



Транзакции

sd@freematiq.com

<https://github.com/freematiq/php>



Соединение с БД

- Открывается на каждый запрос новое
- Отключается после обработки запроса
- Открытое соединение называется Сессия

index.php

SELECT
Обращение к БД

INSERT
Обращение к БД

Сессия БД

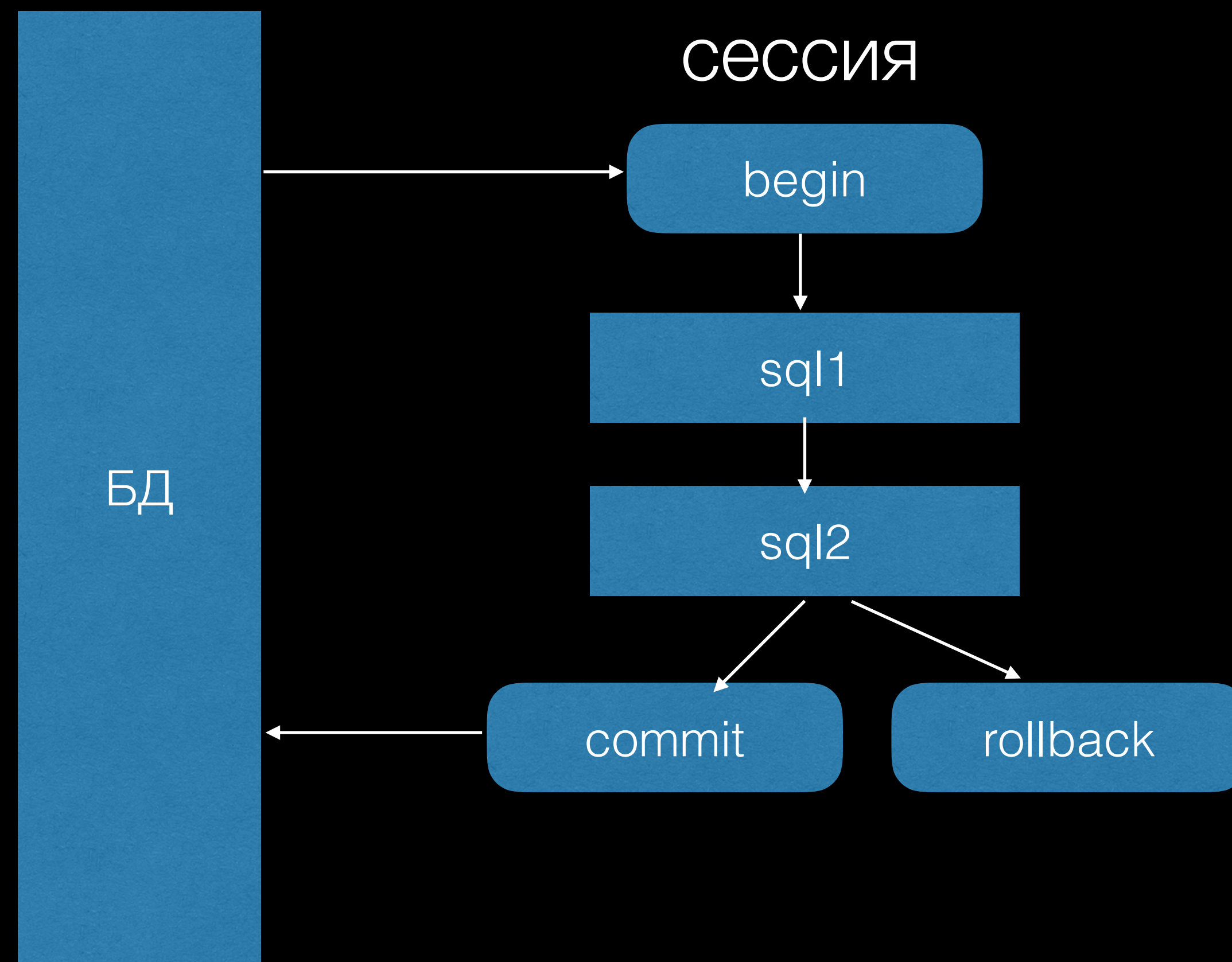



Соединение с БД

- Число активных одновременных соединений ограничено
- Закрывается после завершения работы скрипта

Например в postgresql.conf
`max_connections = 100`

Транзакция





Документация

<http://www.yiiframework.com/doc-2.0/yii-db-transaction.html>

```
// вне блока try catch
$transaction = $connection->beginTransaction();
try {
    $connection->createCommand($sql1)->execute();
    $connection->createCommand($sql2)->execute();
    //.... other SQL executions
    $transaction->commit();
} catch (\Exception $e) {
    $transaction->rollback();
    // throw $e;
} catch (\Throwable $e) {
    $transaction->rollback();
    // throw $e;
}
```

