Погружение в СУБД. Сезон 2017

СУБД и приложение. Часть I

Дмитрий Барашев

Computer Science Center

Санкт-Петербург 2017

Сверстано в Папирии

PAPEERIA

papeeria.com

онлайн редактор для LaTeX и Markdown• совместное редактирование в реальном времени • интеграция с Git репозиториями • графики

подсветка синтаксиса • автодополнение • проверка орфографии • предпросмотр математических формул • галерея шаблонов

Информационная система

Информационная система

СУБД + приложения

Информационная система

- выполняет свою функцию;

Информационная система

- выполняет свою функцию;
- работает эффективно;

Информационная система

- выполняет свою функцию;
- работает эффективно;
- соблюдает политику безопасности;

Информационная система

- выполняет свою функцию;
- работает эффективно;
- соблюдает политику безопасности;

- является поддерживаемой;

Информационная система

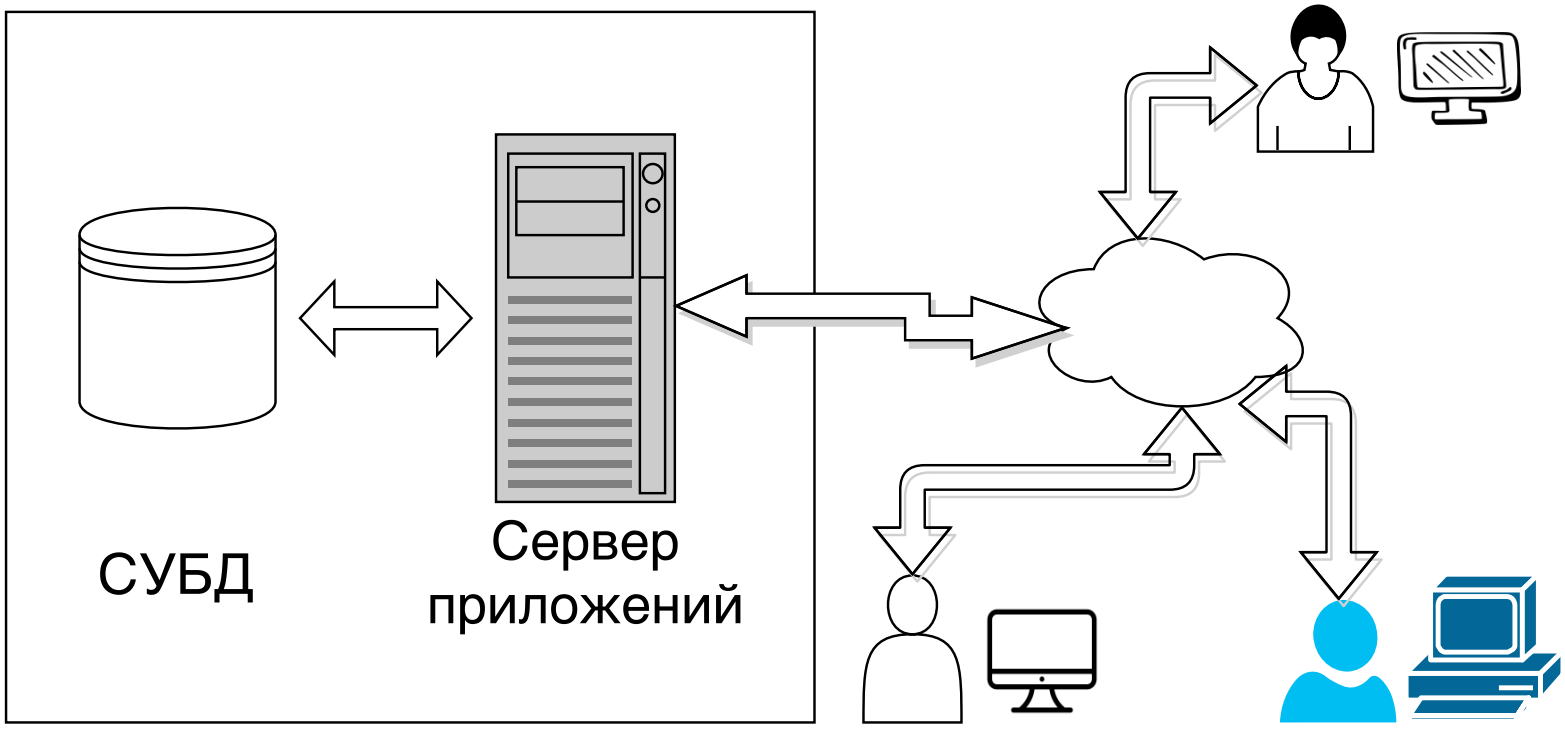
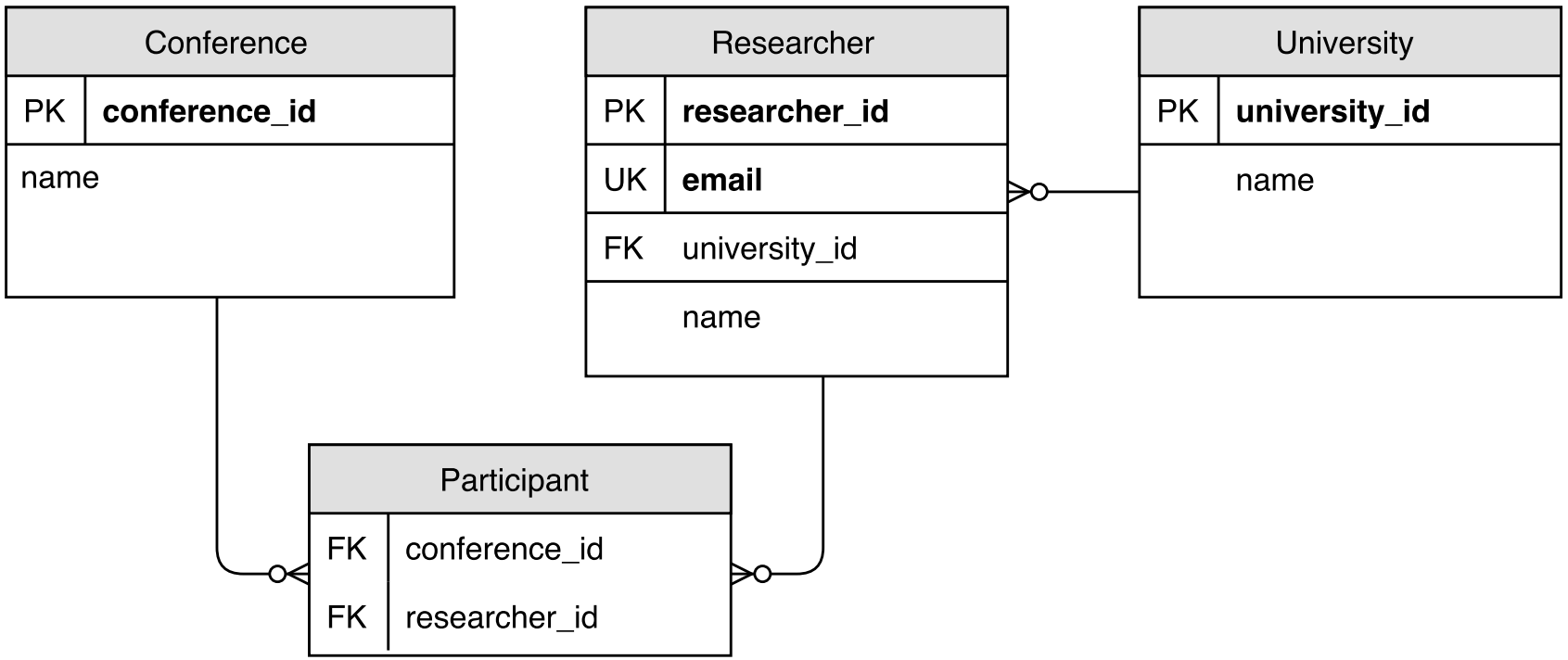


