Лабораторная работа №5.
Транзакции. Индексы. Дополнительные функции SQL.

Базы данных.
2 курс. 5 группа.
Кушнеров А.В. 2021-2022 г.
Уровень сложности: Сложный
Формат работы: Индивидуальная по вариантам.
Срок выполнения: ***2 недели.***

Цель работы

Изучить механизмы транзакций в MySQL и области их применения. Получить навыки использования индексации на различных объёмах и типах данных. Овладеть базовыми знаниями и умениями в плане шифрования данных и организации многопользовательского доступа к БД.

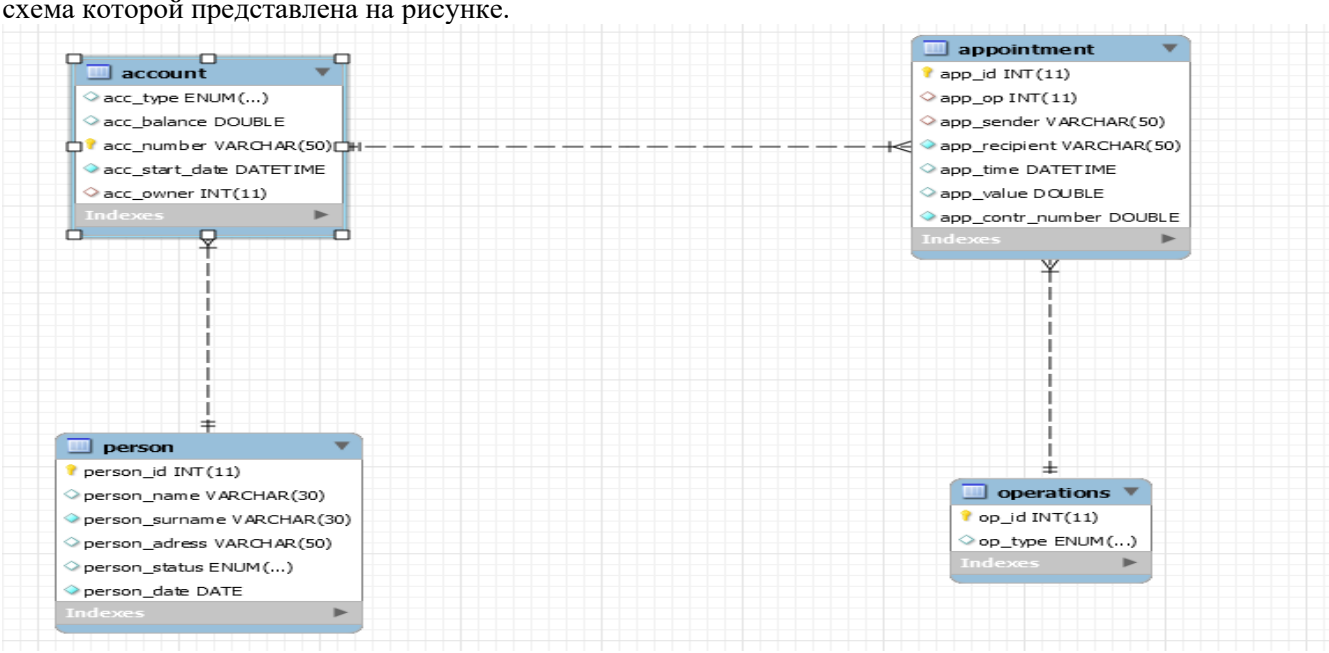
Минимальные теоретические сведения

Транзакции

Транзакция – последовательность SQL-команд, которая выполняется как «одно целое» без прерывания другими пользователями и процессами. Основное назначение транзакций – выполнение одной функции пользовательской или серверной логики, которая состоит из нескольких команд DML. В случае, если один из этапов (команд) не будет выполнен, происходит отмена всей последовательности и сервер БД вернётся к исходному состоянию.
Ключевые операторы:

- START TRANSACTION – оператор начала транзакции. После его вызова все изменения в базу будут доставлены не автоматически, а только при вызове оператора COMMIT.
- COMMIT – оператор, доставляющий изменения на сервер БД.
- ROLLBACK – оператор для отката изменений до последнего COMMIT, START TRANSACTION или SAVEPOINT.
- SAVEPOINT – оператор, позволяющий создать именованную точку сохранения, к которой потом можно вернуться при помощи ROLLBACK.

