Using the PostgreSQL Extension Ecosystem for Advanced Analytics

sales@chartio.com
(855) 232-0320
CHARTIO

Agenda

- The problem
  - The prevailing view vs. the practical reality
- A possible solution
  - Or just building blocks?
- Nearness
  - Near at hand, near to our skill set, near to our capabilities
- A more complete solution
  - The PostgreSQL extension ecosystem

The Problem

The Prevailing View
vs.
The Practical Reality

The Prevailing View - Logical

Dimension	Relational	Non-Relational
Schema objects	• Structured rows and columns • Schema on write • Referential integrity • Painful migrations	• Unstructured files, docs, etc • Schema on read • No referential integrity • No migrations
Query languages	• SQL • Declarative • Easy enough for non-tech users	• Various • Procedural • Requires some programming skills
Exploratory analysis	• Native support for joins • Interactive/low execution overhead	• No native support for joins • OLAP - Batch processing
Data science and ML	• Only descriptive statistics • Requires exporting dumps/samples	• Robust ecosystem • Does not require exports

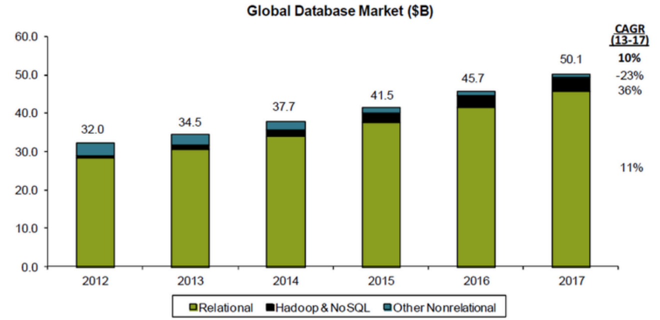
The Prevailing View - Physical

Dimension	Relational	Non-Relational
Parallel query processing	• Single node system • Single process per query	• Multiple node system • Multiple processes per query
Concurrency	• High concurrency • Single process per connection	• OLAP - low concurrency/high scheduling overhead
High Availability & Replication	• Async and sync replication • HA may not be native	• Async and sync replication • HA likely to be native
Sharding	• Sharding may not be native • Difficult to manage	• Sharding likely to be native • Easy to manage

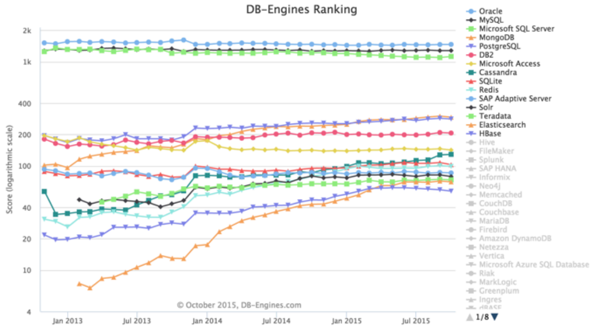
The Prevailing View - Summary

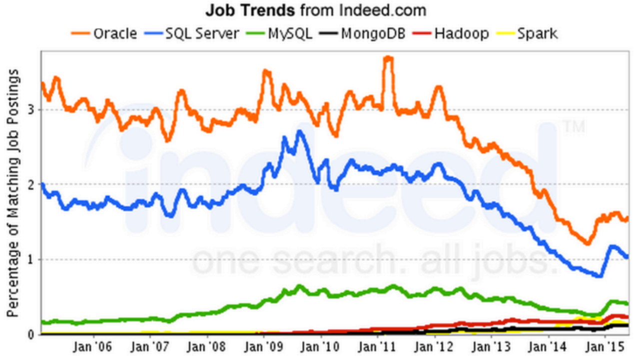
- RDBMS have nice properties for producing rich data
  - ACID, relational integrity, constraints, strong data types
- Easier for non-tech users and exploratory analysis
- Probably don’t meet the needs of today’s analysts
  - Data science & Machine Learning
  - Parallel processing
- Definitely don’t meet the needs of today’s apps
  - Schema migrations
  - Replication and sharding

The Practical Reality



Source: IDC, Bernstein analysis





The Practical Reality

But we still want more advanced functionality.

A Possible Solution

Or Just Building Blocks?

