Querying Massive Data Sets in the Cloud with Google BigQuery and Java

Kon Soulianidis
JavaOne 2014

Agenda

- ‘Massive Data’ Qualification
- A Big Data Problem
- What is BigQuery?
- BigQuery Java API
- Getting data in & out
- BigQuery streaming meets Java 8 Streams

Massive Data Credentials

- Have you ever written a report?
- Has it taken a long time to run?
  - > 1 minutes
  - > 5 minutes
  - > 20 minutes
  - > 1 hour
  - > 1 day
- Hadoop & Other BigData tools?

Our Project

And why BigQuery over everything else?

Client Problem Domain

- Client runs a number of popular Australian websites
- Knows who is logged in
- 3rd party analytics service tracks site visits
- Reports of what their users were looking at, and encoded demographic data
- A years worth of logs was 30 billion rows, 15 TB

Open Questions

- Which groups (demographics) were looking at which sites?
- Were there patterns?
- Could they adapt their content?
- Could they change advertising shown based on who used the site most?
- Can we have real time analysis and live dashboards of KPIs?

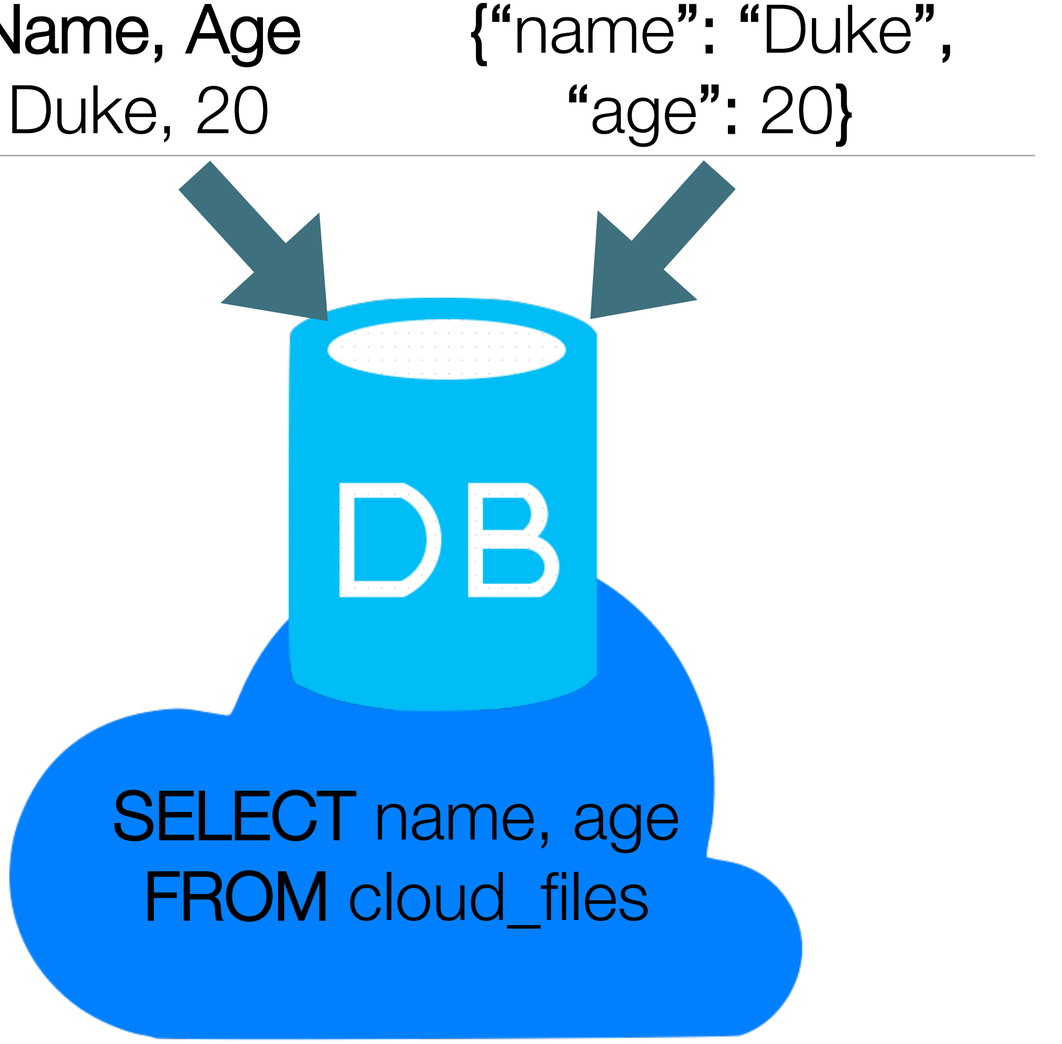
Solutions

- Batch processes to pull new logs into BQ
- Single Page WebApp for reports (JS)
- Dashboard WebApp
  - Spring Boot backend app polling BQ via the API
  - Clients connecting via web-sockets, receiving updates every 3 seconds.

What is BigQuery?

What is BigQuery?

- A DataStore
- Hosted in the Cloud
- Uses SQL
- Queries Hierarchical Data Structures
  - Sourced from JSON and CSVs
- Blazingly Fast



Just how fast?

- Analysing a months worth of web impressions
  - 1.7 Billion Rows
  - 700 GB
  - **10 seconds**
- Aggregating a single TB of data
  - **25 seconds**
- Analysing a years worth of web impressions
  - 30 Billion Rows
  - 15 Terabytes
  - **330 seconds**

Demo of the BigQuery Web UI

Google bigquery

COMPOSE QUERY

Query History

Job History

JavaOneTalk

- InsertTest
  - wikipedia_title_edit_count
- gdelt-bq:full
- gdelt-bq:sample_views
- publicdata:samples
  - github_nested
  - github_timeline
  - gsod
  - natality
  - shakespeare
  - trigrams
