БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ и ИНФОРМАТИКИ
Кафедра многопроцессорных сетей и систем

Рафеенко Е.Д.

Web- программирование

Доступ к базам данных

Содержание

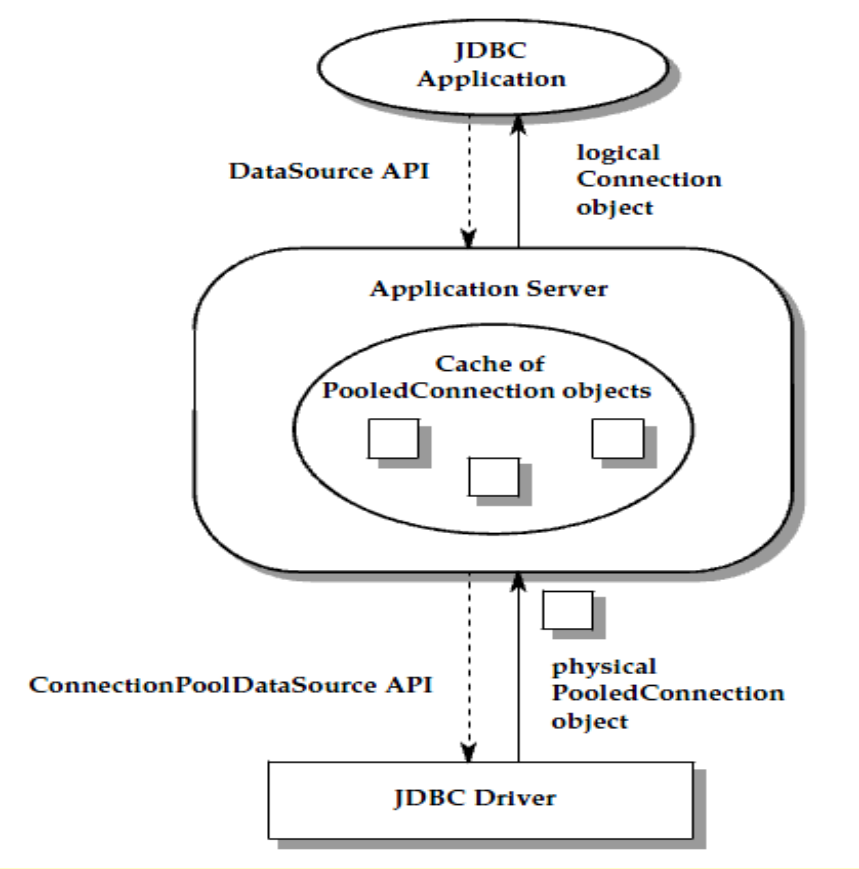
Пул соединений. Выделение ресурсов соединениям

Использование Connection в Web-приложениях

- Стандартный подход - при каждом обращении клиента создавать новое соединение, обмениваться данными с СУБД и закрывать соединение.

- При использовании технологии Servlet, которая позволяет хранить данные между обращениями пользователя, можно создать класс - пул соединений (DB Connection Pool).

Пул соединений



- Пул соединений реализуется согласно шаблону Singleton. В нем необходимо создать свойство -коллекцию соединений, в котором будут хранится все свободные соединения с СУБД.

- По завершении работы с БД следует вернуть соединение в пул, вызвав соответствующий метод.

Пул соединений

- Для работы с БД сначала необходимо открыть соединение к ней и получить объект типа Connection.

  Пул соединений представляет собой класс в виде набора объектов JDBC Connection и методов доступа к ним.
- По своей сути это – *контейнер с простейшим интерфейсом для контроля и управления над производимыми соединениями к базе данных*.
- Стандартная реализация пула выполнена в виде класса – DataSource, который даёт возможность:

  -загрузить необходимые драйвера для конкретной базы данных;

  - получить ссылку на объект типа DataSource;

  - получить доступное соединение типа Connection из хранилища;

  - возвратить соединение обратно в хранилище;

  - уничтожить все ресурсы и закрыть все соединения из хранилища;

Транзакции

Транзакции - Пример

Пример: *перечисление денег с одного счета на другой*.

- Если сбой произошел в тот момент, когда операция снятия с одного счета деньги уже произведена, а операция зачисления на другой счет еще не произведена, то система позволяющие такие ситуации должна быть признана не отвечающей требованиям заказчика.
- Такие операции должны выполняться обе или не выполняться вовсе.
- В этом случае такие две операции трактуют как одну и называют **транзакцией**.

Транзакции

Транзакции должны удовлетворять условиям:

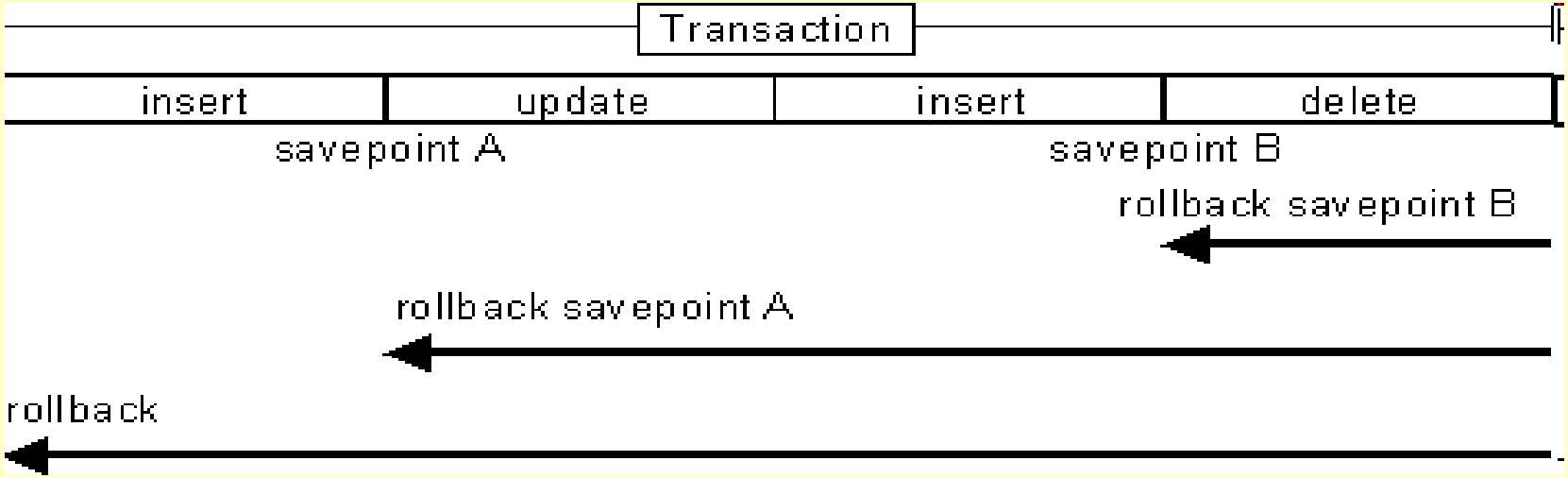
- **Атомарность (Atomicity)** – две или более операций выполняются все или не выполняется ни одна. Успешно завершенные транзакции фиксируются, в случае неудачного завершения происходит откат всей транзакции.
- **Согласованность (Consistency)** – если происходит сбой, то система возвращается в состояние до начала неудавшейся транзакции. Если транзакция завершается успешно, то проверка согласованности проверяет успешное завершение всех операций транзакции.
- **Изолированность (Isolation)** – во время выполнения транзакции все объекты-сущности, участвующие в ней, должны быть синхронизированы.
- **Долговечность (Durability)–** все изменения, произведенные с данными во время транзакции, записываются в базу данных. Это позволяет восстанавливать систему.

Транзакции

- Чтобы все операции SQL выполняли транзакции, в БД используется ключевое слово **COMMIT**.
- В JDBC эта операция выполняется по умолчанию после каждого вызова методов **executeQuery()** и **executeUpdate()**.
- Включение режима неавтоматического подтверждения операций: вызывается метод **setAutoCommit()** 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.
- Метод **rollback()** отменяет действия всех запросов SQL, начиная от последнего вызова **commit()**.

Точки сохранения

- Начиная с версии 3.0, JDBC поддерживает точки сохранения.
- Интерфейс **Savepoint** позволяет разделить транзакцию на логические блоки, дающие возможность откатывать совершённые изменения не к последнему вызову **commit()**, а лишь к заранее установленной точке сохранения.



Точки сохранения - Пример

```
cn.setAutoCommit(false);

Statement st = cn.createStatement();
int rows = st.executeUpdate("INSERT INTO Employees " +
            "(FirstName, LastName) VALUES " + "('Игорь', 'Цветков')");
// Устанавливаем именованную точку сохранения.
Savepoint svpt = cn.setSavepoint("NewEmp");

// ...
rows = st.executeUpdate("UPDATE Employees
            set Address = 'ул. Седых, 19-34' " +
             "WHERE LastName = 'Цветков'");

// ...
cn.rollback(svpt);
// ...
// Запись о работнике вставлена, но адрес не обновлен.
cn.commit();
```

Транзакции

Для транзакций существует несколько типов чтения:

- Грязное чтение (dirty reads) происходит, когда транзакциям разрешено видеть несохраненные изменения данных. Изменения, сделанные в одной транзакции, видны вне ее до того, как она была сохранена.
- Неповторяющееся чтение (nonrepeatable reads) происходит, когда транзакция А читает строку, транзакция Б изменяет эту строку, транзакция А читает ту же строку и получает обновленные данные.
- Фантомное чтение (phantom reads) происходит, когда транзакция А считывает все строки, удовлетворяющие WHERE-условию, транзакция Б вставляет новую или удаляет одну из строк, которая удовлетворяет этому условию, транзакция А еще раз считывает все строки, удовлетворяющие WHERE-условию, уже вместе с новой строкой или недосчитавшись старой.

Транзакции

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.

Уровни изоляции транзакций

- Установка уровня изоляции - `setTransactionIsolation(`*`level`*`)`

Transaction Level	Permitted Phenomena			Impact
	Dirty Reads	Non-Repeatable Reads	Phantom Reads	
TRANSACTION_NONE	-	-	-	FASTEST
TRANSACTION_READ_UNCOMMITED	YES	YES	YES	FASTEST
TRANSACTION_READ_COMMITED	NO	YES	YES	FAST
TRANSACTION_REPEATABLE_READ	NO	NO	YES	MEDIUM
TRANSACTION_SERIALIZABLE	NO	NO	NO	SLOW

Преимущества JDBC

- Лёгкость разработки: разработчик может не знать специфики базы данных, с которой работает
- Код не меняется, если компания переходит на другую базу данных
- Не нужно устанавливать громоздкую клиентскую программу
- К любой базе можно подсоединиться через легко описываемый URL

Информационные ресурсы
Организационные вопросы