Схема БД


Conference
PK
conference_id
name
Researcher
PK
researcher_id
UK
email
FK
university_id
name
University
PK
university_id
name
Participant
FK
conference_id
FK
researcher_id

Задача

Для каждой пары

$$(конференция, университет)$$

найти суммарное количество исследователей из данного университета, участвовавших в данной конференции.

Количество записей в таблицах

- 200 записей в `University`
- 1000 записей в `Conference`
- 20000 записей в `Researcher`
- 100000 записей в `Participant`

Возможное решение

не очень эффективное

```
# db.execute выполняет запрос и возвращает результат
# в виде массива словарей
conferences = db.execute('SELECT * FROM Conference')
for conf in conferences:
  selectParts = '''
    SELECT * FROM Participant
    WHERE conference_id = ''' + conf['conference_id']
  for p in db.execute(selectParts):
    researcher = db.execute('''
      SELECT * FROM Researcher
      WHERE researcher_id=''' + p['researcher_id'])
    uni_id = researcher['university_id']
    uni = db.execute('''
      SELECT * FROM University
      WHERE university_id=''' + uni_id)
    # Инкрементируем счетчик
    inc(conf, uni)
```

Жизнь запроса

Жизнь запроса

- Приложение посылает текст серверу БД

Жизнь запроса

- Сервер делает синтаксический разбор запроса

Жизнь запроса

- Проверяет использованные имена

Жизнь запроса

- Составляет план выполнения запроса

Жизнь запроса

- Выполняет запрос

Жизнь запроса

- Формирует результат и отсылает приложению

План запроса и его выполнение

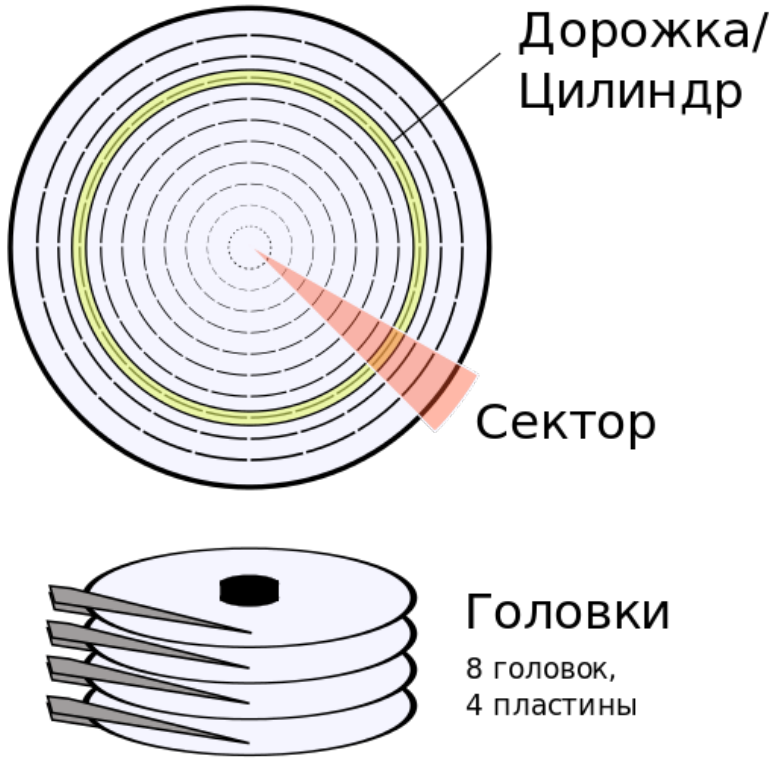
- План включает в себя инструкции по физическому выполнению запроса
- Простейший план: последовательно просмотреть всю таблицу

План запроса и его выполнение

- Таблицы хранятся на дисковых страницах размером 4-8 Кб
- Время чтения одной случайной страницы с диска $\approx$ 5 мс
- Скорость последовательного чтения с диска $\approx$ 200-300 Мб/сек

Устройство жёсткого диска

Устройство жёсткого диска



Стоимость чтения одного сектора

Стоимость чтения
=
стоимость позиционирования (2-10 ms)
+
стоимость вращения (< 10 ms)

