Погружение в СУБД. Сезон 2017

СУБД и приложение. Часть II

Дмитрий Барашев

Computer Science Center

Санкт-Петербург 2017

Сверстано в Папирии

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

Устойчивость приложения к изменениям

Когда-то в Древней Греции

– Всё течёт, всё изменяется!

Всё течёт, всё изменяется

- Меняются требования к приложению

Всё течёт, всё изменяется

- Меняются требования к приложению
- Меняется производитель СУБД

Всё течёт, всё изменяется

- Меняются требования к приложению
- Меняется производитель СУБД
- Появляются новые технологии

Неизменно одно

– Я не жадный, я домовитый!

В этом модуле

- Представления

В этом модуле

- Представления
- Хранимые процедуры

В этом модуле

- Представления
- Хранимые процедуры
- Абстракции в приложении

Формулировка проблемы

Информационная система

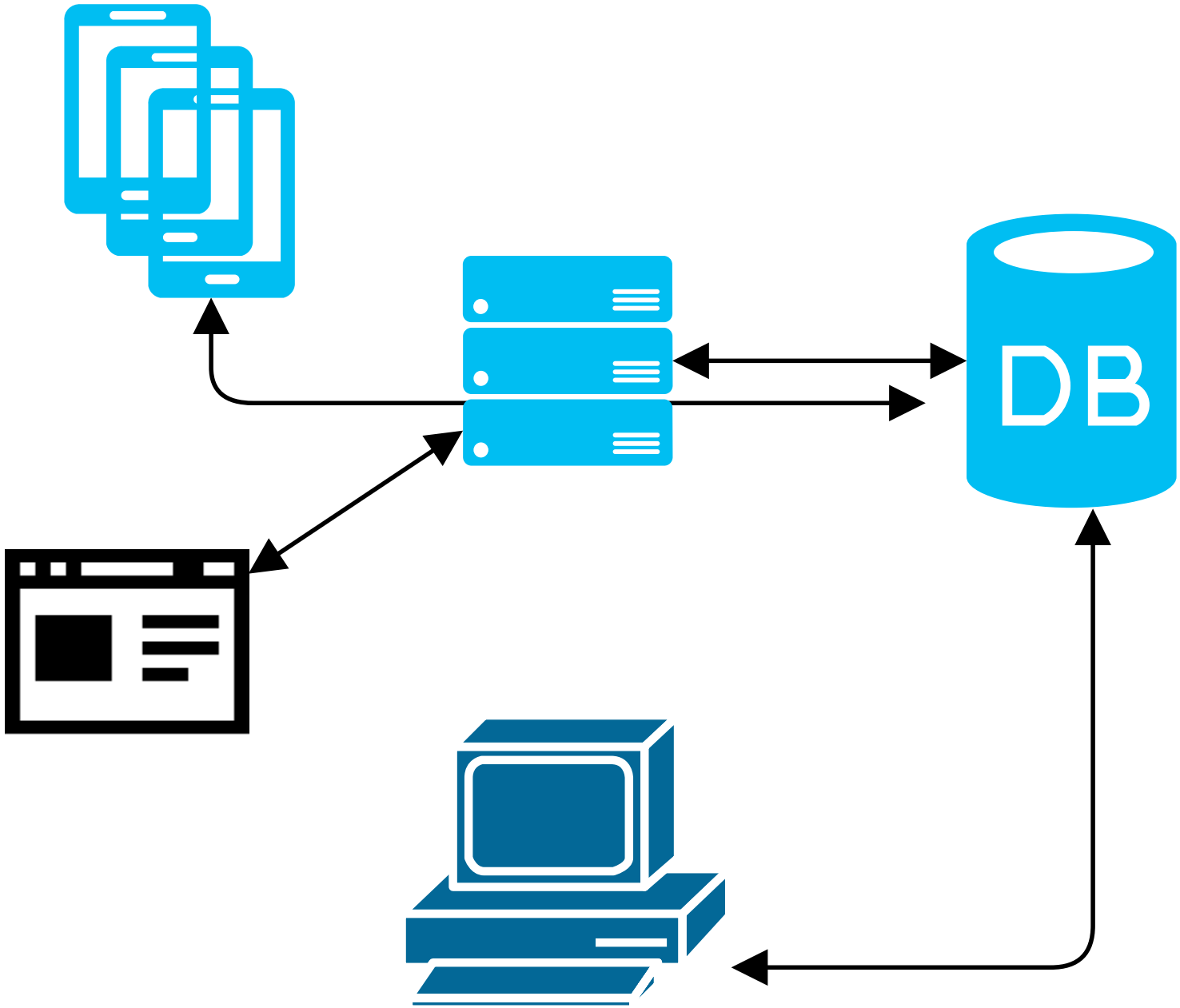


Схема БД

```
CREATE TABLE Conference(
  id SERIAL PRIMARY KEY,
  name TEXT UNIQUE);


CREATE TABLE Paper(
  id SERIAL PRIMARY KEY,
  title TEXT,
  conference_id INT REFERENCES Conference,
  keywords TEXT[],
  accepted BOOLEAN);
```

Код приложения

отображение таблицы

```
def print_papers(conference, out):
  db.execute('''
    SELECT title, keywords, name AS conf_name, accepted
    FROM Paper JOIN Conference
      ON(Paper.conference_id = Conference.id)
    WHERE Conference.name=?
    ''', conference)
  for paper in db.fetchall():
    out.add('''
    <tr>
      <td>{}</td><td>{}</td><td>{}</td><td>{}</td>
    </tr>
    '''.format(
      paper.title, ','.join(paper.keywords),
      paper.conf_name, paper.accepted
    )
```

Эпоха перемен

Хотим каталог ключевых слов

Окей, каталог...

```
CREATE TABLE Conference(
  id SERIAL PRIMARY KEY,
  name TEXT UNIQUE);

CREATE TABLE Keyword(
  id SERIAL PRIMARY KEY, value TEXT UNIQUE);

CREATE TABLE Paper(
  id SERIAL PRIMARY KEY,
  title TEXT,
  conference_id INT REFERENCES Conference,
  accepted BOOLEAN);

CREATE TABLE PaperKeyword(
  paper_id INT REFERENCES Paper,
  keyword_id INT REFERENCES Keyword);
```

Эпоха перемен

Хотим подавать статью на разные конференции

Окей, много конференций...