Modern SQL

- Many people still think of SQL in terms of SQL-92
- Since then we've had: SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011
- http://use-the-index-luke.com/blog/2015-02/modern-sql
  - Common Table Expressions (CTEs) / Recursive CTEs
  - Window Functions
  - Ordered-set Aggregates
  - Lateral joins
  - Temporal support
  - The list goes on...

Procedural Languages

- Native

pgSQL | Tcl | Perl | Python

- Community

Java | PHP | R | Javascript | Ruby | Scheme | sh

sales@chartio.com
(855) 232-0320
CHARTIO

Building Blocks

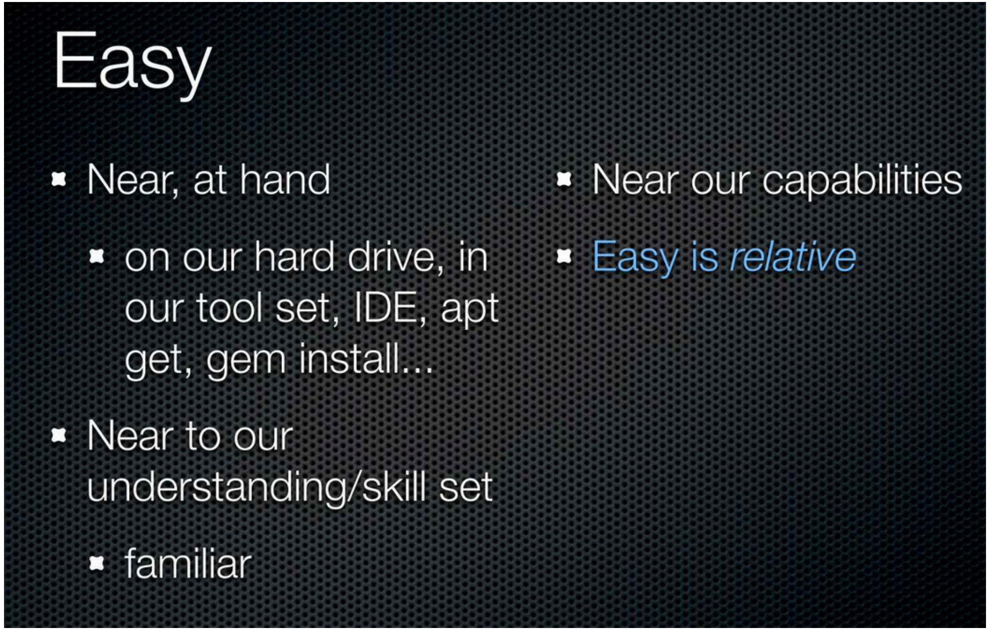
These solve some problems.
For others, they are just building blocks.

Nearness

Near at Hand
Near to Our Skill Set
Near to Our Capabilities

Nearness

- http://www.infoq.com/presentations/Simple-Made-Easy



sales@chartio.com
(855) 232-0320

Nearness Drives Adoption

- Near at hand
  - Easily installable
- Near to our skill set
  - Familiar tool/language/abstraction
  - Modular and composable
- Near to our capabilities
  - Capable of solving a problem in our domain

A More Complete Solution

The PostgreSQL Extension Ecosystem

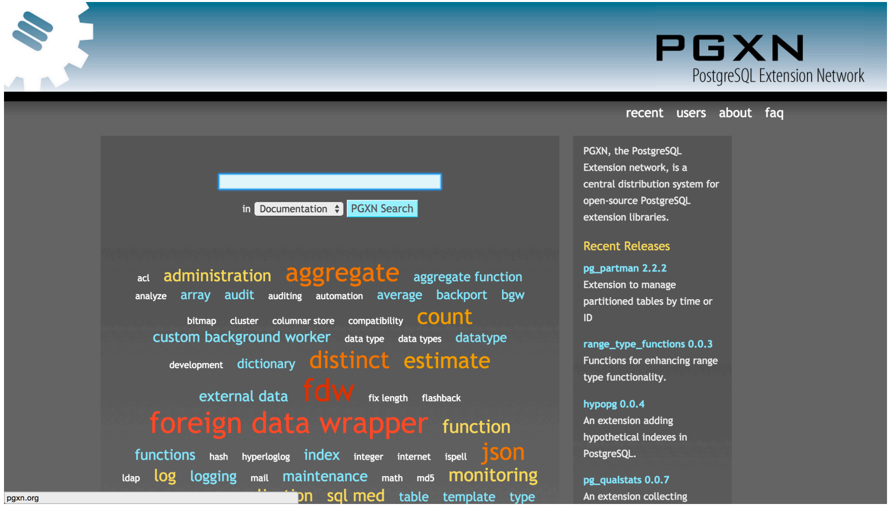
Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & operators: https://github.com/eulerto/pg_similarity
- UDAs & data types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

