**Введение**

“Разработка базы данных подсистемы расчета с клиентами в банке”

**Актуальность темы исследования** – данная тема очень актуальна, потому что с очень много людей пользуется услугами банков, которым нужно хранить очень много информации и получать данные по клиентам в кратчайшие сроки для того, чтобы оказывать свои услуги как можно быстрее и удобнее для клиентов. С течением времени всё больше людей предпочитают использовать интернет для управления своими банковскими счетами, нежели через офисы банков. Пандемия COVID-19 еще сильнее подтолкнула людей к прекращению посещения банков лично.

**Цель и задачи курсовой работы:**

Цель: разработка базы данных подсистемы расчета с клиентами в банке.

Задачи:

1. Разработать базу данных подсистемы расчета с клиентами в банке

2. Разработать приложение для управления базой данных.

3. Связать приложение с базой данных.

**Предмет исследования –** разработка базы данных для подсистемы расчета с клиентами в банке и реализация приложения для управления ей.

**Объект исследования –** база данных.

**Методы исследования:** разработка, тестирование.

**Структура работы:** Структура курсовой работы состоит из введения, двух разделов, приложений, списка использованных источников.

**Основная часть**

**Раздел 1. Анализ бизнес-процессов предметной области**

**1.1 Описание предметной области**

**1.1.1 Общее описание предметной области, функционал БД**

Областью применения БД является банковская сфера. Банковская сфера делится на розничный банкинг (обслуживание физических лиц), корпоративный банкинг (обслуживание организаций) и инвестиционный банкинг (инвестирование средств). В случае данного проекта больше всего подходит розничный банкинг. Розничный банкинг – это обслуживание физических лиц, предоставление им возможности для вкладов, кредитов, инвестирования, залогов, обмена валюты и обслуживания их банковских карт, дебетовых или кредитных.

В этом курсовом проекте была разработана база данных для банка “CyberBank”, в которой хранятся данные об аккаунтах пользователей, их дебетовых картах, заявки по кредитам и данные о кредитных картах. Для упрощения работы с базой данных, информация выводится в приложение, которое связано с базой данных. Данные, которые не являются строго конфиденциальными, администратор может видеть в приложении, если он зайдет под своим аккаунтом. Администратор также может просматривать заявки на кредитные карты, которые оставляют пользователи, через приложение, и подтверждать, либо отклонять их. Строго конфиденциальная информация зашифрована и недоступна сотрудникам банка, для защиты от утечек пользовательских аккаунтов, в случае если взломают аккаунт администратора.

Для пользователей реализована возможность перевода денег на счет других пользователей, а также возможность заказать кредитную карту, а затем закрыть её, если выполнены условия банка. При одобрении получения кредитной карты пользователю приходит письмо на почту, которую он указывает при оформлении заявки. Также имеется возможность регистрации через приложение, вследствие чего пользователь автоматически будет добавлен в базу данных, а затем сможет авторизоваться в приложении.

**1.1.2 Описание входных документов**

В результате разработки базы данных в ней используются следующие входные документы:  
 -Таблица “users”;

-Таблица “e\_carts”;

-Таблица “credit\_requests”;

-Таблица “credit\_cards”;

**1.1.3 Описание выходных документов и сообщений**

Для работы с базой данных в приложении были разработаны формы, которые выводят информацию на экран и упрощает работу с базой данных для администраторов.

База данных и приложение предоставляют следующие возможности:

-Закрытый доступа: только для администраторов банка

-Просмотр информации в формах.

-Добавление новой информации в базу данных.

**1.1.4 Описание запросов к базе данных**

При работе с базой данных через приложение используется много различных запросов по типу SELECT, INSERT, UPDATE, DELETE.

1. Простой запрос на выборку по ID.

Код запроса на языке SQL: SELECT {what} FROM {from} WHERE {where} = @id;

Данный запрос делает выборку нужной информации из нужного места основываясь на id пользователя. Данные {what}, {from}, {where} и @id указываются для конкретного случая, в зависимости от того какая информация нужна для выборки.

2. Простой запрос на выборку по логину и паролю.

Код запроса на языке SQL: SELECT {what} FROM {from} where u\_username = @login and u\_password = @pass;

Данный запрос делает выборку нужной информации из нужного места основываясь на логине и пароле пользователя. Данные {what}, {from}, @login и @pass указываются для конкретного случая, в зависимости от того какая информация нужна для выборки.