  - **wikipedia**

Table Details: wikipedia

Schema | Details | Query Table

Schema

title	STRING	REQUIRED	Describe this field...
id	INTEGER	NULLABLE	Describe this field...
language	STRING	REQUIRED	Describe this field...
wp_namespace	INTEGER	REQUIRED	Describe this field...
is_redirect	BOOLEAN	NULLABLE	Describe this field...
revision_id	INTEGER	NULLABLE	Describe this field...
contributor_ip	STRING	NULLABLE	Describe this field...
contributor_id	INTEGER	NULLABLE	Describe this field...
contributor_username	STRING	NULLABLE	Describe this field...
timestamp	INTEGER	REQUIRED	Describe this field...
is_minor	BOOLEAN	NULLABLE	Describe this field...
is_bot	BOOLEAN	NULLABLE	Describe this field...
reversion_id	INTEGER	NULLABLE	Describe this field...
comment	STRING	NULLABLE	Describe this field...
num_characters	INTEGER	REQUIRED	Describe this field...

BigQuery History & Use

- **Dremel** - internal project used at Google
- Google use it across all their product suite.
    - Crawled Web Documents, Android Install Data, Spam Analysis, Crash Reporting, Builds, Resource Monitoring
- BigQuery is the public implementation

Datatypes

- String
- Float
- Integer
- Boolean
- Timestamp
- **Record (Nested / Repeatable)**

Accessing BigQuery

- WebUI
- Command Line
- JDBC connectors (Starschema-SQL Driver)
- Reporting Tools (Tableau = Happy Clients)
- API

Pricing: Loading and Streaming

- Pay as you go pricing model
  - Storage + Compute
- Transferring CSV/JSON data in is free
- Storage $0.026 per GB, per month
- Streaming data is $0.01 per 100,000 inserts *(Free until Jan 1st 2015)*

Pricing: Queries

- Queries $5 per TB
- Charged on the amount of data in the *columns used*
- Not charged for Cached Queries or those that fail
- LIMITing rows does nothing for reducing cost. Column based
- Table Decorators restrict based on age of data and *do reduce cost*

Query Quotas

- 20,000 queries per day
- Interactive and Batch
- Limit on the number of concurrent interactive queries
- https://cloud.google.com/bigquery/quota-policy

Storage and Scalability

How does it get to be so fast?

