**

**студента....**

**Геннадия Балан**

В рамках этой курсовой работы была разработана база данных "Ресторан", предназначенная для хранения данных работы ресторана.

**В БД всего создано 13 таблиц:**

1. Меню
   1. **Menus** - меню (Основное, дессерты, винная карта)
   2. **MenusGroups** - разделы меню (закуски, холодные блюда, напитки и т.д.)
   3. **MenusItems** - наименования блюд
2. Заказы
   1. **Orders** - заказы
   2. **Ordersitems** - блюда в заказах
   3. **Payments** - платежи по заказам
3. Скидки
   1. **Discounts** - скидки
   2. **Discountscards** - дисконтные карты
   3. **Discountgroup** - виды дисконтных карт
4. Сотрудники
   1. **Employes** - сотрудники
   2. **Positions** - должности
   3. **Timesheet** - табель учета рабочего времени сотрудников ресторана
5. Прочее
   1. **Log** - лог сообщений

Меню ресторана представлено тремя таблицами, связанными внешними ключами:

**CREATE TABLE `Menus` (**

**`id` INT NOT NULL, -- уникальный идентификатор меню**

**`name` LONGTEXT NOT NULL, -- наименование меню**

**`active` TINYINT NOT NULL DEFAULT 0, -- меню в работе**

**PRIMARY KEY (`id`));**

**CREATE TABLE `MenusGroups` (**

**`id` INT NOT NULL, -- уникальный идентиф. группы**

**`menu` INT NOT NULL,** **-- внешний ключ к Menus**

**`name` VARCHAR(45) NULL, -- название группы**

**PRIMARY KEY (`id`),**

**INDEX `fk\_MenuGroups\_Menu\_idx` (`menu` ASC) VISIBLE,**

**CONSTRAINT `fk\_MenuGroups\_Menu`**

**FOREIGN KEY (`menu`)**

**REFERENCES `Menus` (`id`));**

**CREATE TABLE `MenusItems` (**

**`id` INT NOT NULL, -- уникальный идентиф. блюда**

**`group` INT NOT NULL, -- внешний ключ к MenusGroups**

**`name` LONGTEXT NOT NULL, -- название блюда**

**`picture` LONGTEXT NULL, -- имя файла с картинкой блюда**

**`weight` DOUBLE NOT NULL, -- выход (вес) блюда**

**`price` DECIMAL NOT NULL, -- цена за порцию**

**PRIMARY KEY (`id`),**

**INDEX `fk\_MenuItems\_MenuGroups1\_idx` (`group` ASC) VISIBLE,**

**CONSTRAINT `fk\_MenuItems\_MenuGroups1`**

**FOREIGN KEY (`group`)**

**REFERENCES `MenusGroups` (`id`));**

Для учета и ведения заказов предназначены следующие таблицы:

