Дмитрий Дорофеев
Группа Компаний Luxms

PGDAY'17 RUSSIA
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Использование возможностей Greenplum в PostgreSQL проекте

Luxms

О Группе Компаний Luxms

ГК Luxms – группа компаний, свыше 12 лет работающая в области высоких технологий.

Лидер в области решений по визуальному управленческому контролю – аналитике и визуализации для руководителей.

Наши R&D центр и центр внедрения находятся в Санкт-Петербурге.

Флагманский продукт: Luxms BI – платформа визуального управленческого контроля

Luxms Business Intelligence

Уникальная аналитическая платформа для лиц, принимающих решения.

Новый уровень комфорта при контроле ключевых показателей бизнеса.

Наш опыт – 16 внедрений в 6 отраслях

- Финансы
- Здравоохранение
- Информационные технологии
- Нефть, Газ, Энергетика и ЖКХ
- Телеком
- Транспорт

О докладчике

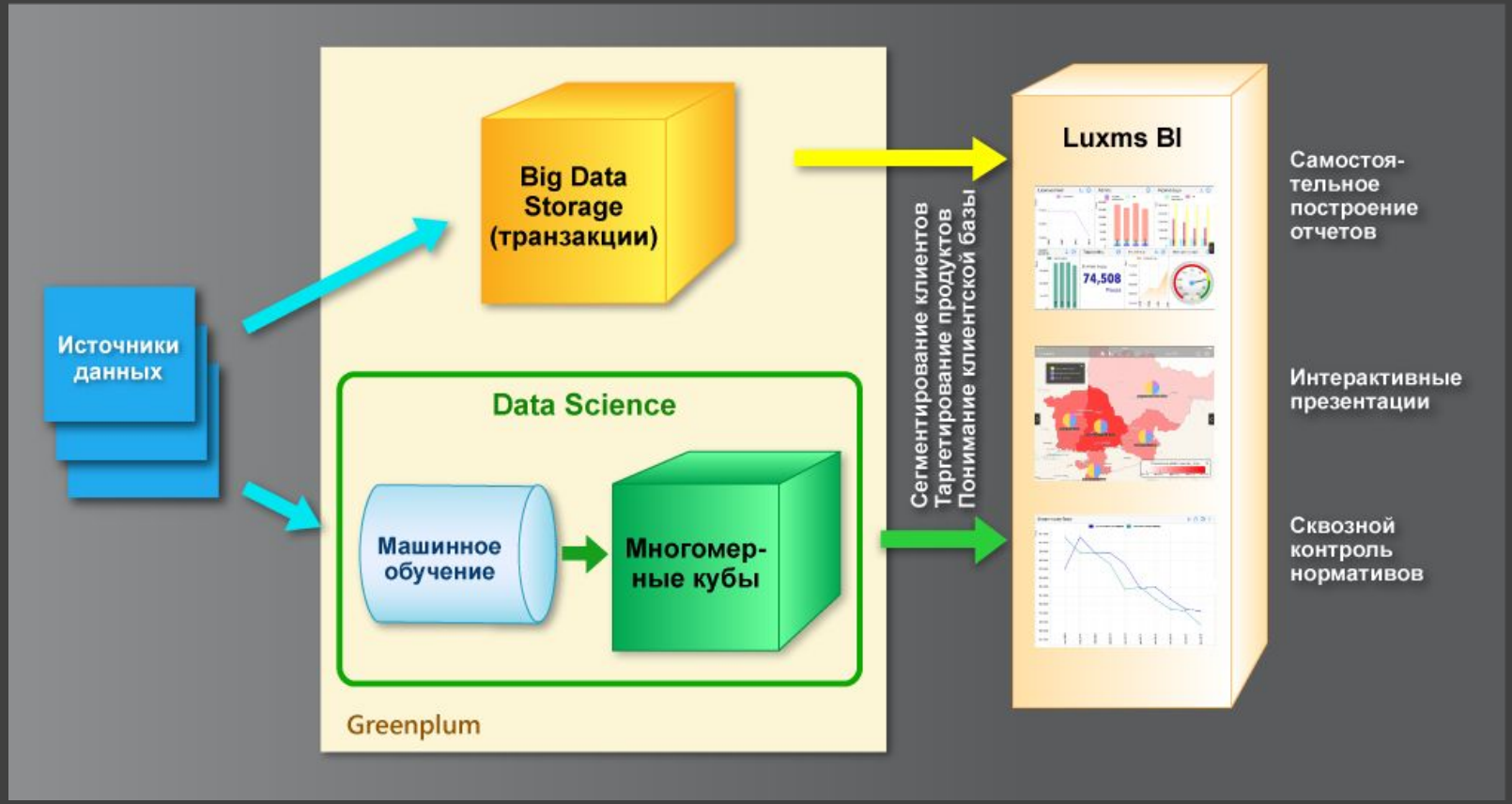
Дмитрий Дорофеев,
Главный архитектор ГК Luxms