Стоимость просмотра одной таблицы

- В таблице `Participant` 2-3 мегабайта и примерно 200-300 страниц

Стоимость просмотра одной таблицы

- Если бы страницы читались только с диска то один просмотр занял бы 1-2 секунды

Стоимость просмотра одной таблицы

- 1000 просмотров выполнились бы за 1000 секунд

Стоимость просмотра одной таблицы

- В самом хорошем случае за 10 секунд

Кеш страниц

- СУБД держит недавно использованные страницы в кеше в оперативной памяти

Кеш страниц

- Наша БД скорее всего целиком помещается в память

Схема итераций

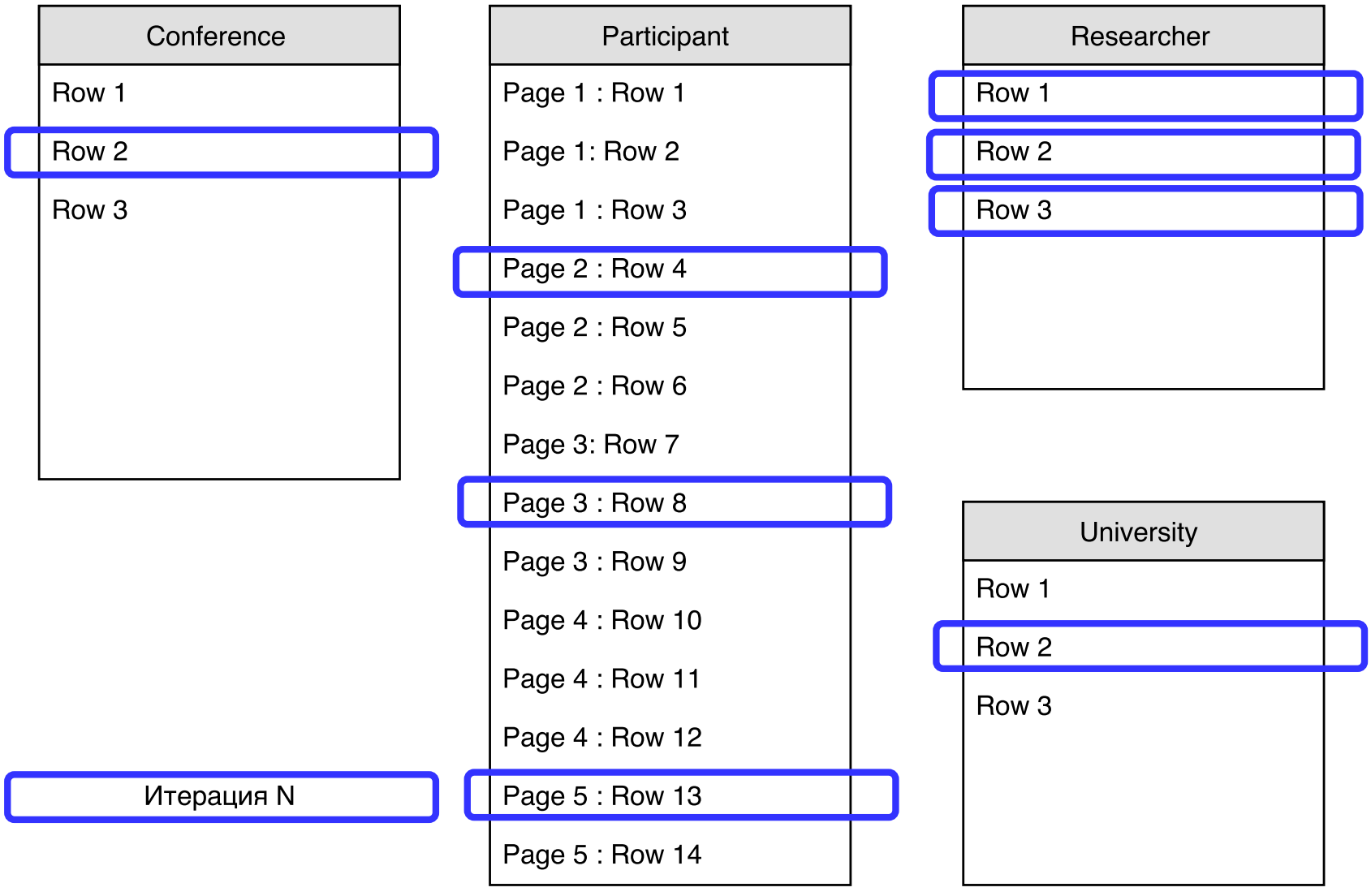
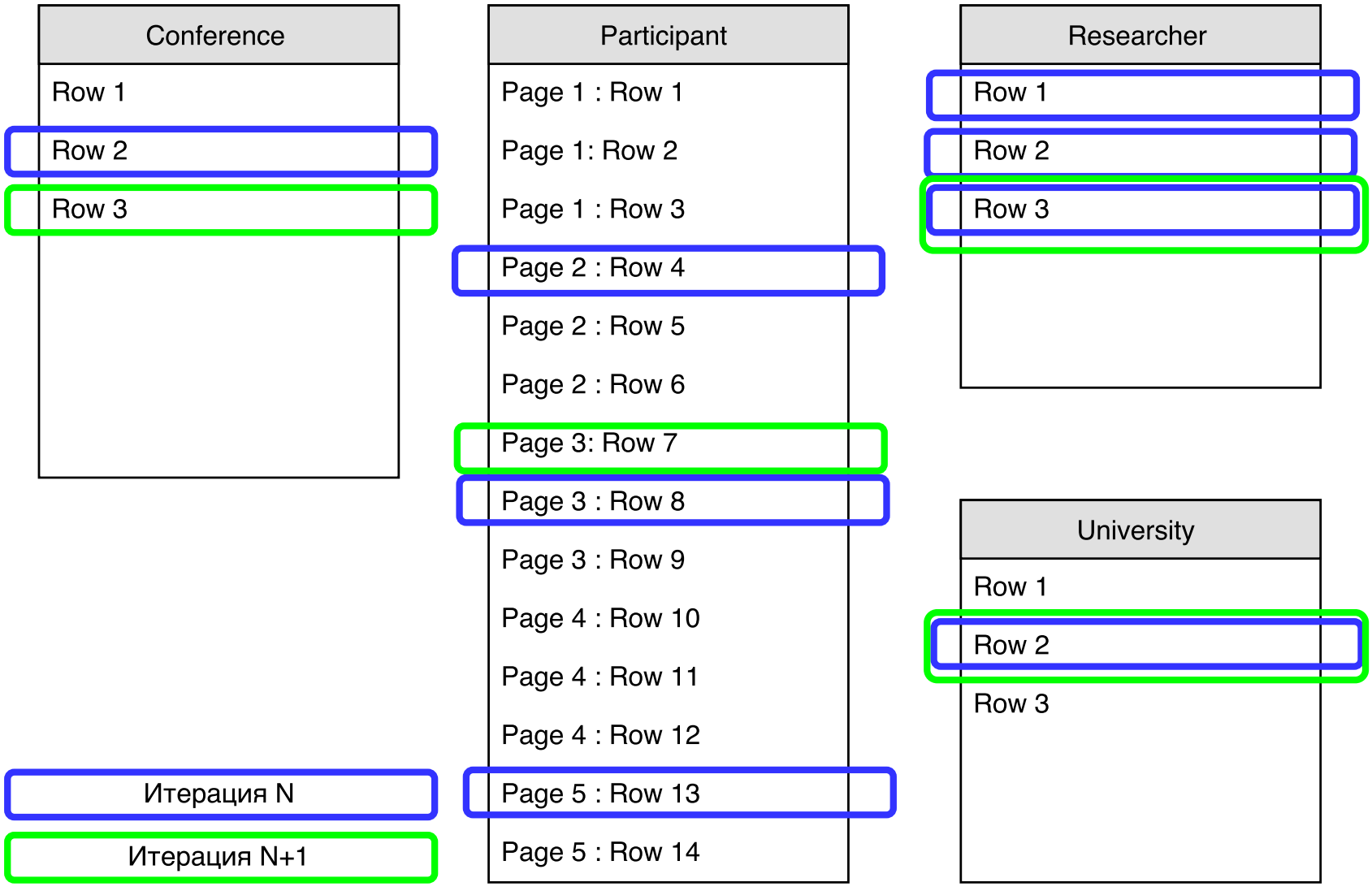


