Механизм транзакций

Курс «Базы данных»

Вадим Цесько

Санкт-Петербургский государственный политехнический университет

5 апреля 2012 г.

Содержание

CAP-теорема Брюера

Выберите два из трёх:

- Consistency
- Availability
- Partition Tolerance

[1] http://blog.nahurst.com/visual-guide-to-nosql-systems

CAP-теорема Брюера

Выберите два из трёх:

- Consistency
- Availability
- Partition Tolerance

Consistency + Availability [1]:

- RDBMS: Firebird, MySQL, PostgreSQL, etc.

[1] http://blog.nahurst.com/visual-guide-to-nosql-systems

CAP-теорема Брюера

Выберите два из трёх:

- Consistency
- Availability
- Partition Tolerance

Consistency + Availability [1]:

- RDBMS: Firebird, MySQL, PostgreSQL, etc.

Consistency + Partition Tolerance:

- BigTable, HBase, MongoDB, etc.

[1] http://blog.nahurst.com/visual-guide-to-nosql-systems

CAP-теорема Брюера

Выберите два из трёх:

- Consistency
- Availability
- Partition Tolerance

Consistency + Availability [1]:

- RDBMS: Firebird, MySQL, PostgreSQL, etc.

Consistency + Partition Tolerance:

- BigTable, HBase, MongoDB, etc.

Availability + Partition Tolerance:

- Dynamo, Cassandra, CouchDB, Riak, etc.

[1] http://blog.nahurst.com/visual-guide-to-nosql-systems

Транзакции

Транзакция

Группа логически связанных операторов SQL, которая может выполниться либо **целиком**, либо не выполниться вообще.

Транзакции

Транзакция

Группа логически связанных операторов SQL, которая может выполниться либо **целиком**, либо не выполниться вообще.

Управление транзакциями в SQL

```
START TRANSACTION; -- неявный запуск транзакции
COMMIT;
ROLLBACK;
```

Транзакции

Транзакция

Группа логически связанных операторов SQL, которая может выполниться либо **целиком**, либо не выполниться вообще.

Управление транзакциями в SQL

```
START TRANSACTION; -- неявный запуск транзакции
COMMIT;
ROLLBACK;
```

Пример

```
START TRANSACTION;
OPERATOR1;
OPERATOR2;
...
COMMIT;
```

ACID-свойства транзакций

Atomicity

Транзакция выполнена либо целиком, либо не выполнена вообще.

ACID-свойства транзакций

Atomicity

Транзакция выполнена либо целиком, либо не выполнена вообще.

Consistency

Система переходит из согласованного состояния в начале транзакции в согласованное состояние в конце транзакции.

ACID-свойства транзакций

Atomicity

Транзакция выполнена либо целиком, либо не выполнена вообще.

Consistency

Система переходит из согласованного состояния в начале транзакции в согласованное состояние в конце транзакции.

Isolation

Промежуточные состояния не видны параллельным транзакциям.

ACID-свойства транзакций

Atomicity

Транзакция выполнена либо целиком, либо не выполнена вообще.

Consistency

Система переходит из согласованного состояния в начале транзакции в согласованное состояние в конце транзакции.

Isolation

Промежуточные состояния не видны параллельным транзакциям.

Durability

Изменения подтверждённой транзакции долговечны.

Назначение транзакций

Назначение транзакций

Транзакции обеспечивают целостность БД при **возникновении ошибок** и **многопользовательском доступе**.

Контроль целостности при ошибках

Основная проблема

Частичное выполнение действия, состоящего из нескольких операторов.

Контроль целостности при ошибках

Основная проблема

Частичное выполнение действия, состоящего из нескольких операторов.

Перевод денег между счетами

- Снять деньги со счёта A
- Зачислить деньги на счёт B

Контроль целостности при многопользовательском доступе

Потерянные изменения

Данные меняются сначала одним, а потом другим пользователем.

Контроль целостности при многопользовательском доступе

Потерянные изменения

Данные меняются сначала одним, а потом другим пользователем.

Чтение незафиксированных изменений

Данные меняются одним пользователем, читаются другим пользователем, а затем первый пользователь восстанавливает исходные данные.