Perl, C, C++, Obj-C, Smalltalk, Swift, Haskell, Java, Go, Scheme, JavaScript, PL/SQL, Julia, Python, Nim, Elm, Lisp

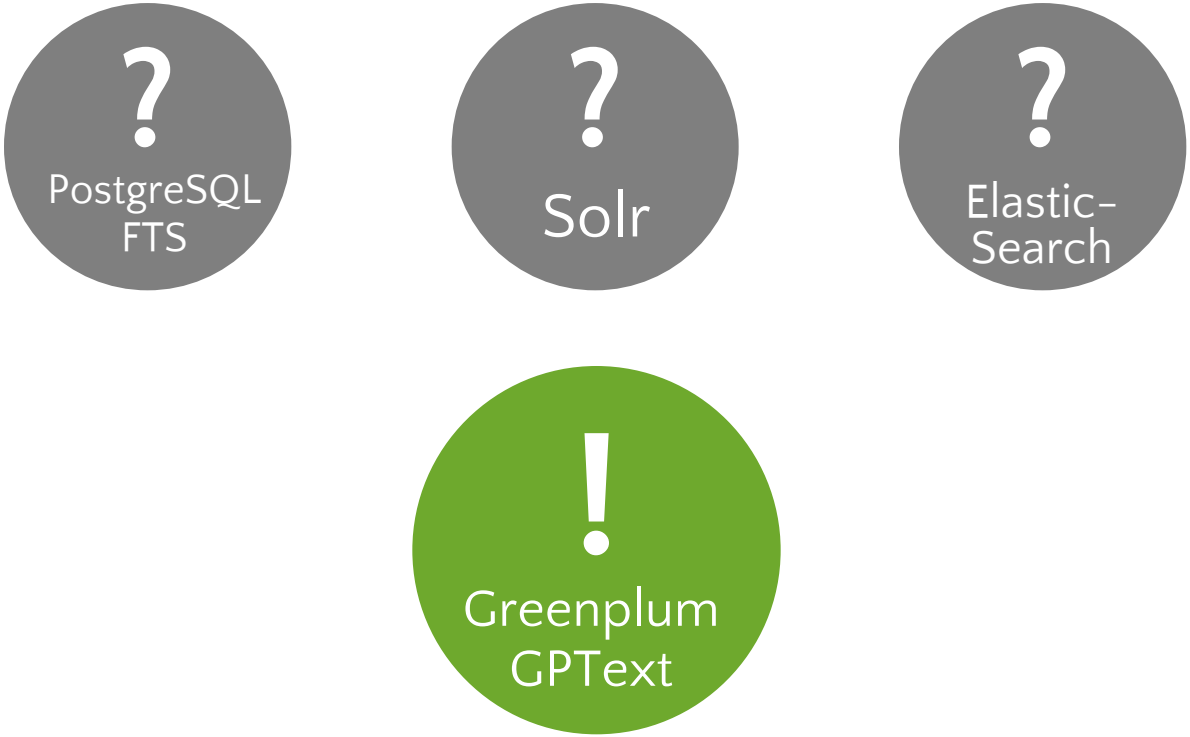
PostgreSQL, Greenplum, MySQL, Oracle, SiriDB, MongoDB, ClickHouse, ScyllaDB, Realm