Postgres Extension Ecosystem Examples

- **PostgreSQL Extension Network: http://pgxn.org/**
- UDFs & operators: https://github.com/eulerto/pg_similarity
- UDAs & data types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

The PostgreSQL Extension Network



- Package Manager: pgxn
- Index/Network: http://pgxn.org/
- PyPI, RubyGems, CPAN, CRAN

The PostgreSQL Extension Network

- Near at hand
  - pgxn search semver
  - pgxn info semver
  - pgxn install semver
  - pgxn load –d somedb semver
  - pgxn unload –d somedb semver
  - pgxn uninstall semver

- Search github? google? mailing list?
- Github README?
- git clone; make; make install;
- psql –c “CREATE EXTENSION IF NOT EXISTS”
- psql –c “DROP EXTENSION IF EXISTS”
- make uninstall?

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- **UDFs & operators: https://github.com/eulerto/pg_similarity**
- UDAs & data types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

UDFs & Operators: pg_similarity

- Near to our capabilities
- Similarity coefficient algorithms
  - L1 Distance
  - Cosine Distance
  - Dice Coefficient
  - Euclidean Distance
  - Hamming Distance
  - Jaccard Coefficient
  - Jaro Distance
  - Jaro-Winkler Distance
  - Levenshtein Distance
  - Matching Coefficient
  - Monge-Elkan Coefficient
  - Needleman-Wunsch Coefficient
  - Overlap Coefficient
  - Q-Gram Distance
  - Smith-Waterman Coefficient
  - Smith-Waterman-Gotoh Coefficient
  - Soundex Distance

UDFs & Operators: pg_similarity

- Near to our skill set

```
mydb=# select a, b, cosine(a,b), jaro(a, b), euclidean(a, b) from foo, bar;
            a             |           b            |  cosine  |   jaro   | euclidean
--------------------------+------------------------+----------+----------+-----------
 Euler                    | Euler T. de Oliveira   |      0.5 |     0.75 |  0.579916
 Euler                    | Euller                 |        0 | 0.944444 |         0
 Euler                    | Oliveira, Euler Taveira|  0.57735 | 0.605797 |  0.552786
 Euler                    | Sr. Oliveira           |        0 | 0.505556 |  0.225403
 Oiler                    | Euler T. de Oliveira   |        0 | 0.472222 |  0.457674
 Oiler                    | Euller                 |        0 |      0.7 |         0
 Oiler                    | Oliveira, Euler Taveira|        0 | 0.672464 |  0.367544
 Oiler                    | Sr. Oliveira           |        0 | 0.672222 |  0.225403
 Euler Taveira de Oliveira | Euler T. de Oliveira   |     0.75 |  0.79807 |      0.75
 Euler Taveira de Oliveira | Euller                 |        0 | 0.677778 |  0.457674
 Euler Taveira de Oliveira | Oliveira, Euler Taveira| 0.866025 | 0.773188 |       0.8
 Euler Taveira de Oliveira | Sr. Oliveira           | 0.353553 | 0.592222 |  0.552786
 Maria Taveira dos Santos | Euler T. de Oliveira   |        0 |  0.60235 |       0.5
 Maria Taveira dos Santos | Euller                 |        0 | 0.305556 |  0.457674
 Maria Taveira dos Santos | Oliveira, Euler Taveira| 0.288675 | 0.535024 |  0.552786
 Maria Taveira dos Santos | Sr. Oliveira           |        0 | 0.634259 |  0.452277
 Carlos Santos Silva      | Euler T. de Oliveira   |        0 | 0.542105 |   0.47085
 Carlos Santos Silva      | Euller                 |        0 | 0.312865 |  0.367544
 Carlos Santos Silva      | Oliveira, Euler Taveira|        0 | 0.606662 |   0.42265
 Carlos Santos Silva      | Sr. Oliveira           |        0 | 0.507728 |  0.379826
(20 rows)
```

```
mydb=# set pg_similarity.qgram_threshold to 0.7;
SET
mydb=# show pg_similarity.qgram_threshold;
 pg_similarity.qgram_threshold
-------------------------------
 0.7
(1 row)

mydb=# select a, b,qgram(a, b) from foo, bar where a ~~~ b;
            a             |           b            |  qgram
--------------------------+------------------------+----------
 Euler                    | Euller                 |      0.8
 Euler Taveira de Oliveira | Euler T. de Oliveira   |  0.77551
 Euler Taveira de Oliveira | Oliveira, Euler Taveira | 0.807692
(3 rows)

mydb=# set pg_similarity.qgram_threshold to 0.35;
SET
mydb=# select a, b,qgram(a, b) from foo, bar where a ~~~ b;
            a             |           b            |  qgram
--------------------------+------------------------+----------
 Euler                    | Euler T. de Oliveira   | 0.413793
 Euler                    | Euller                 |      0.8
 Oiler                    | Euller                 |      0.4
 Euler Taveira de Oliveira | Euler T. de Oliveira   |  0.77551
 Euler Taveira de Oliveira | Oliveira, Euler Taveira | 0.807692
 Euler Taveira de Oliveira | Sr. Oliveira           | 0.439024
(6 rows)
```