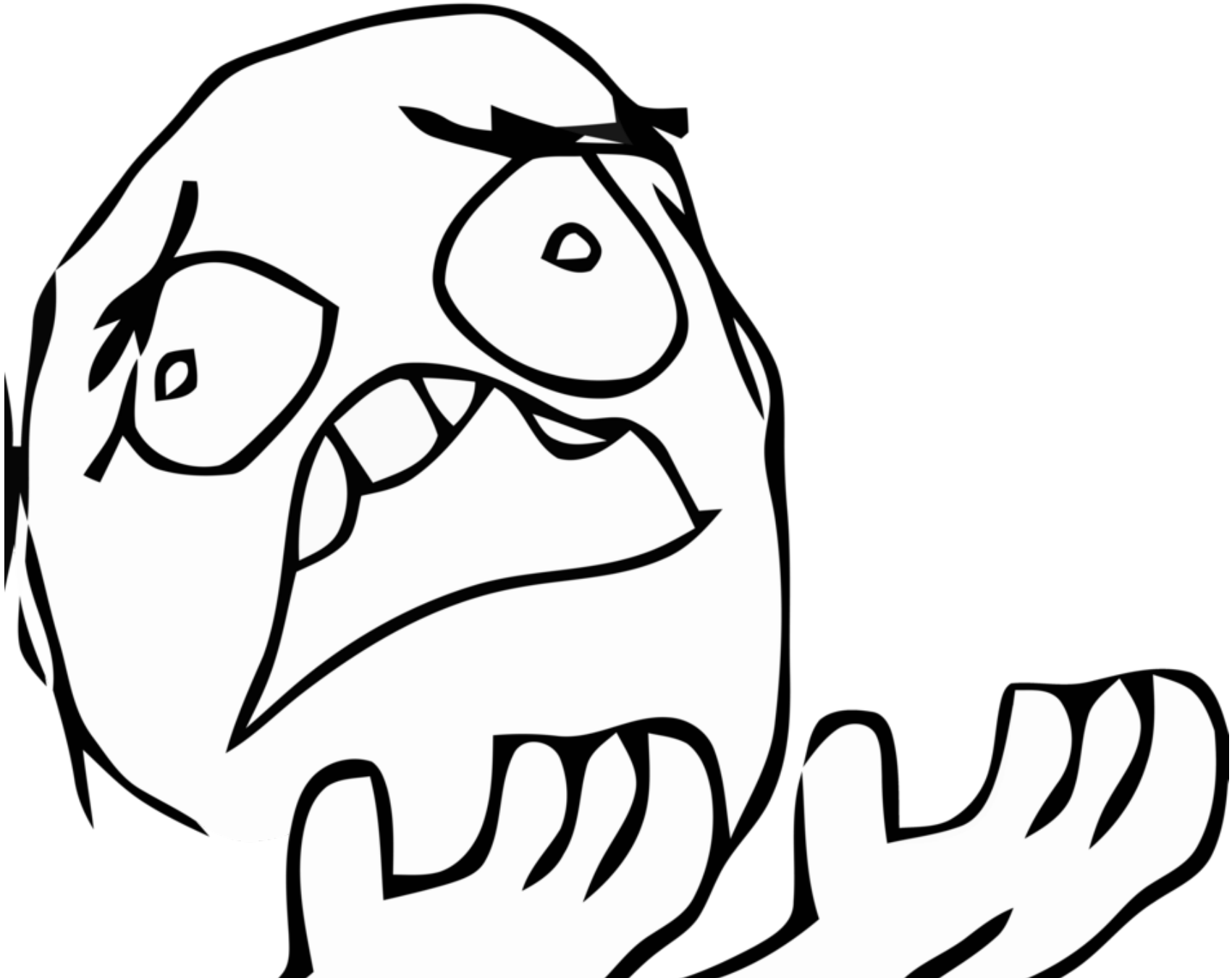
```
CREATE TABLE Conference(
  id SERIAL PRIMARY KEY, name TEXT UNIQUE);

CREATE TABLE Keyword(
  id SERIAL PRIMARY KEY, value TEXT UNIQUE);

CREATE TABLE Paper(
  id SERIAL PRIMARY KEY, title TEXT);

CREATE TABLE PaperKeyword(
  paper_id INT REFERENCES Paper,
  keyword_id INT REFERENCES Keyword);

CREATE TABLE PaperConference(
  paper_id INT REFERENCES Paper,
  conference_id INT REFERENCES Conference,
  accepted BOOLEAN,
  UNIQUE(paper_id, conference_id));
```

Эпоха перемен

Хотим, чтобы рецензенты ставили оценки.

Статья принята, если средняя оценка больше 3

Oooo...



Код приложения

добавили каталог ключевых слов

```
def print_papers(conference, out):
  db.execute('''
    SELECT title, array_agg(K.value),
           C.name AS conf_name, P.accepted
    FROM Paper P
    JOIN Conference C ON(P.conference_id = C.id)
    JOIN PaperKeyword PK ON (P.id = PK.paper_id)
    JOIN Keyword K ON (PK.keyword_id = K.id)
    WHERE C.name=?
    GROUP BY P.id, C.id
    ''', conference)

  print_table(db.fetchall())
```

Код приложения

статья на несколько конференций

```
def print_papers(conference, out):
  db.execute('''
    SELECT title, array_agg(K.value),
           C.name AS conf_name, PC.accepted
    FROM Paper P
    JOIN PaperConference PC ON (PC.paper_id = P.id)
    JOIN Conference C ON(PC.conference_id = C.id)
    JOIN PaperKeyword PK ON (P.id = PK.paper_id)
    JOIN Keyword K ON (PK.keyword_id = K.id)
    WHERE C.name=?
    GROUP BY P.id, C.id, PC.accepted
    ''', conference)

  print_table(db.fetchall())
```

Представление

Создание

```
CREATE VIEW PaperSubmission AS
    SELECT title, keywords,
           name AS conf_name, accepted
    FROM Paper JOIN Conference
      ON(Paper.conference_id = Conference.id);
```