Nats.io, Kafka, Riemann, Flink, Beam

Luxms BI + Greenplum



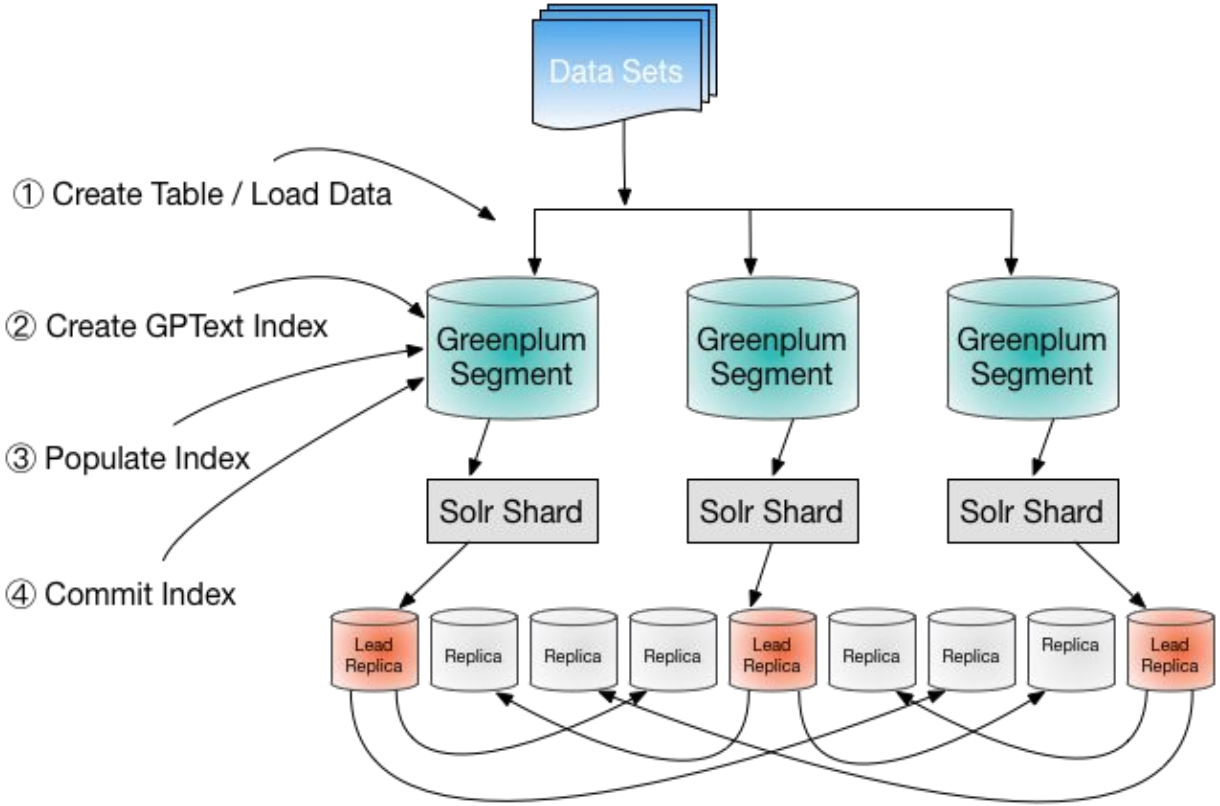
Полнотекстовый поиск



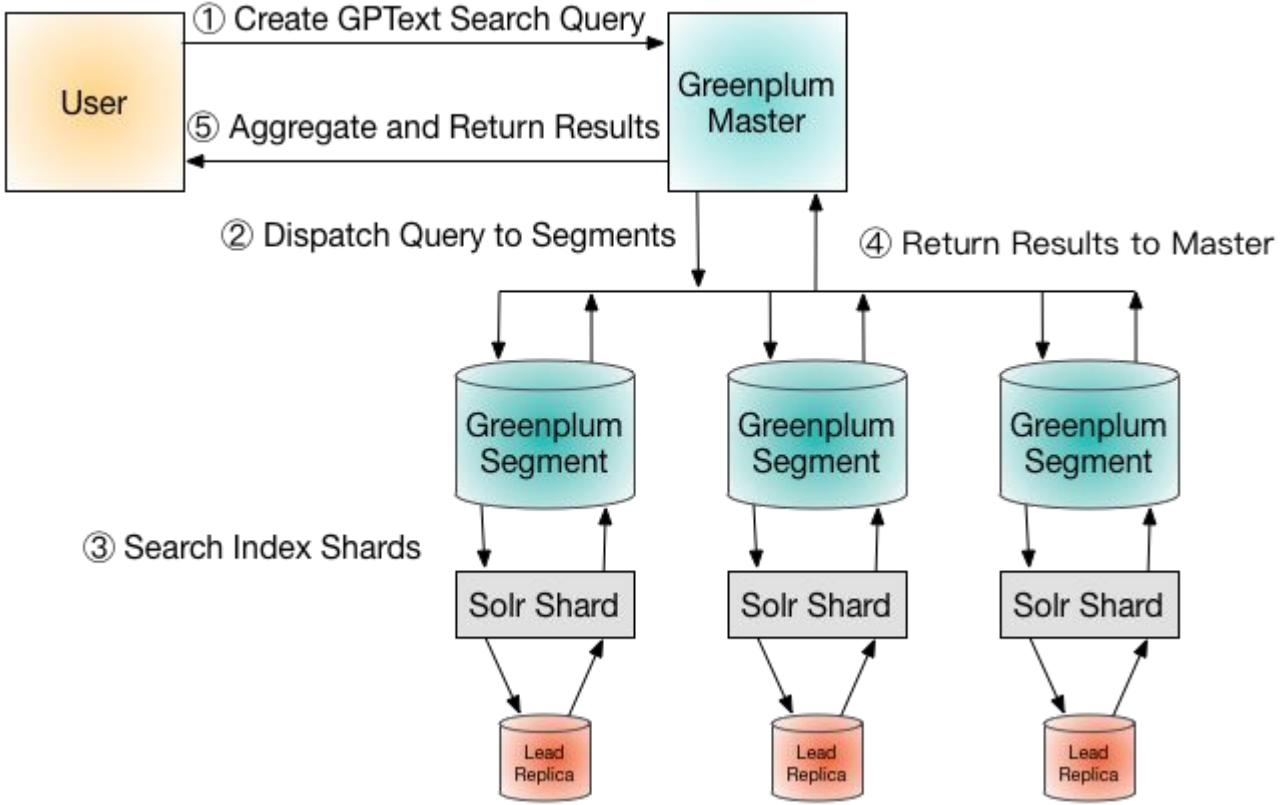
Pivotal GPText

- Greenplum only
- SQL интерфейс к Solr (Apache Lucene)
- Solr сервер работает на каждом сегменте Greenplum
- Можно указать сколько копий Solr (JVM) запускать на каждом сегменте
- Встроенные лексеры и алгоритмы индексации

Pivotal GPText: Index



Pivotal GPText: Query



Подготовка текста: таблица

```
CREATE TABLE enron.emails (
    id          bigint,
    email_id    character varying(15),
    folder      character varying(50),
    "to"        character varying(600),
    "from"      character varying(400),
    cc          character varying(300),
    bcc         character varying(100),
    subject     text,
    content     text,
    date        timestamp without time zone,
    ignorecode integer
) DISTRIBUTED BY (id);
```

Подготовка текста: индекс

```
select gptext.create_index('enron', 'emails', 'id', 'content',
true);

### добавляем поле subject в индекс
##<field name="subject" type="text_intl" indexed="true"
stored="true"/>
# gptext-config -i 'demo.enron.emails' -f managed-schema

// включение опции для генерации термов
select gptext.enable_terms('demo.enron.emails', 'content');
select gptext.enable_terms('demo.enron.emails', 'subject');

select * from gptext.index(table(select * from enron.emails),
'demo.enron.emails');

select gptext.commit_index('demo.enron.emails');
```