Контроль целостности при многопользовательском доступе

Потерянные изменения

Данные меняются сначала одним, а потом другим пользователем.

Чтение незафиксированных изменений

Данные меняются одним пользователем, читаются другим пользователем, а затем первый пользователь восстанавливает исходные данные.

Неповторяемое чтение и фантомные записи

При повторной выборке данных получается другой результат.

Механизм блокировок

Сериализуемость транзакций

Параллельное выполнение заданного множества транзакций будет верным, если при его выполнении будет получен такой же результат, как и при последовательном выполнении тех же транзакций.

Механизм блокировок

Сериализуемость транзакций

Параллельное выполнение заданного множества транзакций будет верным, если при его выполнении будет получен такой же результат, как и при последовательном выполнении тех же транзакций.

Сериализуемость обеспечивается с помощью **механизма блокировок** и определения **способов взаимодействия** транзакций.

Механизм блокировок

Сериализуемость транзакций

Параллельное выполнение заданного множества транзакций будет верным, если при его выполнении будет получен такой же результат, как и при последовательном выполнении тех же транзакций.

Сериализуемость обеспечивается с помощью **механизма блокировок** и определения **способов взаимодействия** транзакций.

Последовательное выполнение транзакций

Блокировка на уровне всей БД.

Реализация транзакций

Блокировка приводит к остановке транзакции.

Реализация транзакций

Блокировка приводит к остановке транзакции.
Типы блокировок:

- Для чтения — shared-блокировка
- Для записи — exclusive-блокировка

Реализация транзакций

Блокировка приводит к остановке транзакции.
Типы блокировок:

- Для чтения — shared-блокировка
- Для записи — exclusive-блокировка

Уровни блокировок:

- БД
- **Таблиц**
- **Записей**
- Полей

Реализация транзакций

Блокировка приводит к остановке транзакции.
Типы блокировок:

- Для чтения — shared-блокировка
- Для записи — exclusive-блокировка

Уровни блокировок:

- БД
- **Таблиц**
- **Записей**
- Полей

Двухфазный механизм выполнения транзакций:

1. Накопление (захват) блокировок
2. Выполнение операций и освобождение блокировок

Синтаксис настройки транзакций в SQL

Настройка транзакций

```
SET TRANSACTION
    [READ WRITE | READ ONLY]
    [WAIT | NO WAIT]
    [{SNAPSHOT [TABLE STABILITY] |
     READ COMMITTED [[NO] RECORD_VERSION]}]
    [RESERVING <reserving_clause>];
```

Синтаксис настройки транзакций в SQL

Настройка транзакций

```
SET TRANSACTION
    [READ WRITE | READ ONLY]
    [WAIT | NO WAIT]
    [{SNAPSHOT [TABLE STABILITY] |
     READ COMMITTED [[NO] RECORD_VERSION]}]
    [RESERVING <reserving_clause>];
```

Пример

```
SET TRANSACTION READ WRITE WAIT SNAPSHOT;
```

Уровни изоляции транзакций

DIRTY READ (не поддерживается)

Чтение незафиксированных изменений.

Уровни изоляции транзакций

DIRTY READ (не поддерживается)

Чтение незафиксированных изменений.

READ COMMITTED

Чтение всех изменений своей транзакции и зафиксированных изменений других транзакций.

Уровни изоляции транзакций

DIRTY READ (не поддерживается)

Чтение незафиксированных изменений.

READ COMMITTED

Чтение всех изменений своей транзакции и зафиксированных изменений других транзакций.

SNAPSHOT (CONCURRENCY)

Чтение всех изменений своей транзакции. Изменения, сделанные в других транзакциях, недоступны.

Уровни изоляции транзакций

DIRTY READ (не поддерживается)

Чтение незафиксированных изменений.

READ COMMITTED

Чтение всех изменений своей транзакции и зафиксированных изменений других транзакций.

SNAPSHOT (CONCURRENCY)

Чтение всех изменений своей транзакции. Изменения, сделанные в других транзакциях, недоступны.

SNAPSHOT TABLE STABILITY (CONSISTENCY)

Запрещает вносить изменения другим транзакциям.