Регистрация

```
public function register()
{
    if (!$this->validate()) {
        return null;
    }

    $user = new User();
    $user->username = $this->username;
    $user->email = $this->email;
    $user->setPassword($this->password);
    $user->generateAuthKey();

    if ($user->save()) {
        $rbac = \Yii::$app->authManager;
        $studentRole = $rbac->getRole('student');
        $rbac->assign($studentRole, $user->id);
        return $user;
    }

    return null;
}
```

```
public function register()
{
    if (!$this->validate()) {
        return null;
    }

    $user = new User();
    $user->username = $this->username;
    $user->email = $this->email;
    $user->setPassword($this->password);
    $user->generateAuthKey();

    * $rbac = \Yii::$app->authManager;
    * $studentRole = $rbac->getRole('student');

    $transaction = \Yii::$app->db->beginTransaction();

    try {
        if ($user->save()) {
            $rbac->assign($studentRole, $user->id);
            $transaction->commit();
            return $user;
        }
    } catch (\Throwable $e) {
        $transaction->rollBack();
    }

    return null;
}
```



Транзакции

- Любой запрос - неявная транзакция
- Защищают текущие данные от изменений (Lock)
- Запоминают новые изменения: не нужно делать транзакции изменяющие “много” данных по сравнению с возможностями БД и hardware
- Могут конфликтовать друг с другом (Deadlock)



ACID

Atomicity (атомарность) — выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.

Consistency (согласованность) — гарантирует, что по мере выполнения транзакций, данные переходят из одного согласованного состояния в другое, то есть транзакция не может разрушить взаимной согласованности данных.

Isolation (изолированность) — локализация пользовательских процессов означает, что конкурирующие за доступ к БД транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит, как будто они выполняются параллельно.

Durability (долговечность) — устойчивость к ошибкам — если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах.



Уровни изоляции транзакций

1. Read uncommitted
2. Read committed
3. Repeatable read
4. Serializable

Влияет на получение/обновление/удаление строк таблицы



АНОМАЛИИ ВСТАВКИ

Потеряное обновление

Проявляется при одновременном изменении одной записи разными транзакциями, когда работа одной транзакции теряется

Транзакция 1	Транзакция 2
UPDATE t SET b = b + 1 WHERE id = 1	UPDATE t SET b = b + 10 WHERE id = 1
SELECT b FROM t WHERE id = 1	SELECT b FROM t WHERE id = 1
b = 2	b = 11

Решение: READ UNCOMMITTED



АНОМАЛИИ ВСТАВКИ

Грязное чтение

Чтение данных из транзакции, которая затем не подтвердится и откатится.

Транзакция 1	Транзакция 2
UPDATE t SET b = b + 1 WHERE id = 1	
	SELECT b FROM t WHERE id = 1

Решение: READ COMMITTED (по умолчанию)



АНОМАЛИИ ВСТАВКИ

Неповторяющееся чтение

Когда запросы одной транзакции в процессе работы возвращают разные результаты из-за изменившихся значений

Транзакция 1	Транзакция 2
	SELECT b FROM t WHERE id = 1
UPDATE t SET b = b + 1 WHERE id = 1; COMMIT	
	SELECT b FROM t WHERE id = 1

Решение: REPEATABLE READ

```
$transaction->setIsolationLevel(Transaction::REPEATABLE_READ);
```



АНОМАЛИИ ВСТАВКИ

Чтение “фантомов”

Когда запросы одной транзакции в процессе работы возвращают разные результаты из-за изменившихся строк (удалены, добавлены)

Транзакция 1	Транзакция 2
	SELECT sum(b) FROM t
INSERT INTO t; COMMIT	
	SELECT sum(b) FROM t

Решение: SERIALIZABLE



УРОВНИ ИЗОЛЯЦИИ

SQL

Уровень изоляции	Фантомное чтение	Неповторяющееся чтение	«Грязное» чтение	Потерянное обновление
SERIALIZABLE	+	+	+	+
REPEATABLE READ	-	+	+	+
READ COMMITTED	-	-	+	+
READ UNCOMMITTED	-	-	-	+
NULL	-	-	-	-

PostgreSQL

<https://www.postgresql.org/docs/current/transaction-iso.html>

Table 13-1. Transaction Isolation Levels

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible



Домашнее задание

- Ознакомиться с материалом о транзакциях
- Использовать для целостности данных