Подготовка текста: генерация термов

```
CREATE TABLE email_terms AS SELECT * FROM
gptext.terms(TABLE(SELECT 1 SCATTER BY 1),
'demo.enron.emails', 'content', '*:*', null);

CREATE TABLE subject_terms AS SELECT * FROM
gptext.terms(TABLE(SELECT 1 SCATTER BY 1),
'demo.enron.emails', 'subject', '*:*', null);

demo=# \d subject_terms
   Table "enron.subject_terms"
  Column   |   Type    | Modifiers
-----------+-----------+-----------
 id        | bigint    |
 term      | text      |
 positions | integer[] |
```

Подготовка текста: стоп-слова

```
CREATE TABLE stopwords (word TEXT);

COPY stopwords FROM /tmp/stopwords.txt;
```

ТОП-20 слов на дату

```
CREATE TABLE dict (
  id          BIGINT,
  term        TEXT,
  date        timestamp,
  cnt         INT,
  period_type INT
);

// period_type: 4=дни, 5=недели, 6=месяцы, 7=кварталы,
8=годы
```

ТОП-20 слов термов на дату

```
INSERT INTO dict
SELECT id, term, date, cnt, 4 from (
   SELECT id, term, date, cnt, row_number() OVER
     (partition by date order by cnt desc) AS cnt1 FROM
  ( SELECT min(t.id) AS id, t.term, date_trunc('day', date) AS date,
          sum(array_upper(t.positions,1)) AS cnt
    FROM emails as e
     LEFT JOIN
 (SELECT * FROM email_terms UNION ALL SELECT * FROM subject_terms)
AS t
     ON e.id = t.id
    WHERE t.term NOT IN (select word from stopwords)
        AND length(t.term) > 1 AND t.term !~ $_$^[0-9]+$$_$
    GROUP BY t.term, date_trunc('day', date)
    HAVING sum(array_upper(t.positions,1)) > 2 // более 2-х вхождений
    ORDER BY date_trunc('day', date), sum(array_upper(t.positions,1))
desc
  ) AS s
) AS x WHERE cnt1 < 21;
```

ТОП-20 слов термов на дату

```
CREATE TABLE words (
  id          BIGINT,
  term        TEXT,
  period_id   BIGINT,
  date        timestamp without time zone,
  cnt         INT,
  word        TEXT,
  period_type INT
--PRIMARY KEY (period_id, term)
) WITH (appendonly=true, orientation=column)
DISTRIBUTED BY (period_id);

CREATE RULE "words_on_duplicate_ignore" AS ON INSERT TO "words"
  WHERE EXISTS(SELECT 1 FROM words
               WHERE (period_id, term)=(NEW.period_id, NEW.term))
  DO INSTEAD NOTHING;
```

Восстановление слов из термов

```
-- поиск терма по subject и по content
CREATE OR REPLACE FUNCTION
enron.recover_word(_id BIGINT, _term TEXT)
RETURNS TEXT LANGUAGE 'sql' AS $body$
SELECT
coalesce(
  substring(gptext.highlight(content, 'content', hs) from
$$%<em>#"[^<]+#"</em>%$$ for '#'),
  substring(gptext.highlight(subject, 'subject', hs) from
$$%<em>#"[^<]+#"</em>%$$ for '#')
)
FROM gptext.search(table(select 1 scatter by 1), 'demo.enron.emails',
'id:' || $1 || ' AND (content:"' || $2 || '" OR subject:"'|| $2 ||'")', null,
'hl=true&hl.fl=content,subject') l, emails r
WHERE l.id=r.id;
$body$;
// 'id:1234 AND (content:"shar" OR subject:"shar")'
```

Восстановление слов из термов

```
#!/bin/sh

export PGOPTIONS='-c search_path=enron,public';
for a in {0..1200000..10000}; do

(echo '\timing on'; echo "select enron.recover_words($a, 10000);") |
psql -d demo

done
```

Восстановление слов из термов

```
// enron.recover_words(_start BIGINT, _rows BIGINT)

FOR _r IN SELECT * FROM dict WHERE id > _start AND id <= (_start +
_rows)
  LOOP
   _t := enron.recover_word(_r.id, _r.term);
   IF _t IS NULL THEN
      _t := enron.recover_word(_r.id, _r.term || 'e');
   END IF;
   IF _t IS NOT NULL THEN
      INSERT INTO words
      VALUES (_r.id, _r.term, enron.generate_period_id_v1(_r.date,
_r.period_type, 1), _r.date, _r.cnt, _t, _r.period_type);
   END IF;
END LOOP;
```

