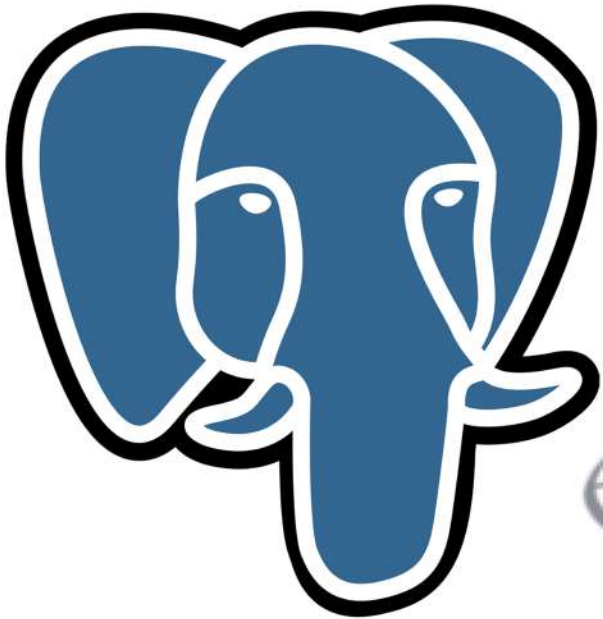
Как хранить данные в JSON*

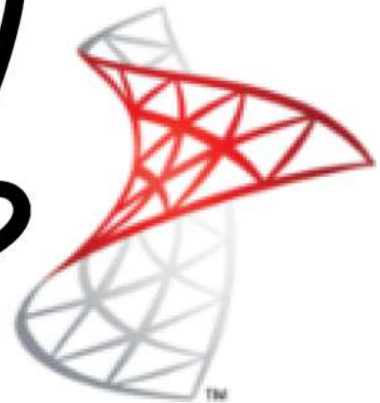
Как хранить данные в JSON*

*не привлекая внимания DBA

Как хранить данные в JSO

*не привлекая внимания DBA





Microsoft®
SQL Server®

ORACLE®

SQL

Диаграммы сущность-связь

HIBERNATE

Entity Framework

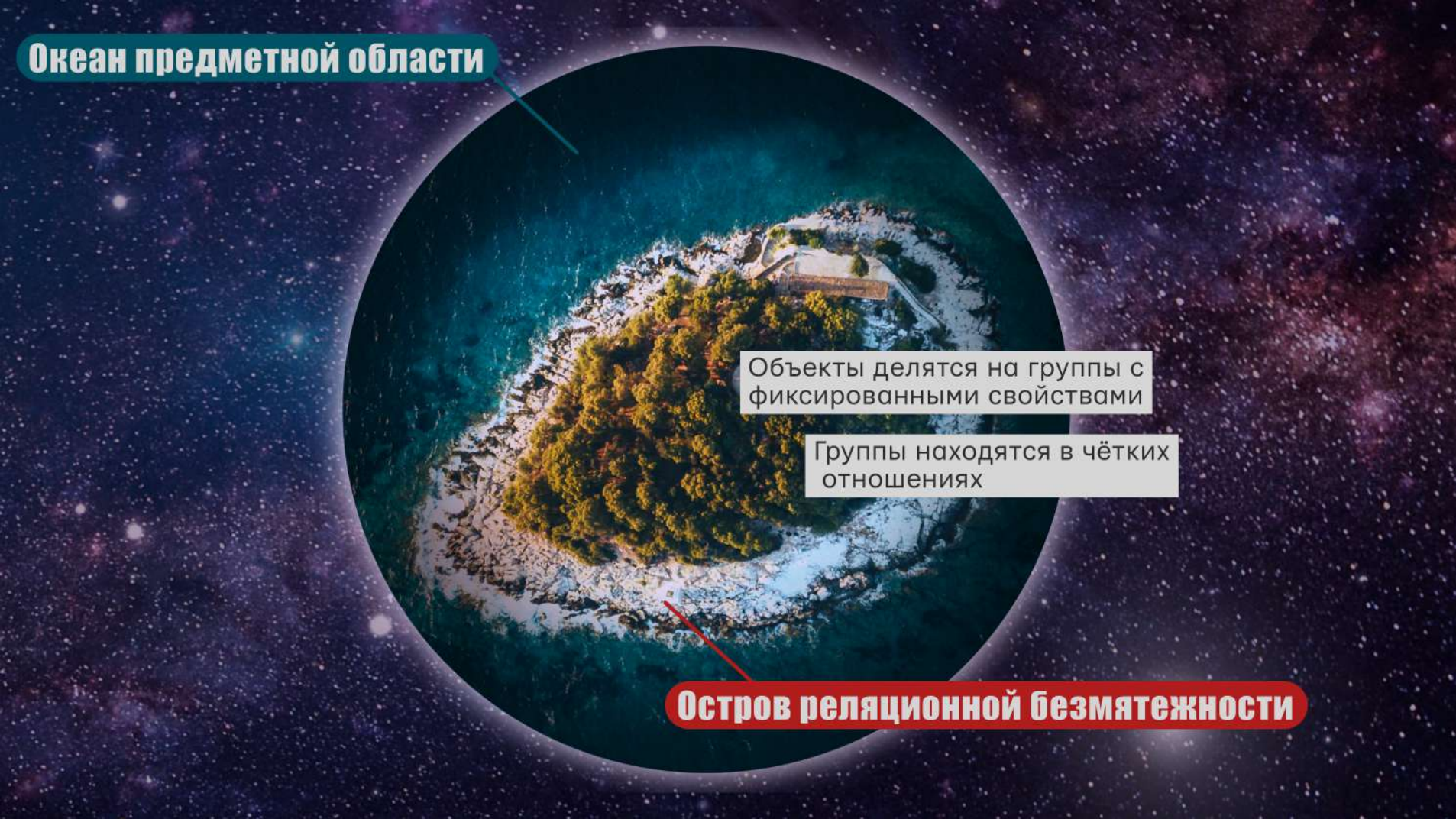
JDBC

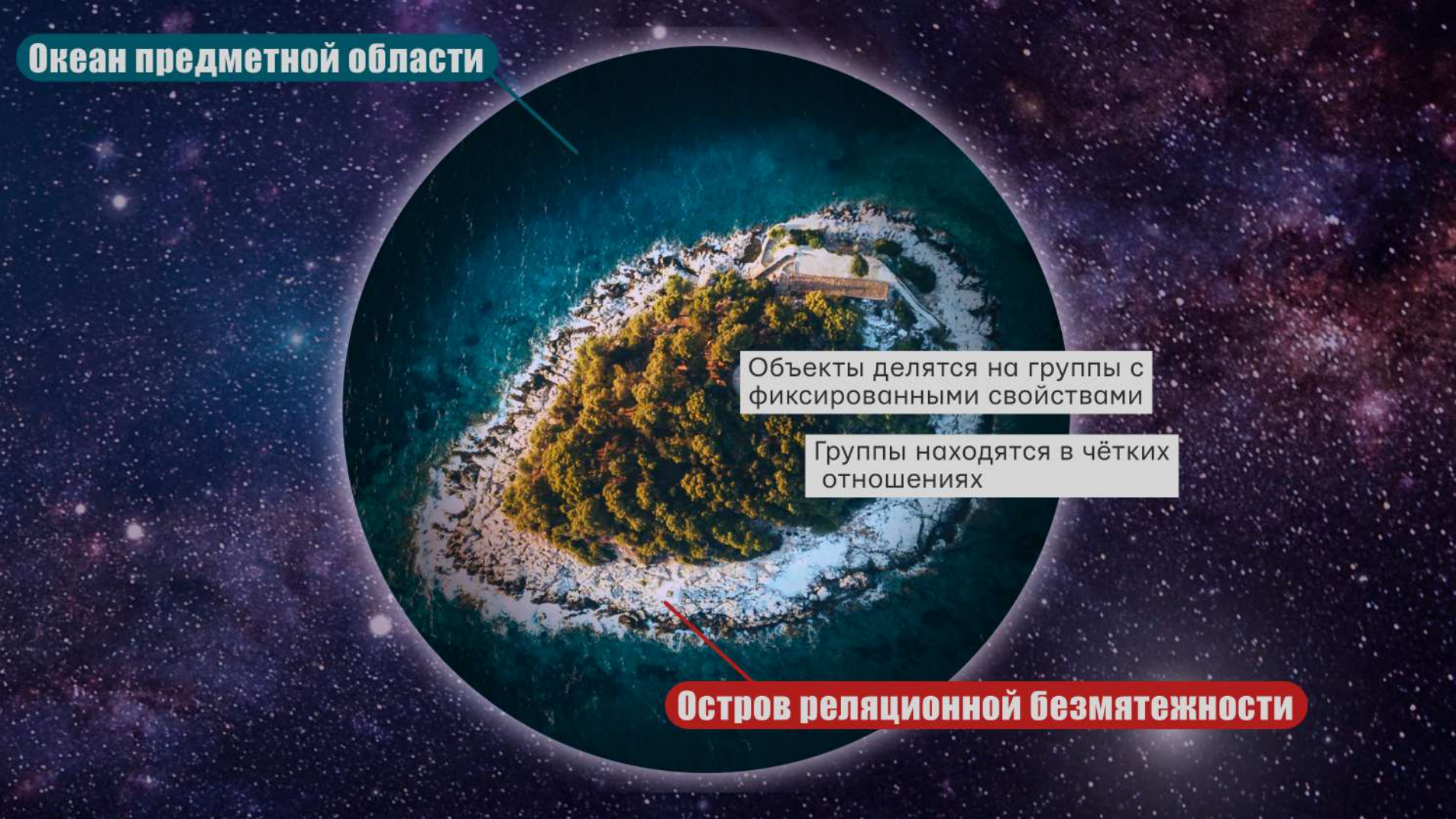
ODBC

doctrine

Океан предметной области

Океан предметной области
Остров реляционной безмятежности

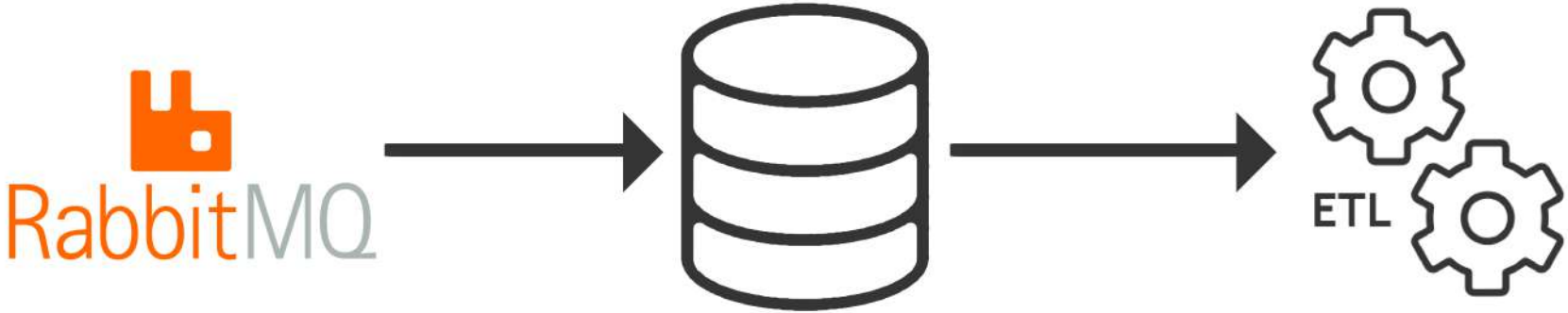

Океан предметной области
Объекты делятся на группы с фиксированными свойствами
Группы находятся в чётких отношениях
Остров реляционной безмятежности