UDFs & Operators: pg_similarity

- Implementation

block.c	updating copyright year.	4 years ago
cosine.c	updating copyright year.	4 years ago
dice.c	updating copyright year.	4 years ago
euclidean.c	updating copyright year.	4 years ago
hamming.c	updating copyright year.	4 years ago
jaccard.c	updating copyright year.	4 years ago
jaro.c	updating copyright year.	4 years ago
levenshtein.c	Fix outstanding defects.	7 months ago
matching.c	updating copyright year.	4 years ago
mongeelkan.c	Fix outstanding defects.	7 months ago
needlemanwunsch.c	updating copyright year.	4 years ago
overlap.c	updating copyright year.	4 years ago
pg_similarity--1.0.sql	add support to CREATE EXTENSION.	3 years ago
pg_similarity--unpackaged--1…	add support to CREATE EXTENSION.	3 years ago
pg_similarity.conf.sample	updating sample configuration file	4 years ago
pg_similarity.control	add support to CREATE EXTENSION.	3 years ago
pg_similarity.sql.in	add support to CREATE EXTENSION.	3 years ago
qgram.c	updating copyright year.	4 years ago
similarity.c	updating copyright year.	4 years ago
similarity.h	Add another hamming function. The former are for varbit parameters. The	4 years ago
similarity_gin.c	updating copyright year.	4 years ago
smithwaterman.c	Fix outstanding defects.	7 months ago

Branch: master ▾ pg_similarity / pg_similarity--1.0.sql

eulerto add support to CREATE EXTENSION. c12cc00 on Sep 25, 2012

1 contributor

381 lines (318 sloc) | 9.17 KB Raw Blame History

```
-- keep this file in sync with the pg_similarity.sql.in legacy install file

-- complain if script is sourced in psql, rather than via CREATE EXTENSION
\echo Use "CREATE EXTENSION pg_similarity" to load this file. \quit

-- Block
CREATE FUNCTION block (text, text) RETURNS float8
AS 'MODULE_PATHNAME', 'block'
LANGUAGE C IMMUTABLE STRICT;

CREATE FUNCTION block_op (text, text) RETURNS bool
AS 'MODULE_PATHNAME', 'block_op'
LANGUAGE C STABLE STRICT;

CREATE OPERATOR ~++ (
        LEFTARG = text,
        RIGHTARG = text,
        PROCEDURE = block_op,
        COMMUTATOR = '~++',
        RESTRICT = contsel,
        JOIN = contjoinsel
);

-- Cosine
CREATE FUNCTION cosine (text, text) RETURNS float8
AS 'MODULE_PATHNAME', 'cosine'
```

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- **UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll**
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