TOP 20 слов


Included
system
Enron
Power
manage
analysis
Project
generation
Energy
year
Services
price
modelling
Henwood
India
markets
electricity
Data
forecasts
Development

Полнотекстовый поиск по одному столбцу

```
SELECT
    gptext.highlight(content, 'content', hs),
    "to", "from", "cc", "bcc"
FROM demo.enron.emails r
LEFT JOIN
  gptext.search(
    table(select 1 scatter by 1),
    'demo.enron.emails',
    'id:1234 AND content:"share"',
    null,
    'hl=true&hl.fl=content') l
ON l.id=r.id
WHERE r.id = 1234;
```

Полнотекстовый поиск по комбинации столбцов

```
SELECT
    gptext.highlight(content, 'content', hs),
    gptext.highlight(subject, 'subject', hs),
    "to", "from", "cc", "bcc"
FROM demo.enron.emails r
LEFT JOIN
  gptext.search(
    table(select 1 scatter by 1),
    'demo.enron.emails',
    'id:1234 AND (content:"share" OR subject:"share")',
    null,
    'hl=true&hl.fl=content,subject') l
ON l.id=r.id
WHERE r.id = 1234;
```

Фасетный полнотекстовый поиск (агрегация)

```
SELECT
 range_value,  // date
 value_count   // сколько вхождений на эту дату
FROM gptext.faceted_range_search(
        'demo.enron.emails',
        'subject:"share" OR content:"share"',
        null,
        'date',
        '2000-01-01T00:00:00Z',
        '2000-02-01T00:00:00Z',
        '+1DAY'
);
```

Как PostgreSQL может общаться с внешним миром?

- FDW
- dblink
- PL/* (unsafe versions)
- COPY TO|FROM PROGRAM
- extensions

PostgreSQL – FDW - Greenplum

```
postgres=# select * from predictions;
ERROR:  Greenplum Database does not support REPEATABLE READ transactions.
CONTEXT:  Remote SQL command: START TRANSACTION ISOLATION LEVEL
REPEATABLE READ
postgres=# SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION
LEVEL SERIALIZABLE;
SET
postgres=# select count(*) from predictions;
 count
-------
   994
(1 row)

postgres=# SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION
LEVEL READ COMMITED;
```