Columnar Datastore

- Traditional DB
  - Loads in each row, all columns
  - Regardless of if it needs it or not

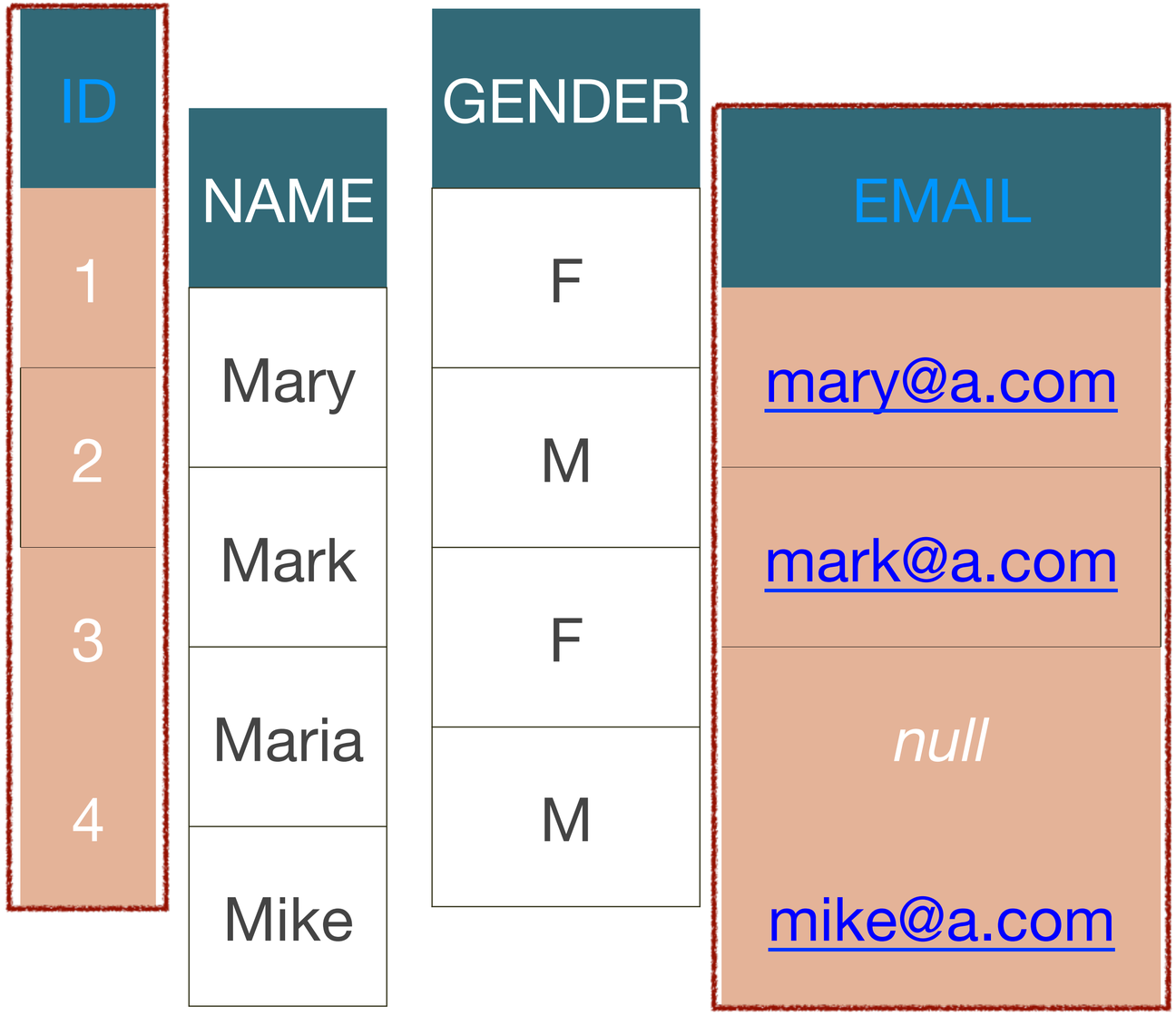
```
SELECT id, email
FROM table;
```

ID	NAME	GENDER	EMAIL
1	Mary	F	mary@a.com
2	Mark	M	mark@a.com
3	Maria	F	null
4	Mike	M	mike@a.com

Columnar Datastore

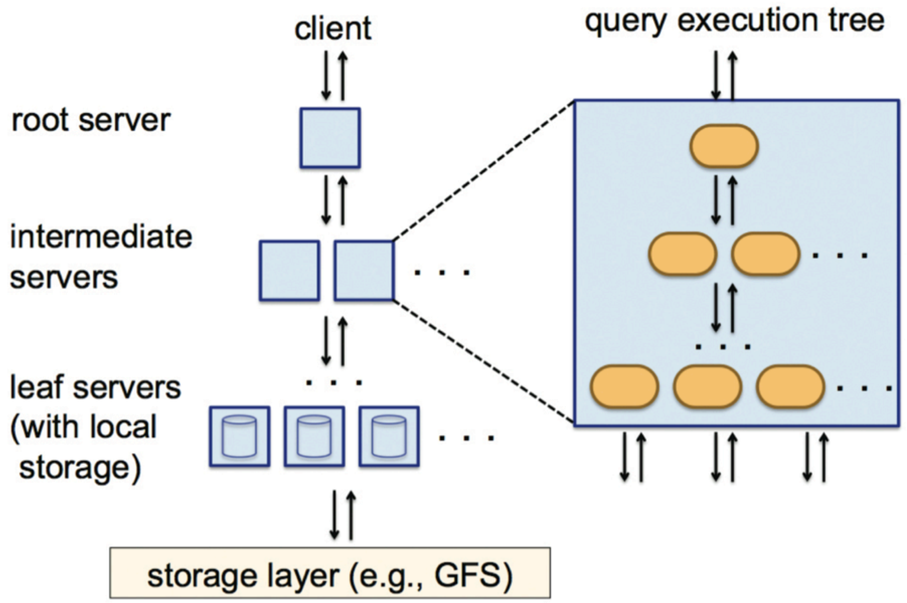
- BigQuery Columnar Storage
  - Columns stored separately
  - Only loads data queried
  - Better compression achieved

```
SELECT id, email
FROM table;
```



Tree Architecture

- Query runs on thousands nodes simultaneously
- Combines the results back in seconds
- Creates a massively parallel
  tree & aggregating results from the leaves.