Использование

```
SELECT * FROM PaperSubmission
WHERE conf_name = 'SIGMOD''15';
```

Функции представлений

Функции представлений

- Настройка схемы БД для разных ролей пользователей

Функции представлений

- Сокращение текста запросов

Функции представлений

- Разграничение прав доступа

Функции представлений

- Устойчивость приложений к изменениям

Представление

добавили каталог ключевых слов

```
CREATE VIEW PaperSubmission AS
    SELECT title,
           array_agg(K.value) AS keywords,
           C.name AS conf_name,
           P.accepted
    FROM Paper P
    JOIN Conference C ON(P.conference_id = C.id)
    JOIN PaperKeyword PK ON (P.id = PK.paper_id)
    JOIN Keyword K ON (PK.keyword_id = K.id)
    GROUP BY P.id, C.id;
```

Представление

статья на несколько конференций

```
CREATE VIEW PaperSubmission AS
    SELECT title,
           array_agg(K.value) AS keywords,
           C.name AS conf_name,
           PC.accepted
    FROM Paper P
    JOIN PaperConference PC ON (PC.paper_id = P.id)
    JOIN Conference C ON(PC.conference_id = C.id)
    JOIN PaperKeyword PK ON (P.id = PK.paper_id)
    JOIN Keyword K ON (PK.keyword_id = K.id)
    GROUP BY P.id, C.id, PC.accepted;
```

Устойчивость операций записи

Типичный спор о ХП

Код приложения

добавление статьи, изменение ключевых слов

```
class SubmitPaperHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    title, conference_id, keywords = self.parseQueryArg()
    db = pool.getconn()
    db.execute('''
      INSERT INTO Paper(title, conference_id, keywords)
      VALUES(?, ?, ?)
    ''', (title, conference_id, keywords))

class UpdateKeywordsHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    paper_id, keywords = self.parseQueryArg()
    db = pool.getconn()
    db.execute('''
      UPDATE Paper SET keywords=?
      WHERE id=?
    ''', (keywords, paper_id))
```

Эпоха перемен

добавление статьи

```
class SubmitPaperHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    title, conference_id, keywords = self.parseQueryArg()
    db = pool.getconn()
    paper_id = db.execute('''
      INSERT INTO Paper(title) VALUES(?) RETURNING id
    ''', title).fetchone()[0]
    db.execute('''
      INSERT INTO PaperConference(paper_id, conference_id)
      VALUES(?, ?)
    ''', (paper_id, conference_id))
    for k in keywords:
      keyword_id = db.execute('''
        SELECT id FROM Keyword WHERE value=?''', k)
        .fetchone()[0]
      db.execute('''
        INSERT INTO PaperKeyword(paper_id, keyword_id)
        VALUES(?,?)
      ''', (paper_id, keyword_id))
```

Эпоха перемен

изменение ключевых слов

```
class UpdateKeywordsHandler(BaseHTTPRequestHandler):
  def do_GET(self):
    paper_id, keywords = self.parseQueryArg()
    db = pool.getconn()
    db.execute('''
      DELETE FROM PaperKeyword WHERE paper_id=?''', paper_id)
    for k in keywords:
      keyword_id = db.execute('''
        SELECT id FROM Keyword WHERE value=?''', k)
        .fetchone()[0]
      db.execute('''
        INSERT INTO PaperKeyword(paper_id, keyword_id)
        VALUES(?,?)
      ''', (paper_id, keyword_id))
```

Интерфейс операций

```
SubmitPaper(
  title TEXT, conference_id INT, keywords TEXT[]);

UpdateKeywords(
  paper_id INT, keywords TEXT[]);
```

Хранимые процедуры

Хранимые процедуры

- SQL + императивные конструкции
  переменные, условные операторы, циклы, вот это вот всё