**CREATE TABLE `Orders` (**

**`id` INT NOT NULL, -- уникальный идентиф. заказа**

**`date\_time` DATETIME NOT NULL, -- дата и время заказа**

**`table` LONGTEXT NOT NULL, -- номер столика**

**`waiter` INT NOT NULL, -- сотрудник, обсл. заказ**

**`invoice` DATETIME NULL, -- дата и время выдачи счета**

**`discount\_card` INT NULL, -- карта скидок (внеш.ключ)**

**PRIMARY KEY (`id`),**

**INDEX `fk\_Orders\_Employee1\_idx` (`waiter` ASC) VISIBLE,**

**INDEX `fk\_Orders\_DiscountsCards1\_idx` (`discount\_card` ASC) VISIBLE,**

**CONSTRAINT `fk\_Orders\_Employee1`**

**FOREIGN KEY (`waiter`)**

**REFERENCES `Employees` (`id`),**

**CONSTRAINT `fk\_Orders\_DiscountsCards1`**

**FOREIGN KEY (`discount\_card`)**

**REFERENCES `DiscountsCards` (`id`));**

**CREATE TABLE `OrdersItems` (**

**`id` INT NOT NULL, -- уникальный идентиф. блюда**

**`order` INT NOT NULL, -- внеш.ключ заказа**

**`menu\_item` INT NOT NULL, -- внеш.ключ блюда**

**`quantity` INT NOT NULL, -- количество порций**

**PRIMARY KEY (`id`),**

**INDEX `fk\_OrderItems\_MenuItems1\_idx` (`menu\_item` ASC) VISIBLE,**

**INDEX `fk\_OrderItems\_Orders1\_idx` (`order` ASC) VISIBLE,**

**CONSTRAINT `fk\_OrderItems\_MenuItems1`**

**FOREIGN KEY (`menu\_item`)**

**REFERENCES `MenusItems` (`id`),**

**CONSTRAINT `fk\_OrderItems\_Orders1`**

**FOREIGN KEY (`order`)**

**REFERENCES `Orders` (`id`))**

**CREATE TABLE `Payments` (**

**`id` INT NOT NULL, -- уникальный идентиф. платежа**

**`order` INT NOT NULL, -- внеш.ключ заказа**

**`date\_time` DATETIME NOT NULL, -- время и дата платежа**

**`sum` DECIMAL NOT NULL, -- сумма платежа**

**PRIMARY KEY (`id`),**

**INDEX `fk\_Payments\_Orders1\_idx` (`order` ASC) VISIBLE,**

**CONSTRAINT `fk\_Payments\_Orders1`**

**FOREIGN KEY (`order`)**

**REFERENCES `Orders` (`id`))**

Для учета сотрудников ресторана предназначены следующие таблицы:

**CREATE TABLE `Employees` (**

**`id` INT NOT NULL, -- уникальный идентиф. сотрудника**

**`name` LONGTEXT NOT NULL, -- имя сотрудника**

**`patronymic` LONGTEXT NULL, -- отчество сотрудника**

**`surname` LONGTEXT NOT NULL, -- фамилия сотрудника**

**`employment\_date` DATETIME NOT NULL, -- дата найма**

**`dismiss\_date` DATETIME NULL, -- дата увольнения**

**`position` INT NOT NULL, -- вн.ключ должности**

**PRIMARY KEY (`id`),**

**INDEX `fk\_Employee\_Positions1\_idx` (`position` ASC) VISIBLE,**

**CONSTRAINT `fk\_Employee\_Positions1`**

**FOREIGN KEY (`position`)**

**REFERENCES `Positions` (`id`)**

**)**

**CREATE TABLE `Positions` (**

**`id` INT NOT NULL, -- уникальный идентиф. должности**

**`name` LONGTEXT NOT NULL, -- название должности**

**PRIMARY KEY (`id`))**

**CREATE TABLE `Timesheet` (**

**`id` INT NOT NULL, -- уникальный идентиф. записи**

**`entry\_time` DATETIME NOT NULL, -- время начала работы**

**`exit\_time` DATETIME NULL, -- время окончания**

**`employee` INT NOT NULL, -- вн.ключ сотрудника**

**PRIMARY KEY (`id`),**

**INDEX `fk\_Timesheet\_Employee1\_idx` (`employee` ASC) VISIBLE,**

**CONSTRAINT `fk\_Timesheet\_Employee1`**

**FOREIGN KEY (`employee`)**

**REFERENCES `Employees` (`id`))**

Таблицы учета скидок:

**CREATE TABLE `Discounts` (**

**`id` INT NOT NULL, -- уникальный идентиф. скидки**

**`discount\_group` INT NOT NULL, -- вн.ключ группы скидок**

**`discount` DECIMAL NOT NULL, -- размер скидки**

**`expiration` DATETIME NOT NULL, -- срок действия скидки**

**PRIMARY KEY (`id`),**

**INDEX `fk\_Discounts\_DiscountGroups1\_idx`   
 (`discount\_group` ASC) VISIBLE,**

**CONSTRAINT `fk\_Discounts\_DiscountGroups1`**

**FOREIGN KEY (`discount\_group`)**

**REFERENCES `DiscountsGroups` (`id`);**

**CREATE TABLE `DiscountsGroups` (**

**`id` INT NOT NULL, -- уникальный идентиф. группы скидок**

**`name` LONGTEXT NOT NULL, -- название**

**PRIMARY KEY (`id`));**

**CREATE TABLE `DiscountsCards` (**

**`id` INT NOT NULL, -- уникальный идентиф. карты**

**`card\_no` LONGTEXT NOT NULL, -- номер карты**

**`holder\_name` LONGTEXT NOT NULL, -- имя держателя**

**`discount\_group` INT NOT NULL, -- вн.ключ группы скидок**

**PRIMARY KEY (`id`),**

**INDEX `fk\_DiscountsCards\_DiscountGroups1\_idx`**

**(`discount\_group` ASC) VISIBLE,**

**CONSTRAINT `fk\_DiscountsCards\_DiscountGroups1`**

**FOREIGN KEY (`discount\_group`)**

**REFERENCES `DiscountsGroups` (`id`));**

Для фиксации фактов нарушения финансовой дисциплины и хранения других сообщений предназначена таблица логов:

**CREATE TABLE `restaurant`.`Log` (**

**`date\_time` DATETIME NOT NULL DEFAULT NOW(), --время сообщения**

**`employee` INT NULL, -- вн.ключ сотрудника**

**`order` INT NULL, -- вн.ключ заказа**

**`message` LONGTEXT NOT NULL, -- текст сообщения**

**INDEX `fk\_log\_Employees1\_idx` (`employee` ASC) VISIBLE,**

**INDEX `fk\_Log\_Orders1\_idx` (`order` ASC) VISIBLE,**

**CONSTRAINT `fk\_log\_Employees1`**

**FOREIGN KEY (`employee`)**

**REFERENCES `restaurant`.`Employees` (`id`)**

**ON DELETE NO ACTION**

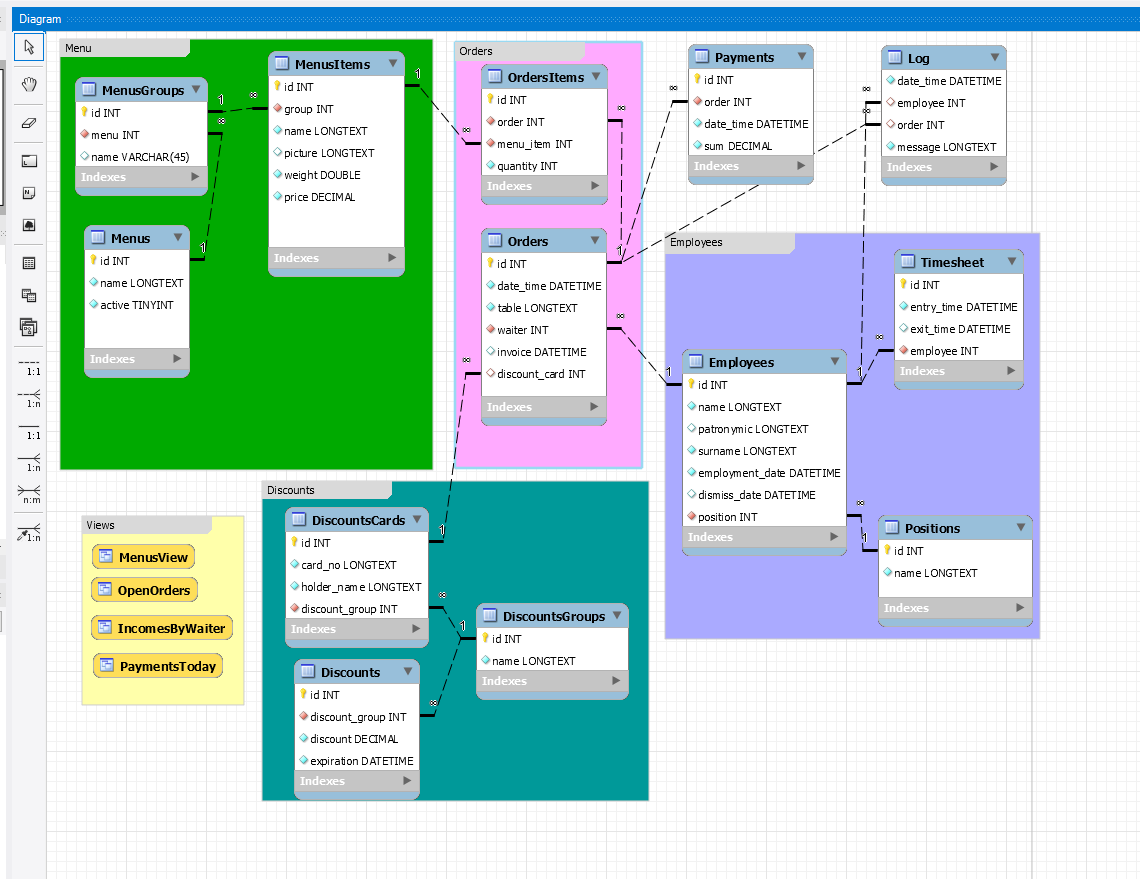
**ON UPDATE NO ACTION,**

**CONSTRAINT `fk\_Log\_Orders1`**

**FOREIGN KEY (`order`)**

**REFERENCES `restaurant`.`Orders` (`id`));**

**Entity Relationship Diagram (ERD) :**



**Скрипты характерных выборок:**

1. Полное меню с группами и блюдами:

**SELECT**

**m.name menu\_name,**

**mg.name group\_name,**

**mi.name dish,**

**weight,**

**price**

**FROM**

**Menus m inner join**

**MenusGroups mg on m.id=mg.menu inner join**

**MenusItems mi on mg.id=mi.`group`;**

1. Запрос, выдающий платежи по заказам за последний день работы. Запрос использует вложенные запросы:

**SELECT \*, SUM(sum) OVER() total**

**FROM Payments p**

**WHERE DATE(p.date\_time) IN (**

**SELECT DATE(MAX(p.date\_time)) as last\_date**

**FROM Payments p**

**);**

1. Запрос, показывающий открытые заказы. Включает использование хранимой процедуры «**fio**»:

**WITH m AS (**

**SELECT m.id menu\_id,**

**m.`name` menu\_name,**

**mg.id group\_id,**

**mg.`name` group\_name,**

**mi.id item\_id,**

**mi.`name` item\_name,**

**mi.weight,**

**mi.price**

**FROM Menus m**

**INNER JOIN MenusGroups mg on m.id = mg.menu**

**INNER JOIN MenusItems mi on mg.id = mi.`group`**

**),**

**o AS (**

**SELECT**

**o.id as order\_id,**

**o.date\_time,**

**o.`table`,**

**o.waiter,**

**o.invoice,**

**o.discount\_card,**

**oi.menu\_item,**

**oi.quantity**

**FROM**

**restaurant.Orders o**

**INNER JOIN OrdersItems oi on oi.order=o.id**

**),**

**e AS (**

**SELECT**

**e.id employee\_id,**

**fio(e.surname, e.name, e.patronymic) fio\_name,**

**e.name,**

**e.patronymic,**

**e.surname,**

**p.name position\_name**

**FROM Employees e**

**INNER JOIN Positions p on e.position = p.id**

**)**

**SELECT**

**o.order\_id,**

**o.date\_time as opened,**

**o.`table`,**

**e.position\_name,**

**e.fio\_name waiter,**

**m.item\_name,**

**m.price,**

**o.quantity,**

**m.price \* o.quantity as sum,**

**SUM(price \* o.quantity) OVER (PARTITION BY order\_id) order\_sum**

**FROM o**

**INNER JOIN m ON o.menu\_item = m.item\_id**

**INNER JOIN e ON o.waiter = e.employee\_id**

**WHERE o.invoice IS NULL;**

1. Поступления от выполненных заказов по официантам за все время работы ресторана (используется группировка)