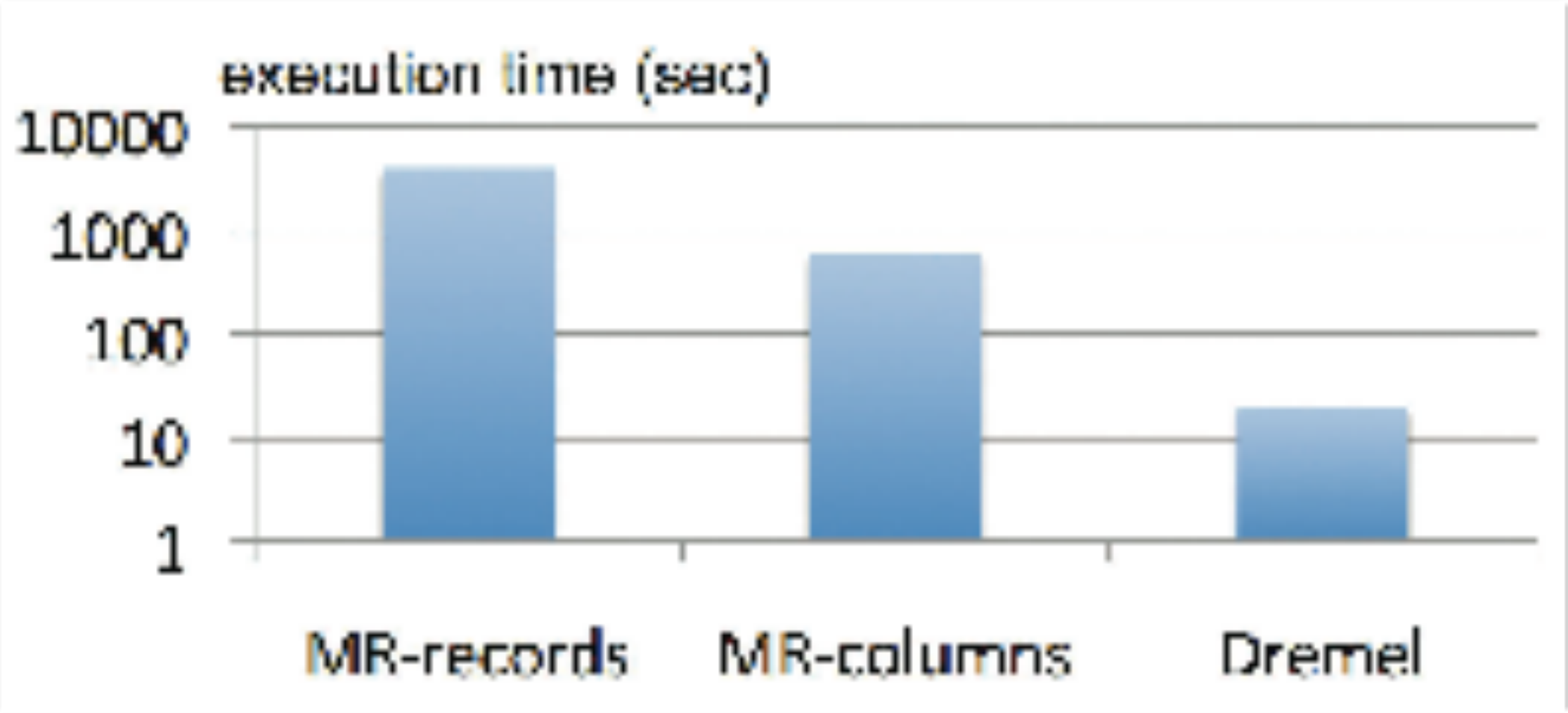
Source: Google BigQuery Technical White-paper

Alternatives

- Oracle RDBMS
- Amazon Redshift
- Hadoop MapReduce
- Apache Drill

Why BQ?

- Easily imports data from Google Cloud Storage
- No setup fees or hardware required.
- Easy account setup.
- Already know SQL.
- API in Java, JS, Python, .NET, and more



Source: Google BigQuery Technical White-paper

MapReduce vs Dremel

Java API

- *Connecting and Authenticating*
- Reading the Schema
- Loading a table
- Querying results
- Using Table Decorators

API Authentication

- OAuth2 - usually a client email and secret
- Google has a way of setting this up via the cloud console.
- Different OAuth workflows based on whether your app is web based, server to server, desktop or mobile only.
- Cloud console allows you to set these up independently and provide credentials for each scenario

