Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Транзакции

Предпосылки для появления транзакций

Проблема конкуренции

Сервер баз данных может обслуживать нескольких клиентов одновременно и каждый из них может вносить изменения одновременно с другими.

Проблема целостности

Все операторы SQL должны выполниться одновременно, т.к. они являются частью одной задачи.

Транзакции

Транзакция — это совокупность операций над базой данных, которые вместе образуют логически целостную процедуру, и могут быть либо выполнены все вместе, либо не будет выполнена ни одна из них.

Транзакция фиксируется, тем самым сохраняя изменения в базе данных, либо отменяется, удаляя все изменения и восстанавливая данные до их прежнего состояния до запуска транзакции.

Результат выполнения транзакции

Транзакция может иметь два исхода:

изменения данных, успешно зафиксированы в базе данных (фиксация результата)

транзакция отменяется, и отменяются все изменения, выполненные в ее рамках (отмена результата)

Многоверсионная модель MVCC

Многоверсионное управление конкурентным доступом (Multiversion Concurrency Control, MVCC)

Эта модель предполагает, что каждый SQL-оператор видит так называемый снимок данных (snapshot), т. е. то согласованное состояние (версию) базы данных, которое она имела на определенный момент времени вне зависимости от текущего состояния данных.

Преимущества MVCC

Основное преимущество использования модели MVCC по сравнению с блокированием заключается в том, что блокировки MVCC, полученные для чтения данных, не конфликтуют с блокировками, полученными для записи, и поэтому чтение никогда не мешает записи, а запись чтению.

Важное следствие применения MVCC — операции чтения никогда не блокируются операциями записи, а операции записи никогда не блокируются операциями чтения.

Модели конкурентного доступа

Существуют следующие модели одновременного конкурентного доступа:

пессимистический одновременный конкурентный доступ

оптимистический одновременный конкурентный доступ

Свойства транзакций

Транзакции обладают следующими свойствами, которые все вместе обозначаются сокращением ACID:

атомарность (Atomicity)

согласованность (Consistency)

изолированность (Isolation)

долговечность (Durability)

Виды транзакций

Существует два вида транзакций.

Неявная транзакция - задает любую отдельную инструкцию INSERT, UPDATE или DELETE как единицу транзакции.

Явная транзакция - обычно это группа инструкций языка SQL, начало и конец которой обозначаются такими инструкциями, как BEGIN, COMMIT и ROLLBACK.

Подготовка к транзакции

Когда требуется транзакция? В случае предельно важных изменений в записях БД, требуется контроль точности выполнения операций.

```
UPDATE accounts SET balance = balance - 100.00
 WHERE name = ‘Peter';
UPDATE branches SET balance = balance - 100.00
 WHERE name = (SELECT branch_name FROM accounts WHERE name = ‘Peter');
UPDATE accounts SET balance = balance + 100.00
 WHERE name = ‘Robert';
UPDATE branches SET balance = balance + 100.00
 WHERE name = (SELECT branch_name FROM accounts WHERE name = ‘Robert’);
```

Очевидно, что без контроля может случится сбой.

Начало и завершение транзакции

Во многих реляционных СУБД транзакция определяется набором команд, окружённым командами BEGIN и COMMIT. Таким образом, наша транзакция должна была бы выглядеть так:

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00 WHERE name = ‘Peter’;
-- ...
COMMIT;
```

Полноценная транзакция

Предположим, что списывается 100 долларов со счёта Питера, добавляется на счёт Роберта, и вдруг оказывается, что деньги нужно было перевести Мартину.

В данном случае можно применить точки сохранения:

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
 WHERE name = Peter';
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00
 WHERE name = Robert';
-- ошибочное действие... забыть его и использовать счёт Уолли
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00
 WHERE name = ‘Martin';
COMMIT;
```

Журнал транзакций

Реляционные системы баз данных создают запись для каждого изменения, которые они выполняют в базе данных в процессе транзакции. Это требуется на случай ошибки при выполнении транзакции. В такой ситуации все выполненные инструкции транзакции необходимо отменить, осуществив для них откат. Как только система обнаруживает ошибку, она использует сохраненные записи, чтобы возвратить базу данных в согласованное состояние, в котором она была до начала выполнения транзакции.

Идентификаторы команд и транзакций

В PostgreSQL идентификаторы команд 32-битные. Это создаёт жёсткий лимит на 2^32 (4 миллиарда) команд SQL в одной транзакции.

Идентификаторы транзакций также являются 32-битными.

Распределённые базы данных

Распределённой базой данных называют систему взаимодействующих между собой баз данных, которую пользователь информационной системы воспринимает как одну базу данных.

Вопросы

Список литературы

1. Шёниг, Г. -. PostgreSQL 11. Мастерство разработки / Г. -. Шёниг ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 352 с. — ISBN 978-5-97060-671-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131714 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Список литературы

2. MySQL 8 для больших данных / Ш. Чаллавала, Д. Лакхатария, Ч. Мехта, К. Патель ; перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2018. — 226 с. — ISBN 978-5-97060-653-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131684 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Список литературы

3. Джуба, С. Изучаем PostgreSQL 10 / С. Джуба, А. Волков. — Москва : ДМК Пресс, 2019. — 400 с. — ISBN 978-5-97060-643-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/116125 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

1. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник для вузов / В. К. Волк. — 3-е изд., стер. — Санкт-Петербург : Лань, 2022. — 244 с. — ISBN 978-5-8114-9368-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/193373 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

2. Сьоре, Э. Проектирование и реализация систем управления базами данных / Э. Сьоре ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2021. — 466 с. — ISBN 978-5-97060-488-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/190718 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

3. Брэдшоу, Ш. Mongo DB Полное руководство : руководство / Ш. Брэдшоу, Й. Брэзил, К. Ходоров ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2020. — 540 с. — ISBN 978-5-97060-792-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179483 (дата обращения: 11.02.2022). — Режим доступа: для авториз. пользователей.

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Изоляция транзакций

Реализация уровней изоляции

Существует два глобально различных подхода к реализации изолированности: блокирование и версионирование.

Версионирование (snapshot)

Блокирование (lock)

Для блокировок характерно понятие оптимистичной и пессимистичной блокировки.

Уровни изоляции транзакций

Стандарт SQL определяет четыре уровня изоляции транзакций:

Read uncommited (Чтение незафиксированных данных)

Read committed (Чтение зафиксированных данных)

Repeatable read (Повторяемое чтение)

Serializable (Сериализуемость)

Особые условия изоляции транзакций

При работе транзакций существуют особые условия

«Грязное» чтение (Dirty Reads)

Неповторяемое чтение (None-Repeatable Reads)

Фантомное чтение (Phantom Reads)

Аномалия сериализации/Потерянное обновление (Lost Update)

Уровни изоляции и особые условия

Уровень изоляции	Особые условия	«Грязное» чтение (Dirty Reads)	Неповторяемое чтение (None-Repeatable Reads)	Фантомное чтение (Phantom Reads)	Аномалия сериализации (Lost Update)
Read uncommited (Чтение незафиксированных данных)		Да, но не в PG	Да	Да	Да
Read committed (Чтение зафиксированных данных)		Нет	Да	Да	Да
Repeatable read (Повторяемое чтение)		Нет	Нет	Да, но не в PG	Да
Serializable (Сериализуемость)		Нет	Нет	Нет	Нет

Особенности в PostgreSQL

Команды определены в стандарте SQL, за исключением режима транзакции DEFERRABLE и формы SET TRANSACTION SNAPSHOT, которые являются расширениями PostgreSQL.