Схема итераций



Возможное решение

более эффективное

```
# Выполняем несколько соединений и итерируемся по результату
pairs = db.execute('''
    SELECT Conference.name, University.name
    FROM Conference JOIN Participant ON (Conference.conference
    JOIN Researcher ON (Participant.researcher_id = Researcher
    JOIN University ON (Researcher.university_id = University.
    ''')
for pair in pairs:
   inc(pair['Conference.name'], pair['University.name'])
```

JOIN: операция соединения

- Бинарная операция, работает с двумя таблицами T_1 и T_2
- Из всех возможных пар строк

$$(t_1, t_2) : t_1 \in T_1, t_2 \in T_2$$

оставляет те, в которых выполнилось указанное условие

JOIN: SQL синтаксис

```
SELECT * FROM T1 JOIN T2 ON(condition)
```

Пример соединения

Fibonacci

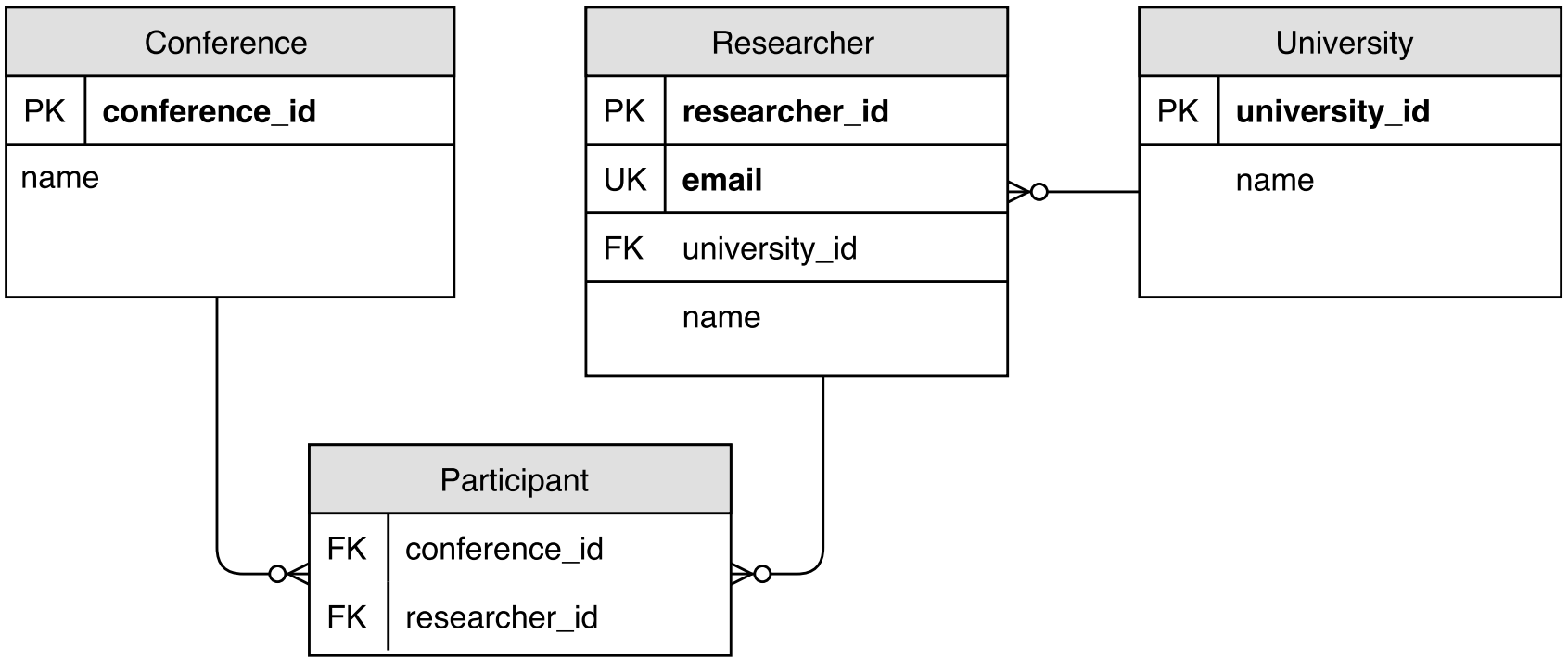
num	value
1	1
2	1
3	2
4	3
5	5

Catalan

num	value
4	5
2	1
3	2
1	1

```
SELECT * FROM Fibonacci JOIN Catalan
ON (Fibonacci.num = Catalan.num)
```

Схема БД


Conference
PK
conference_id
name
Researcher
PK
researcher_id
UK
email
FK
university_id
name
University
PK
university_id
name
Participant
FK
conference_id
FK
researcher_id

Задача

Для каждой пары

$$(конференция, университет)$$

найти суммарное количество исследователей из данного университета, участвовавших в данной конференции.