Authenticating

```
/**
 * Builders for Credentials and BigQuery objects
 */
public class BigQueryServiceAccountAuth {

    private static final String SCOPE = "https://www.googleapis.com/auth/bigquery";
    private static final String SCOPE_INSERTDATA = "https://www.googleapis.com/auth/cloud-platform";
    private static final HttpTransport TRANSPORT = new NetHttpTransport();
    private static final JsonFactory JSON_FACTORY = new JacksonFactory();

    private static final ServiceAccountCredentials CREDENTIALS = new ServiceAccountCredentials();

    public static GoogleCredential newCredential() throws GeneralSecurityException, IOException, URISyntaxException {
        return new GoogleCredential.Builder().setTransport(TRANSPORT)
                    .setJsonFactory(JSON_FACTORY)
                    .setServiceAccountId(CREDENTIALS.getServiceAccountId())
                    .setServiceAccountScopes(Arrays.asList(SCOPE, SCOPE_INSERTDATA))
                    .setServiceAccountPrivateKeyFromP12File(CREDENTIALS.getP12File())
                    .build();
    }

    public static Bigquery newBigQuery(GoogleCredential credential) {
        return new Bigquery.Builder(TRANSPORT, JSON_FACTORY, credential)
                .setApplicationName(CREDENTIALS.getApplicationUserAgent())
                .setHttpRequestInitializer(credential).build();
    }

}
```

Java API

- Connecting and Authenticating
- *Reading the Schema*
- Loading a table
- Querying results
- Using Table Decorators

Query Metadata with Java 8 Streams

```
Bigquery.Projects.List projectList = bigquery.projects().list();

 final ProjectList list = projectList.execute();
 final List<ProjectList.Projects> projects = list.getProjects();

 projects.parallelStream()
         .peek(p -> out.format("PROJECT Id:%s Name:%s%n", p.getId(), p.getFriendlyName()))
         .map((ProjectList.Projects project) ->
                 LambdaExceptionWrapper.propagate(
                         () -> bigquery.datasets().list(project.getId()).execute()))
         .flatMap((datasetList) -> datasetList.getDatasets().parallelStream())
         .peek(ds -> out.format("%nDATASET id:%s %n", ds.getId()))
         .map(ds -> LambdaExceptionWrapper.propagate(
                 () -> bigquery.tables().list(
                         ds.getDatasetReference().getProjectId(),
                         ds.getDatasetReference().getDatasetId())
                         .execute())
         )
         .map(tableList -> Optional.ofNullable(tableList.getTables()))
         .flatMap(tableList -> tableList.orElse(Collections.emptyList()).parallelStream())
         .forEach(table -> out.format("TABLE id:%s %n", table.getId()));
```

Query Metadata with Java 8 Streams

- `PROJECT Id:java-one-talk Name:JavaOneTalk`

- `DATASET id:java-one-talk:InsertTest`
- `TABLE id:java-one-talk:InsertTest.stream_in`
- `TABLE id:java-one-talk:InsertTest.transactions_20140928_22240`
- `TABLE id:java-one-talk:InsertTest.wikipedia_edit_stats`
- `TABLE id:java-one-talk:InsertTest.wikipedia_title_edit_count`

Java API

- Connecting and Authenticating
- Reading the Schema
- *Loading a table*
- Querying results
- Using Table Decorators

Creating a table - Java

```
Table table = new Table();
table.setDescription("CSVLoadTest.create a table");

String tableName = “YOUR_TABLE_NAME”;
table.setTableReference(new TableReference().setDatasetId(DATASET_ID).setTableId(tableName));
TableSchema schema = new TableSchema().setFields(
        Arrays.asList(new TableFieldSchema().setName("Date").setType("TIMESTAMP"),
                      new TableFieldSchema().setName("Amount").setType("FLOAT"),
                      new TableFieldSchema().setName("Subscriber_ID").setType("INTEGER")));

table.setSchema(schema);
table = bq.tables().insert(PROJECT_ID, DATASET_ID, table).execute();
```

Creating a table - Groovy

```
def table = new Table()
table.setDescription("CSVLoadTest.create a table")

String tableName = newTableName()
table.setTableReference(new TableReference(datasetId: DATASET_ID, tableId: tableName))
def schema = new TableSchema(
                 fields: [new TableFieldSchema(name: 'Date', type: 'TIMESTAMP'),
                          new TableFieldSchema(name: 'Amount', type: 'FLOAT'),
                          new TableFieldSchema(name: 'Subscriber_ID', type: 'INTEGER')])
table.setSchema(schema)
table = bq.tables().insert(PROJECT_ID, DATASET_ID, table).execute()
```

