**Техническое задание:**

**Разработка финансовых отчетов и создание процедур для финансового сервиса управления платежами**

Вы претендуете на позицию MySQL разработчика, и в рамках этого технического задания мы предлагаем вам продемонстрировать свои навыки работы с реляционными базами данных. Задание включает разработку финансовых отчетов, создание хранимых процедур для обработки транзакций и обеспечение целостности данных.

Основная цель задания — оценить вашу способность эффективно проектировать запросы и оптимизировать их.

**Стукрутра таблиц:**

***# Операции пользователей***

Содержит в себе список денежных операций пользователей финансовго сервиса

*CREATE TABLE payment.operations (*

*id\_operation BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT,*

*dt TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*id\_user INT UNSIGNED NOT NULL,*

*id\_type\_oper SMALLINT UNSIGNED NOT NULL,*

*move TINTINT NOT NULL COMMENT 'направление движения (-1: списание со счёта, 1: начисление на счёт)',*

*amount\_oper DECIMAL(19, 5) NOT NULL COMMENT 'Сумма операции в валюте пользователя',*

*PRIMARY KEY (id\_operation)*

*) ENGINE = INNODB;*

***# Справочник типов операций***

Содержит в себе список всевозможных типов операций пользователей. Поле *id\_type\_oper в таблице payment.operations*

Наример:

* Депозит средств на счет (Deposit)
* Снятие средств со счета (Withdrawal)
* Возврат средств (Refund)
* Штрафное списание (Penalty Charge)

*CREATE TABLE payment.type\_opers (*

*id\_type\_oper SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,*

*name\_oper VARCHAR(255) DEFAULT NULL COMMENT 'Название операции',*

*comission DECIMAL(5, 2) DEFAULT 0.00 COMMENT 'Процент комиссии за операцию',*

*PRIMARY KEY (id\_type\_oper)*

*) ENGINE = INNODB;*

***# Логи действий пользователя***

Содержит в себе список всевозможных действий пользователей финансового сервиса.

Наример:

* Пополнение счета (*idAction* 1);
* Вывод со счета (*idAction* 2);
* Авторизация в ситеме (*idAction* 3);
* Просмотр истории операций (*idAction* 4);

(поле в таблице *idAction*)

*CREATE TABLE payment.log\_users (*

*dt TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*idUser INT UNSIGNED NOT NULL,*

*idAction INT UNSIGNED NOT NULL,*

*Params JSON DEFAULT NULL*

*) ENGINE = INNODB;*

***# Таблица с пользователями финансового сервиса***

Содержит в себе список пользователей, с их свойствами, пользователи выполняют денежные операции на сервисе *финансов*

*CREATE TABLE payment.users (*

*id\_user INT UNSIGNED NOT NULL AUTO\_INCREMENT,*

*user\_balance DECIMAL(14, 2) DEFAULT 0.00 COMMENT ‘Текущий баланс в валюте игрока’',*

*id\_currency SMALLINT UNSIGNED NOT NULL,*

*id\_country SMALLINT UNSIGNED NOT NULL,*

*PRIMARY KEY (id\_user)*

*) ENGINE = INNODB;*

***# Справочник валют***

Содержит в себе список всех возможных валют с курсами к рублю. Используется как свойство пользователя.

*CREATE TABLE payment.currencies (*

*id\_currency SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,*

*name\_currency VARCHAR(255) DEFAULT NULL COMMENT 'ISO код валюты (EUR / USD / BTC / TON ...)',*

*base\_rate DECIMAL(15, 5) DEFAULT NULL COMMENT 'Курс к рублю',*

*PRIMARY KEY (id\_currency)*

*) ENGINE = INNODB;*

*Примечание для “base\_rate”: чтобы получить эквивалент валюты в рублях, достаточно разделить базовую валюту на “base\_rate”. Пример: валюта - EUR, base\_rate - 0,010119000. 1 EUR / 0,010119000 = 98,8240 RUB*

***# Справочник стран***

Содержит в себе список всех возможных стран. Используется как свойство пользователя

*CREATE TABLE payment.countries (*

*id\_country SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,*

*name\_country VARCHAR(50) DEFAULT NULL,*

*PRIMARY KEY (id\_country)*

*) ENGINE = INNODB;*

***Аннотация к таблицам и хранящимся в них данным:***

1. Архитектура БД финансового сервиса, над которым вам предстоит работать, обрабатывает большое количество транзакций в режиме реального времени.
2. Таблица *payment.operations* содержит от 5 до 10 миллионов строк в день и хранит в себе данные за несколько лет.
3. Таблица *payment.users* содержит записи о 7 миллионах пользователей финансового сервиса.
4. Таблица *payment.log\_users* содержит от 5 до 10 миллионов строк в день и хранит в себе данные за 10 лет.
5. Таблицы *payment.type\_opers, payment.currencies, payment.currencies* - справочники с количеством записей до 500.

***В этой части технического задания нужно будет добавить или не добавлять индексы к таблицам, при условии, что:***

1. Необходимость индексов определяется исполнителем, исходя из условий данного технического задания;
2. Наличие или отсутствие индекса должно быть аргументирован исполнителем.
3. Все индексы, если они необходимы, нужно добавить в текстовый файл с решением.

В рамках данного технического задания предлагается решит 6 заданий. В качестве решения ожидается только листинг кода решения каждого из заданий. Все задания собираются в один текстовый файл и отправляются на проверку в чат группы для созвона. Решение нужно выполнить на MySQL 8.

**Задание 1**

Создайте хранимую процедуру, которая будет выполнять операцию списания или зачисления на счет пользователя.