Количество записей в таблицах

- 200 записей в `University`
- 1000 записей в `Conference`
- 20000 записей в `Researcher`
- 100000 записей в `Participant`

Возможное решение

более эффективное

```
result = defaultdict(int)
def inc(c_name, u_name):
  result[(c_name, u_name)] += 1

# Выполняем несколько соединений и итерируемся по результату
pairs = db.execute('''
    SELECT Conference.name, University.name
    FROM Conference JOIN Participant ON
      (Conference.conference_id = Participant.conference_id)
    JOIN Researcher ON
      (Participant.researcher_id = Researcher.researcher_id)
    JOIN University ON
      (Researcher.university_id = University.university_id)
    ''')
for pair in pairs:
   inc(pair['Conference.name'], pair['University.name'])
```

Что сделала СУБД

```
EXPLAIN ANALYZE
    SELECT Conference.name, University.name
    FROM Conference JOIN Participant ON
      (Conference.conference_id = Participant.conference_id)
    JOIN Researcher ON
      (Participant.researcher_id = Researcher.researcher_id)
    JOIN University ON
      (Researcher.university_id = University.university_id)
```

Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

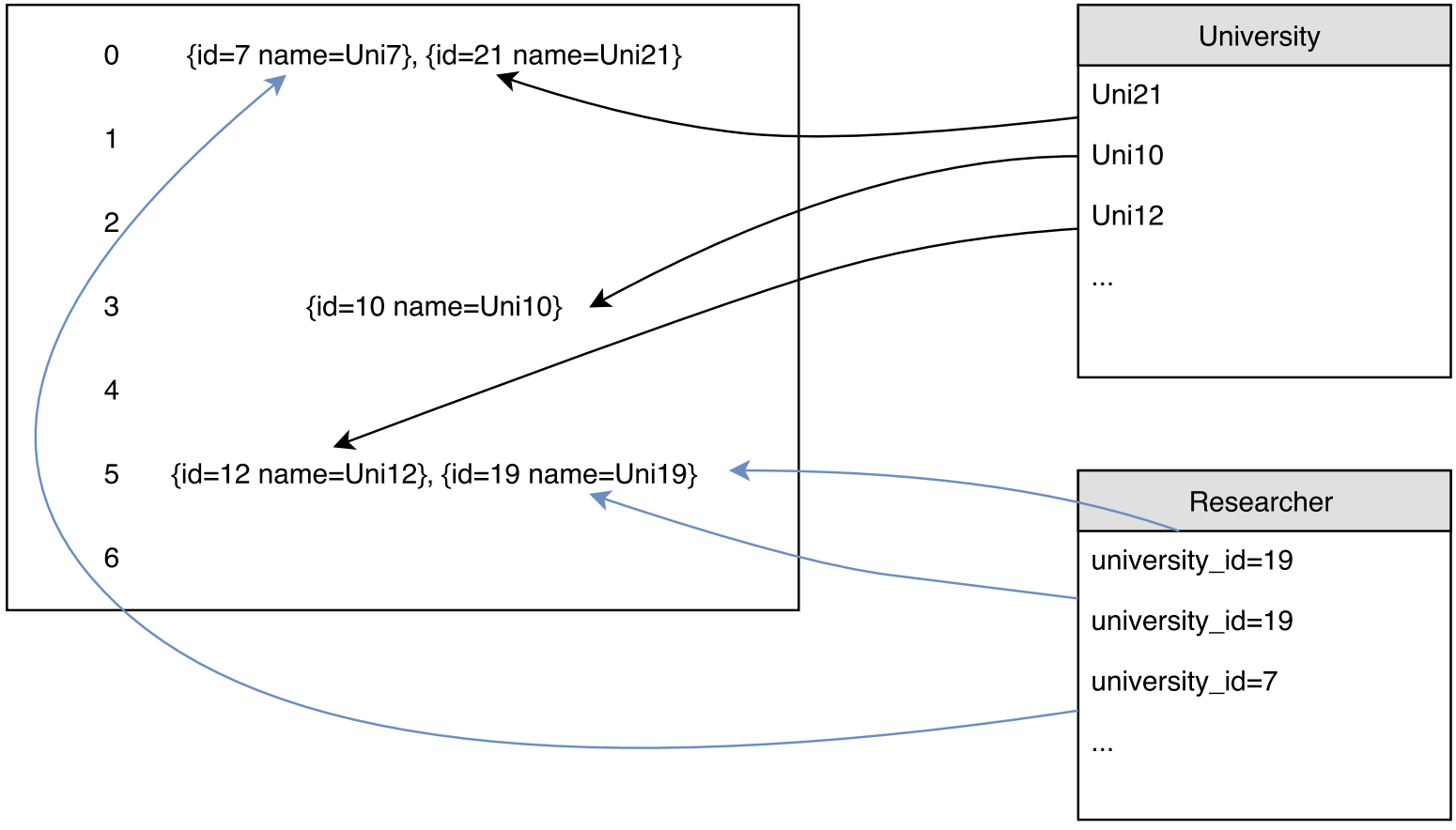
Что сделала СУБД

```
Hash Join
  Hash Cond: (p.conference_id = c.conference_id)
  ->  Hash Join
        Hash Cond: (p.researcher_id = r.researcher_id)
        ->  Seq Scan on Participant p
        ->  Hash
              ->  Hash Join
                    Hash Cond: (
                        r.university_id = u.university_id
                    )
                    ->  Seq Scan on Researcher r
                    ->  Hash
                          ->  Seq Scan on University u
  ->  Hash
        ->  Seq Scan on Conference c
Planning time: 1.033 ms
Execution time: 93.119 ms
```