Loading a CSV File into a table

```
def table = new Table(description: "CSVLoadTest.load a csv into BQ")

def (TableSchema schema, TableReference tableReference) = configureTableRefAndSchema(table)

// The tricky bit is setting the correct mime type and setting
// the sourceURIs as an emptyList (null will error)
def insertJob = bq.jobs().insert(PROJECT_ID,
        new Job(configuration: new JobConfiguration(
                load: new JobConfigurationLoad()
                        .setSourceUris(Collections.emptyList())
                        .setSchema(schema)
                        .setDestinationTable(tableReference)
                        .setSkipLeadingRows(1))),
        new FileContent('application/octet-stream', file))
        .execute();

// queryJobUntilDone
while (job.getStatus().getState() == "PENDING") {
    print "."
    job = bq.jobs().get(PROJECT_ID, job.getJobReference()?.getJobId()).execute()
}
```

Java API

- Connecting and Authenticating
- Reading the Schema
- Loading a table
- *Querying results*
- Using Table Decorators

Querying a Table

```
QueryResponse response = bq.jobs().query(PROJECT_ID,
        new QueryRequest(
                defaultDataset: new DatasetReference(projectId: PROJECT_ID, datasetId: DATASET_ID),
                // queries in SQL are great for existing Tooling to work with
                query: "SELECT Subscriber_ID, COUNT(*) as Transactions FROM transactions_20140929_162727 GROUP BY Subscriber_ID",

                // defaults worth mentioning

              // Used to test if query has any errors.  Will return an empty result if none.
                dryRun: false,
              // How long for the API call to wait before handing back.  Job Complete may be false and job need to be polled.
                timeoutMs: 10_000,
              // save money & use the results from cached queries in the last 24 hrs
                useQueryCache: true,
              // rows to return each page. For large result sets its recommended to keep page size down (1000) to improve reliability.
                maxResults: null,     // This is also an interesting way to batch pages of results back
        )).execute()
queryResponseUntilComplete(response)
```