Уровень изоляции Read Uncommitted

Уровень, имеющий самую плохую согласованность данных, но самую высокую скорость выполнения транзакций. Каждая транзакция видит незафиксированные изменения другой транзакции (феномен грязного чтения).

Выполняются две транзакции T1 и T2. T1 вносит изменения в данные с помощью операторов `INSERT`, `DELETE`, `UPDATE`. При этом T2 видит данные другой транзакции, которые еще не были зафиксированы.

При этом уровне изоляции, в случае отката T1 данные полученные T2 окажутся ошибочными. Таким образом, наблюдается феномен грязного чтения.

Уровень изоляции Read committed

На этом уровня параллельно исполняющиеся транзакции видят только зафиксированные изменения из других транзакций. Таким образом, данный уровень обеспечивает защиту от грязного чтения.

Выполняются две транзакции T1 и T2. T1 вносит изменения в данные с помощью операторов INSERT, DELETE, UPDATE. При этом T2 **НЕ** видит данные другой транзакции, которые еще не были зафиксированы. **НО** в T1 на данном этапе видны внесенные изменения.

После выполнения COMMIT, т.е. фиксации изменений транзакцией T1, другая транзакция T2 теперь увидит внесенные изменения.

Уровень изоляции Repeatable read

Уровень, позволяющий предотвратить феномен неповторяющегося чтения. При этом не видно в исполняющейся транзакции T2 измененные и удаленные записи другой транзакцией T1.

Выполняются две транзакции T1 и T2. В T1 выполняем запросы `INSERT`, `UPDATE` и `DELETE`. Затем, в T2 пытаемся обновить ту же самую строку, которую обновили в T1.

В такой ситуации T2 будет ждать, пока T1 зафиксирует изменения или откатится.

Таким образом, можно читать все изменения только своей транзакции. Данные, измененные другими транзакциями, недоступны.

Уровень изоляции Serializable

Транзакции могут выполняться только одна за другой. Медленная эффективность выполнения. В классическом представлении этот уровень избавляет от эффекта чтения фантомов.

Выполняются две транзакции T1 и T2. T2 читает таблицу table1 (делает SELECT), затем T1 пытается выполнить `INSERT, UPDATE и DELETE` для той же таблицы прочитанные T2 в рамках транзакции.

В такой ситуации T1 не может изменить данные, прочитанные T2. Поэтому T1 будет ждать, пока T2 завершит работу.

Значения по умолчанию для изоляции

В большинстве приложений уровень изолированности редко меняется и используется значение по умолчанию (например, в MySQL это repeatable read, в Oracle, MSSQL Server, PostgreSQL — read committed).

Однако на уровне операторов (SELECT, UPDATE и т.д.) в Oracle по умолчанию уже есть REPEATABLE READ, т.е. в рамках одного оператора всегда получается согласованное чтение, что достигается конечно же за счет сегмента отката.

Чем сильнее уровень изоляции, тем меньше производительность БД — потому что транзакциям чаще приходится ждать друг друга.

Установка требуемого уровня изоляции

Для выбора нужного уровня изоляции транзакций используется команда SET TRANSACTION.

```
SET TRANSACTION режим_транзакции [, ...]
```

Где режим_транзакции может быть следующим:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE
READ | READ COMMITTED | READ
 UNCOMMITTED }
 READ WRITE | READ ONLY
 [ NOT ] DEFERRABLE
```

Пример SET TRANSACTION

Пример использования:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED, READ WRITE;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ, READ ONLY;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Команда `SET TRANSACTION SNAPSHOT`

Команда `SET TRANSACTION SNAPSHOT` позволяет выполнить новую транзакцию со снимком данных, который имеет уже существующая транзакция.

```
SET TRANSACTION SNAPSHOT id_снимка
```

pg_export_snapshot – функция экспорта снимка возвращает id_снимка (например, 000003A1-1).

Особенности SET TRANSACTION

Если команде `SET TRANSACTION` не предшествует `START TRANSACTION` или `BEGIN`, она выдаёт предупреждение и больше ничего не делает.

Поэтому сначала нужно написать

```
START TRANSACTION
```

или

```
BEGIN
```

а затем

```
SET TRANSACTION
```

Транзакция со снимком данных

Начинаем транзакцию

```
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE
READ;
SELECT pg_export_snapshot();
 pg_export_snapshot
---------------------
00000003-0000001B-1
(1 row)
```

Передаем id команде SET TRANSACTION SNAPSHOT

```
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE
READ;
SET TRANSACTION SNAPSHOT '00000003-0000001B-1';
```

Команда START TRANSACTION

Команда начинает новый блок транзакции

START TRANSACTION — начать блок транзакции (команда описана в стандарте SQL)

```
START TRANSACTION режим_транзакции, [,…]
```

Где режим_транзакции может быть следующим:

```
 ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ
 UNCOMMITTED }
 READ WRITE | READ ONLY
```

Команда BEGIN

Команда начинает блок транзакции

BEGIN — начать блок транзакции

```
BEGIN  режим_транзакции, [,…]
```

Где режим_транзакции может быть следующим:

```
 ISOLATION    LEVEL    {    SERIALIZABLE    |
REPEATABLE READ | READ COMMITTED | READ
 UNCOMMITTED }
 READ WRITE | READ ONLY
```

Команда SAVEPOINT

Команда устанавливает новую точку сохранения в текущей транзакции.

SAVEPOINT — определяет новую точку сохранения в текущей транзакции

```
SAVEPOINT имя_точки_сохранения
```

Параметры:

имя_точки_сохранения

Имя, назначаемое новой точке сохранения.

Команда COMMIT

Команда начинает блок транзакции

COMMIT — начать блок транзакции

```
COMMIT [AND CHAIN]
```

Параметры:

AND CHAIN

начинается новая транзакция с такими же характеристиками транзакции

Команда ROLLBACK

Команда начинает блок транзакции

ROLLBACK — прерывает текущую транзакцию

```
ROLLBACK [AND CHAIN]
```

Параметры:

AND CHAIN

начинается новая транзакция с такими же характеристиками транзакции

Команда `ROLLBACK TO SAVEPOINT`

Команда возвращает изменения в базе данных к точке сохранения

`ROLLBACK TO SAVEPOINT` — откатиться к точке сохранения

```
ROLLBACK TO SAVEPOINT имя_точки_сохранения
```

Параметры:

`имя_точки_сохранения`

Точка сохранения, к которой нужно откатиться.

Команда RELEASE SAVEPOINT

Команда уничтожает точку сохранения, определённую ранее в текущей транзакции.

ROLLBACK TO SAVEPOINT — освобождает ранее определённую точку сохранения