UDAs & Data Types: postgresql-hll

- Near to our capabilities & near to our skill set
- Data type
  - Estimate count distinct with tunable precision
  - 1280 bytes estimates tens of billions of distinct values with few percent error

```
CREATE TABLE facts (
    date            date,
    user_id         integer,
    activity_type   smallint,
    referrer        varchar(255)
);
```

```
-- Create the destination table
CREATE TABLE daily_uniques (
    date            date UNIQUE,
    users           hll
);

-- Fill it with the aggregated unique statistics
INSERT INTO daily_uniques(date, users)
    SELECT date, hll_add_agg(hll_hash_integer(user_id))
    FROM facts
    GROUP BY 1;
```

UDAs & Data Types: postgresql-hll

We're first hashing the `user_id`, then aggregating those hashed values into one `hll` per day. Now we can ask for the cardinality of the `hll` for each day:

```
SELECT date, hll_cardinality(users) FROM daily_uniques;
```

Or the monthly uniques for this year?

```
SELECT EXTRACT(MONTH FROM date) AS month, hll_cardinality(hll_union_agg(users))
FROM daily_uniques
WHERE date >= '2012-01-01' AND
      date <  '2013-01-01'
GROUP BY 1;
```

Or how about a sliding window of uniques over the past 6 days?

```
SELECT date, #hll_union_agg(users) OVER seven_days
FROM daily_uniques
WINDOW seven_days AS (ORDER BY date ASC ROWS 6 PRECEDING);
```

UDAs & Data Types: postgresql-hll

- Implementation

Branch: **master** postgresql-hll / +

timonk Merge (and squash) #23. @cwelton added Travis CI integration. Latest commit 63de8ac on Oct 15, 2014

regress	Merge (and squash) #23. **@cwelton** added Travis CI integration.	a year ago
testdata	First commit, v2.7	3 years ago
.gitignore	Merge (and squash) #23. **@cwelton** added Travis CI integration.	a year ago
.travis.yml	Merge (and squash) #23. **@cwelton** added Travis CI integration.	a year ago
CHANGELOG.markdown	Version bumped to 2.10.0.	2 years ago
DEVELOPER.markdown	Fix Makefile and hll--XYZ.sql to reflect correct version number.	2 years ago
LICENSE	Updated license.	2 years ago
Makefile	Fix Makefile and hll--XYZ.sql to reflect correct version number.	2 years ago
MurmurHash3.cpp	Fix GCC 4.8 compiler warnings.	2 years ago
MurmurHash3.h	First commit, v2.7	3 years ago
README.markdown	Merge (and squash) #23. **@cwelton** added Travis CI integration.	a year ago
REFERENCE.markdown	Added references to new storage spec and java-hll.	2 years ago
hll--2.10.0.sql	Fix Makefile and hll--XYZ.sql to reflect correct version number.	2 years ago
hll.c	Binary input/output for hll type.	2 years ago
hll.control	Fixed #18, put correct version number in hll.control.	2 years ago
postgresql-hll.spec	Fixed bug in spec file that pointed to wrong sql file. Added gitignor…	a year ago

```sql
CREATE FUNCTION hll(hll, integer, boolean)
RETURNS hll
AS 'MODULE_PATHNAME'
LANGUAGE C STRICT IMMUTABLE;

CREATE TYPE hll (
        INTERNALLENGTH = variable,
        INPUT = hll_in,
        OUTPUT = hll_out,
        TYPMOD_IN = hll_typmod_in,
        TYPMOD_OUT = hll_typmod_out,
        RECEIVE = hll_recv,
        SEND = hll_send,
        STORAGE = external
);
```

```sql
-- Union aggregate function, returns hll.
--
CREATE AGGREGATE hll_union_agg (hll) (
       SFUNC = hll_union_trans,
       STYPE = internal,
       FINALFUNC = hll_pack
);
```

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll
- **Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery**
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

Foreign Data Wrappers: API

53.2.1. FDW Routines For Scanning Foreign Tables

```
void
GetForeignRelSize (PlannerInfo *root,
                   RelOptInfo *baserel,
                   Oid foreigntableid);
```