Океан предметной области
Объекты делятся на группы с фиксированными свойствами
Группы находятся в чётких отношениях
Остров реляционной безмятежности

Океан предметной области
Если JSON никому не нужен, откуда взялась MongoDB?
Остров реляционной безмятежности

Океан предметной области
Остров реляционной безмятежности

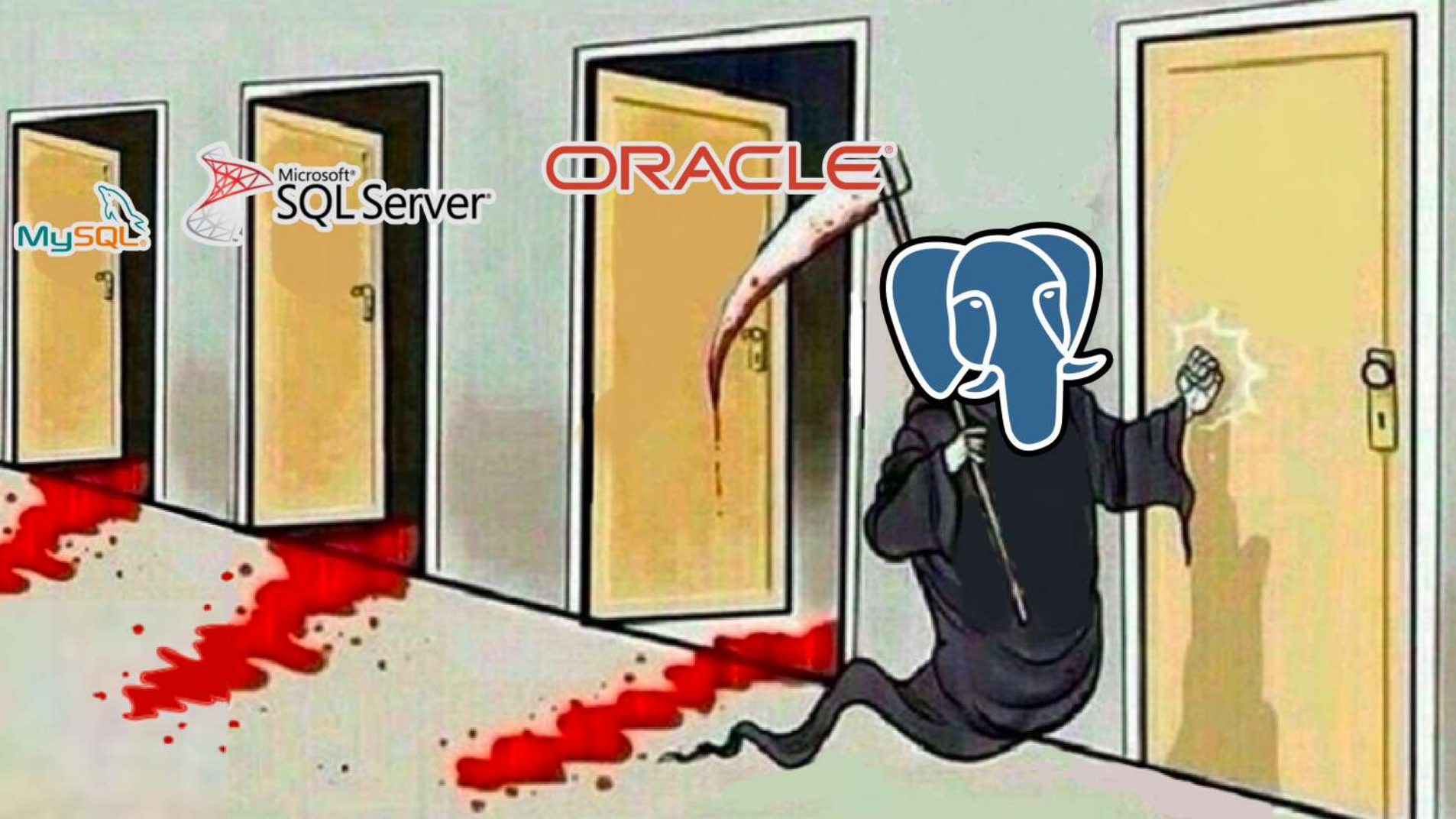
Океан предметной области
Пик произвольных событий
Остров реляционной безмятежности

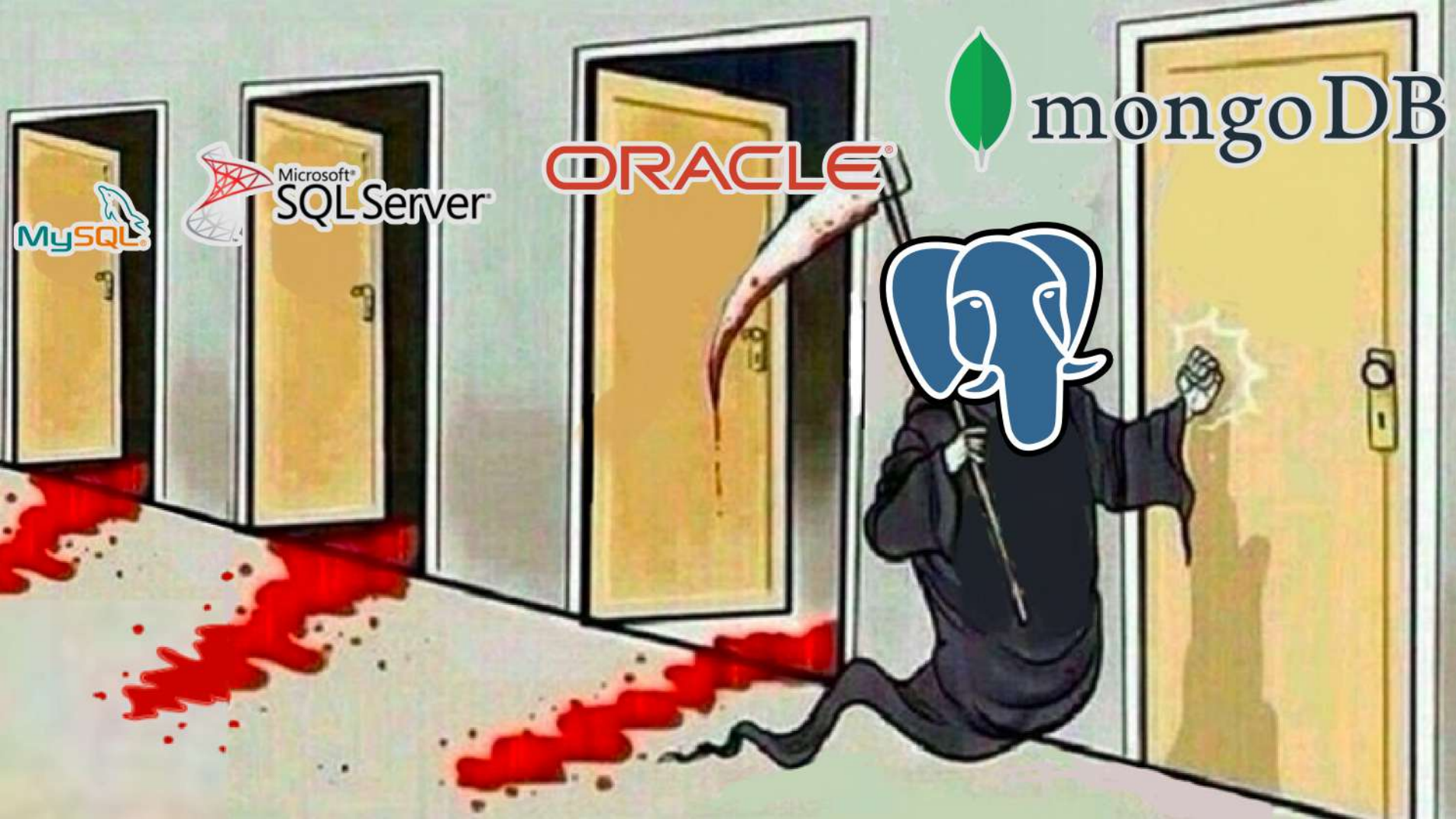



```xml
<changeSet id="8" author="sazonovfm">
    <createTable tableName="events">
        <column name="event_id" type="varchar(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="payload" type="varchar"/>
    </createTable>
</changeSet>
```

```sql
create table events
(
    event_id varchar(36) not null
        primary key,
    payload  varchar
);
```

MySQL
Microsoft SQL Server
ORACLE


MySQL
Microsoft SQL Server
ORACLE


MySQL
Microsoft SQL Server
ORACLE
mongoDB

У Монги нет будущего

У Монги нет будущего
Почему?

У Монги нет будущего
Почему?
В Постгресе появился jsonb

У Монги нет будущего
Почему?
В Постгресе появился jsonb

Олег Бартунов

blogonlymail 8 months ago

Нужно руки оторвать тем, кто придумал в реляционную базу Json пихать. Открыли ящик пандоры для всяких лузер-разработчиков.

2 Reply

blogonlymail 8 months ago

Нужно руки оторвать тем, кто придумал в реляционную базу Json пихать. Открыли ящик пандоры для всяких лузер-разработчиков.