```
RELEASE [ SAVEPOINT ] имя_точки_сохранения
```

Параметры:

имя_точки_сохранения

Имя точки сохранения, подлежащей уничтожению.

Популярный уровень изоляции

На вопрос: какой уровень изоляции в вашем текущем основном проекте?

размещенном на ресурсе https://habr.com/ в одном из постов были получены следующие ответы:

более 80%

Read committed (Чтение зафиксированных данных)

около 13%

Repeatable read (Повторяемое чтение)

около 10%

Serializable (Сериализуемость)

Вопросы

Список литературы

1. Шёниг, Г. -. PostgreSQL 11. Мастерство разработки / Г. -. Шёниг ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 352 с. — ISBN 978-5-97060-671-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131714 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Список литературы

2. MySQL 8 для больших данных / Ш. Чаллавала, Д. Лакхатария, Ч. Мехта, К. Патель ; перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2018. — 226 с. — ISBN 978-5-97060-653-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131684 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Список литературы

3. Джуба, С. Изучаем PostgreSQL 10 / С. Джуба, А. Волков. — Москва : ДМК Пресс, 2019. — 400 с. — ISBN 978-5-97060-643-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/116125 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

1. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник для вузов / В. К. Волк. — 3-е изд., стер. — Санкт-Петербург : Лань, 2022. — 244 с. — ISBN 978-5-8114-9368-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/193373 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

2. Сьоре, Э. Проектирование и реализация систем управления базами данных / Э. Сьоре ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2021. — 466 с. — ISBN 978-5-97060-488-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/190718 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Дополнительная литература

3. Брэдшоу, Ш. Mongo DB Полное руководство : руководство / Ш. Брэдшоу, Й. Брэзил, К. Ходоров ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2020. — 540 с. — ISBN 978-5-97060-792-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179483 (дата обращения: 25.02.2022). — Режим доступа: для авториз. пользователей.

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Аномалии и уровни изоляции транзакций

Аномалия потерянное обновление

При обновлении поля двумя транзакциями одно из изменений теряется.

Транзакция 1	Транзакция 2
SELECT x FROM tbl WHERE y=1;	SELECT x FROM tbl WHERE y=1;
UPDATE tbl SET x=5 WHERE y=1;	
	UPDATE tbl SET x=3 WHERE y=1;

Потерянное обновление не допускается стандартом ни на одном уровне изоляции.

Аномалия грязное чтение

Чтение данных, полученных в результате действия транзакции, которая после этого откатится.

Транзакция 1	Транзакция 2
SELECT x FROM tbl WHERE y=1;	
UPDATE tbl SET x=x+1 WHERE y=1;	
	SELECT x FROM tbl WHERE y=1;
ROLLBACK;	

Грязное чтение допускается на уровне изоляции Read Uncommitted.

Аномалия неповторяющееся чтение

Возникает, когда в течение одной транзакции при повторном чтении данные оказываются перезаписанными.

Транзакция 1	Транзакция 2
SELECT x FROM tbl WHERE y=1;	SELECT x FROM tbl WHERE y=1;
UPDATE tbl SET x=x+1 WHERE y=1;	
COMMIT;	
	SELECT x FROM tbl WHERE y=1;

Неповторяющееся чтение допускается стандартом на уровнях Read Uncommitted и Read Committed.

Аномалия фантомное чтение

Отличие от предыдущей аномалии в том, что при повторном чтении одна и та же выборка дает разные множества строк.

Транзакция 1	Транзакция 2
	SELECT SUM(x) FROM tbl;
INSERT INTO tbl (x, y) VALUES (5, 3);	
	SELECT SUM(x) FROM tbl;

Фантомное чтение допускается стандартом на уровнях Read Uncommitted, Read Committed, Repeatable Read.

Отсутствие аномалий и Serializable

Уровень Serializable должен предотвращать вообще все аномалии.

Уровни изоляции и аномалии по стандарту

	Потерянные изменения	Грязное чтение	Неповторяющееся чтение	Фантомное чтение	Другие аномалии
Read Uncommitted	—	да	да	да	да
Read Committed	—	—	да	да	да
Repeatable Read	—	—	—	да	да
Serializable	—	—	—	—	—

Эволюция блокировок транзакций

2PL – двухфазная блокировка

Snapshot Isolation - протокол изоляции на основе снимков

MVCC (multiversion concurrency control) - управление параллельным доступом посредством многоверсионности. Является расширением над Snapshot Isolation.

Уровни изоляции и аномалии в PostgreSQL

	Потерянные изменения	Грязное чтение	Неповторяющееся чтение	Фантомное чтение	Другие аномалии
Read Uncommitted	—	—	да	да	да
Read Committed	—	—	да	да	да
Repeatable Read	—	—	—	—	да
Serializable	—	—	—	—	—

Уровни изоляции на практике

```
=> CREATE TABLE accounts(
  id integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  number text UNIQUE,
  client text,
  amount numeric
);
=> INSERT INTO accounts VALUES
  (1, '1001', ‘alex', 1000.00),
  (2, '2001', ‘petr', 100.00),
  (3, '2002', ‘petr', 900.00);
```

Уровень изоляции Read Committed

Уровень изоляции по умолчанию

```
=> BEGIN;
=> SHOW transaction_isolation;
 transaction_isolation
-----------------------
 read committed
(1 row)
```

Уровень изоляции по умолчанию

```
=> SHOW default_transaction_isolation;
 default_transaction_isolation
-------------------------------
 read committed
(1 row)
```

Транзакция T1

```
=> UPDATE accounts SET amount = amount - 200 WHERE id = 1;
=> SELECT * FROM accounts WHERE client = ‘alex';
 id | number | client | amount
----+--------+--------+--------
  1 | 1001   | alex   | 800.00
(1 row)
```

Транзакция T2

```
|  => BEGIN;
|  => SELECT * FROM accounts WHERE client = ‘alex';
|   id | number | client | amount
|  ----+--------+--------+---------
|    1 | 1001   | alex   | 1000.00
|  (1 row)
```

Неповторяющееся чтение

=> COMMIT;

| => SELECT * FROM accounts WHERE client = ‘alex’;

| id | number | client | amount
| ----+--------+--------+--------
| 1 | 1001 | alice | 800.00
| (1 row)

| => COMMIT;

Принятие решения

```
IF (SELECT amount FROM accounts WHERE id = 1) >= 1000 THEN
    UPDATE accounts SET amount = amount - 1000 WHERE id = 1;
END IF;
```

Принятие решения