53.2.2. FDW Routines For Updating Foreign Tables

If an FDW supports writable foreign tables, it should provide some or all

```
void
AddForeignUpdateTargets (Query *parsetree,
                         RangeTblEntry *target_rte,
                         Relation target_relation);
```

53.2.3. FDW Routines for EXPLAIN

```
void
ExplainForeignScan (ForeignScanState *node,
                    ExplainState *es);
```

53.2.4. FDW Routines for ANALYZE

```
bool
AnalyzeForeignTable (Relation relation,
                     AcquireSampleRowsFunc *func,
                     BlockNumber *totalpages);
```

Foreign Data Wrappers: multicorn

- Near to our skill set

```python
from multicorn import ForeignDataWrapper

class ConstantForeignDataWrapper(ForeignDataWrapper):

    def __init__(self, options, columns):
        super(ConstantForeignDataWrapper, self).__init__(options, columns
        self.columns = columns

    def execute(self, quals, columns):
        for index in range(20):
            line = {}
            for column_name in self.columns:
                line[column_name] = '%s %s' % (column_name, index)
            yield line
```

```python
def get_rel_size(self, quals, columns):
```

Write API

Since PostgreSQL 9.3, foreign data wrappers ca

In multicorn, this involves defining which column
following methods at your discretion:

```python
def insert(self, new_values)
def update(self, old_values, new_values)
def delete(self, old_values)
```

```python
def commit(self)
def rollback(self)
def pre_commit(self)
```

Foreign Data Wrappers: pgosquery

- Near at hand

```
CREATE SERVER pgosquery_srv foreign data wrapper multicorn options (
    wrapper 'pgosquery.PgOSQuery'
);

CREATE FOREIGN TABLE processes (
    pid integer,
    name character varying,
    username character varying
) server pgosquery_srv options (
    tabletype 'processes'
);

CREATE FOREIGN TABLE listening_ports (
    pid integer,
    address character varying,
    port integer
) server pgosquery_srv options (
    tabletype 'listening_ports'
);
```

```
---------------------------------------------------------------
-- get the name, pid and attached port of all processes
-- which are listening on localhost interfaces
---------------------------------------------------------------
SELECT DISTINCT
    process.name,
    listening.port,
    process.pid
FROM processes AS process
JOIN listening_ports AS listening
ON process.pid = listening.pid
WHERE listening.address = '127.0.0.1';
```

```
   name    | port | pid
-----------+------+------
 postgres | 5432 | 6932

(1 row)
```

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- **Indexes: https://github.com/zombodb/zombodb**
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

Indexes: ZomboDB

```
CREATE INDEX idx_zdb_products
          ON products
       USING zombodb(zdb('products', products.ctid), zdb(products))
        WITH (url='http://localhost:9200/');
```

```
tutorial=# SELECT * FROM products WHERE zdb('products', products.ctid) ==> 'sports or box';
 id |   name   |            keywords             |         short_summary          |
----+----------+---------------------------------+--------------------------------+------------------------------
  4 | Box      | {wooden,box,"negative space"} | Just an empty box made of wood | A wooden container that will e
  2 | Baseball | {baseball,sports}               | It's a baseball                | Throw it at a person with a bi
(2 rows)
```

- Index Access Method API
  - http://www.postgresql.org/docs/9.4/static/indexam.html

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes (GiST, GIN): https://github.com/zombodb/zombodb
- **Composing Extension Methods: http://doc.madlib.net/**
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

Composing Extension Methods: MADlib

Near to our capabilities

- **Supervised Learning**
  - Regression Models
    - Clustered Variance
    - Cox-Proportional Hazards Regression
    - Elastic Net Regularization
    - Generalized Linear Models
    - Linear Regression
    - Logistic Regression
    - Marginal Effects
    - Multinomial Regression
    - Ordinal Regression
    - Robust Variance
  - Tree Methods
    - Decision Tree
    - Random Forest
  - Conditional Random Field

- **Unsupervised Learning**
  - Association Rules
    - Apriori Algorithm
  - Clustering
    - k-Means Clustering
  - Topic Modelling
    - Latent Dirichlet Allocation

- **Time Series Analysis**
  - ARIMA

- **Model Evaluation**
  - Cross Validation

- **Statistics**
  - Descriptive Statistics
    - Summary
    - Pearson's Correlation
  - Inferential Statistics
    - Hypothesis Tests
  - Probability Functions