2 Reply

Океан предметной области
Пик произвольных событий
Остров реляционной безмятежности

Океан предметной области
Шельф составных групп
Пик произвольных событий
Остров реляционной безмятежности

Договоры
Договоры найма
Договоры займа

Table inheritance

Table inheritance

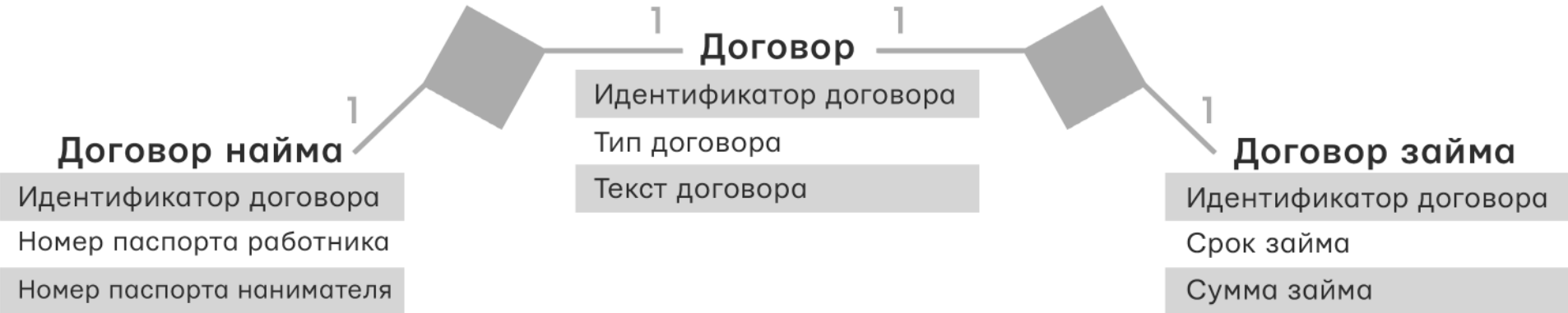


Table inheritance

```java
@Entity
@Table(name = "agreements")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "agr_type")
public class Agreement {
    @Id
    @Column(name = "agreement_id")
    UUID id;

    @Column(name = "descr")
    String descr;

    @Column(name = "agr_type")
    String type;

    //getters, setters
}
```

Table inheritance

```java
@Entity
@Table(name = "hire_agreements")
@DiscriminatorValue(value = "HIRE")
public class HireAgreement extends Agreement{
    @Column(name = "employee_passport")
    String employeePassport;

    @Column(name = "employer_passport")
    String employerPassport;

    //getters, setters, constructor
}
```

```java
@Entity
@Table(name = "loan_agreements")
@DiscriminatorValue(value = "LOAN")
public class LoanAgreement extends Agreement{
    @Column(name = "length")
    int length;

    @Column(name = "amount")
    int amount;

    //getters, setters, constructor
}
```

Table inheritance

```java
@Entity
@Table(name = "agreements")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "agr_type")
public class Agreement {
    @Id
    @Column(name = "agreement_id")
    UUID id;

    @Column(name = "descr")
    String descr;

    @Column(name = "agr_type")
    String type;

    //getters, setters
```

```java
@Entity
@Table(name = "hire_agreements")
@DiscriminatorValue(value = "HIRE")
public class HireAgreement extends Agreement{
    @Column(name = "employee_passport")
    String employeePassport;

    @Column(name = "employer_passport")
    String employerPassport;

    //getters, setters, constructor
}
```

```java
@Entity
@Table(name = "loan_agreements")
@DiscriminatorValue(value = "LOAN")
public class LoanAgreement extends Agreement{
    @Column(name = "length")
    int length;

    @Column(name = "amount")
    int amount;

    //getters, setters, constructor
}
```

Joins!
DBA
DEV
Joins everywhere!
LIGHTYEAR

Table inheritance

```
select
    agreement0_.agreement_id as agreemen1_0_,
    agreement0_.descr as descr2_0_,
    agreement0_.agr_type as agr_type3_0_,
    agreement0_1_.employee_passport as employee1_2_,
    agreement0_1_.employer_passport as employer2_2_,
    agreement0_2_.amount as amount1_3_,
    agreement0_2_.length as length2_3_,
    case
        when agreement0_1_.agreement_id is not null then 1
        when agreement0_2_.agreement_id is not null then 2
        when agreement0_.agreement_id is not null then 0
        end as clazz_
from
    agreements agreement0_
        left outer join
    hire_agreements agreement0_1_
    on agreement0_.agreement_id=agreement0_1_.agreement_id
        left outer join
    loan_agreements agreement0_2_
    on agreement0_.agreement_id=agreement0_2_.agreement_id
```

```
select
    agreement0_.agreement_id as agreemen2_0_,
    agreement0_.descr as descr3_0_,
    agreement0_.agr_type as agr_type1_0_,
    agreement0_1_.employee_passport as employee1_2_,
    agreement0_1_.employer_passport as employer2_2_,
    agreement0_2_.amount as amount1_3_,
    agreement0_2_.length as length2_3_
from
    agreements agreement0_
        left outer join
    hire_agreements agreement0_1_
    on agreement0_.agreement_id=agreement0_1_.agreement_id
        left outer join
    loan_agreements agreement0_2_
    on agreement0_.agreement_id=agreement0_2_.agreement_id
```

Table inheritance

```
{
  "agreement_id": 12345,
  "agreement_type": "HIRE",
  "content": "hello",
  "employer_passport": 12345678,
  "employee_passport": 23456789
}
```

```
{
  "agreement_id": 1234,
  "agreement_type": "LOAN",
  "content": "hello",
  "amount": 789,
  "duration": 60
}
```

Table inheritance

1 1

Договор

Идентификатор договора

Тип договора

Текст договора

Номер паспорта работника

Номер паспорта нанимателя

Срок займа

Сумма займа

1 1

Договор найма

Идентификатор договора

Номер паспорта работника

Номер паспорта нанимателя

```
{
  "agreement_id": 12345,
  "agreement_type": "HIRE",
  "content": "hello",
  "employer_passport": 12345678,
  "employee_passport": 23456789
}
```

1 1

Договор займа

Идентификатор договора

Срок займа

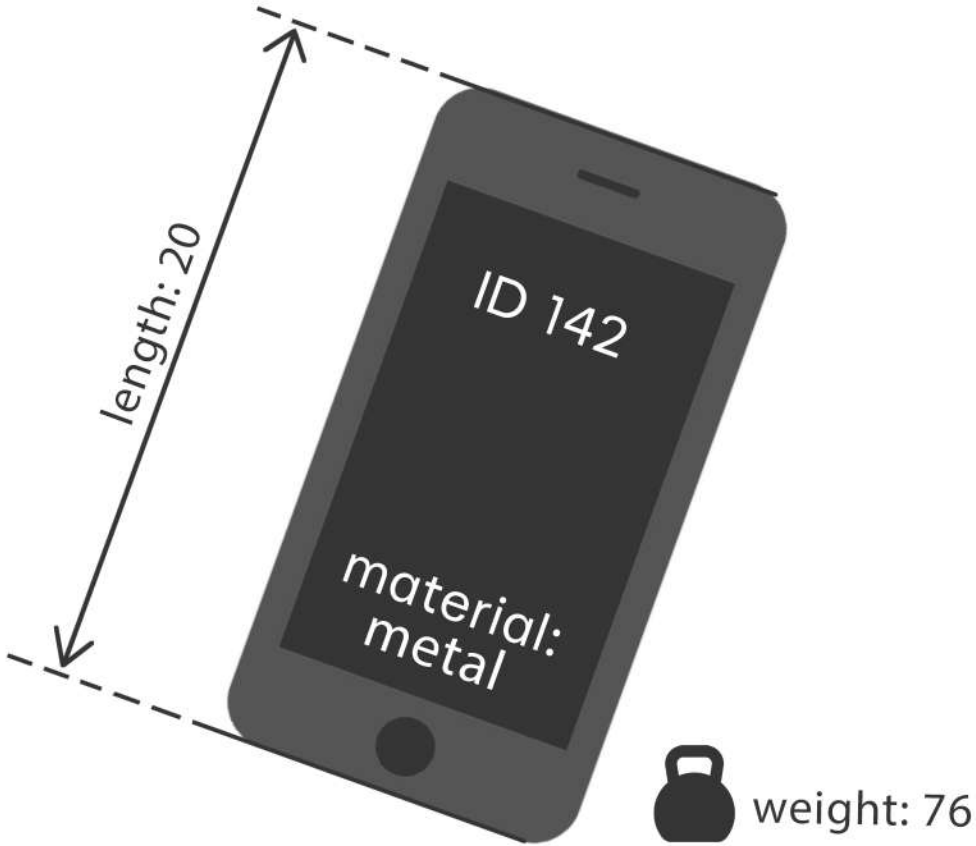
Сумма займа

```
{
  "agreement_id": 1234,
  "agreement_type": "LOAN",
  "content": "hello",
  "amount": 789,
  "duration": 60
}
```

Океан предметной области
Шельф составных групп
Пик произвольных событий
Остров реляционной безмятежности

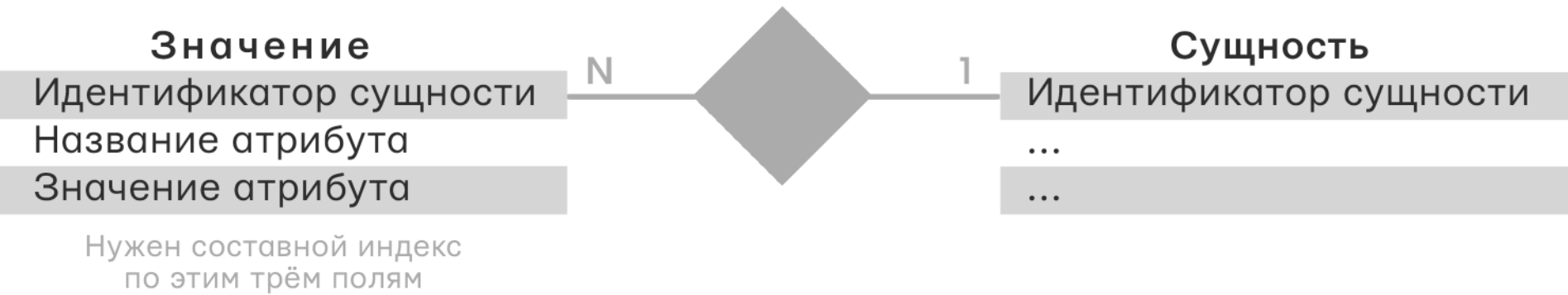
Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности

Каждый товар уникален



EAV (Entity Attribute Value)

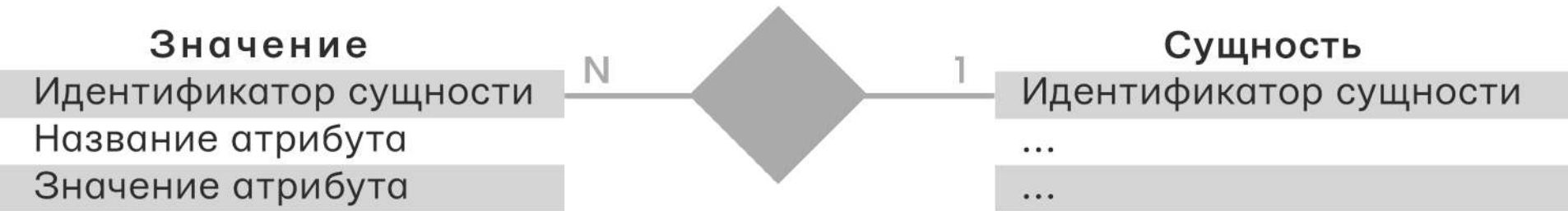
EAV (Entity Attribute Value)



Нужен составной индекс
по этим трём полям

DBA одобряет

EAV (Entity Attribute Value)