Процедура должна:

1. Использовать входящие параметры, такие как: ID Пользователя, Сумма, Тип операции (из справочника *payment.type\_opers*).
2. В таблице операций есть поле *move 'направление движения (-1: списание со счёта, 1: начисление на счёт)',* учесть это.
3. Проверить наличие достаточного баланса на счете пользователя при необходимости.
4. Списать или зачислить указанную сумму на счет пользователя.
5. Сделать запись в таблице операций.
6. Обеспечить логирование в таблице с логами. Поле *Params заполняется в произвольной форме по усмотрению автора.*
7. Процедура должна вернуть ответ о своей работе, вернуть баланс до и после операции.

**Задание 2**

Финансовый отчет за период дат. Процедура с выборкой из БД по следующим условиям:

1. На вход процедура принимает 2 параметра – 2 даты со временем типа datetime.
2. Нужно будет сгруппировать данные по стране и типу операции и вычислить общую сумму, сумму комиссии, итоговую сумму (с вычетом комиссии) в рублях.
3. Отчёт должен содержать промежуточные итоги по каждой стране, а так же строку с общим итогом.

*Пример выполнения:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| country\_name | oper\_name | amount | amount\_comiss | amount\_no\_comiss |
| Cyprus | Deposit | 10 | 2 | 8 |
| Cyprus | Witdrawal | 708 | 6 | 702 |
| *Cyprus* | ***TOTAL:*** | 718 | 8 | 710 |
| Russia | Deposit | 200000 | 2000 | 198000 |
| *Russia* | ***TOTAL:*** | 200000 | 2000 | 198000 |
| *TOTAL:* | ***TOTAL:*** | 200718 | 2008 | 198710 |

**Задание 3**

Процедура для консолидирования данных.

Так как таблица с операциями огромная (смотри аннотацию к таблицам), необходимо сделать евент/джобу/событие в виде хранимой процедуры, которая будет выполнять консолидацию/агрегацию данных и класть эти данные в заранее созданную таблицу. Такая таблица пригодится для финансовых отчетов при выборках за большой период. Такие выборки будут эффективнее, если использовать в запросах часть данных, которая уже консолидирована, что убирает необходимость выбирать много сырых данных из огромной таблицы, и тем самым ускоряет выполнение запроса.

Требования к заданию:

1. Создать таблицу для консолидированных данных. Структура таблицы разрабатывается исполнителем, исходя из условий задачи, добавление колонок и индексов, если они нужны, должны быть аргументированы.
2. Решением задания будет являться код запроса, добавляющий данные в таблицу консолидации.
3. Данные нужно консолидировать по дням.
4. Ожидается, что евент будет запускаться раз в сутки в 3 часа ночи, выбирать данные за предыдущий день, класть их в таблицу консолидации. Время выполнения евента варьируется от 1 до 2 часов в зависимости от количества строк в сутки.
5. Срез данных выполняется с группировкой по стране и типу операции.

**Задание 4**

Финансовый отчет за период дат. Задание 4 объединяет в себе задание 2 и задание 3. Нам нужно показать финансовые данные за период дат, как и в задании 2, но используя при этом таблицу консолидации из задания 3 для ускорения выборки данных за период. Процедура с выборкой из БД по следующим условиям:

1. На вход процедура принимает 2 параметра – 2 даты со временем типа datetime.
2. Нужно сгруппировать данные по стране и типу операции и вычислить общую сумму, сумму комиссии, итоговую сумму (с вычетом комиссии) в рублях.
3. Отчёт должен содержать промежуточные итоги по каждой стране, а так же строку с общим итогом.
4. Здесь мы используем таблицу консолидации из задания 3, чтобы ускорить отчет использованием заранее агрегированных данных.
5. Вариации дат, подающихся на вход в процедуру могут быть любыми, от нескольких лет до нескольких минут.
6. Результатом выполнения задания будет листинг кода, возвращающий данные в формате, указанном в примере второго задания (смотри картинку с таблицей).

Пример: для запроса с 2024-05-01 12:00:00 по 2024-05-05 17:00:00 нужно взять 2, 3, 4 числа из таблицы консолидации (т.к. это полные дни), а неполные дни (2024-05-01 12:00:00 - 2024-05-01 23:59:59 и 2024-05-05 00:00:00 - 2024-05-05 17:00:00) взять из сырых данных, т.е. из таблицы operations.

**Задание 5**

Финансовый отчет по операциям конкретного пользователя за период дат. Процедура с выборкой из БД по следующим условиям:

1. На вход процедура принимает 3 параметра – 2 даты со временем типа datetime и ID Пользователя типа int.
2. Нужно сгруппировать данные по типу операции, вывести общую сумму, сумму комиссии, итоговую сумму (с вычетом комиссии) в рублях.
3. Отчёт должен содержать общую сумму по всем операциям (TOTAL).

**Задание 6**

Партицирование таблицы логов. Нужно разработать процедуру, которая будет осуществлять следующие действия:

1. Необходимо разбить таблицу *payment.log\_users* на партиции по одному дню и написать эвент, осуществляющий ротацию партиций (добавление новых партиций и удаление старых партиций).
2. Ожидается, что евент будет запускаться раз в сутки в 2 часа ночи.
3. Евент должен выполнять 2 функции:

- удаление устаревших партиций. «Старыми» считаются партиции, дата которых меньше 7 дней назад от текущего дня.

- добавление новых партиций. Евент должен обеспечивать добавление партиций с запасом в 1-2 дня. Запас нужен, чтобы предусмотреть внештатную ситуацию, при которой евен не запускался 1-2 дня по какой-либо причине.

Подсказка: один из возможных способов реализации - списки партиций можно получить запросом из таблицы information\_schema.PARTITIONS, а собирать запросы удаления и создания через prepared statement.