- **Utility Functions**
  - Developer Database Functions
  - Linear Solvers
    - Dense Linear Systems
    - Sparse Linear Systems
  - PMML Export
  - Text Analysis
    - Term Frequency

Composing Extension Methods: MADlib

- Near to our skill set

1. Create the training data table.

```
CREATE TABLE patients( id INTEGER NOT NULL,
                       second_attack INTEGER,
                       treatment INTEGER,
                       trait_anxiety INTEGER);
COPY patients FROM STDIN WITH DELIMITER '|';
  1 |             1 |           1 |            70
  3 |             1 |           1 |            50
  5 |             1 |           0 |            40
  7 |             1 |           0 |            75
  9 |             1 |           0 |            70
 11 |             0 |           1 |            65
 13 |             0 |           1 |            45
 15 |             0 |           1 |            40
 17 |             0 |           0 |            55
 19 |             0 |           0 |            50
  2 |             1 |           1 |            80
  4 |             1 |           0 |            60
  6 |             1 |           0 |            65
  8 |             1 |           0 |            80
 10 |             1 |           0 |            60
 12 |             0 |           1 |            50
 14 |             0 |           1 |            35
 16 |             0 |           1 |            50
 18 |             0 |           0 |            45
 20 |             0 |           0 |            60
\.
```

2. Train a regression model.

```
SELECT madlib.logregr_train( 'patients',
                             'patients_logregr',
                             'second_attack',
                             'ARRAY[1, treatment, trait_anxiety]',
                             NULL,
                             20,
                             'irls'
                           );
```

3. View the regression results.

```
-- Set extended display on for easier reading of output
\x on
SELECT * from patients_logregr;
```

Result:

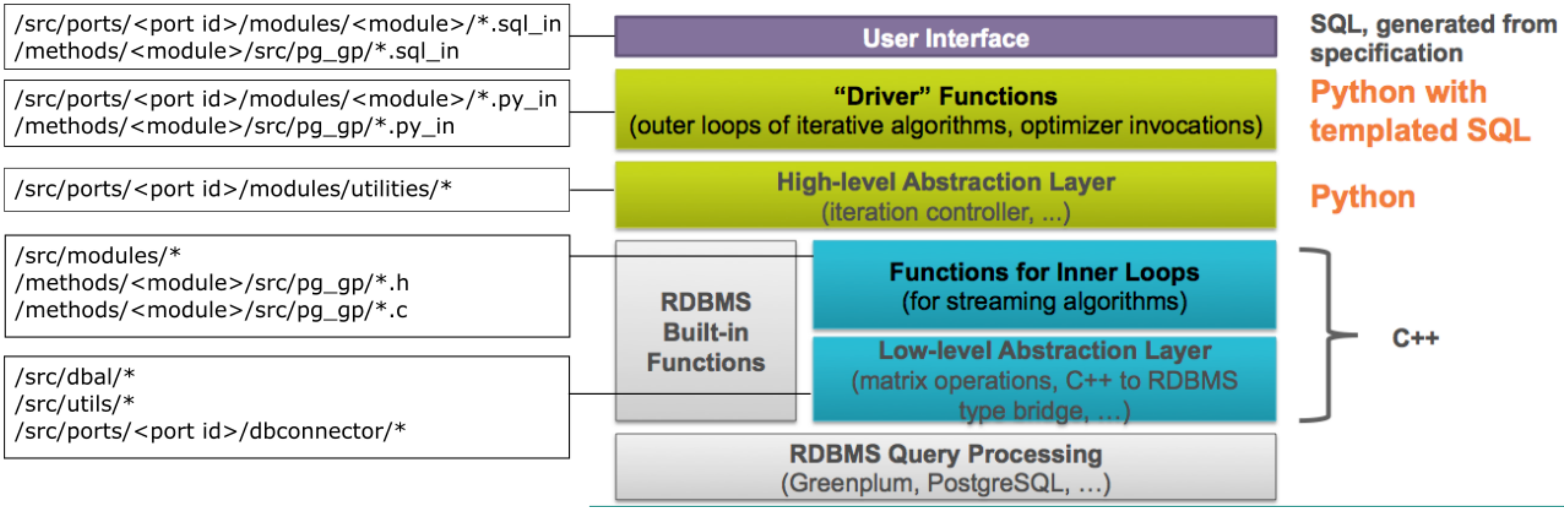
```
coef           | {5.59049410898112,2.11077546770772,-0.237276684606453}
log_likelihood | -467.214718489873
std_err        | {0.318943457652178,0.101518723785383,0.294509929481773}
z_stats        | {17.5281667482197,20.7919819024719,-0.805666162169712}
p_values       | {8.73403463417837e-69,5.11539430631541e-96,0.420435365338518}
odds_ratios    | {267.867942976278,8.2546400100702,0.788773016471171}
condition_no   | 179.186118573205
num_iterations | 9
```