Транзактные возможности, как правило, необходимы любому приложению, но типичные сферы применения: банки, торговля, архивы строгой отчётности. Рассмотрим пример транзакции в БД bankDB, схема которой представлена на рисунке.



Данная БД хранит данные пользователей (person), данные об состоянии их карт-счётов (account) и историю операций (appointment).
Пример 1. Рассмотрим пример элементарной транзакции. Переведём 100 рублей с одного счёта на другой. По сути эта операция – это два оператора UPDATE.

```
START TRANSACTION;

UPDATE account SET acc_balance=acc_balance-100
WHERE acc_number='9111 0001 4578 1134';

UPDATE account SET acc_balance=acc_balance+100
WHERE acc_number='9332 0002 4689 3212';

COMMIT;
```

Только по завершению, всех операций, мы применяем изменения с помощью оператора COMMIT.
Пример 2. Транзакция из примера 1 не совсем корректна. Требуется провести ещё некоторые проверки для корректной работы. Рассмотрим наш процесс более детально и оформим его в виде хранимой процедуры.

```
DELIMITER ;;
CREATE PROCEDURE `transfer1`(IN sender varchar(50),IN rec varchar(50),IN sum double)
begin

start transaction;

update account
set acc_balance=acc_balance-sum
where acc_number=sender;
if row_count()>0
    then
    update account
    set acc_balance=acc_balance+sum
    where acc_number=rec;
    if row_count()>0
        then
        insert into appointment
        (app_op,app_sender,app_recipient,app_time,app_value,app_contr_number)
        values
        (3,sender,rec,now(),sum,rand(10));
        commit;
    else rollback;
end if;

else rollback;

end if;

end ;;
DELIMITER ;

CALL transfer1('9111 0001 4578 1134','9332 0002 4689 3212',300);
```

Ключевое отличие заключается в применении оператора row_count (), который возвращает количество строк, изменённых оператором UPDATE. С его помощью мы продолжаем дальнейшие действия, только в случае если предыдущие были успешно завершены на некоторой строке. Также мы добавили возможность добавления строки в таблицу с историей операций.
Упражнение. Какая серьёзная уязвимость присутствует в данной транзакции?
Заметим, что транзактный подход вполне может быть применён и в триггерах.

Индексы

Индекс – фоновое упорядочивание информации с целью ускорения поиска и выборки. Индексация позволяет повысить скорость поиска в таблицах БД. Индексируемое поле сервер сортирует в фоновом режиме, представляя его в виде дерева, например. Это простое решение позволяет добиться значительного повышения производительности при запросах.

НО! Создание индекса значительно нагружает память, а также замедляет операции INSERT и UPDATE. **Золотое правило: «7 раз подумай – один раз проиндексируй!»**. Индексировать стоит лишь те поля, по которым вероятнее всего будет идти поиск.

Рассмотрим индексацию на простом примере. Продумаем вопрос поиска по фамилии студента в БД mmf2018 из ЛР4.

Пример 3. Выполним элементарный запрос поиска по фамилии.

```
EXPLAIN SELECT * FROM mmf2018.studs
WHERE st_surname='Luka%';

CREATE INDEX myInd ON studs(st_surname(4));

EXPLAIN SELECT * FROM mmf2018.studs
WHERE st_surname='Luka%';
```

С помощью функции EXPLAIN мы можем отследить логику выполнения запроса.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	studs	NULL	ALL	NULL	NULL	NULL	NULL	4	25.00	Using where

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	studs	NULL	ref	myInd	myInd	18	const	1	100.00	Using where

Как видим, после создания индекса количество обработанных записей (столбец rows) сократилось, ведь сервер уже знал где искать.

Полнотекстовый индекс

Полнотекстовый индекс – это специальный тип индекса, предназначенный для поиска без шаблонов по большим объёмам текстовой информации.