**WITH m AS (**

**SELECT m.id menu\_id,**

**m.`name` menu\_name,**

**mg.id group\_id,**

**mg.`name` group\_name,**

**mi.id item\_id,**

**mi.`name` item\_name,**

**mi.weight,**

**mi.price**

**FROM Menus m**

**INNER JOIN MenusGroups mg on m.id = mg.menu**

**INNER JOIN MenusItems mi on mg.id = mi.`group`**

**),**

**o AS (**

**SELECT**

**o.id as order\_id,**

**o.date\_time,**

**o.`table`,**

**o.waiter,**

**o.invoice,**

**o.discount\_card,**

**oi.menu\_item,**

**oi.quantity**

**FROM**

**restaurant.Orders o**

**INNER JOIN OrdersItems oi on oi.order=o.id**

**),**

**e AS (**

**SELECT**

**e.id employee\_id,**

**fio(e.surname, e.name, e.patronymic) fio\_name,**

**e.name,**

**e.patronymic,**

**e.surname,**

**p.name position\_name**

**FROM Employees e**

**INNER JOIN Positions p on e.position = p.id**

**)**

**SELECT**

**e.fio\_name waiter,**

**o.order\_id,**

**o.date\_time as opened,**

**o.`table`,**

**p.sum as paiment\_sum,**

**SUM(m.price \* o.quantity) as order\_sum,**

**SUM(p.sum) OVER (PARTITION BY e.fio\_name) as total\_sum**

**FROM o**

**INNER JOIN m ON o.menu\_item = m.item\_id**

**INNER JOIN e ON o.waiter = e.employee\_id**

**INNER JOIN Payments p ON p.`order` = o.order\_id**

**WHERE o.invoice IS NOT NULL**

**GROUP BY**

**e.fio\_name,**

**o.order\_id,**

**o.date\_time,**

**o.`table`,**

**p.sum**

**order by e.fio\_name;**

**Представления**

Все вышеприведенные выборки выборки оформлены в БД в виде представлений. Таким образом БД содежит 4 представления.

**Хранимые процедуры**

В БД создана хранимая функция для формирования строки, содержащей фамилию и инициалы по полным фамилии, имени и отчеству:

**DELIMITER $$**

**CREATE FUNCTION fio ( sn LONGTEXT, nm LONGTEXT, pn LONGTEXT)**

**RETURNS LONGTEXT**

**DETERMINISTIC NO SQL**

**BEGIN**

**DECLARE res LONGTEXT DEFAULT '';**

**IF sn IS NOT NULL THEN**

**SET res = sn;**

**END IF;**

**IF nm IS NOT NULL THEN**

**IF NOT LENGTH(res) = 0 THEN**

**set res = CONCAT(res, ' ');**

**END IF;**

**set res = CONCAT(res, LEFT(nm,1), '.');**

**IF pn IS NOT NULL THEN**

**set res = CONCAT(res, LEFT(pn,1), '.');**

**END IF;**

**END IF;**

**RETURN res;**

**END$$**

**DELIMITER ;**

**Триггеры**

Представленный в БД триггер запрещает добавление скидки к заказу в случае, если счет по заказу уже был выдан. В результате работы триггера попытка добавления карты блокируется и сообщение об этом заносится в таблицу Log.

**DELIMITER $$**

**CREATE**

**TRIGGER `discount\_after\_invoice`**

**BEFORE UPDATE**

**ON restaurant.Orders**

**FOR EACH ROW**

**BEGIN**

**IF old.invoice IS NOT NULL AND new.discount\_card IS NOT NULL THEN**

**INSERT INTO `Log` (employee, `order`, message)**

**VALUES (**

**old.waiter,**

**old.id,**

**'try to add discount after invoice issued ');**

**SET new.discount\_card = NULL;**

**END IF;**

**END;**

**$$**

**ПРИЛОЖЕНИЯ**

К этой работе прилагается папка «Приложения». В ней находятся:

1. Скрипт создания БД «00. Создание БД.sql»
2. Скрипт «01. Скрипт наполнения БД данными.sql»
3. Скрипт создания триггера «02. Триггер.sql»

Последовательный запуск этих трех скриптов даст БД, с данными, в которой работают все основные выборки, хранимые процедуры, тригеры и представления

Кроме того, в папке находится файл «Ресторан.mwb» - проект базы данных для MySQL Workbench. С его помощью также можно создать необходимую структуру БД. Кроме того, этот файл включает в себя ERD.

1. Представленный в БД триггер запрещает добавление скидки к заказу в случае, если счет по заказу уже был выдан. В результате работы триггера попытка добавления карты блокируется и сообщение об этом заносится в таблицу Log.