5. Predicting dependent variable using the logistic regression model. (This example uses the original data table to perf
Typically a different test dataset with the same features as the original training dataset would be used for prediction.

```
\x off
-- Display prediction value along with the original value
SELECT p.id, madlib.logregr_predict(coef, ARRAY[1, treatment, trait_anxiety]),
       p.second_attack
FROM patients p, patients_logregr m
ORDER BY p.id;
```

sales@chartio.com
(855) 232-0320
CHARTIO

Composing Extension Methods: MADlib



Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- **MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb**
- Composing Extensions
  - Custom Background Workers: https://github.com/no0p/alps
  - Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/

Parallel Processing

citusdata

Pivotal GPDB

Postgres-XL

- Parallel sequential scan
  - http://rhaas.blogspot.com/2015/11/parallel-sequential-scan-is-committed.html

- Columnar FDW:
  - https://github.com/citusdata/cstore_fdw

Postgres Extension Ecosystem Examples

- PostgreSQL Extension Network: http://pgxn.org/
- UDFs & Operators: https://github.com/eulerto/pg_similarity
- UDAs & Data Types: https://github.com/aggregateknowledge/postgresql-hll
- Foreign Data Wrappers: http://multicorn.org/, https://github.com/shish/pgosquery
- Indexes: https://github.com/zombodb/zombodb
- Composing Extension Methods: http://doc.madlib.net/
- MPP: https://www.citusdata.com/, https://github.com/greenplum-db/gpdb
- **Composing Extensions**
  - **Custom Background Workers: https://github.com/no0p/alps**
  - **Record linking: http://no0p.github.io/2015/10/20/record_linking.html#/**

Composing Extensions: Alps

Alps

This extension implements a postgres background worker which builds generic statistical models.

It's kind of like an autovacuum which updates parameter vectors rather than removing dead tuples.

Usage

Alps will modify all tables in the target database and add new fields. The fields are the names of existing fields on a table concatenated by "__predicted". Thus if you have a table of housing prices, ...

```
select id, sq_feet, zipcode, price
```

Alps would add a column *price__predicted*

A common use for filling in nulls is...

```
select id, coalesce(price, price__predicted);
```

This prototype of Alps will attempt to create __predicted columns for any boolean or numeric field (numeric, float, int).

Composing Extensions: Record Linking

Example Training Table

```
robert=# \d modeling.real_humans
        Table "modeling.real_humans"
        Column           |  Type    | Modifiers
-------------------------+----------+-----------
 a_id                    | integer  |
 b_id                    | integer  |
 name_jaro_winkler       | numeric  |
 name_trigram_distance   | numeric  |
 true_pair               | boolean  |
```

Training a Model

```
SELECT madlib.logregr_train(
  'modeling.real_humans',                -- source table
  'modeling.real_humans_linking',        -- out table
  'true_pair',                           -- dependent variable column name
  'ARRAY[ name_jaro_winkler,
          name_trigram_distance]'        -- independent variable columns
);
```

Beyond Analytics

- Web app framework
  - http://blog.aquameta.com/
- REST API
  - https://github.com/begriffs/postgrest
- Unit testing framework
  - http://pgtap.org/
- Firewall
  - https://github.com/uptimejp/sql_firewall
- More every week!

Conclusion

- With PostgreSQL, you get
  - more than rows and columns
  - more than SELECT, FROM, WHERE, GROUP BY, ORDER BY
  - more than a single machine
- Make sure you get the full return on your investment!

Get your Chartio free trial!

sales@chartio.com

(855) 232-0320