3. Запрос на добавление нового пользователя в базу данных.

Код запроса на языке SQL: INSERT INTO `users` (`u\_username`, `u\_password`, `u\_name`, `u\_surname`, `u\_date\_of\_birth`) VALUES (@username, @pass, @name, @surname, @dob);

Данный запрос добавляет нового пользователя в базу данных после успешной регистрации. Параметры @username, @pass, @name, @surname, @dob заполняются пользователем при регистрации в приложении. Каждому пользователю присваивается уникальный id.

4. Запрос на создание дебетовой карты.

Код запроса на языке SQL: INSERT INTO `e\_carts` (`ec\_cartnumber`, `ec\_cache`, `ec\_cvv`, `ec\_cartholder\_id`) VALUES (@cardnumber, @cache, @cvv, @id) ;

Данный запрос добавляет новую дебетовую карту в базу данных. Параметры @cardnumber, @cache, @cvv определяют данные дебетовой карты. Параметр @id является id пользователя и связывает карту с данными пользователя в таблице users.

5. Запрос на изменение данных о владении пользователем дебетовой карты нашего банка.

Код запроса на языке SQL: UPDATE `users` SET `u\_havecart` = 1 where `u\_id` = @id;

Данный запрос изменяет параметр владения дебетовой карты пользователем в базе данных, что в последствии помогает приложению определять это автоматически. Параметр @id меняется в зависимости от id пользователя.

6. Запрос на добавление заявки о кредите.

Код запроса на языке SQL: INSERT INTO `credit\_requests` (`cr\_description`, `cr\_value`, `cr\_email`, `cr\_id\_user`) VALUES (@desc, @value, @email, @id);

Данный запрос добавляет новую заявку на кредитную карту в базу данных. Параметры @desc, @value, @email заполняются пользователем через приложение. Параметр @id является id пользователя, подающего заявку, он же связывает заявку с пользователем в таблице users.

7. Запрос на удаление заявки на кредитную карту из базы данных.

Код запроса на языке SQL: DELETE FROM `credit\_requests` where cr\_id\_user = @id;

Когда администратор одобряет или отклоняет заявку на кредитную карту она автоматически удаляется из базы данных. Параметром @id является id пользователя, подавшего заявку.

8. Запрос на изменение данных о владении пользователем кредитной карты нашего банка.

Код запроса на языке SQL: UPDATE `users` SET `u\_havecredit` = 1 where `u\_id` = @id;

Данный запрос изменяет параметр владения кредитной карты пользователем в базе данных, что в последствии помогает приложению определять это автоматически. Параметр @id меняется в зависимости от id пользователя.

9. Запрос на удаление кредитной карты из базы данных.

Код запроса на языке SQL: DELETE FROM `credit\_cards` where cr\_c\_id\_user= @id;

Когда пользователь решает закрыть кредитную карту банка, выполняется этот запрос, и его карта удаляется из базы данных. Параметр @id меняется в зависимости от id пользователя.

10. Запрос на перевод денег на счет другого пользователя в банке.

UPDATE e\_carts SET ec\_cache=ec\_cache+@value where ec\_cartnumber = @cardnumber

Запрос добавляет к полю денег определенного пользователя параметр @value который является количеством денег, которое хочет перевести пользователь на другой счет. Параметр @cardnumber является номером карты, на который переводятся деньги.

UPDATE e\_carts SET ec\_cache=ec\_cache-@value where ec\_cartnumber = @cardnumber;

Запрос убавляет из поля денег определенного пользователя параметр @value который является количеством денег, которое хочет перевести пользователь на другой счет. Параметр @cardnumber является номером карты, с которого переводятся деньги.

**1.1.4 Список ограничений**

Ограничения представляют собой набор некоторых условий, налагаемых на элементы базы данных (таблицы, столбцы и т.д.) или всю базу данных, гарантирующие, что информация будет подчиняться определенным правилам целостности данных.

В данном курсовом проекте было использовано ограничение ссылочной целостности, т. к. значения одних столбцов таблиц связаны со значениями других столбцов в другой таблице. В каждой из таблиц проектируемой базы данных использовались первичный и внешний ключи, содержащие уникальные значения столбцов. Благодаря обеспечению ссылочной целостности данных была исключена возможность дублирования записей в базе данных, обеспечено каскадное обновление, вставка и удаление записей БД.

**1.2 Выбор и описание используемой среды разработки**