Пример 4. Используем БД litdb, которая содержит текст с цитатами различных авторов. База содержит одну таблицу следующего вида.

sk_id	sk_aut	sk_text
1	Липницкий	Ниже рассматриваются норменные методы коррекции многократных ошибок БЧХ-кодами. Они реали...
2	Липницкий	Анализ рассматриваемого метода декодирования и его реализаций показывает, что операции при в...
3	Конопелько	При построении декодера на основе норм синдромов имеется возмож-ность использования ПЛИС для...
4	Конопелько	Я всякий раз чувствую жестокое угрызение совести, — сказал мне однажды Пушкин в откровенном ...
5	Белинский	Пушкин был совершенным выражением своего времени. Одаренный высоким поэтическим чувством и...
6	Гоголь	Пушкин пользуется своею художественною виртуозностью, как средством посвятить всю читающу...
7	Достоевский	Пушкин как раз приходит в самом начале правильного самосознания нашего, едва лишь начавшегося...
8	Пушкин	Можем смело сказать, что мы ни единой минуты не усумнились в исполнении планов г. Каченовского...
9	Пушкин	Странные требования! В летах «Вестника Европы» уже не учатся и не бросают предрассудков зако...
10	Пушкин	Успокоясь насчет ужасного смысла вышепомянутого примечания, мы сожалели о бесполезном действ...
11	Оксимирон	Я сам себе режиссер, я тебя создал и снял. Я Брюс Уэйн и Кристофер Нолан в одном, роли готовы да...
12	Джигарханян	В театре - как в природе. Извините за натурализм, но вот бегут две собаки. Понюхали друг друга - ...
13	Джигарханян	Взаимоотношения режиссера и актера - это страшнее, чем у женщины и мужчины.
14	СлаваКПСС	Че, так болел за Россию, что на нервах потерял ганглии? Но пока тут проходили митинги, где ты си...
15	СлаваКПСС	Я везу вам революцию, как встарь, по дороге из Тушино. У этой телеги вращаются оси. Футуристы ...
16	Пушкин	На самом деле. Лизавета Ивановна была пренесчастное создание. Горек чужой хлеб, говорит Данте...

Определим и применим полнотекстовую индексацию по полю sk_text.

```
CREATE FULLTEXT INDEX myInd1 ON litsketch (sk_text);

SELECT * FROM litsketch
WHERE MATCH(sk_text) AGAINST ('Пушкин');
```

Конструкция MATCH(), AGAINST() осуществляет полнотекстовый поиск по индексированному полю. Приведённый запрос возвращает все цитаты, в которых содержится слово Пушкин.

```
SELECT sk_text,
MATCH(sk_text) AGAINST ('Пушкин')
FROM litsketch
WHERE MATCH(sk_text) AGAINST ('Пушкин');
```

Для более удобной работы сервер при таком поиске вычисляет специальный коэффициент релевантности и возвращает нам слова, для которых этот коэффициент превосходит 0. Детали вычисления коэффициента не так важны, но он показывает «важность слова» в контексте большой фразы. Тогда результат запроса будет следующим.

sk_text	MATCH(sk_text) AGAINST ('Пушкин')
Пушкин был совершенным выражением своего времени. Одаренный высоким поэтическим чувством и...	1.1595041751861572
Пушкин пользуется своею художественною виртуозностью, как средством посвятить всю читающу...	0.9332535266876221
Пушкин как раз приходит в самом начале правильного самосознания нашего, едва лишь начавшегося...	0.836798369884491
Я всякий раз чувствую жестокое угрызение совести, — сказал мне однажды Пушкин в откровенном ...	0.6948787569999695

Пример 6. Для более гибкой работы с критериями поиска используют опцию полнотекстового поиска IN BOOLEAN MODE.

```
SELECT * FROM litsketch
WHERE MATCH(sk_text) AGAINST ('Пушкин +всякий' IN BOOLEAN MODE);

SELECT * FROM litsketch
WHERE MATCH(sk_text) AGAINST ('Пушкин -своего' IN BOOLEAN MODE);
```

Первый запрос вернёт нам записи содержащие слово “Пушкин” и “всякий” одновременно, второй, в свою очередь, вернёт лишь те записи, в которых есть слово “Пушкин”, но нет слова “своего”.

Управление учётными записями пользователей

Как мы помним из формального определения, любая реляционная БД – это в принципе многопользовательский объект. Язык SQL предоставляет набор методов для управления учётными записями пользователей. Все данные об учётных записях хранятся в таблице mysql.users.
По умолчанию такая таблица имеет следующий вид.

Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shu
localhost	mysql.infoschema	Y	N	N	N	N	N	N	N
localhost	mysql.session	N	N	N	N	N	N	N	N
localhost	mysql.sys	N	N	N	N	N	N	N	N
localhost	root	Y	Y	Y	Y	Y	Y	Y	Y
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Основной и базовой учётной записью является root@localhost – это родительская запись, которая создаётся при установке сервера на машину. Эта запись обладает всеми привилегиями. К привилегиям мы относим возможность создавать, обновлять таблицы, осуществлять запросы на выборку, создавать новых пользователей, удалять записи и таблицы, вызывать хранимые процедуры и т.д.
Чёткое разграничение пользователей по уровню доступа к БД позволяет грамотно оганизовать бизнес-логику приложения. Рассмотрим простой пример.
Пример 7. Создадим пользователя с базовыми привилегиями для БД mmf2018.

```
CREATE USER base_user IDENTIFIED BY  '12345';

GRANT SELECT ON mmf2018.* TO base_user;
```

Для создания пользователя используют оператор CREATE USER. Для наделения его определённым набором привилегий нужен оператор GRANT.
Важно! Пользователь может дать другому пользователю лишь те привелегии, которыми сам обладает!

Шифрование данных

В любой БД могут хранится данные, которые было бы странно хранить в явном виде. Как правило, это пароли, логины, личные данные. Соременные требования информационной эпохи заставляют применять шифрование. Рассмотрим небольшой пример.
Пример 8. Зашифруем фамилию одного из студентов по методе шифрования AES. Для этого существуют встроенные методы AES_ENCRYPT() и AES_DECRYPT().

```
SELECT  aes_encrypt(st_surname,'pas') INTO @x FROM studs
WHERE st_id='1622161';

SELECT cast(@x AS CHAR),@x;
```

Вторым параметром функции является ключ шифрования. В силу особенностей блочного алгоритма AES-128 функции AES_ENCRYPT() и AES_DECRYPT() возвращает бинарные строки. Для получения информации в текстовом виде необходимо привести её в привычный строковый вид с помощью функции CAST.

cast(@x AS CHAR)	@x
◆◆Lÿ◆◆ ◆"◆□◆◆Y◆	BLOB

```
SELECT cast(aes_decrypt(@x,'pas') AS CHAR);
```

Если же вам необходимо зашифровать целый столбец, то это можно сделать с помощью курсора или оператора UPDATE. К примеру, зашифруем столбец с именем преподавателя в таблице предмет.

```
ALTER TABLE subjects MODIFY sub_teacher varbinary(50);

UPDATE subjects SET sub_teacher=aes_encrypt(sub_teacher,'mmf2018');
```

Доступ к данным осуществляем соответственно следующим образом.

```
SELECT cast(aes_decrypt(sub_teacher,'mmf2018') AS CHAR) FROM subjects;
```

Оконные функции.

Оконные функции позволяют проводить аггрегацию без потери данных.

```
select  CountryCode, sum(Population) from city
group by CountryCode;
```

Обычный запрос на группировку позволяет получить данные по аггрегированным группам. Важно понимать, что в этом случае мы теряем информацию о кокретных представителях группы.

CountryCode	sum(Population)
ABW	29034
AFG	2332100
AGO	2561600
AIA	1556
ALB	270000
AND	21189
ANT	2345
ARE	1728336
ARG	19996563

В результате видно, что мы получили суммарное население в городах страны, но потеряли информацию о количестве и названиях и в целом о самих городах.

Иногда это может быть проблемой, поэтому в таких случая пользуюся оконными функциями, которые проволят аггрегацию без удаления данных в группе.

```
select  CountryCode, name, Sum(population) OVER (partition by CountryCode)  from city;
```

Результат запароса.

CountryCode	name	Sum(population) OVER (partition by CountryCode)
ABW	Oranjestad	29034
AFG	Kabul	2332100
AFG	Qandahar	2332100
AFG	Herat	2332100
AFG	Mazar-e-Sharif	2332100
AGO	Luanda	2561600
AGO	Huambo	2561600
AGO	Lobito	2561600
AGO	Benguela	2561600

Также оконные функции просто незааменимое средство для упорядочивания элементов в группе по некоторум параметру. Это позволяет фильтровать данные по рейтингу некоторого показателя в рамках группы. Например следующий запрос позволяет отсыкать сумму населения в двух самых крупных городах страны.

```
select * , Rank() over (partition by CountryCode order by population desc) as r from city;


select CountryCode, sum(population) from(
select * , Rank() over (partition by CountryCode order by population desc) as r from city) as t
where t.r<=2
group by CountryCode
;
```

Первый запрос иллюстрирует работу второго. В нём города получат своеобразный рейтинговый номер по населению в рамках своей страны. Этот рейтинг можно использовать, чтобы отфильтровать топ-2.