```
IF (SELECT amount FROM accounts WHERE id = 1) >= 1000 THEN
      -----
|     UPDATE accounts SET amount = amount - 200 WHERE id = 1;
|  COMMIT;
      -----
      UPDATE accounts SET amount = amount - 1000 WHERE id = 1;
END IF;
```

Корректный код

Как написать код корректно

Не писать код, который приведет двоякому смыслу. (можно использовать ограничение целостности)

ALTER TABLE accounts ADD CHECK amount >= 0;

Использовать один SQL-оператор (общие табличные выражения (CTE) или INSERT ON CONFLICT)

Пользовательские блокировки (SELECT FOR UPDATE, LOCK TABLE)

Вопросы

Список литературы

1. Gerhard Weikum,Gottfried Vossen. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery / Morgan Kaufmann Publishers Inc., 340 Pine Street, Sixth Floor, San Francisco, CA, United States, May 2001, 852 p. – ISBN 978-0-08-051956-2. — Текст : электронный // Доступна для чтения и скачивания в pdf формате на англ. языке с ресурса http://nozdr.ru/ по запросу в Яндекс: Transactional Information Systems

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Менеджеры транзакций

Предназначение менеджера транзакций

Менеджер транзакций контролирует и координирует выполнение транзакций в базе данных.

Менеджер транзакций LIXA

LIXA – Libre XA. Менеджер транзакций, построенные на стандарте XA.

Мониторы транзакций

Менеджер транзакций может быть подмножеством монитора транзакций. Большинство коммерческих продуктов объединяют функции в один пакет

Это плюс, если нужен монитор транзакций. Это минус, если нужен только диспетчер транзакций.

IBM TXSeries —монитор транзакций со встроенным диспетчером транзакций.

Oracle (BEA) Tuxedo — монитор транзакций со встроенным менеджером транзакций.

JBoss — сервер приложений JEE (Java Enterprise Edition); это монитор транзакций со встроенным диспетчером транзакций для приложений на основе Java.

Менеджеры транзакций

JOTM и BTM являются менеджерами транзакций, но они не являются мониторами транзакций

JOTM (Java Open Transaction Manager)

BTM (Bitronix JTA Transaction Manager)

XTA в LIXA

LIXA реализует диспетчер распределенных транзакций, который прикладные программы могут использовать для координации распределенной транзакции между двумя или более прикладными программами, работающими в разных системах.

XTA (XA Transaction API) - интерфейс прикладного программирования, разработанный для обеспечения двухфазной фиксации транзакций ACID для многоязычных приложений, ориентированных на микросервисы.

Интерфейс JTA (Java Transaction API) специфичен для одного языка и среды выполнения (Java/JRE)

Особенности LIXA

LIXA реализует автономный диспетчер транзакций, который прикладная программа может использовать разными способами.

Архитектура LIXA

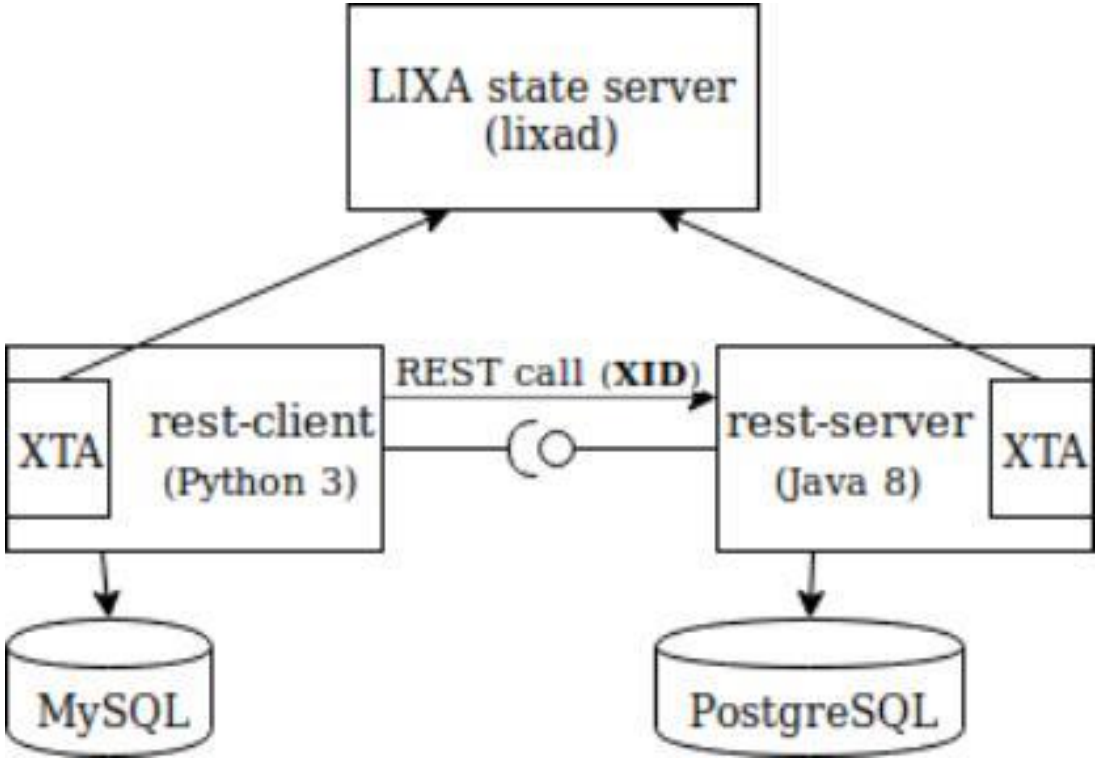
В LIXA прикладная программа связывает клиентскую библиотеку LIXA и включает логику менеджера транзакций.

Системные Требования LIXA

LIXA разработан под Ubuntu и портирован на различные версии Linux.

Клиентские библиотеки для языков

LIXA имеет библиотеки поддержки для C++, python, java



Менеджеры ресурсов и LIXA

Менеджеры ресурсов, которые были протестированы в сочетании с менеджером транзакций LIXA:

IBM DB2

MySQL

Oracle XE

Oracle SE

PostgreSQL

WebSphere MQ

MySQL и XTA для Python

Если нужно использовать XTA для Python, тогда нужен драйвер mysqlclient-python с включенной функцией _get_native_connection.

Команды установки:

```
git clone https://github.com/PyMySQL/mysqlclient-python.git

cd mysqlclient-python

python setup.py build

sudo python setup.py install
```

MySQL и XTA для Java

Если нужно использовать XTA для Java, тогда нужен драйвер JDBC, поддерживающий стандарт MySQL и JTA.

./configure --with-mysql --with-mysql-jdbc=/usr/share/java/mysql.jar

PostgreSQL и XTA для Python

Если нужно использовать XTA для Python, тогда нужен драйвер psycopg2c включенной функцией get_native_connection.

Команды для установки драйвера из исходного кода:

```
git clone https://github.com/psycopg/psycopg2.git

cd psycopg2

python setup.py build

sudo python setup.py install
```

PostgreSQL и XTA для Java

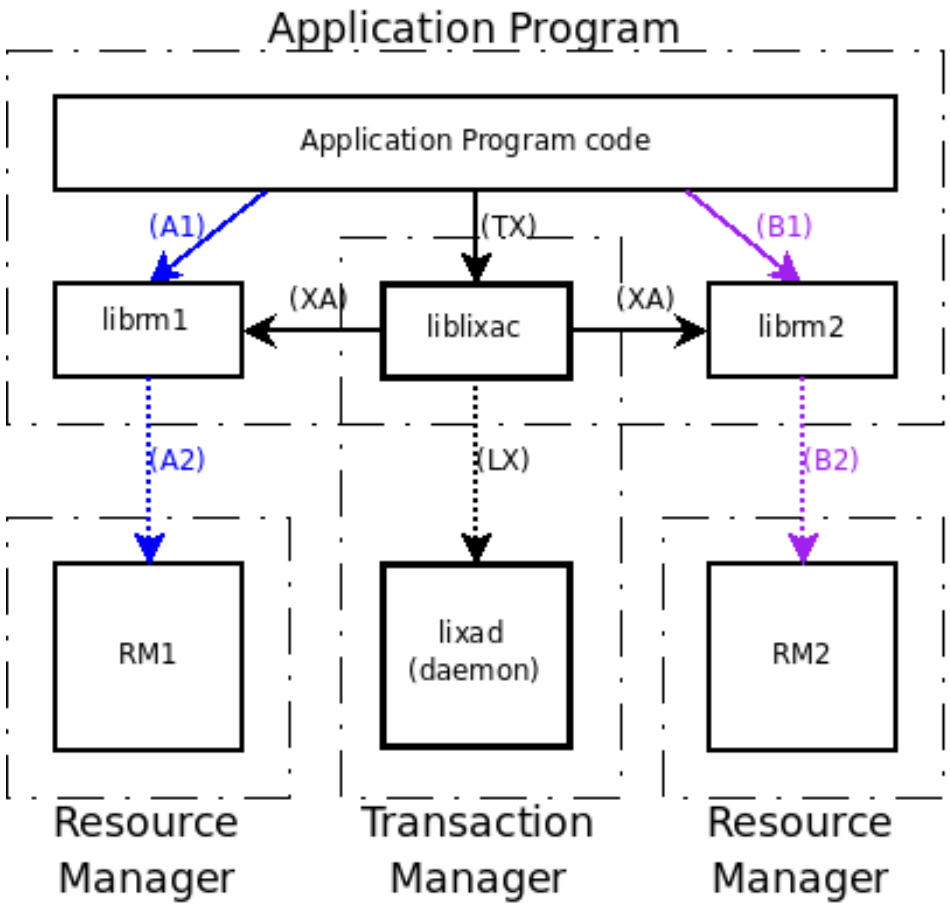
Если требуется использовать XTA для Java, нужен драйвер JDBC, поддерживающий стандарт PostgreSQL и JTA

Потребуется следующая конфигурационная строка

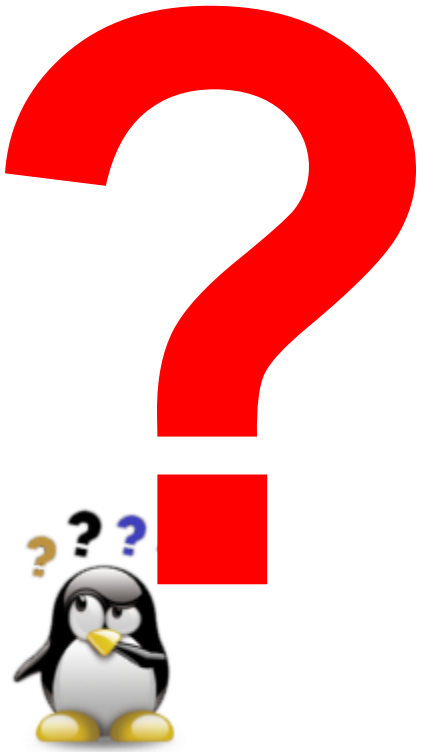
./configure --with-postgresql --with-postgresql-jdbc=/opt/postgresql/postgresql.jar

Архитектура LIXA

Компоненты LIXA



Вопросы



Список литературы

1. LIXA Reference Guide — URL: http://lixa.sourceforge.net/lixa-doc/html/index_all_in_one.html (дата обращения: 06.05.2022). — Режим доступа: открытый доступ

Дополнительная литература

2. Java Transaction API (JTA) — URL: https://www.oracle.com/java/technologies/jta.html (дата обращения: 06.05.2022). — Режим доступа: открытый доступ

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Спецификация XA для распределенных транзакций

Спецификация XA

Спецификация eXtended Architecture (XA), разработана X/Open Company, Ltd.

Представляет собой спецификацию для распределённых транзакций

XA-транзакция — распределённая транзакция

Параллельно развивались следующие стандарты распределённых транзакций (у каждого нет API) :

ROSE

OSI-CCR

OSI-TP

Компоненты XA

В XA используется модель распределенных транзакций, состоящая из трёх компонент:

1. Прикладная программа (AP)
2. Менеджер распределенных транзакций (TM)
3. Менеджеры ресурсов (RM)

Спецификация XA определяет взаимодействие только между менеджерами ресурсов (RM) и транзакций (TM).

XA представляет собой API на основе языка Си.

Спецификация XA+

Спецификация XA+, определяет транзакции, распределенные относительно приложений.

1. Прикладная программа (AP)
2. Менеджер распределенных транзакций (TM)
3. Менеджеры ресурсов (RM)
4. Менеджер по ресурсному взаимодействию (Communication Resource Manager, CRM)

Стандарты, основанные на XA

На спецификации XA основаны некоторые стандарты:

1. Java Transaction API
2. .Net Framework, поддержка распределённых транзакций

Спецификация Object Transaction Service

Спецификация разработана Object Management Group, в которой связываются объекты CORBA с транзакциями в базах данных.

CORBA использует XA.

Благодаря XA приложения CORBA могут принимать участие в распределенных транзакциях.

Диспетчер транзакций может использовать XA (который представляет собой API на основе языка Си) для взаимодействия с диспетчером ресурсов.

Object Transaction Service (OTS) — это служба CORBA

Типы транзакций

Транзакции делят на:

- Локальные
- Распределенные

Служба объектных транзакций CORBA

Object Transaction Service (OTS) — это служба CORBA.

OTS состоит из:

1. нескольких интерфейсов IDL
2. дополнительного библиотечного кода, связанного с клиентскими и серверными приложениями
3. диспетчера транзакций.

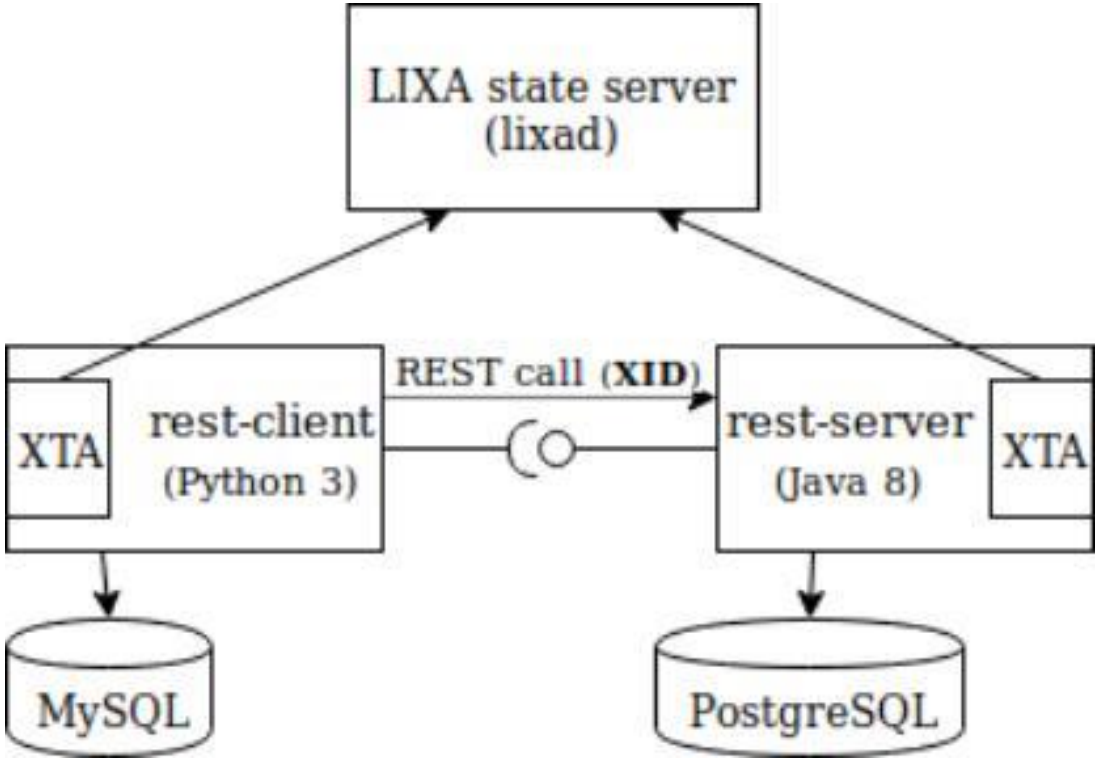
IDL - язык определения интерфейсов (Interface Definition Language)

API интерфейс OTS