Хранимые процедуры

- Хранятся в базе данных и выполняются ядром СУБД

Хранимые процедуры

- Могут принимать аргументы и возвращать значения

Хранимые процедуры

- Могут быть вызваны из приложения через стандартизированный API

Процедура SubmitPaper

```
CREATE OR REPLACE FUNCTION SubmitPaper(
 _title TEXT, _conference_id INT, _keywords TEXT[])
 RETURNS VOID AS $$
DECLARE
  _paper_id INT;
  k TEXT;
BEGIN
    INSERT INTO Paper(title) VALUES(_title)
      RETURNING id INTO _paper_id;
    INSERT INTO PaperConference(paper_id, conference_id)
      VALUES (_paper_id, _conference_id);
    FOREACH k IN ARRAY _keywords LOOP
      INSERT INTO PaperKeyword(paper_id, keyword_id)
      SELECT _paper_id, Keyword.id
      FROM Keyword
      WHERE value = k;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Хранимые процедуры. Плюсы.

- Нет накладных расходов на передачу данных по сети

Хранимые процедуры. Плюсы.

- Возможен более изощренный контроль над правами доступа

Хранимые процедуры. Плюсы.

- Код и схема БД рядом $\Rightarrow$ есть шансы, что изменения будут синхронизированы

Хранимые процедуры. Плюсы.

- Приложение получает интерфейс для действий над данными

Процедура SubmitPaper

вызов из Java

```
import java.sql.*;