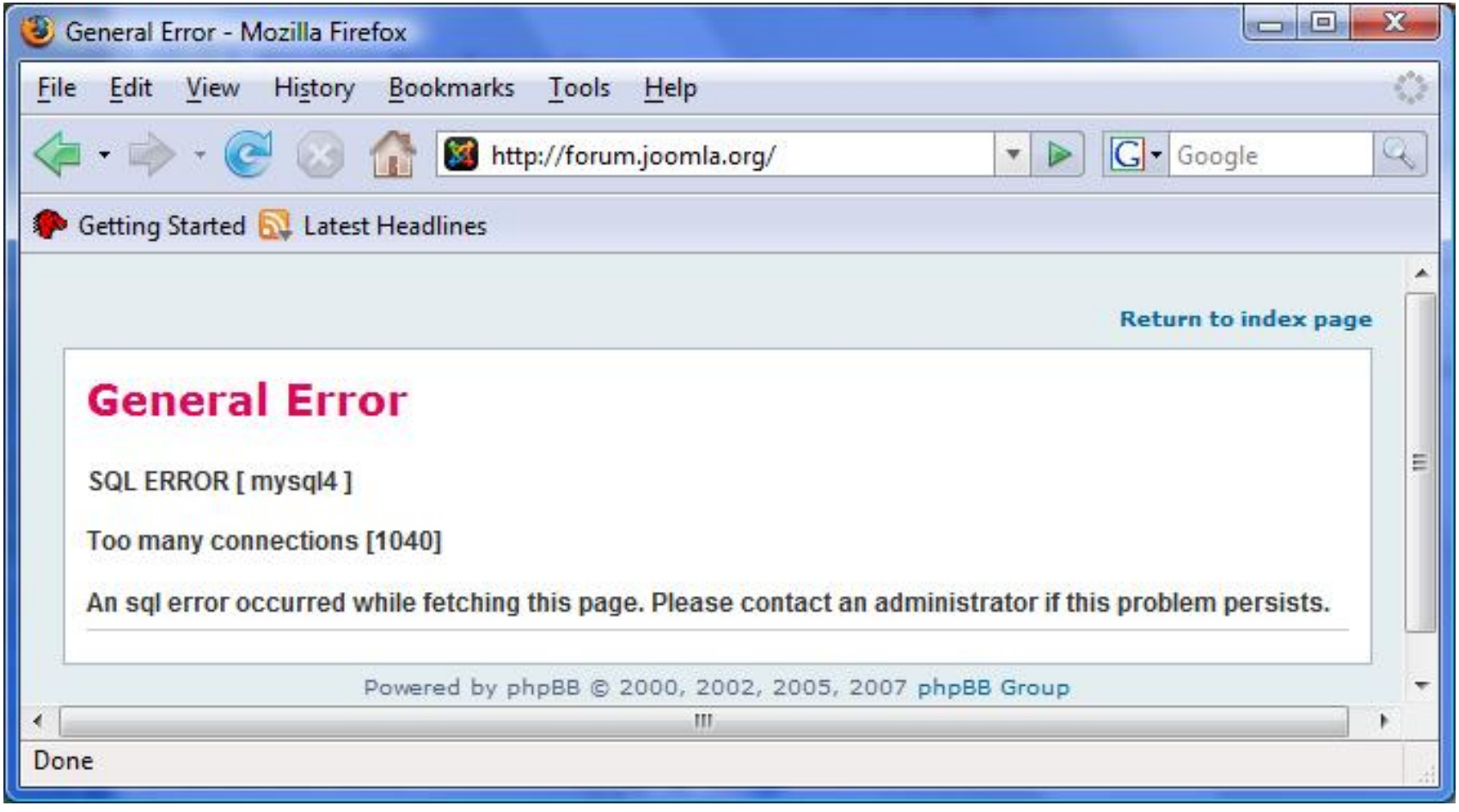
Hash Join

Выполняет соединение $R \bowtie S$ за $B(R + S)$ операций чтения диска, где $B(R + S)$ – суммарное количество дисковых страниц в обеих таблицах, при условии что одна из таблиц помещается в памяти

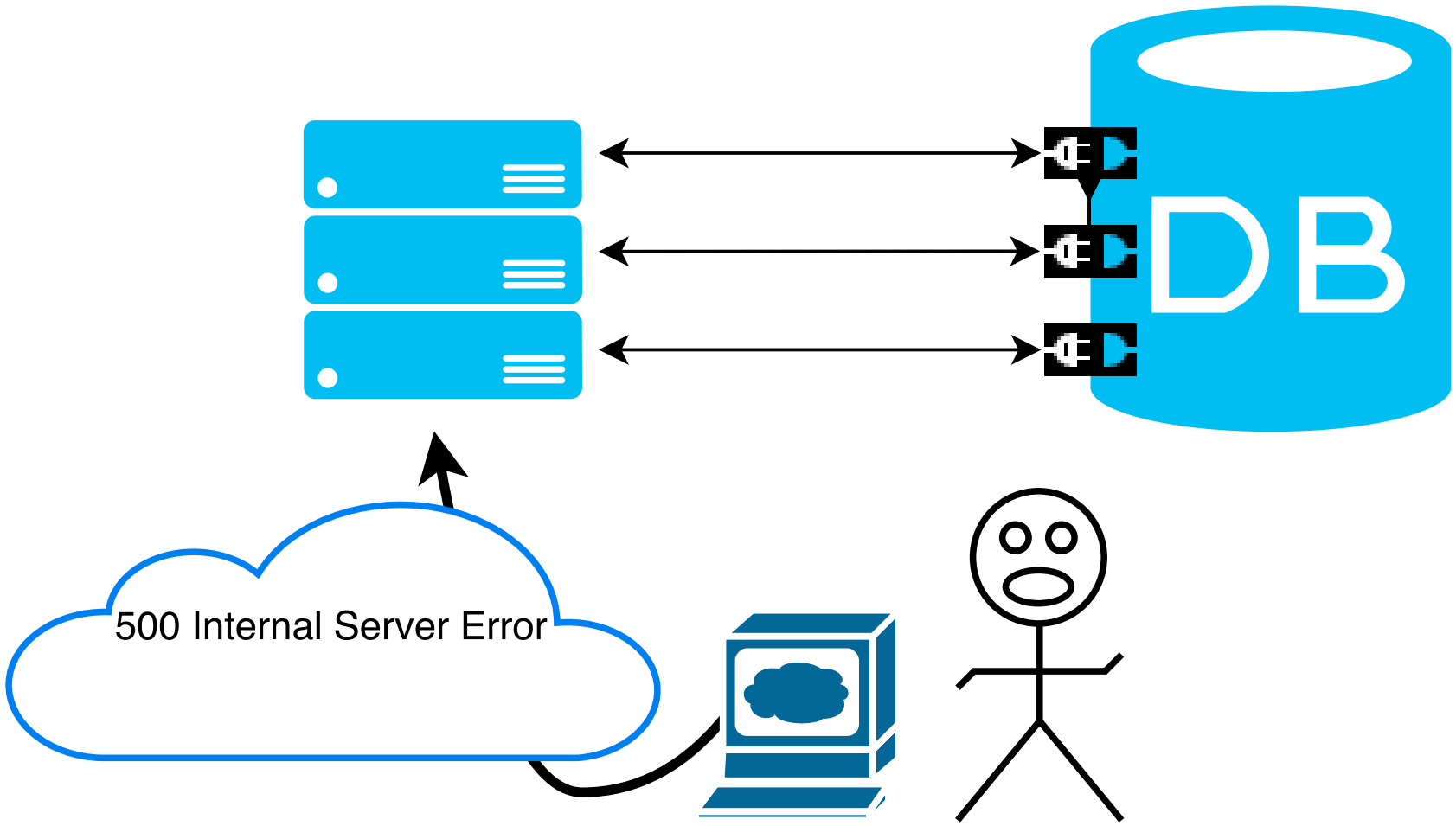
Hash Join


0
{id=7 name=Uni7}, {id=21 name=Uni21}
1
2
3
{id=10 name=Uni10}
4
5
{id=12 name=Uni12}, {id=19 name=Uni19}
6
University
Uni21
Uni10
Uni12
...
Researcher
university_id=19
university_id=19
university_id=7
...