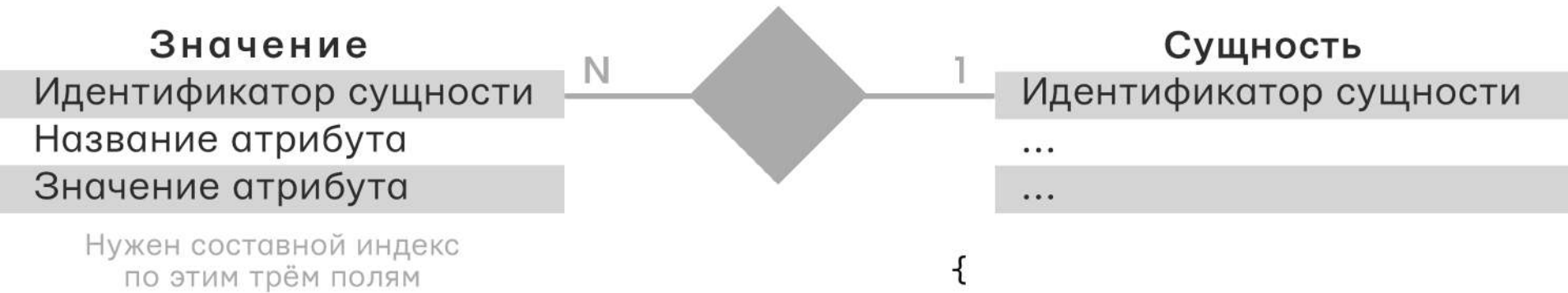


Нужен составной индекс
по этим трём полям

```
{
  "Идентификатор сущности" : 1234,
  "Название аттрибута 1" : "Значение аттрибута 1",
  "Название аттрибута 2" : "Значение аттрибута 2",
  "Название аттрибута 3" : "Значение аттрибута 3",
  "Название аттрибута 4" : "Значение аттрибута 4",
  "Название аттрибута 5" : "Значение аттрибута 5"
}
```

DBA одобряет

EAV (Entity Attribute Value)



Нужен составной индекс
по этим трём полям

```
{
  "id": 1234,
  "length": 20,
  "weight": 76,
  "material": "metal"
}
```

DBA одобряет

EAV (Entity Attribute Value)

```java
@Entity
@Table(name = "products")
public class Product {

    @Id
    private String id;

    private String name;

    @OneToMany(cascade = CascadeType.ALL
            , fetch = FetchType.LAZY
            , mappedBy = "attributeId.entityId"
            , orphanRemoval = true
    )
    private List<Attribute> attributes;

    // getters, setters
}
```

```java
@Entity
public class Attribute {
    @EmbeddedId
    AttributeId attributeId;
    String attrValue;

    //getters, setters
}
```

```java
@Embeddable
public class AttributeId implements Serializable {

    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "entity_id")
    Product entityId;

    String attrName;

   // getters, setters
}
```

EAV (Entity Attribute Value)

ID сущности	Название	Значение
123	weight	70
123	height	30
123	material	metal
123	age	21
123	city	Agraba
127	weight	70
127	height	30

Будет преобразован в JSON массив

≠

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

EAV (Entity Attribute Value)

ID сущности	weight	height	material	age	city
123	70	30	metal	21	Agraba
127	70	30	null	null	null

JSON friendly =

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

EAV (Entity Attribute Value)

БД

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

FrontEnd

```
{
  "id": 123,
  "weight": 70,
  "height": 30,
  "material": "metal",
  "age": 21,
  "city": "Agroba"
}
```

EAV (Entity Attribute Value)

```
{
  "id": 1234,
  "length": 20,
  "weight": 76,
  "material": "metal"
}
```

Почему бы не хранить JSON в строке?

Не получится искать по значению одного атрибута

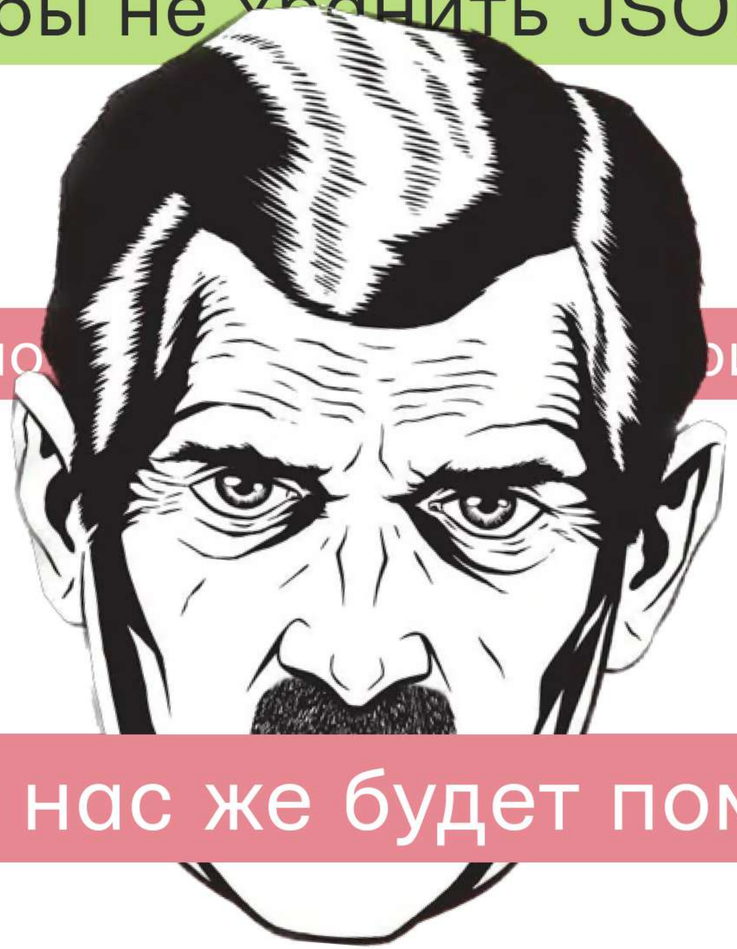
Помойка-помойка, у нас же будет помойка!

EAV (Entity Attribute Value)

```
{
  "id": 1234,
  "length": 20,
  "weight": 76,
  "material": "metal"
}
```

Почему бы не хранить JSON в строке?

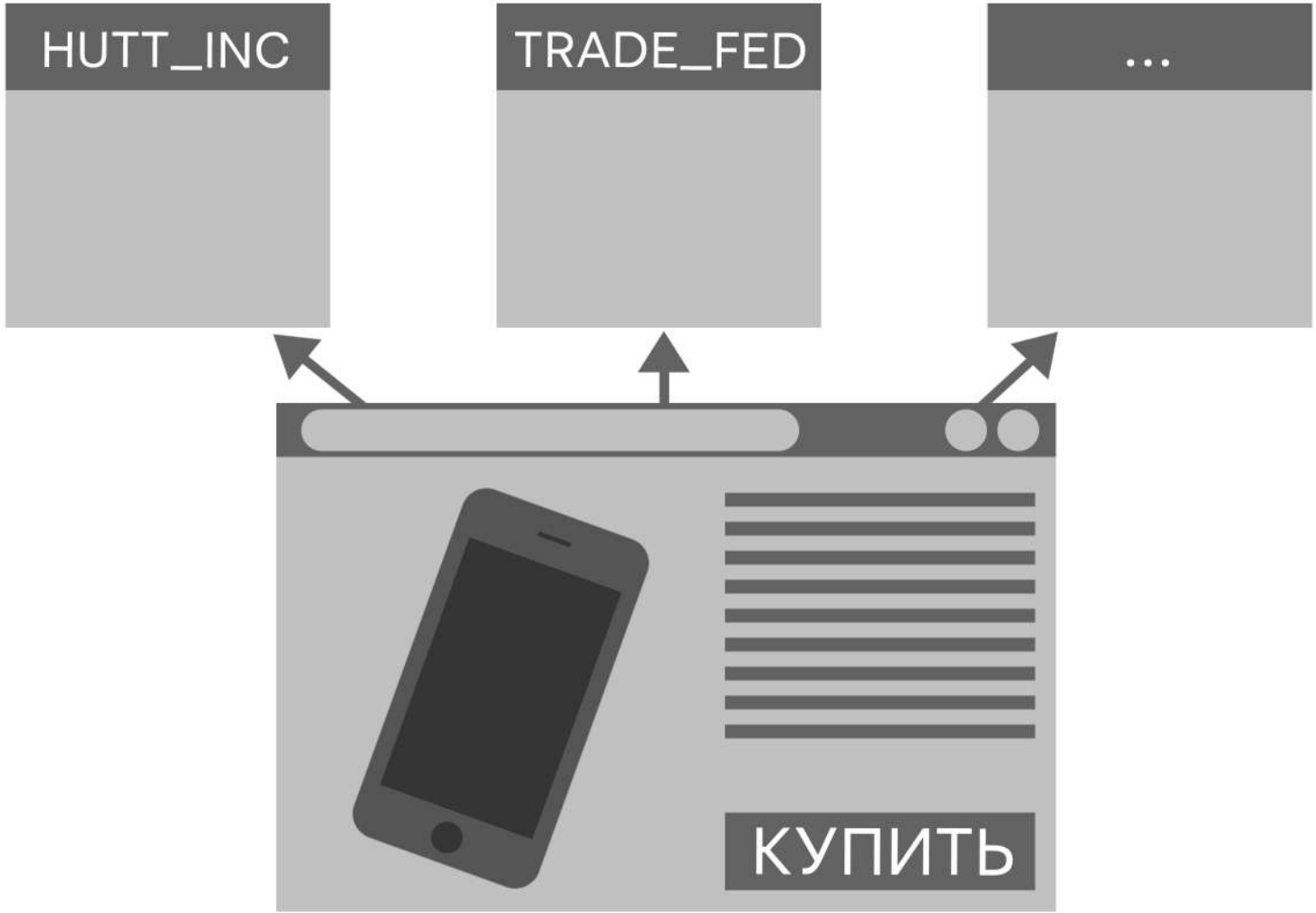
Не получится искать по [illegible]ибута



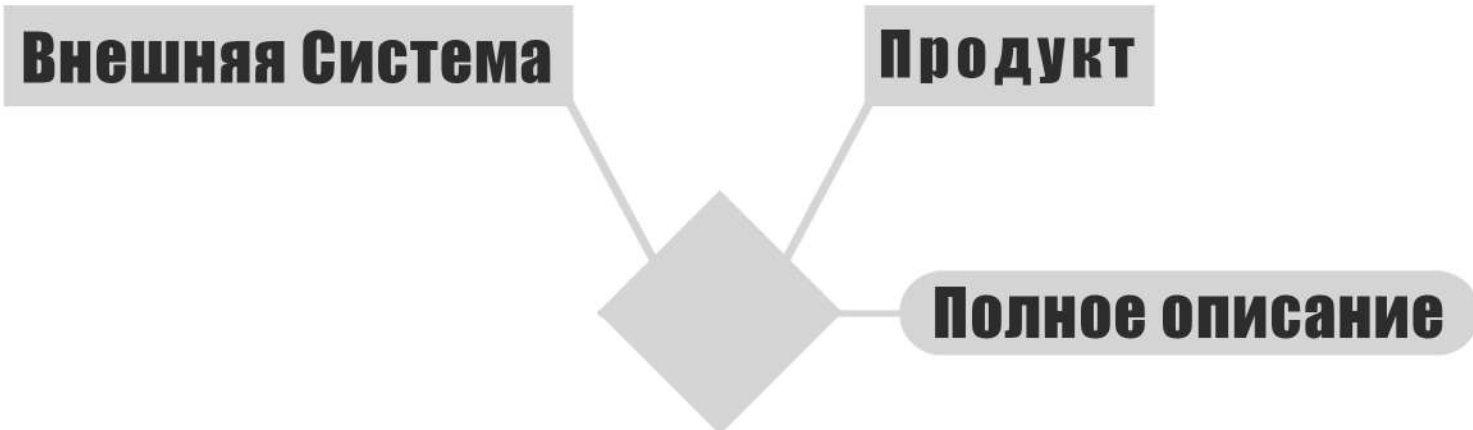
Помойка-помойка, у нас же будет помойка!

Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности

Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов


HUTT_INC
TRADE_FED
...
КУПИТЬ

Внешние коды продуктов



EAV

sys_id	product_id	ext_id
HUTT_INK	123	BEST_PHONE
TRADE_FED	123	ULTRA_PHONE

JSON

product_id	ext_codes
123	{ "HUTT_INC" : "BEST_PHONE", "TRADE_FED" : "ULTRA_PHONE" }

Внешние коды продуктов

```java
@Entity
@TypeDef(
        typeClass = JsonBinaryType.class,
        defaultForType = ExternalIds.class
)
@Table(name = "product_connections")
public class ProductConnection {

    @Id
    private String productId;

    @Column(columnDefinition = "jsonb")
    private ExternalIds externalIds;

    // getters, setters
}
```

```java
@JsonIgnoreProperties(ignoreUnknown = true)
public class ExternalIds {
    @JsonProperty("HUTT_INC")
    private String huttInc;
    @JsonProperty("TRADE_FED")
    private String tradeFed;

    // getters, setters
}
```

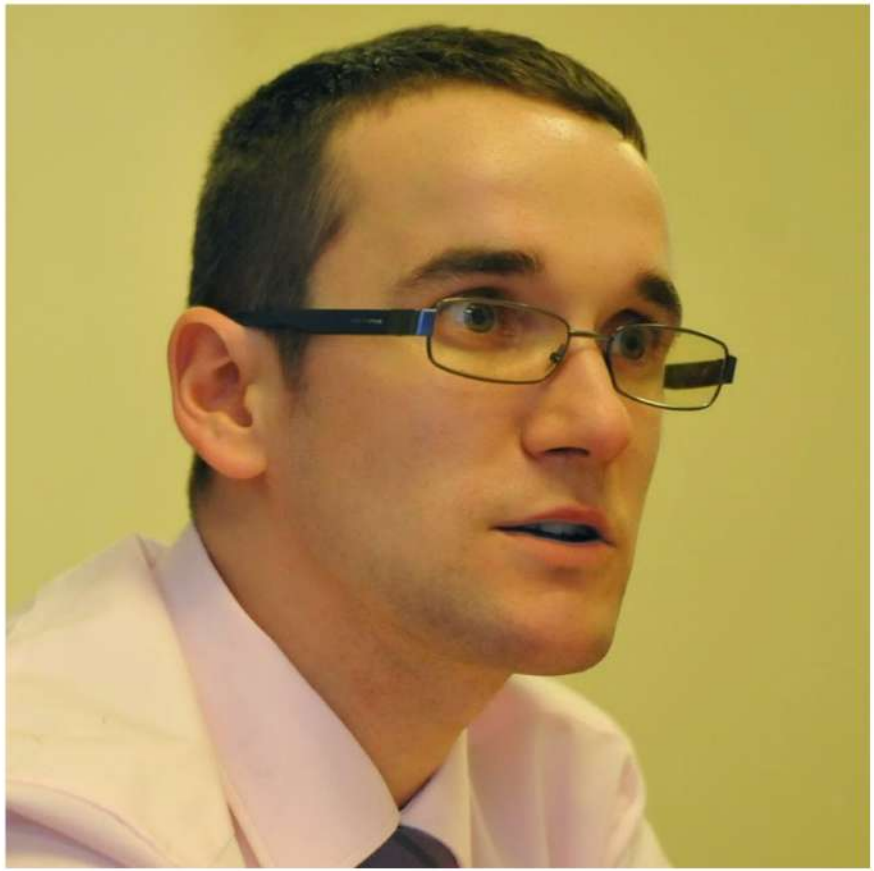
Внешние коды продуктов

```java
@Entity
@TypeDef(
        typeClass = JsonBinaryType.class,
        defaultForType = Map.class
)
@Table(name = "product_connections")
public class AnotherProductConnection {

    @Id
    private String productId;

    @Column(columnDefinition = "jsonb")
    private Map<String, String> externalIds;

    // getters, setters
}
```



```xml
<dependency>
    <groupId>com.vladmihalcea</groupId>
    <artifactId>hibernate-types-55</artifactId>
    <version>2.19.2</version>
</dependency>
```

https://github.com/vladmihalcea/hibernate-types

Внешние коды продуктов (JSON)

Не надо мучаться с ленивостью

Никаких N + 1

Никаких дополнительных запросов

Аналитики переживают, что мы не сможем делать поиск

DBA переживают, что у нас же будет помойка

QA в панике, идентификатор во внешней системе нельзя использовать в запросах

Проблемы с ленивостью

Если забыть про транзакцию

```
org.hibernate.LazyInitializationException: failed to lazily initializ
   a collection of role: com.example.demojson.entity.Product
  .attributes, could not initialize proxy - no Session

    at org.hibernate.collection.internal.AbstractPersistentCollection
      .throwLazyInitializationException(AbstractPersistentCollection
      .java:614)
    at org.hibernate.collection.internal.AbstractPersistentCollection
      .withTemporarySessionIfNeeded(AbstractPersistentCollection
      .java:218)
    at org.hibernate.collection.internal.AbstractPersistentCollection
      .readSize(AbstractPersistentCollection.java:162)
    at org.hibernate.collection.internal.PersistentBag.size
      (PersistentBag.java:371)
    at com.example.demojson.repository.ProductRepositoryTest
      .testProductByAttrSearch(ProductRepositoryTest.java:73)
       <31 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
       <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
       <27 internal lines>
```

```java
@Entity
@Table(name = "products")
public class Product {

    @Id
    private String id;

    private String name;

    @OneToMany(cascade = CascadeType.ALL
            , fetch = FetchType.LAZY
            , mappedBy = "attributeId.entityId"
            , orphanRemoval = true
    )
    private List<Attribute> attributes;

    // getters, setters
}
```

N + 1

```
@Query(value = """
select
  p
from
  Product p join p.attributes as atr
where
      atr.attrValue = :attrValue
  and atr.attributeId.attrName = :attrName
""")
List<Product> findAllByAttrValueAndAttrName(
        @Param("attrValue") String attrValue,
        @Param("attrName") String attrName
);
```

```
@Entity
@Table(name = "products")
public class Product {

    @Id
    private String id;

    private String name;

    @OneToMany(cascade = CascadeType.ALL
            , fetch = FetchType.LAZY
            , mappedBy = "attributeId.entityId"
            , orphanRemoval = true
    )
    private List<Attribute> attributes;

    // getters, setters
}
```

N + 1

```
2022-11-13 19:35:18.941 DEBUG 61013
--- [           main] org.hibernate.SQL
    select
        product0_.id as id1_7_,
        product0_.name as name2_7_
    from
        products product0_
    inner join
        attribute attributes1_
            on product0_.id=attributes1_.entity_id
    where
        attributes1_.attr_value=?
        and attributes1_.attr_name=?
```

```
2022-11-13 19:35:18.950 DEBUG 61013
--- [           main] org.hibernate.SQL
    select
        attributes0_.entity_id as entity_i3_2_0_,
        attributes0_.attr_name as attr_nam1_2_0_,
        attributes0_.attr_name as attr_nam1_2_1_,
        attributes0_.entity_id as entity_i3_2_1_,
        attributes0_.attr_value as attr_val2_2_1_
    from
        attribute attributes0_
    where
        attributes0_.entity_id=?
```

join

N + 1

```
2022-11-13 19:35:18.941 DEBUG 61013
--- [           main] org.hibernate.SQL
    select
        product0_.id as id1_7_,
        product0_.name as name2_7_
    from
        products product0_
    inner join
        attribute attributes1_
            on product0_.id=attribu
    where
        attributes1_.attr_value=
        and attributes1_.attr_nam
```

```
2022-11-13 19:35:18.950 DEBUG 61013
--- [           main] org.hibernate.SQL

    select
        attributes0_.entity_id as entity_i3_2_0_,
        attributes0_.attr_name as attr_nam1_2_0_,
        ibutes0_.attr_name as attr_nam1_2_1_,
        s0_.entity_id as entity_i3_2_1_,
        _.attr_value as attr_val2_2_1_

        ttributes0_

        0_.entity_id=?
```

N + 1

```
@Query(value = """
select
  p
from
  Product p join fetch p.attributes as atr
where
      atr.attrValue = :attrValue
  and atr.attributeId.attrName = :attrName
""")
List<Product> findAllByAttrValueAndAttrNameJoinFetch(
        @Param("attrValue") String attrValue,
        @Param("attrName") String attrName
);
```

```
@Entity
@Table(name = "products")
public class Product {

    @Id
    private String id;

    private String name;

    @OneToMany(cascade = CascadeType.ALL
            , fetch = FetchType.LAZY
            , mappedBy = "attributeId.entityId"
            , orphanRemoval = true
    )
    private List<Attribute> attributes;

    // getters, setters
}
```

N + 1

```
select
    product0_.id as id1_7_0_,
    attributes1_.attr_name as attr_nam1_2_1_,
    attributes1_.entity_id as entity_i3_2_1_,
    product0_.name as name2_7_0_,
    attributes1_.attr_value as attr_val2_2_1_,
    attributes1_.entity_id as entity_i3_2_0__,
    attributes1_.attr_name as attr_nam1_2_0__
from
    products product0_
inner join fetch
    attribute attributes1_
        on product0_.id=attributes1_.entity_id
where
    attributes1_.attr_value=?
    and attributes1_.attr_name=?
```

N + 1

```
select
    product0_.id as id1_7_0_,
    attributes1_.attr_name as attr_nam1_2_1_,
    attributes1_.entity_id as entity_i3_2_1_,
    product0_.name as name2_7_0_,
    attributes1_.attr_value as attr_val2_2_1_,
    attributes1_.entity_id as entity_i3_2_0__,
    attributes1_.attr_name as attr_nam1_2_0__
from
    products product0_
inner join fetch
    attribute attributes1_
        on product0_.id=attributes1_.entity_id
where
    attributes1_.attr_value=?
    and attributes1_.attr_name=?
```

Понятное дело
это не работает

join fetch

N + 1

```
@Query(value = """
select
  p
from
  Product p join p.attributes as atr
where
      atr.attrValue = :attrValue
  and atr.attributeId.attrName = :attrName
""")
List<Product> findAllByAttrValueAndAttrName(
        @Param("attrValue") String attrValue,
        @Param("attrName") String attrName
);
```

```
@Entity
@Table(name = "products")
public class Product {

    @Id
    private String id;

    private String name;

    @BatchSize(size = 2)
    @OneToMany(cascade = CascadeType.ALL
            , fetch = FetchType.LAZY
            , mappedBy = "attributeId.entityId"
            , orphanRemoval = true
    )
    private List<Attribute> attributes;

    // getters, setters
}
```

