Transactions. Savepoints.

JDBC 4.0

- При проектировании распределенных систем часто возникают ситуации, когда сбой в системе или какой-либо ее периферийной части может привести к потере информации или к финансовым потерям.

Транзакции

Транзакцию (деловую операцию) определяют как единицу работы, обладающую свойствами ACID:

- Атомарность – две или более операций выполняются все или не выполняется ни одна. Успешно завершенные транзакции фиксируются, в случае неудачного завершения происходит откат всей транзакции.
- Согласованность – при возникновении сбоя система возвращается в состояние до начала неудавшейся транзакции. Если транзакция завершается успешно, то проверка согласованности удостоверяется в успешном завершении всех операций транзакции.

- Изолированность – во время выполнения транзакции все объекты сущности, участвующие в ней, должны быть синхронизированы.
- Долговечность – все изменения, произведенные с данными во время транзакции, сохраняются, например, в базе данных. Это позволяет восстанавливать систему.

Транзакции и JDBC

Для фиксации результатов работы SQL-операторов, логически выполняемых в рамках некоторой транзакции, используется SQL-оператор COMMIT.

- В API JDBC эта операция выполняется по умолчанию после каждого вызова методов executeQuery() и executeUpdate(). Если же необходимо сгруппировать запросы и только после этого выполнить операцию COMMIT, сначала вызывается метод setAutoCommit(boolean param) интерфейса Connection с параметром false

- Подтверждает выполнение SQL-запросов метод commit() интерфейса Connection, в результате действия которого все изменения таблицы производятся как одно логическое действие.
- Если же транзакция не выполнена, то методом rollback() отменяются действия всех запросов SQL, начиная от последнего вызова commit().

Изоляция транзакций

- JDBC удовлетворяет четырем уровням изоляции транзакций, определенным в стандарте SQL:2003.
- Уровни изоляции транзакций определены в виде констант интерфейса **Connection** (по возрастанию уровня ограничения):

- TRANSACTION_NONE – информирует о том, что драйвер не поддерживает транзакции;
- TRANSACTION_READ_UNCOMMITTED – позволяет транзакциям видеть несохраненные изменения данных, что разрешает грязное, непроверяющееся и фантомное чтения;
- TRANSACTION_READ_COMMITTED – означает, что любое изменение, сделанное в транзакции, не видно вне неё, пока она не сохранена. Это предотвращает грязное чтение, но разрешает непроверяющееся и фантомное;

- TRANSACTION_REPEATABLE_READ – запрещает грязное и непроверяющееся, но фантомное чтение разрешено;
- TRANSACTION_SERIALIZABLE – определяет, что грязное, непроверяющееся и фантомное чтения запрещены.

Dirty Read

Transaction 1

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
```

Transaction 2

```sql
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
/* No commit here */
```

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
```

```sql
ROLLBACK; /* lock-based DIRTY READ */
```

Non repeatable read

Transaction 1

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
```

Transaction 2

```sql
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
COMMIT; /* in multiversion concurrency
   control, or lock-based READ COMMITTED */
```

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
COMMIT; /* lock-based REPEATABLE READ */
```

Phantom Read

Transaction 1

```sql
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
```

Transaction 2

```sql
/* Query 2 */
INSERT INTO users VALUES ( 3, 'Bob', 27 );
COMMIT;
```

```sql
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
```

Методы

- Метод boolean supportsTransactionIsolationLevel(int level) интерфейса DatabaseMetaData определяет, поддерживается ли заданный уровень изоляции транзакций.
- Методы интерфейса Connection определяют доступ к уровню изоляции:
- int getTransactionIsolation() – возвращает текущий уровень изоляции;
- void setTransactionIsolation(int level) – устанавливает нужный уровень.

Точки сохранения

- Точки сохранения дают дополнительный контроль над транзакциями. Установкой точки сохранения обозначается логическая точка внутри транзакции, которая может быть использована для отката данных.
- Таким образом, если произойдет ошибка, можно вызвать метод rollback()для отмены всех изменений, которые были сделаны после точки сохранения.

Методы

- Метод boolean supportsSavepoints() интерфейса DatabaseMetaData используется для того, чтобы определить, поддерживает ли точки сохранения драйвер JDBC и сама СУБД.

- Методы setSavepoint(String name) и setSavepoint() (оба возвращают объект Savepoint) интерфейса Connection используются для установки именованной или неименованной точки сохранения во время текущей транзакции.
- При этом новая транзакция будет начата, если в момент вызова **setSavepoint()** не будет активной транзакции.

Задание

“Упаковать” в одну транзакцию обновление таблицы и выборку данных из таблицы. Вывести результат выборки. ***Примечание:*** перед началом выполнения установите уровень изоляции TRANSACTION_SERIALIZABLE

«Упаковать» в транзакцию два запроса на обновления один из которых ошибочный. В обработчике исключения вызвать метод rollback. Вывести данные из таблицы до и после обновления.