PostgreSQL – FDW - Greenplum

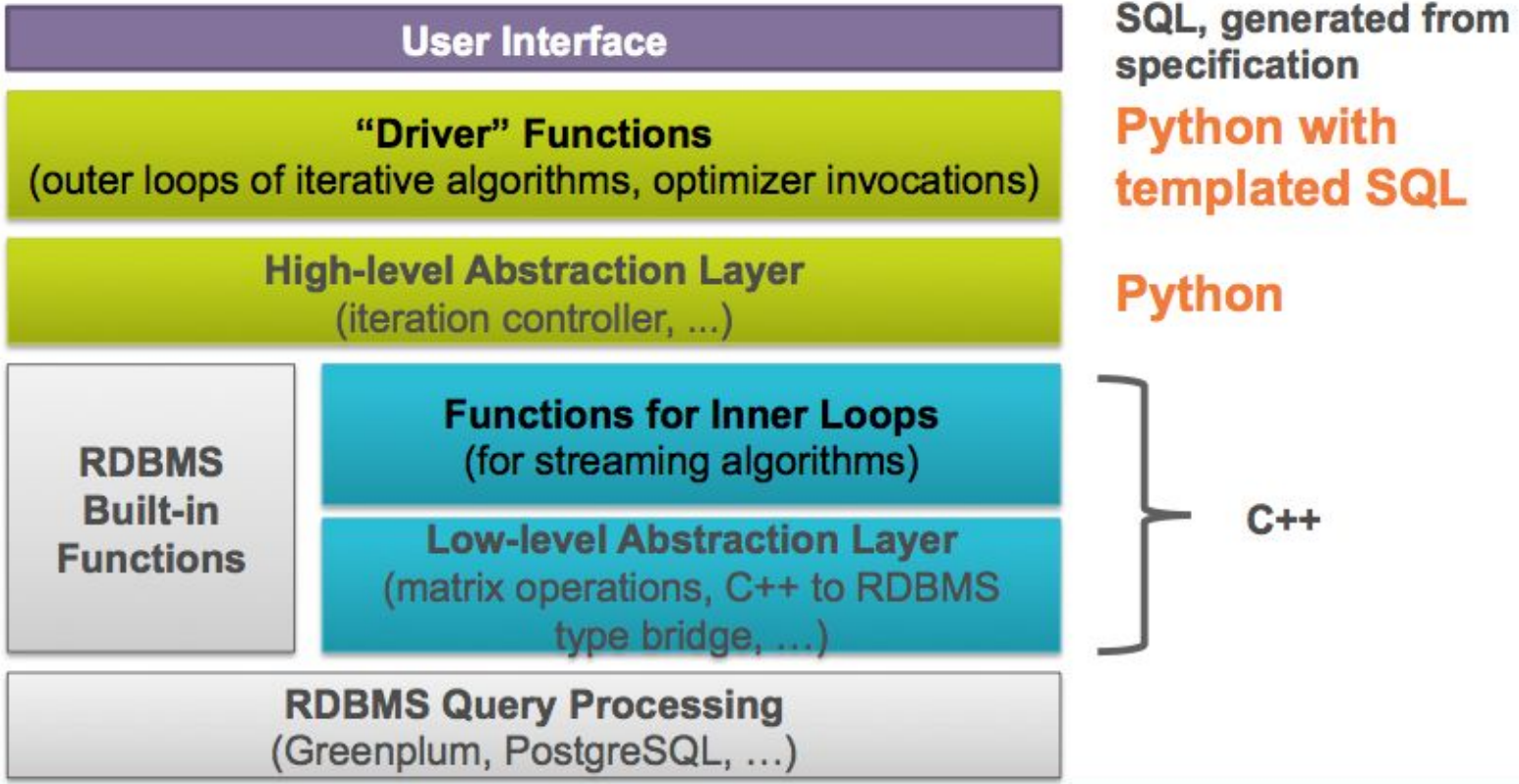
```
postgres=# START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
START TRANSACTION
postgres=# select count(*) from predictions;
 count
-------
   994
(1 row)

postgres=# commit;
COMMIT
postgres=# SHOW transaction_isolation;
 transaction_isolation
-----------------------
 read committed
```

JOIN & dblink

```
SELECT words.value_count as val, p.id::TEXT as period_id
FROM periods AS p LEFT JOIN dblink(
 'dbname=demo port=5432 host=gp user=gp password=gp',
 format($$SELECT range_value, value_count FROM
gptext.faceted_range_search(
  'demo.enron.emails',
  'subject:"%1$s" OR content:"%1$s"',null, 'date', %2$L, %3$L, %4$L)$$,
        'share',
        '2000-01-01T00:00:00Z',
        '2000-02-01T00:00:00Z',
        '+1DAY'
 )
) AS words(
        range_value TIMESTAMP,
        value_count INT)
ON (p.start_time::TIMESTAMP = words.range_value::TIMESTAMP)
WHERE p.period_type = 4 AND p.id >= '2000-01-01' AND p.id <=
'2000-02-01'
ORDER BY p.id;
```

MADlib 1.11


User Interface
"Driver" Functions
(outer loops of iterative algorithms, optimizer invocations)
High-level Abstraction Layer
(iteration controller, ...)
RDBMS Built-in Functions
Functions for Inner Loops
(for streaming algorithms)
Low-level Abstraction Layer
(matrix operations, C++ to RDBMS type bridge, ...)
RDBMS Query Processing
(Greenplum, PostgreSQL, ...)
SQL, generated from specification
Python with templated SQL
Python
C++

PGDAY'17 RUSSIA
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Вопросы и ответы ;-)