```
interface TransactionFactory {
  Control create(in unsigned long time_out);
  ...
};
interface Control {
  Terminator  get_terminator();
  Coordinator get_coordinator();
};
interface Terminator {
  void commit(...);
  void rollback();
};
interface Coordinator {
  RecoveryCoordinator register_resource(in Resource r);
  ...
};
interface RecoveryCoordinator {
  Status replay_completion(in Resource r);
};
interface Resource {
  Vote prepare();
  void rollback();
  void commit();
  ...
};
local interface Current : CORBA::Current {
  void begin();
  void commit();
  void rollback();
  void set_timeout(in unsigned long seconds);
  unsigned long get_timeout();
  Control get_control();
  Control suspend();
  void resume(in Control which);
};
```

LIXA (LIbre XA)

Менеджер транзакций с открытым исходным кодом, реализующий распределенную обработку транзакций XA specification



Вопросы

Список литературы

1. Ciaran McHale Chapter 21 Object Transaction Service. — URL: http://www.ciaranmchale.com/corba-explained-simply/object-transaction-service.html (дата обращения: 22.04.2022). — Режим доступа: открытый доступ

Дополнительная литература

3. LIXA Reference Guide — URL: http://lixa.sourceforge.net/lixa-doc/html/index_all_in_one.html (дата обращения: 22.02.2022). — Режим доступа: открытый доступ

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Аномалии уровня изоляции Read Committed

Понимание термина транзакции

Транзакцией называется множество операций, выполняемое приложением, которое переводит базу данных из одного корректного состояния в другое корректное состояние (условие согласованное — consistent) при условии, что транзакция выполнена полностью (условие атомарности — atomicity) и без помех со стороны других транзакций (условие изоляции — isolation).

При этом система подтверждает (условие надёжности — durability) пользователю, что транзакция выполнена.

Т.о. выполняются требования ACID

Несогласованное чтение

Из определения транзакции следует, что согласованность наступает тогда, когда транзакция переводит базу данных из одного корректного состояния в другое корректное состояние, т.е. состояние базы данных до выполнения транзакции будет абсолютно точно соответствовать состоянию базы данных после выполнения транзакции.

Несогласованное чтение наступает тогда, когда транзакция переводит базу данных из одного корректного состояния в любое некорректное состояние, т.е. будут нарушены логические связки между данными.

Известные уровни изоляции

Изоляция / Аномалия	Потерянные изменения	Грязное чтение	Неповторяющееся чтение	Фантомное чтение	Другие аномалии
Read Uncommitted	—	да	да	да	да
Read Committed	—	—	да	да	да
Repeatable Read	—	—	—	—	да
Serializable	—	—	—	—	—

Исходные данные

Возьмем уже использовавшийся пример данных

```
=> CREATE TABLE accounts(
  id integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  number text UNIQUE,
  client text,
  amount numeric
);
=> INSERT INTO accounts VALUES
  (1, '1001', ‘alex', 1000.00),
  (2, '2001', ‘petr', 100.00),
  (3, '2002', ‘petr', 900.00);
```

Проверка значений в БД

=> SELECT * FROM accounts WHERE client = ‘petr’;

```
 id | number | client | amount
----+----------+---------+--------
  2 | 2001     | petr    | 100.00
  3 | 2002     | petr    | 900.00
(2 rows)
```

Транзакция T1: уменьшаем значение

Начинаем транзакцию, которая уменьшает баланс petr:

=> BEGIN;

=> UPDATE accounts SET amount = amount - 100 WHERE id = 3;

```
|    amount
|   --------
|    800.00
|   (1 row)
```

Параллельная транзакция T2

В это же время транзакция T2 выполняется одновременно с транзакцией T1 и начисляет проценты на все счета клиентов с общим балансом, равным или превышающим 1000:

```
|  => UPDATE accounts SET amount = amount * 1.01
|  WHERE client IN (
|    SELECT client
|    FROM accounts
|    GROUP BY client
|    HAVING sum(amount) >= 1000
|  );
```

Транзакция T1 завершается

=> COMMIT;

Смотрим результат

=> SELECT * FROM accounts WHERE client = ‘petr';

```
Id  | number | client |  amount
----+--------+--------+----------
  2 | 2001   | petr   | 101.0000
  3 | 2002   | petr   | 808.0000
(2 rows)
```

Итоги работы транзакций T1 и T2

Транзакция T2 получает некорректные данные: часть строк видна на один момент времени, часть — на другой.

ТЕМА

Аномалии уровня изоляции Repeatable Read

Отсутствие неповторяющегося и фантомного чтений

Название уровня изоляции говорит о том, что чтение является повторяемым.

Проверим такое поведение и убедимся и в отсутствии фантомных чтений. Для этого в первой транзакции вернем счета petr в прежнее состояние и создадим новый счет для victor

Известные уровни изоляции

Изоляция \ Аномалия	Потерянные изменения	Грязное чтение	Неповторяющееся чтение	Фантомное чтение	Другие аномалии
Read Uncommitted	—	да	да	да	да
Read Committed	—	—	да	да	да
Repeatable Read	—	—	—	—	да
Serializable	—	—	—	—	—

Транзакция T1

```
=> BEGIN;
=> UPDATE accounts SET amount = 100.00 WHERE id = 2;
=> UPDATE accounts SET amount = 900.00 WHERE id = 3;
=> INSERT INTO accounts VALUES
  (4, '3001', 'victor', 100.00);
=> SELECT * FROM accounts ORDER BY id;
```

```
 id | number | client  | amount
----+--------+---------+--------
  1 | 1001   | alex    | 1000.00
  2 | 2001   | petr    | 100.00
  3 | 2002   | petr    | 900.00
  4 | 3001   | victor  | 100.00
```

Транзакция T2

```
| => BEGIN ISOLATION LEVEL REPEATABLE READ;
| => SELECT * FROM accounts ORDER BY id;
 |   id | number | client |  amount
|  ----+--------+--------+----------
|    1 | 1001   | alice  | 1000.00
|    2 | 2001   | bob    | 101.0000
|    3 | 2002   | bob    | 808.0000
|  (3 rows)
```

Продолжение работы T1 и T2

T1

=> COMMIT;

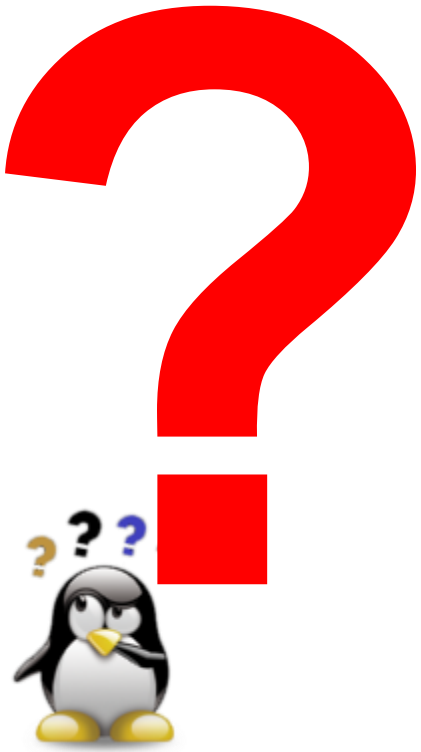
T2

| => SELECT * FROM accounts ORDER BY id;