N + 1

```
2022-11-13 19:40:58.985 DEBUG 61348
-- [           main] org.hibernate.SQL
    select
        product0_.id as id1_7_,
        product0_.name as name2_7_
    from
        products product0_
    inner join
        attribute attributes1_
            on product0_.id=attributes1_.entity_id
    where
        attributes1_.attr_value=?
        and attributes1_.attr_name=?
```

```
2022-11-13 19:40:58.993 DEBUG 61348
--- [           main] org.hibernate.SQL
    select
        attributes0_.entity_id as entity_i3_2_1_,
        attributes0_.attr_name as attr_nam1_2_1_,
        attributes0_.attr_name as attr_nam1_2_0_,
        attributes0_.entity_id as entity_i3_2_0_,
        attributes0_.attr_value as attr_val2_2_0_
    from
        attribute attributes0_
    where
        attributes0_.entity_id in (
            ?, ?
        )
```

N + 1

```
2022-11-13 19:40:58.985 DEBUG 61348
-- [           main] org.hibernate.SQL
    select
        product0_.id as id1_7_,
        product0_.name as name2_7_
    from
        products product0_
    inner join
        attribute attributes1_
            on product0_.id=attributes1_.entity_id
```

```
2022-11-13 19:40:58.993 DEBUG 61348
--- [           main] org.hibernate.SQL
    select
        attributes0_.entity_id as entity_i3_2_1_,
        attributes0_.attr_name as attr_nam1_2_1_,
        attributes0_.attr_name as attr_nam1_2_0_,
        attributes0_.entity_id as entity_i3_2_0_,
        attributes0_.attr_value as attr_val2_2_0_
    from
```

vladmihalcea commented on Dec 1, 2020

@mshima @ctamisier I don't advise people to use `@BatchSize` as, in my opinion, it's a hack. That's because it can reduce the number of additional queries, but it does not necessarily do it in a single query.

Внешние коды продуктов (JSON)

Не надо мучаться с ленивостью

Никаких N + 1

Никаких дополнительных запросов

Аналитики переживают, что мы не сможем делать поиск

DBA переживают, что у нас же будет помойка

QA в панике, идентификатор во внешней системе нельзя использовать в запросах

Внешние коды продуктов (JSON)

Не надо мучаться с ленивостью

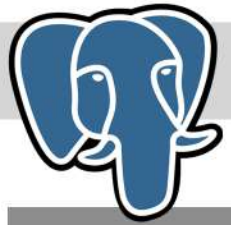
Никаких N + 1

Никаких дополнительных запросов

Аналитики переживают, что мы не сможем делать поиск

DBA переживают, что у нас же будет помойка

QA в панике, идентификатор во внешней системе нельзя использовать в запросах



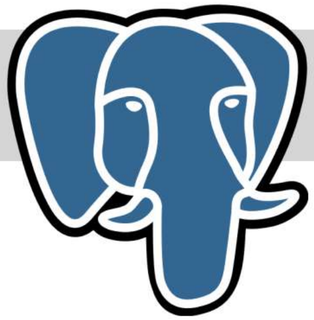
JSON в SQL

поиск товара поставщика с конкретным айди

```
select
    *
from
    product_connections
where
    external_ids @>
    '{"HUTT_INC":"BEST_PHONE"}';
```

поиск всех товаров поставщика

```
select
    *
from
    product_connections
where
    external_ids ?
        'HUTT_INC';
```



JSON в SQL

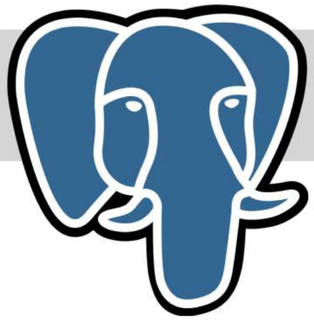
```sql
select
    *
from
    product_connections
where
    external_ids @>
    '{"HUTT_INC":"BEST_PHONE"}';
```

```
Seq Scan on product_connections (cost=0.00..346.40 rows=1 width=145)
        (actual time=0.009..8.656 rows=1 loops=1)
Filter: (external_ids @> '{"'HUTT_INC": "BEST_PHONE"}': : jsonb)
Rows Removed by Filter: 9951
PLanning Time: 0.144 mS
Execution Time: 8.671 ms
```

JSON в SQL

```sql
select
    *
from
    product_connections
where
    external_ids @>
    '{"HUTT_INC":"BEST_PHONE"}';
```

```
Seq Scan on product_connections (cost=0.00..346.40 rows=1 width=145)
        (actual time=0.009..8.656 rows=1 loops=1)
Filter: (external_ids @> '{"'HUTT_INC": "BEST_PHONE"}': : jsonb)
Rows Removed by Filter: 9951
PLanning Time: 0.144 mS
Execution Time: 8.671 ms
```

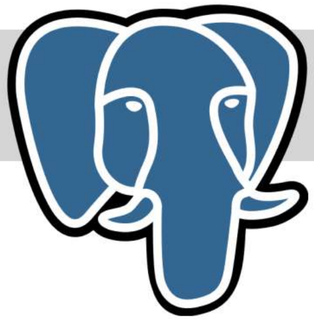


JSON в SQL

```sql
select
    *
from
    product_connections
where
    external_ids @>
    '{"HUTT_INC":"BEST_PHONE"}';
```

```sql
CREATE INDEX idx_pc_ext_ids
    ON product_connections
USING GIN (external_ids);
```

```
Bitmap Heap Scan on product_connections (cost=20.01..24.02 rows=1 width=145)
          (actual time=2.502..2.504 rows=1 loops=1)
Recheck Cond: (external_ids @> ' {"HUTT_INC": "BEST_PHONE"}': :jsonb)
Heap Blocks: exact=1
- > Bitmap Index Scan on id_pc_ext_ids (cost=0.00..20.01 rows=1 width=0)
             (actual time=2.178..2.178 rows=1 Loops=1)
Index Cond: (external_ids d> ' {"HUTT_INC": "BEST_PHONE"}':: jsonb)
Planning Time: 2.744 ms
Execution Time: 2.561 ms
```



JSON в SQL

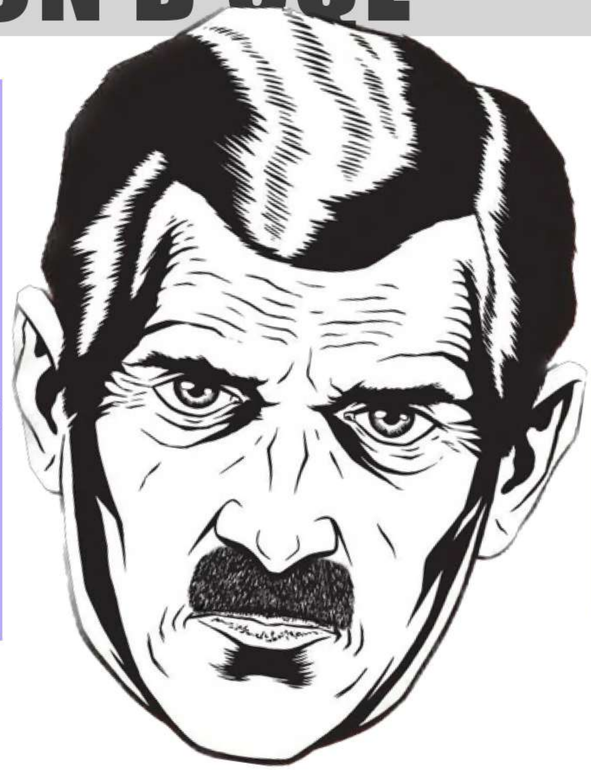
```sql
select
    *
from
    product_connections
where
    external_ids ?
        'HUTT_INC';
```

```sql
CREATE INDEX idx_pc_ext_ids
    ON product_connections
USING GIN (external_ids);
```

```
Seq Scan on product_connections  (cost=0.00..346.40 rows=9951 width=145)
                                 (actual time=0.012..2.401 rows=9952 loops=1)
  Filter: (external_ids ? 'HUTT_INC'::text)
Planning Time: 0.087 ms
Execution Time: 2.814 ms
```

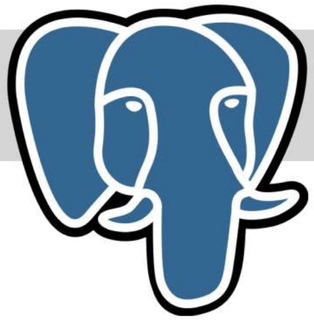
JSON в SQL

```sql
select
    *
from
    product_connections
where
    external_ids ?
        'HUTT_INC';
```



```sql
CREATE INDEX idx_pc_ext_ids
    ON product_connections
USING GIN (external_ids);
```

```
Seq Scan on product_connections  (cost=0.00..346.40 rows=9951 width=145)
                                 (actual time=0.012..2.401 rows=9952 loops=1)
  Filter: (external_ids ? 'HUTT_INC'::text)
Planning Time: 0.087 ms
Execution Time: 2.814 ms
```



JSON в SQL

```sql
select
    *
from
    product_connections
where
    external_ids ?
      'SOME_INC';
```

```sql
CREATE INDEX idx_pc_ext_ids
    ON product_connections
USING GIN (external_ids);
```

```
Bitmap Heap Scan on product_connections  (cost=16.67..251.21 rows=603 width=149)
                                        (actual time=0.171..1.159 rows=664 loops=1)
  Recheck Cond: (external_ids ? 'SOME_INC'::text)
  Heap Blocks: exact=227
  ->  Bitmap Index Scan on idx_pc_ext_ids  (cost=0.00..16.52 rows=603 width=0)
                                           (actual time=0.132..0.133 rows=664 loops=1)
        Index Cond: (external_ids ? 'SOME_INC'::text)
Planning Time: 0.336 ms
Execution Time: 1.237 ms
```

Внешние коды продуктов (JSON)

Не надо мучаться с ленивостью

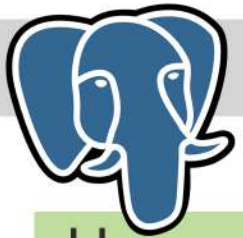
Никаких N + 1

Никаких дополнительных запросов

Аналитики переживают, что мы не сможем делать поиск

DBA переживают, что у нас же будет помойка

QA в панике, идентификатор во внешней системе нельзя использовать в запросах



Внешние коды продуктов (JSON)