Результат первого запроса с рейтингом.

ID	Name	CountryCode	District	Population	r
129	Oranjestad	ABW	Â–	29034	1
1	Kabul	AFG	Kabol	1780000	1
2	Qandahar	AFG	Qandahar	237500	2
3	Herat	AFG	Herat	186800	3
4	Mazar-e-Sharif	AFG	Balkh	127800	4
56	Luanda	AGO	Luanda	2022000	1
57	Huambo	AGO	Huambo	163100	2
58	Lobito	AGO	Benguela	130000	3
59	Benguela	AGO	Benguela	128300	4
60	Namibe	AGO	Namibe	118200	5

Результат запроса на топ-2 города.

CountryCode	sum(population)
ABW	29034
AFG	2017500
AGO	2185100
AIA	1556
ALB	270000
AND	21189
ANT	2345
ARE	1067876

Задания для самостоятельной работы.

1. Реализуйте образ базы данных bankDB согласно схеме. Выполните задания.
   1.1. Доработайте транзакцию для перевода денежных средств из примера 2. Выполните недостающие проверки.
   1.2. Создайте аналогичные транзактные методы для пополнения счёта и снятия средств со счёта.
   1.3. Смоделируйте возникновение взаимной блокировки (deadlock) при работе транзакций в разных подключениях. Предложите и реализуйте в коде обход таких блокировок.
   1.4. Создайте аналогичный образ bankDB, в котором некоторые проверки из транзакций вынесены в триггеры.
   1.5. Основные данные счёта должны хранится в зашифрованном виде. Произведите шифрование. Переработайте основные транзакции под новую модель хранения данных.
   1.6. Создайте представление для отображения истории счёта, конкретного пользователя за определённый период.
   1.7. Добавьте в вашу БД отдельную функциональность с кредитами пользователя.
   1.8. Реализуйте хранимые процедуры для получения кредитной истории, получения и пошагового погашения кредита с различным типом процентных ставок в транзактном режиме.
2. Реализуйте небольшую БД магазина с ключевыми сущностями (склад, торговые точки, продавцы, продажи)
   2.1. Реализуйте процедуры покупок в транзактном режиме. Реализуйте соответствующие проверочные триггеры.
   2.2. Предусмотрите возможность автоматического заказа товара со склада при достижении некоторого порога.
   2.3. Добавьте автономную систему бонусов для клиентов с помощью триггеров. Личные данные пользователей шифруйте.
3. Выполните задания для вашей БД из ЛР1 согласно вариантам.

3.1. Разработайте ряд ключевых учётных записей пользователей системы. Продумайте различные уровни привилегий.
3.2. Реализуйте не менее 2-ух транзакций в хранимых процедурах.
3.3. Создайте необходимые индексы для вашей БД. Тщательно обоснуйте их использование.
3.4. Обеспечьте шифрование необходимых данных в вашей БД. Переработайте функционал с учётом этого факта.

4. Реализуйте собственную БД подобную litDB.
4.1. Заполните БД своими цитатами. Не менее 50. (БД может быть одна на группу).
4.2. Продемонстрируйте работу полнотекстового индекса на различных примерах.
4.3. Порассуждайте о том, как вычисляется коэффициент релевантности? Сделайте предположение, подкреплённое примерами.
5. Протестируйте два ключевых движка таблиц (ENGINE) в MySQL (InnoDB и MyISAM). Дайте осознанный и вдумчивый ответ о работе индексов и транзакций в таблицах данного типа.*
6. Выполните несколько запросов к БД world, используя оконные функции в запросах.

Литература:

1. Кузнецов, Симдянов – MySQL 5.0.
2. Линн Бейли – Изучаем SQL.
3. Кронке – Теория и практика построения баз данных.
4. Коннолли, Берг – Базы данных.