```
 |   id | number | client |  amount
|  ----+--------+--------+----------
|    1 | 1001   | alex   | 1000.00
|    2 | 2001   | petr   | 101.0000
|    3 | 2002   | petr   | 808.0000
|  (3 rows)
```

| => COMMIT; //T2

Вопросы



Список литературы

1. Практическое применение команд PostgreSQL http://snakeproject.ru/python/Postgres.html

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Аномалии и уровни изоляции транзакций

Взаимная блокировка транзакций

В английской терминологии – deadlocks.

В некоторых случаях, две транзакции могут в ходе их обработки пытаться получить доступ к одной и той же части базы данных в одно и то же время, таким образом, что это будет препятствовать их совершению.

Особенности сериализации транзакций

Результат успешной фиксации группы транзакций, выполняющихся параллельно, не совпадает с результатом ни одного из возможных вариантов упорядочения этих транзакций, если бы они выполнялись последовательно.

Уровень изоляции Read Committed

Известные уровни изоляции

Изоляция / Аномалия	Потерянные изменения	Грязное чтение	Неповторяющееся чтение	Фантомное чтение	Другие аномалии
Read Uncommitted	—	да	да	да	да
Read Committed	—	—	да	да	да
Repeatable Read	—	—	—	да	да
Serializable	—	—	—	—	—

Исходные данные

Возьмем уже использовавшийся пример данных

```
=> CREATE TABLE accounts(
  id integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  number text UNIQUE,
  client text,
  amount numeric
);
=> INSERT INTO accounts VALUES
  (1, '1001', ‘alex', 1000.00),
  (2, '2001', ‘petr', 100.00),
  (3, '2002', ‘petr', 900.00);
```

Транзакция T1

Аномалия несогласованное чтение

```
=> BEGIN;
=> UPDATE accounts SET amount = amount - 100 WHERE id = 2;
```

Транзакция T2

```
|  => BEGIN;
|  => SELECT amount FROM accounts WHERE id = 2;

|    amount
|   --------
|    100.00
|   (1 row)
```

Завершение транзакции T1

В этот момент первая транзакция успешно завершается:

```
=> UPDATE accounts SET amount = amount + 100 WHERE id = 3;
=> COMMIT;
```

Транзакция T2 продолжает выполнение

```
| => SELECT amount FROM accounts WHERE id = 3;

|    amount
|   ---------
|    1000.00
|   (1 row)

| => COMMIT;
```

Устранение несогласованного чтения

SELECT sum(amount) FROM accounts WHERE client = ‘petr’;

=> SELECT amount, pg_sleep(2) FROM accounts WHERE client = ‘petr’;

Продолжение примера

Пока эта конструкция выполняется, в другой транзакции выполняем перенос средств обратно:

| => BEGIN;

| => UPDATE accounts SET amount = amount + 100 WHERE id = 2;

| => UPDATE accounts SET amount = amount - 100 WHERE id = 3;

| => COMMIT;

```
amount  | pg_sleep
---------+----------
    0.00 |
 1000.00 |
(2 rows)
```

Особенности выполнения в PostgreSQL

PostgreSQL позволяет определять функции, а у функций есть понятие категории изменчивости.

Транзакция 1

```
=> CREATE FUNCTION get_amount(id integer) RETURNS numeric AS $$
 SELECT amount FROM accounts a WHERE a.id = get_amount.id;
$$ VOLATILE LANGUAGE sql;

=> SELECT get_amount(id), pg_sleep(2)
FROM accounts WHERE client = ‘petr';
```

Транзакция 2

| => BEGIN;

| => UPDATE accounts SET amount = amount + 100 WHERE id = 2;

| => UPDATE accounts SET amount = amount - 100 WHERE id = 3;

| => COMMIT;

```
 get_amount | pg_sleep
------------+----------
     100.00 |
     800.00 |
(2 rows)
```

Параметр VOLATILE

```
CREATE [ OR REPLACE ] FUNCTION
    имя ( [ [ режим_аргумента ] [ имя_аргумента ] тип_аргумента [ { DEFAULT | = } выражение_по_умолчанию ] [, ...] ] )
    [ RETURNS тип_результата
      | RETURNS TABLE ( имя_столбца тип_столбца [, ...] ) ]
  { LANGUAGE имя_языка
    | TRANSFORM { FOR TYPE имя_типа } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | COST стоимость_выполнения
    | ROWS строк_в_результате
    | SET параметр_конфигурации { TO значение | = значение | FROM CURRENT }
    | AS 'определение'
    | AS 'объектный_файл', 'объектный_символ'
  } ...
    [ WITH ( атрибут [, ...] ) ]
```

Вопросы

Список литературы

1. Практическое применение команд PostgreSQL http://snakeproject.ru/python/Postgres.html

Спасибо за внимание!

Технологии обработки транзакций клиент-серверных приложений

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

Менеджеры транзакций

Предназначение менеджера транзакций

Менеджер транзакций контролирует и координирует выполнение транзакций в базе данных.

Менеджер транзакций LIXA

LIXA – Libre XA. Менеджер транзакций, построенные на стандарте XA.

Мониторы транзакций

Менеджер транзакций может быть подмножеством монитора транзакций. Большинство коммерческих продуктов объединяют функции в один пакет

Это плюс, если нужен монитор транзакций. Это минус, если нужен только диспетчер транзакций.

IBM TXSeries —монитор транзакций со встроенным диспетчером транзакций.

Oracle (BEA) Tuxedo — монитор транзакций со встроенным менеджером транзакций.

JBoss — сервер приложений JEE (Java Enterprise Edition); это монитор транзакций со встроенным диспетчером транзакций для приложений на основе Java.

Менеджеры транзакций

JOTM и BTM являются менеджерами транзакций, но они не являются мониторами транзакций

JOTM (Java Open Transaction Manager)

BTM (Bitronix JTA Transaction Manager)

XTA в LIXA

LIXA реализует диспетчер распределенных транзакций, который прикладные программы могут использовать для координации распределенной транзакции между двумя или более прикладными программами, работающими в разных системах.

XTA (XA Transaction API) - интерфейс прикладного программирования, разработанный для обеспечения двухфазной фиксации транзакций ACID для многоязычных приложений, ориентированных на микросервисы.

Интерфейс JTA (Java Transaction API) специфичен для одного языка и среды выполнения (Java/JRE)

Особенности LIXA

LIXA реализует автономный диспетчер транзакций, который прикладная программа может использовать разными способами.

Архитектура LIXA

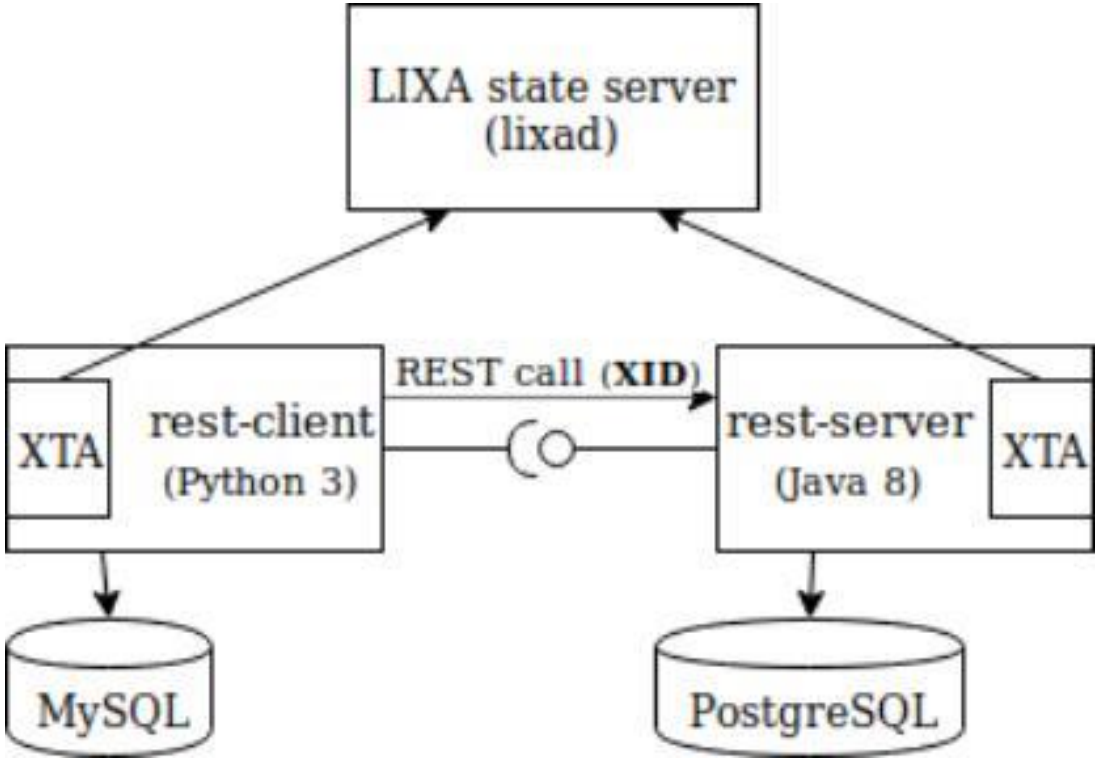
В LIXA прикладная программа связывает клиентскую библиотеку LIXA и включает логику менеджера транзакций.

Системные Требования LIXA

LIXA разработан под Ubuntu и портирован на различные версии Linux.

Клиентские библиотеки для языков

LIXA имеет библиотеки поддержки для C++, python, java



Менеджеры ресурсов и LIXA

Менеджеры ресурсов, которые были протестированы в сочетании с менеджером транзакций LIXA:

IBM DB2

MySQL

Oracle XE

Oracle SE

PostgreSQL

WebSphere MQ

MySQL и XTA для Python

Если нужно использовать XTA для Python, тогда нужен драйвер mysqlclient-python с включенной функцией _get_native_connection.

Команды установки:

```
git clone https://github.com/PyMySQL/mysqlclient-python.git

cd mysqlclient-python

python setup.py build

sudo python setup.py install
```

MySQL и XTA для Java

Если нужно использовать XTA для Java, тогда нужен драйвер JDBC, поддерживающий стандарт MySQL и JTA.

./configure --with-mysql --with-mysql-jdbc=/usr/share/java/mysql.jar

PostgreSQL и XTA для Python

Если нужно использовать XTA для Python, тогда нужен драйвер psycopg2c включенной функцией get_native_connection.

Команды для установки драйвера из исходного кода:

```
git clone https://github.com/psycopg/psycopg2.git

cd psycopg2

python setup.py build

sudo python setup.py install
```

PostgreSQL и XTA для Java

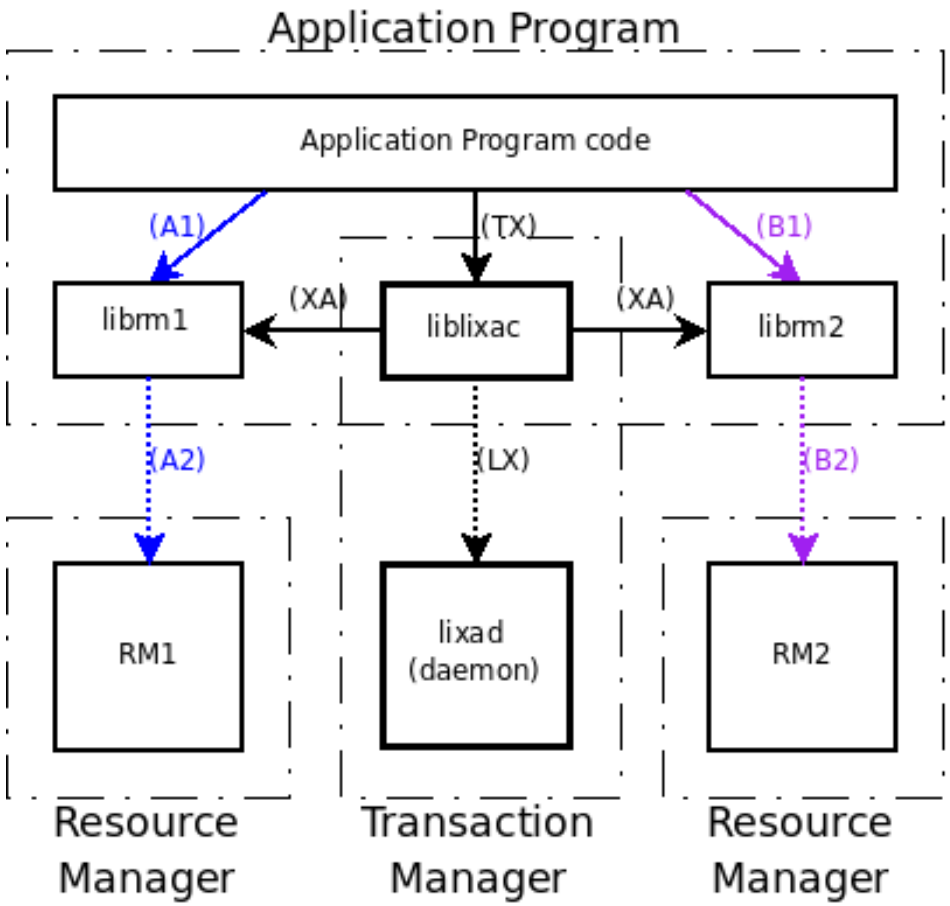
Если требуется использовать XTA для Java, нужен драйвер JDBC, поддерживающий стандарт PostgreSQL и JTA

Потребуется следующая конфигурационная строка

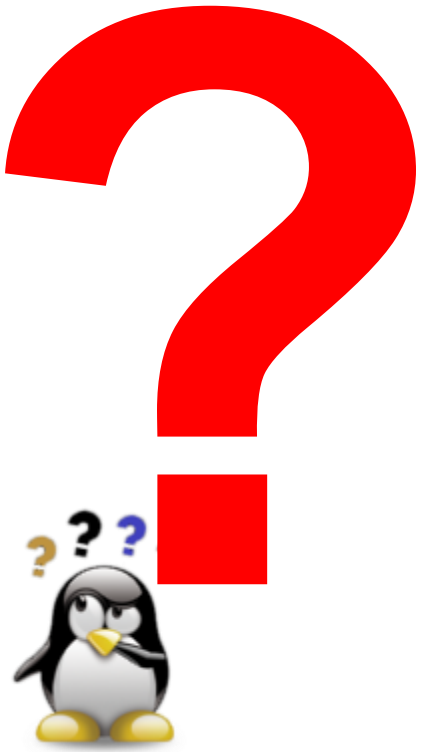
./configure --with-postgresql --with-postgresql-jdbc=/opt/postgresql/postgresql.jar

Архитектура LIXA

Компоненты LIXA



Вопросы



Список литературы

1. LIXA Reference Guide — URL: http://lixa.sourceforge.net/lixa-doc/html/index_all_in_one.html (дата обращения: 06.05.2022). — Режим доступа: открытый доступ

Дополнительная литература

2. Java Transaction API (JTA) — URL: https://www.oracle.com/java/technologies/jta.html (дата обращения: 06.05.2022). — Режим доступа: открытый доступ

Спасибо за внимание!