Не надо мучаться с ленивостью

Никаких N + 1

Никаких дополнительных запросов

+

~~Аналитики переживают, что мы не сможем делать поиск~~

DBA переживают, что у нас же будет помойка

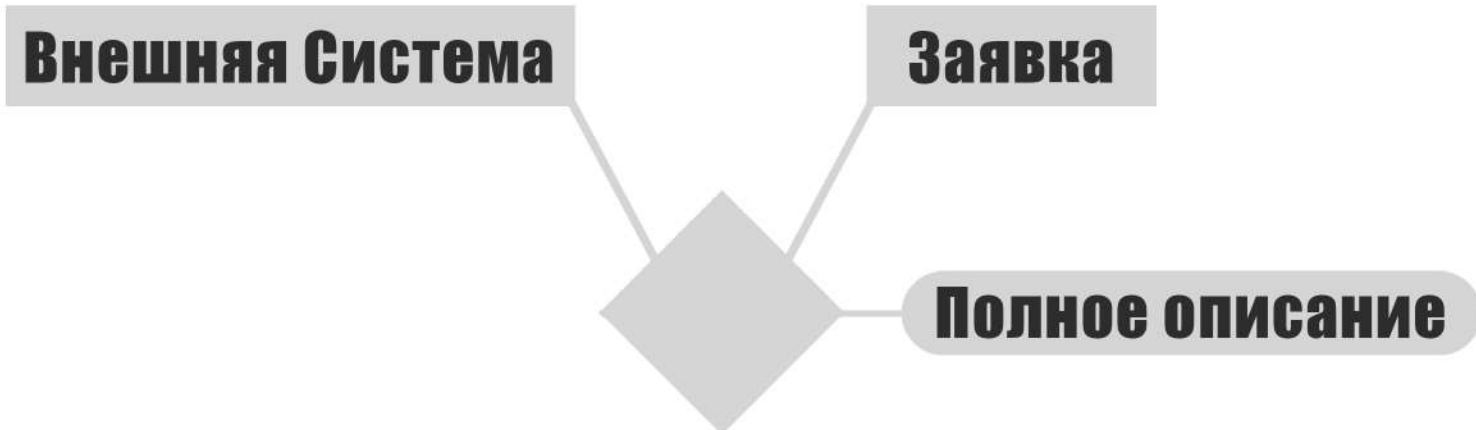
~~QA в панике, идентификатор во внешней системе нельзя использовать в запросах~~

-

Океан предметной области
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов

Океан предметной области
Риф полных описаний
Шельф составных групп
Пик произвольных событий
Отмель произвольных атрибутов
Остров реляционной безмятежности
Пятно внешних кодов

Полные описания заявок



EAV

app_id	attr_name	attr_val
143	smurf	km_232_f12

JSON

app_id	ext_data
143	CENSORED

Полные описания продуктов

EAV

app_id	attr_name	attr_val
143	smurf	km_232_f12

JSON

app_id	ext_data
143	CENSORED

Полные описания заявок (JSON)

Не надо анализировать предметную область

Не надо писать код для преобразования json <-> Entity

Аналитики с QA в панике, что не смогут использовать эти данные в селектах

DBA напоминают про ограничение на размер строки

Полные описания заявок (JSON)

Не надо анализировать предметную область

Не надо писать код для преобразования json <-> Entity

Аналитики с QA в панике, что не смогут использовать эти данные в селектах

DBA напоминают про ограничение на размер строки

Не надо анализировать предметную область

Не надо писать код для преобразования json <-> Entity

Аналитики с QA в панике, что не смогут использовать эти данные в селектах

DBA напоминают про ограничение на размер строки

Не надо анализировать предметную обл

Не надо писать код для преобразования
json <-> Entity

Аналитики с QA в панике, что не смогут
использовать эти данные в селектах

DBA напоминают про ограничение
на размер строки

Не надо писать код для преобразов
json <-> Entity

Аналитики с QA в панике, что не см
использовать эти данные в селект

DBA напоминают про ограничение
на размер строки

DBA напоминают про ограничение на размер строки

Как хранятся большие объекты

PERSON

Name	About
Илья	Весёлый
Фёдор	Умный
Александр	Творческий
Оксана	Лучшая
Олег	Согласно «Повести временных лет», Олег был родичем (соплеменником) Рюрика. В.Н.Татищев со ссылкой на Иоакимовскую летопись считает его шурином — братом жены Рюрика, которую называет Ефандой[7]. Точное происхождение Олега в «Повести временных лет» не указывается. Предания, связанные с его личностью, сохранились также в полумифической...

10Mb

Как хранятся большие объекты

PERSON

Name	About
Илья	Весёлый
Фёдор	Умный
Александр	Творческий
Оксана	Лучшая
Олег	

10Mb

TOAST

Согласно «Повести временных лет»,	(8 kb)
Олег был родичем (соплеменником)	(8 kb)
Рюрика. В.Н.Татищев со ссылкой на	(8 kb)
Иоакимовскую летопись считает его	(8 kb)
шурином — братом жены Рюрика,	(8 kb)
которую называет Ефандой[7]. Точное	(8 kb)
происхождение Олега в «Повести	(8 kb)
временных лет» не указывается.	(8 kb)
Предания, связанные с его личностью,	(8 kb)
сохранились также в полумифической	(8 kb)

Мегабайты текста
(до одного гигабайта)

Как хранятся большие объекты

PERSON

Name	About
Илья	Весёлый
Фёдор	Умный
Александр	Творческий
Оксана	Лучшая
Олег	

10Mb

TOAST

Согласно «Повести временных лет», (8 kb)
Олег был родичем (соплеменником) (8 kb)
Рюрика. В.Н.Татищев со ссылкой на (8 kb)
Иоакимовскую летопись считает его (8 kb)
шурином — братом жены Рюрика, (8 kb)
которую называет Ефандой[7]. Точное (8 kb)
происхождение Олега в «Повести (8 kb)
временных лет» не указывается. (8 kb)
Предания, связанные с его личностью, (8 kb)
сохранились также в полумифической (8 kb)

Мегабайты текста
(до одного гигабайта)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. During an UPDATE operation, values of unchanged fields are normally preserved as-is; so an UPDATE of a row with out-of-line values incurs no TOAST costs if none of the out-of-line values change.

The TOAST management code recognizes four different strategies for storing TOAST-able columns on disk:

- `PLAIN` prevents either compression or out-of-line storage; furthermore it disables use of single-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. During an UPDATE operation, values of unchanged fields are normally preserved as-is; so an UPDATE of a row with out-of-line values incurs no TOAST costs if none of the out-of-line values change.

The TOAST management code recognizes four different strategies for storing TOAST-able columns on disk:

- `PLAIN` prevents either compression or out-of-line storage; furthermore it disables use of single-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. Durin DATE of a row with out-

https://github.com/postgres/postgres/blob/master/src/include/access/heaptoast.h

```
#define TOAST_TUPLES_PER_PAGE   4

#define TOAST_TUPLE_THRESHOLD   MaximumBytesPerTuple(TOAST_TUPLES_PER_PAGE)

#define TOAST_TUPLE_TARGET      TOAST_TUPLE_THRESHOLD
```

The TOAST m sk:

- Pl gle-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compression and out-of-line storage. This is the default for most TOAST-able data types. Compression will be attempted first, then out-of-line storage if the row is still too big.
- `EXTERNAL` allows out-of-line storage but not compression. Use of `EXTERNAL` will make substring operations on wide `text` and `bytea` columns faster (at the penalty of increased storage space) because these operations are optimized to fetch only the required parts of the out-of-line value when it is not compressed.
- `MAIN` allows compression but not out-of-line storage. (Actually, out-of-line storage will still be performed for such columns, but only as a last resort when there is no other way to make the row small enough to fit on a page.)

The TOAST management code is triggered only when a row value to be stored in a table is wider than `TOAST_TUPLE_THRESHOLD` bytes (normally 2 kB). The TOAST code will compress and/or move field values out-of-line until the row value is shorter than `TOAST_TUPLE_TARGET` bytes (also normally 2 kB, adjustable) or no more gains can be had. Durin DATE of a row with out-

https://github.com/postgres/postgres/blob/master/src/include/access/heaptoast.h

```
#define TOAST_TUPLES_PER_PAGE   4

#define TOAST_TUPLE_THRESHOLD   MaximumBytesPerTuple(TOAST_TUPLES_PER_PAGE)

#define TOAST_TUPLE_TARGET      TOAST_TUPLE_THRESHOLD
```

The TOAST m sk:

- Pl gle-byte headers for varlena types. This is the only possible strategy for columns of non-TOAST-able data types.
- `EXTENDED` allows both compressi the default for most TOAST-able data types. Compression will be if the row is still too big.
- `EXTERNAL` allows out-of ERNAL will make substring operations on wide t reased storage space) because these operations are ut-of-line value when it is not compressed.
- `MAIN` allows compress e storage will still be performed for such columns, but o make the row small enough to fit on a page.)

+

HIBERNATE

Используйте ленивую загрузку

```java
@Entity
@Table(name = "app_users")
public class AppUser {
    @Id
    @Column(name = "user_id")
    UUID id;

    @Column(name = "user_num")
    Integer number;

    @Column(name = "user_name")
    String name;

    @Basic(fetch = FetchType.LAZY)
    @Column(name = "additional_info")
    String additionalInfo;

    //getters, setters
}
```

Используйте ленивую загрузку

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.hibernate.orm.tooling</groupId>
            <artifactId>hibernate-enhance-maven-plugin</artifactId>
            <version>${hibernate.version}</version>
            <executions>
                <execution>
                    <configuration>
                        <enableLazyInitialization>true</enableLazyInitialization>
                    </configuration>
                    <goals>
                        <goal>enhance</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Используйте ленивую загрузку

```
<build>
    <plugins>
        <plugin>
            <groupId>org.hibernate.orm.tooling</groupId>
            <artifactId>hibernate-enhance-maven-plugin</artifactId>
            <version>${hibernate.version}</version>
            <executions>
                <execution>
                    <configuration>
                        <enableLazyInitialization>true</enableLazyInitialization>
                    </configuration>
                    <goals>
                        <goal>enhance</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Полагайтесь на dirty checking

```
@Service
@RequiredArgsConstructor
public class UpdateAppServiceImpl
        implements UpdateAppService {

    private final AppRepository repository;

    @Transactional
    @Override
    public void updateNumber(String id, String newNumber) {
        repository.findById(id)
                .ifPresent(app → app.setNumber(newNumber));
    }
}
```

Не обязательно мапить JSON в объекты

```java
@Entity
@Table(name = "app_users")
public class AppUser {
    @Id
    @Column(name = "user_id")
    UUID id;

    @Column(name = "user_num")
    Integer number;

    @Column(name = "user_name")
    String name;

    @Basic(fetch = FetchType.LAZY)
    @Column(name = "additional_info")
    String additionalInfo;

    //getters, setters
}
```

Не обязательно мапить JSON в объекты

```
@Entity
@TypeDef(
        typeClass = JsonBinaryType.class,
        defaultForType = JsonNode.class
)
public class App {

    @Id
    String id;

    String number;

    @Basic(fetch = FetchType.LAZY)
    @Column(columnDefinition = "jsonb")
    JsonNode externalData;

   //getters, setters
}
```

Полные описания заявок (JSON)

Полные описания заявок (JSON)

```
@Query(
   """
   select
      p
   from
      ProductConnection p fetch all properti
   """)
List<ProductConnection> findAllEagerLoad(
        Pageable page);
```