public class  Module4 {
  public static void main(String[] args) throws Exception {
    Class.forName("org.postgresql.Driver");
    try (Connection db = DriverManager.getConnection(
        "jdbc:postgresql://localhost:5432/db", "db", "q")) {
      try (CallableStatement s = db.prepareCall(
          "{call SubmitPaper(?, ?, ?)}")) {
        s.setString(1, "Lorem ipsum");
        s.setInt(2, 1);
        s.setArray(3, db.createArrayOf("text",
            new String[] {"transactions", "indexes"}));
        s.execute();
      }
    }
  }
}
```

Хранимые процедуры. Минусы.

- Синтаксис и поведение плохо стандартизированы

Хранимые процедуры. Минусы.

- Отладка несколько затруднена

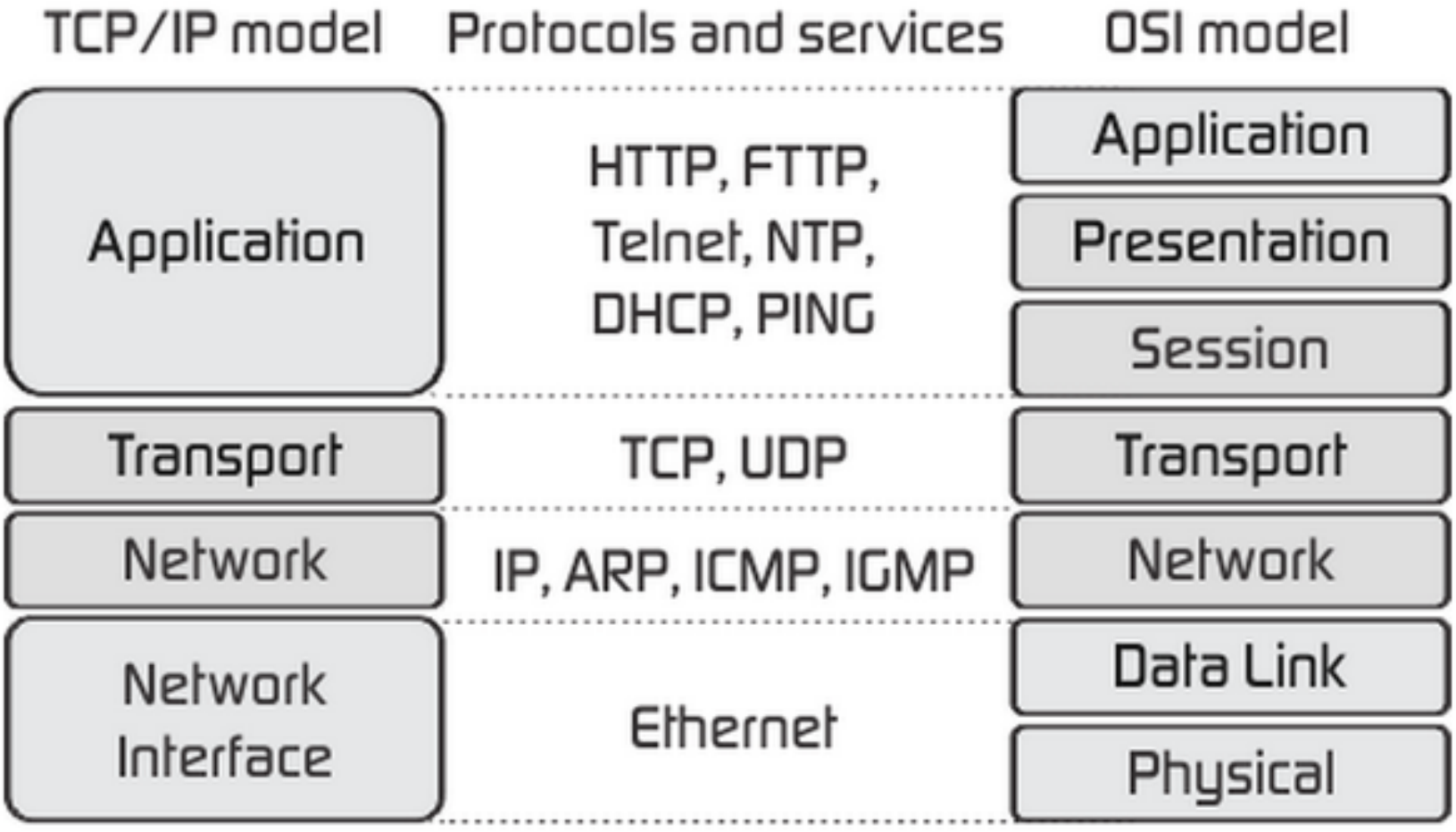
Хранимые процедуры. Минусы.

- Код и схема БД в отрыве от приложения $\Rightarrow$ есть шансы, что изменения будут рассинхронизированы


НЕЛЬЗЯ ПРОСТО ТАК ВЗЯТЬ
И ПЕРЕНЕСТИ ВСЮ
БИЗНЕС-ЛОГИКУ В БАЗУ ДАННЫХ

Между приложением и базой данных

Слои абстракции



JDBC: самый низкий уровень