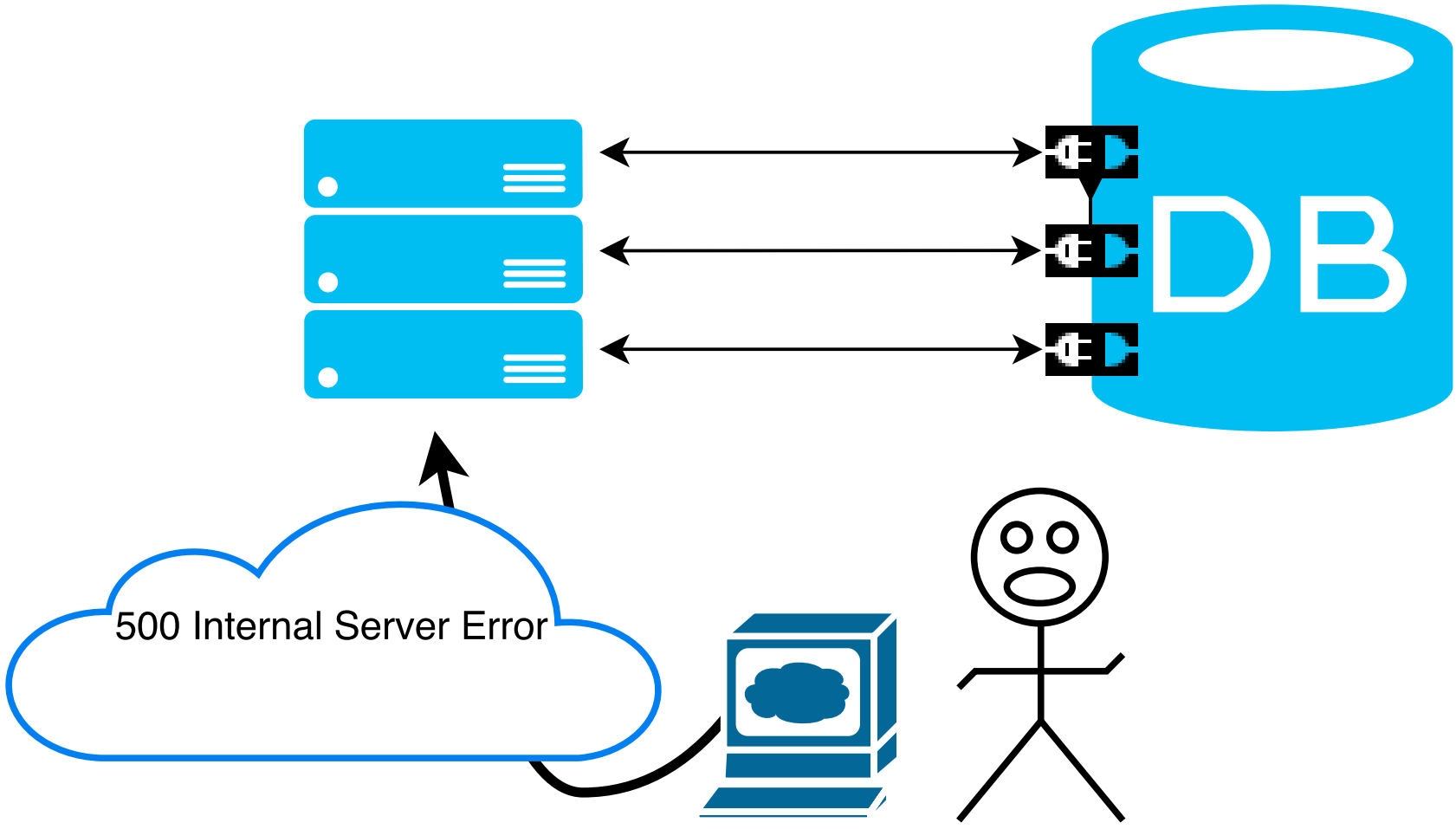
Server Error



Соединение с СУБД

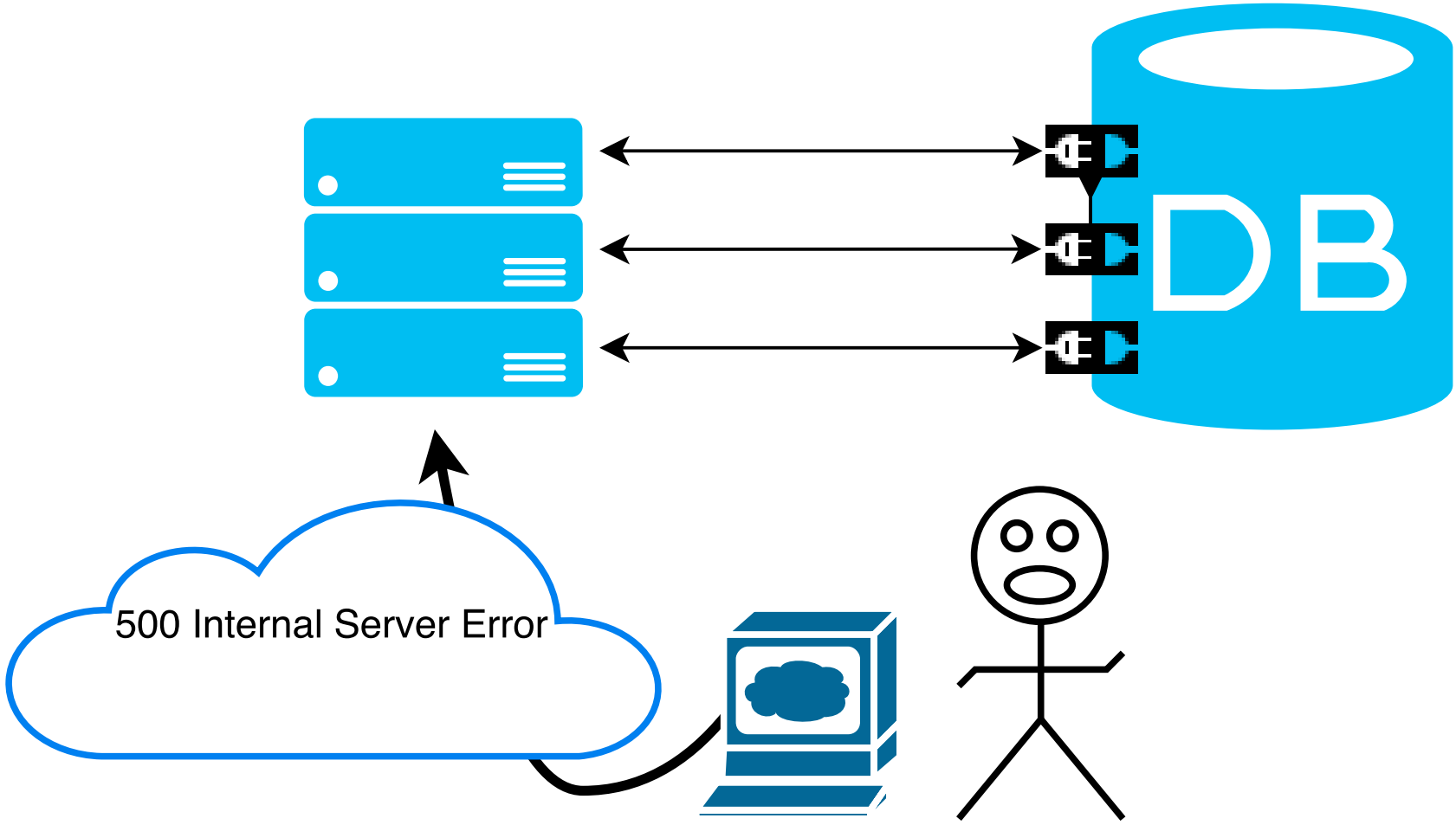


Соединение с СУБД



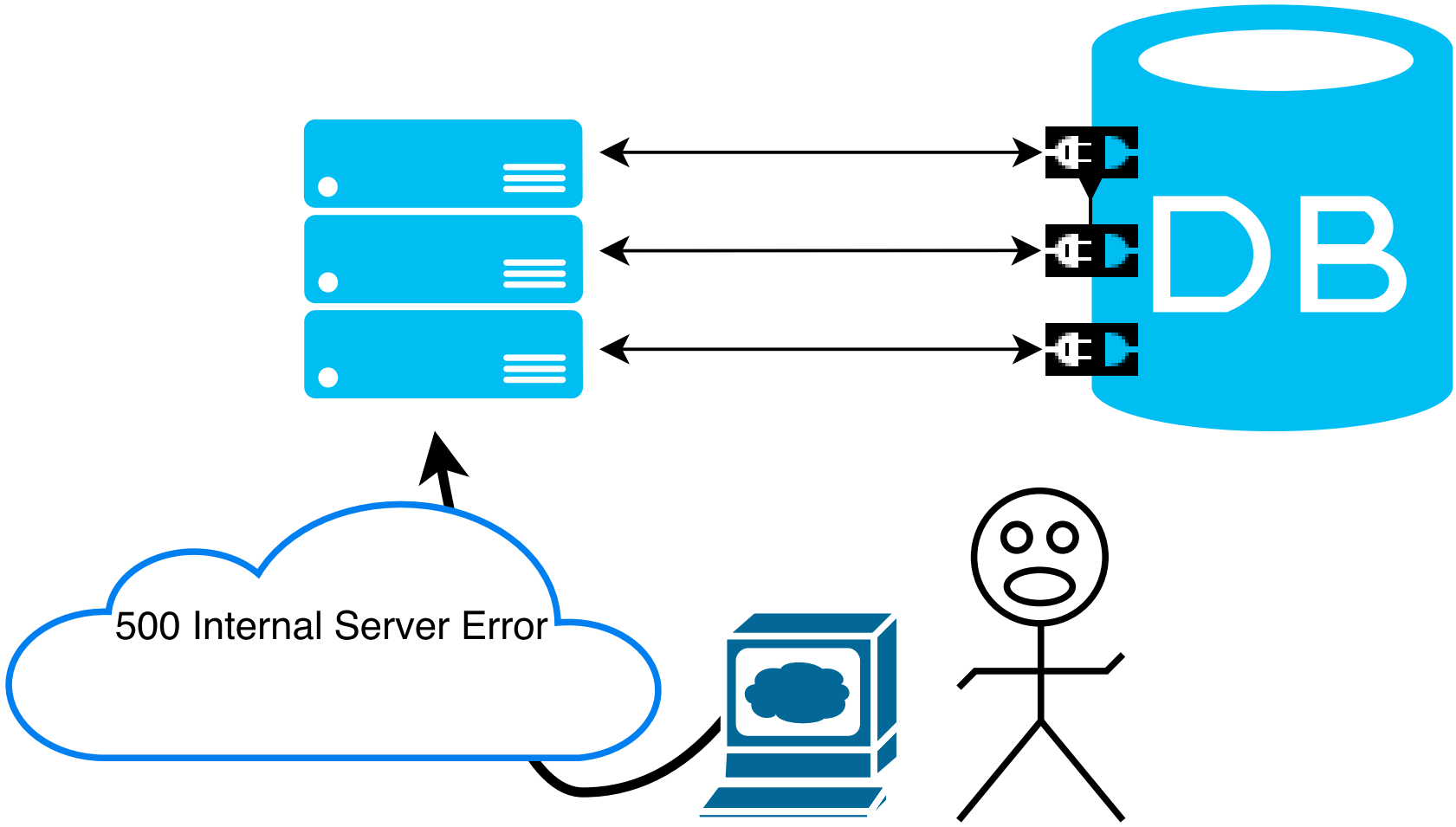
Connection: соединение, подключение, коннекция

Соединение с СУБД



Session: сессия, сеанс

Соединение с СУБД



Бывает подключение == сессия, бывает много сессий на одно подключение

Сессии: как быть?

Сессии: как быть?

- Закрывать сессии

Сессии: как быть?

- Закрывать сессии
- Использовать пул сессий (session pool, connection pool)

Сессии: как быть?

- Закрывать сессии
- Использовать пул сессий (session pool, connection pool)
- Использовать контекстные объекты

Закрывать сессии

Python	`try-finally, with`
Java	`try-finally`, try with resources
C++	деструкторы, умные указатели

Открытие сессии – дорогая операция

Несколько объектов, одна сессия

Открытие сессии – дорогая операция

СУБД и приложение

Не перекладывай работу СУБД на приложение

СУБД и приложение

Закрывай сессии

СУБД и приложение

Пользуйся пулом сессий

СУБД и приложение

Параметризуй контекстом объекты доступа к данным