```
select
    productcon0_.product_id as product_1_6_,
    productcon0_.external_ids as external2_6_
from
    product_connections productcon0_ limit ?
```

Полные описания заявок (JSON)

Не надо анализировать предметную область

Не надо писать код для преобразования json <-> Entity

Аналитики с QA в панике, что не смогут использовать эти данные в селектах

DBA напоминают про ограничение на размер строки

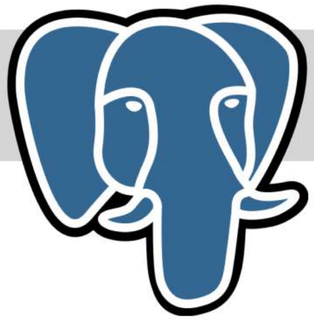
Полные описания заявок (JSON)

Не надо анализировать предметную область

Не надо писать код для преобразования json <-> Entity

Аналитики с QA в панике, что не смогут использовать эти данные в селектах

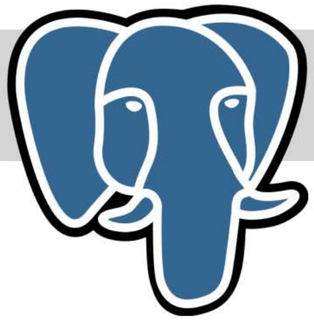
~~DBA напоминают про ограничение на размер строки~~



JSON в SQL

```sql
select
  *
from
  apps
where
  external_data @>
  '{
    "event": {
      "descr": {
        "text": "some"
      } } }';
```

```
Seq Scan on apps (cost=0.00..414.44 rows=905 width=199)
     (actual time=0.011..4.387 rows=830 Loops=1)
Filter: (external_data
 @> '{""event"": {""descr"": {""text"": ""some""}}}' ::jsonb)"
Rows Removed by Filter: 9125
PLanning Time: 0.213 ms
Execution Time: 4.439 ms
```



JSON в SQL

```
CREATE INDEX idx_app_ext_data
    ON apps
USING GIN (external_data);
```

```
select
  *
from
  apps
where
  external_data @>
  '{
    "event": {
      "descr": {
        "text": "some"
      } } }';
```

```
Bitmap Heap Scan on apps (cost=43.01..344.33 rows=905 width=199)
       (actual time=0.131..0.544 rows=830 Loops=1)
Recheck Cond: (external_data
 @> '("event": ("descr": ("text": "some"}}}': :jsonb)
Heap Blocks: exact=290
Bitmap Index Scan on addatagin (cost=0.00..42.79 rows=905 width=0)
          (actual time=0.109..0.109 rows=830 loops=1)
Index Cond: (external_data
 @> ("event": {"descr": {"text": "some"}}}':: jsonb)
Planning Time: 0.076 ms
Execution Time: 0.577 ms
```

Полные описания заявок (JSON)

Не надо анализировать предметную область

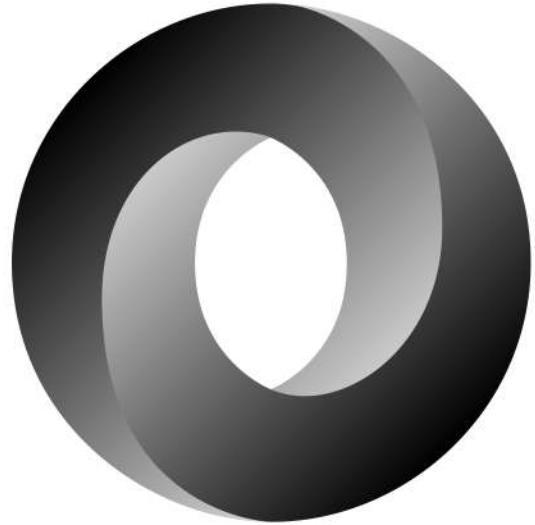
Не надо писать код для преобразования json <-> Entity

+

~~Аналитики с QA в панике, что не смогут использовать эти данные в селектах~~

~~DBA напоминают про ограничение на размер строки~~

−

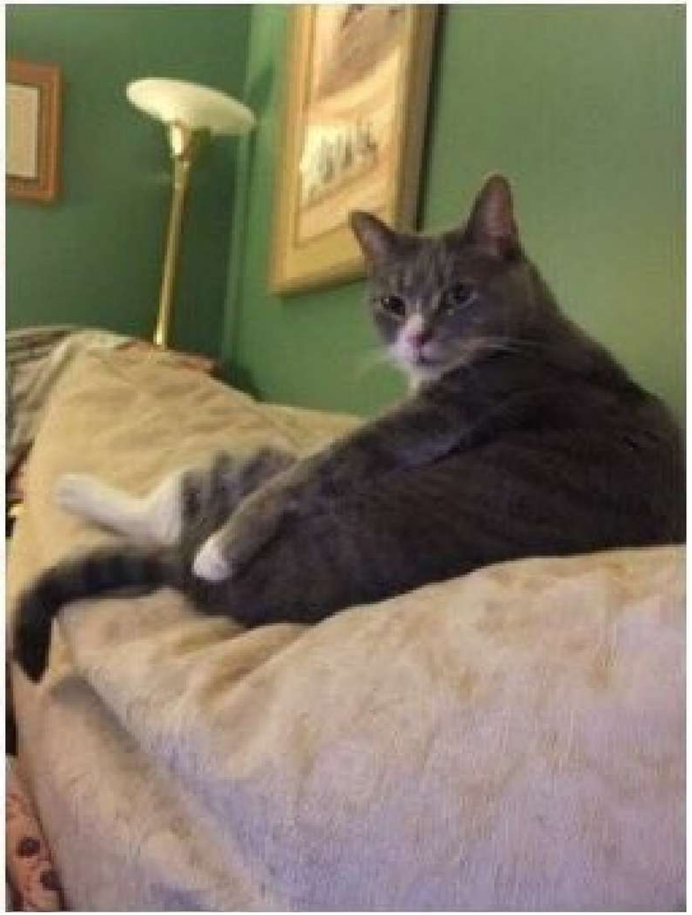


+

HIBERNATE

Что сделать в Hibernate 6.0?
Jakarta!
ANTLR 4!
JSON в HQL!

А как же с нативными запросами?



```
    select
        *
    from
        product_connections
    where external_ids ->
          'HUTT_INC' = '"BEST_PHONE"';
    """)
```

Нативные запросы работают

```
public interface ProductConnectionRepository
        extends JpaRepository<ProductConnection, String> {
    @Query(nativeQuery = true,
      value =
      """
      select
          *
      from
          product_connections
      where external_ids ->
            'HUTT_INC' = '"BEST_PHONE"';
      """)
    List<ProductConnection> findByExternalCode();
}
```

JSON c Hibernate

```
@Query(nativeQuery = true,
  value =
  """
  select
      *
  from
      product_connections
  where
      external_ids @>
          :needle
  """)
List<ProductConnection> findByExternalCode(
        @Param("needle") String needle);
```

JSON с Hibernate

```
@Query(nativeQuery = true,
  value =
  """
  select
      *
  from
      product_connections
  where
      external_ids @>
          :needle
  """)
List<ProductConnection> findByExternalCode(
        @Param("needle") String needle);
```

```
Caused by: org.postgresql.util.PSQLException: ERROR:
 operator does not exist: jsonb @> character varying
  Подсказка: No operator matches the given name and argumen
    types. You might need to add explicit type casts.
```

JSON с Hibernate

vlsi commented on May 25, 2016 — Member

> Following query works perfectly from PSQL console:
> `SELECT id, data FROM table_name WHERE data->'objects' @> '[{"id": "id1"}]'`

Note that `'[{"id": "id1"}]'` is a string literal.

So you have two options:

1. `SELECT id, data FROM table_name WHERE data->'objects' @> ?::jsonb` -- basically you would have to use bind for the whole item. E.g. as string.
2. `SELECT id, data FROM table_name WHERE data->'objects' @> '[{"id": ' || ? || '}]'`. Be careful with "json-injection" kind of attack vector.
3. Implement jsonb type support in pgjdbc, then use `...data->'objects' @> ?::jsonb`

Does that make sense?

👍 1

mented on May 25, 2016

wing query works perfectly from PSQL console:

```
CT id, data FROM table_name WHERE data->'objects' @> '[{"id": "id1"}]'
```

at `'[{"id": "id1"}]'` is a string literal.

have two options:

`LECT id, data FROM table_name WHERE data->'objects' @> ?::jsonb` -- basically u would have to use bind for the whole item. E.g. as string.

`ELECT id, data FROM table_name WHERE data->'objects' @> '[{"id": ' || ? || ]'` . Be careful with "json-injection" kind of attack vector.

plement jsonb type support in pgjdbc, then use `...data->'objects' @> ?::jsonb`

hat make sense?

`ERE data->'objects' @> ?::jsonb` -- basically

ole item. E.g. as string.

`ERE data->'objects' @> '[{"id": ' || ? ||`

kind of attack vector.

bc, then use `...data->'objects' @> ?::jsonb`

JSON с Hibernate

```
@Query(nativeQuery = true,
  value =
  """
  select
      *
  from
      product_connections
  where
      external_ids @>
          :needle \\:\\:jsonb
  """)
List<ProductConnection> findByExternalCode(
        @Param("needle") String needle);
```

~~Caused by: org.postgresql.util.PSQLException: ERROR:~~
~~operator does not exist: jsonb @> character varying~~
~~Подсказка: No operator matches the given name and argumen~~
~~types. You might need to add explicit type casts.~~

JSON с Hibernate

```
@Query(nativeQuery = true,
  value =
  """
  select
      *
  from
      product_connections
  where
      external_ids @>
          :needle \\:\\:jsonb
  """)
List<ProductConnection> findByExternalCode(
        @Param("needle") String needle);
```

~~Caused by: org.postgresql.util.PSQLException: ERROR:~~
~~operator does not exist: jsonb @> character varying~~
~~Подсказка: No operator matches the given name and argumen~~
~~types. You might need to add explicit type casts.~~

Итоги

JSON украшает реляционные базы данных

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

JSON там, где раскладывать предметную область бессмысленно

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

JSON там, где раскладывать предметную область бессмысленно

JSON в новом Postgres — скажите аналитикам, DBA и QA

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

JSON там, где раскладывать предметную область бессмысленно

JSON в новом Postgres — скажите аналитикам, DBA и QA

Размер имеет значение, делайте большой JSON ленивым

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

JSON там, где раскладывать предметную область бессмысленно

JSON в новом Postgres — скажите аналитикам, DBA и QA

Размер имеет значение, делайте большой JSON ленивым

Не стесняйтесь использовать нативные запросы в Hibernate

Итоги

JSON украшает реляционные базы данных

JSON там, где вам предлагают EAV

JSON там, где раскладывать предметную область бессмысленно

JSON в новом Postgres — скажите аналитикам, DBA и QA

Размер имеет значение, делайте большой JSON ленивым

Не стесняйтесь использовать нативные запросы в Hibernate

Не увлекайтесь

```
{
  "Илья Сазонов": {
    "ТГ": "@imsazonov",
    "Почта": "poxvuibr@gmail.com"
  },
  "Фёдор Сазонов": {
    "ТГ": "@sazonovfm",
    "Почта": "sazonovfm@gmail.com"
  }
}
```

{

"Спа": "сибо"

}