Остальные параметры

Режим обращения:

- `READ WRITE`
- `READ ONLY`

Остальные параметры

Режим обращения:

- `READ WRITE`
- `READ ONLY`

Режим обработки конфликтов:

- `WAIT`
- `NO WAIT`

Остальные параметры

Режим обращения:

- `READ WRITE`
- `READ ONLY`

Режим обработки конфликтов:

- `WAIT`
- `NO WAIT`

Чтение неподтверждённых версий (только `READ COMMITTED`):

- `RECORD VERSION` — использует последнюю подтверждённую версию записи
- `NO RECORD VERSION` — требует, чтобы все версии записи были подтверждены

Резервирование таблиц

Защита от действий других транзакций

```
<reserving_clause> = table [, table ...]
    [FOR [SHARED | PROTECTED] {READ | WRITE}]
[, <reserving_clause>]
```

[2] http://www.ibase.ru/devinfo/ibtrans.htm

Резервирование таблиц

Защита от действий других транзакций

```
<reserving_clause> = table [, table ...]
    [FOR [SHARED | PROTECTED] {READ | WRITE}]
[, <reserving_clause>]
```

В двух словах [2]:

- SHARED READ — разрешение другим транзакциям читать и изменять данные
- SHARED WRITE — разрешает другим транзакциям READ COMMITTED и CONCURRENCY обновлять данные, в то время как другие транзакции могут читать данные только в режиме READ ONLY
- PROTECTED READ — разрешает другим транзакциям только читать данные
- PROTECTED WRITE — разрешает читать данные только транзакциям READ COMMITTED и CONCURRENCY

[2] http://www.ibase.ru/devinfo/ibtrans.htm

Примеры настроек транзакций

READ ONLY

- READ COMMITTED
- WAIT
- RECORD_VERSION

Примеры настроек транзакций

READ ONLY

- READ COMMITTED
- WAIT
- RECORD_VERSION

Добавление/изменение/удаление

- READ COMMITTED
- NO WAIT/WAIT
- NO RECORD_VERSION/RECORD_VERSION

Примеры настроек транзакций

READ ONLY

- READ COMMITTED
- WAIT
- RECORD_VERSION

Добавление/изменение/удаление

- READ COMMITTED
- NO WAIT/WAIT
- NO RECORD_VERSION/RECORD_VERSION

Длительные операции чтения

- SNAPSHOT
- NO WAIT

Примеры настроек транзакций

READ ONLY

- READ COMMITTED
- WAIT
- RECORD_VERSION

Добавление/изменение/удаление

- READ COMMITTED
- NO WAIT/WAIT
- NO RECORD_VERSION/RECORD_VERSION

Длительные операции чтения

- SNAPSHOT
- NO WAIT

Изменение большого количества данных

- SNAPSHOT TABLE STABILITY
- NO WAIT

Взаимные блокировки

Конфликт блокировок транзакций

Возникает при попытке двух транзакций установить несовместимые блокировки на один объект.

Взаимные блокировки

Конфликт блокировок транзакций

Возникает при попытке двух транзакций установить несовместимые блокировки на один объект.

Взаимная блокировка (Deadlock)

Обычно возникают при попытке двух транзакций установить конфликтующие блокировки на два объекта одновременно.

Взаимные блокировки

Конфликт блокировок транзакций

Возникает при попытке двух транзакций установить несовместимые блокировки на один объект.

Взаимная блокировка (Deadlock)

Обычно возникают при попытке двух транзакций установить конфликтующие блокировки на два объекта одновременно.

Пример взаимной блокировки

- T_1: блокировка X для записи A, блокировка X для записи B
- T_2: блокировка X для записи B, блокировка X для записи A

Взаимные блокировки

Конфликт блокировок транзакций

Возникает при попытке двух транзакций установить несовместимые блокировки на один объект.

Взаимная блокировка (Deadlock)

Обычно возникают при попытке двух транзакций установить конфликтующие блокировки на два объекта одновременно.

Пример взаимной блокировки

- T_1: блокировка X для записи A, блокировка X для записи B
- T_2: блокировка X для записи B, блокировка X для записи A

Разрешение взаимных блокировок

Сервер прекращает одну из транзакций.