- Набор стандартных Java-интерфейсов: подключение, запрос, результат
- Драйвер для конкретной реализации СУБД
- URL для подключения к конкретному экземпляру СУБД
- Приложение посылает текст SQL запросов

Объектно-реляционные отображения

- ORM - это библиотеки/фреймфорки
- Сохранение Java-объектов в базе данных
- Упрощённый язык запросов, транслирующийся в диалекты SQL
- Автоматическая генерация схемы

Объектно-реляционные отображения

- Java Persistence Architecture (JPA) - стандартный интерфейс
- Разные реализации: Hibernate, EclipseLink, TopLink, SpringJPA
- Реализации могут расширять JPA

ORM в других языках

- SQLAlchemy, DjangoORM, ActiveRecord, Exposed...

Между приложением и БД

- Данные таблично-ориентированы, приложение объектно-ориентированы

  а ваша СУБД поддерживает наследование? а в ваших классах есть двусторонние ассоциации?

Между приложением и БД

- Низкоуровневые интерфейсы

Между приложением и БД

- Объектно-реляционные отображения

Между приложением и БД

- Ваш собственный интерфейс

Независимость от реализации БД

Независимость от реализации БД

Архитектура приложения

- способного работать как с реляционной СУБД так и с NoSQL хранилищем
- эффективного
- надежного

К.О. предупреждает

- это банально и очевидно

К.О. предупреждает

- это высокоуровневая упрощенная схема

К.О. предупреждает

- это лишь одно из возможных решений

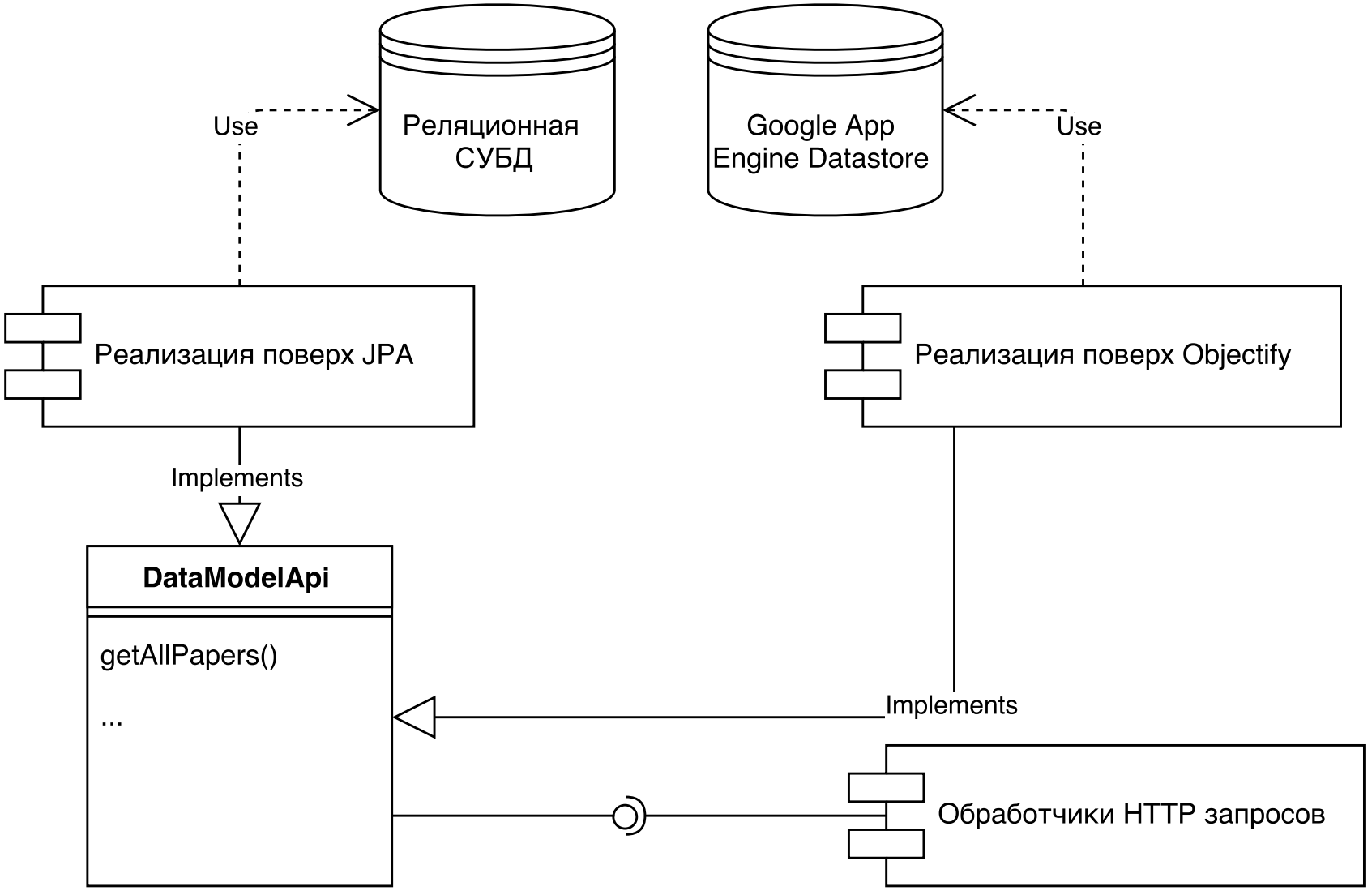
К.О. предупреждает

- это решение может работать плохо, не работать совсем, или быть избыточным

К.О. предупреждает

- только ваша голова в ответе за качество вашей системы

Компоненты приложения



Использованные приемы

- Короткоживущие объекты доступа к данным
- Фабрики объектов
- Общий неявный контекст
- Объектный интерфейс к массивным операциям

Что нужно запомнить

- Сделай приложению интерфейс для чтения данных
  CREATE VIEW

Что нужно запомнить

- И для записи тоже сделай

  хранимые процедуры или объекты бизнес-логики в приложении

Что нужно запомнить

- Минимизируй зависимость от реализации хранилища данных

  ты же не знаешь, какой SQL будет в моде завтра