Java API

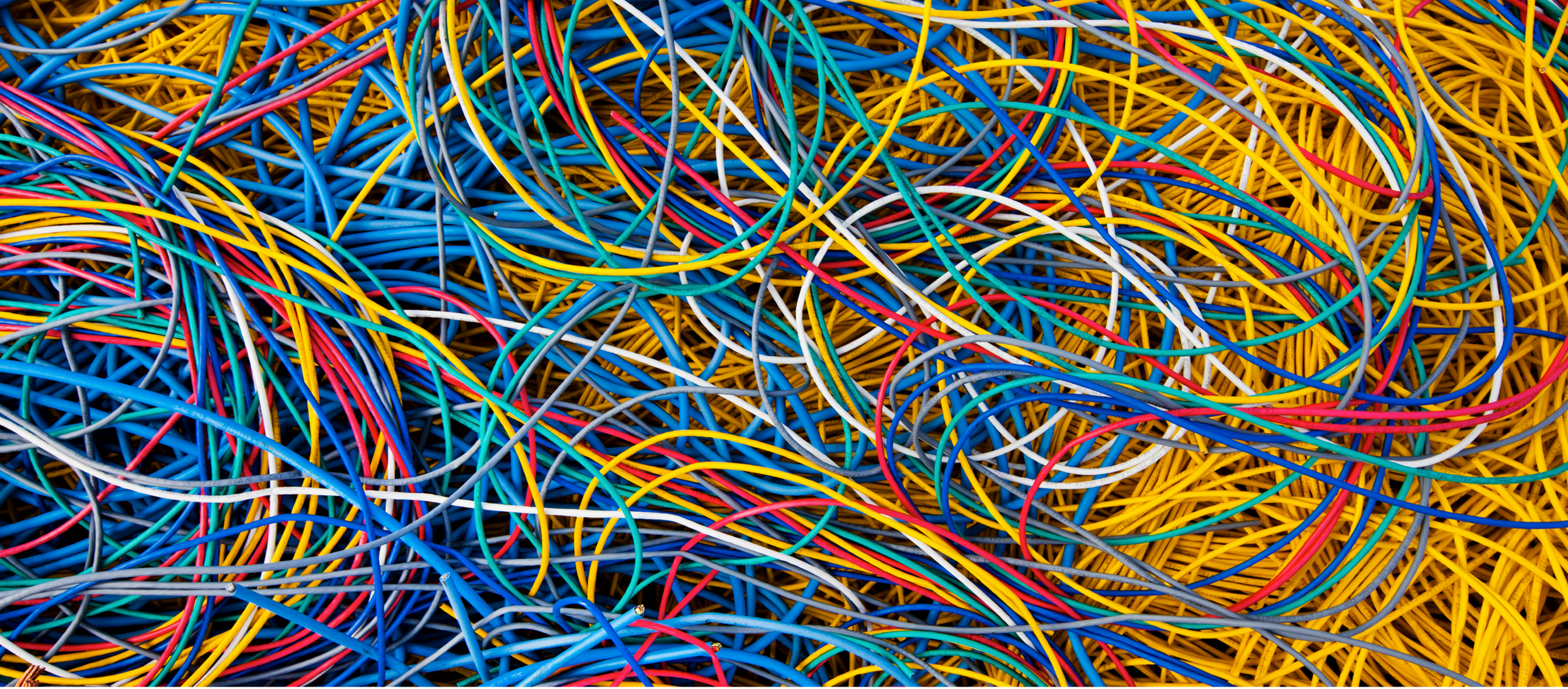
- Connecting and Authenticating
- Reading the Schema
- Loading a table
- Querying results
- *Using Table Decorators*

Table Decorators

We can provide a time range for how long ago to get results from.

For example, results in the last hour (3,600,000 milliseconds ago)

```
def query = "SELECT * FROM [$DATASET_ID:stream_in@-3600000--1000]"

Job job = bq.jobs().insert(PROJECT_ID, new Job(configuration:
        new JobConfiguration(query: new JobConfigurationQuery(
                query: query,

          // we want to cover if the query returns a large dataset.
                allowLargeResults: true,
                destinationTable: new TableReference(projectId: PROJECT_ID, datasetId: DATASET_ID, tableId: newTableName()),

          // Use Batch mode queries if you don't want to fill your pipeline
                priority: “INTERACTIVE",

          // Use cached results but doesnt work when a dest table is specified.
                useQueryCache: true,

          // There is also WRITE EMPTY, and WRITE TRUNCATE
                writeDisposition: “WRITE_APPEND",
       )))).execute()

queryJobUntilDone(job)
```



Gotchyas

and other annoyances

Annoyances, Running SQL Queries

- Prefers denormalized tables
  - Joining to many small tables impacts performance
- When to use the EACH keyword in GROUP BY, JOIN BY
- Uncertain error messages ‘Resources Exceeded’

Gotchyas

- The Shuffle error: Try and de-duplicate a large table using a GROUP BY EACH for all rows

- Hitting the Concurrent Rate Limit when our ETL processes were running alongside with large adhoc queries in our reporting tools.

Annoyances API

API error messages sometimes suck

```
400 Bad Request
{
  "code" : 400,
  "errors" : [ {
    "domain" : "global",
    "message" : "Required parameter is missing",
    "reason" : "required"
  } ],
  "message" : "Required parameter is missing"
}
```

Conclusion

- BigQuery - everyone can use ‘big data’
- Fast means of accessing the data
- API provides integration with other tools and frameworks
- SQL-like integration means older reporting tools can run queries against it
- Enables the use of Java 8 features

References

- An inside look at Google BigQuery
- parleys.com - How Google BigQuery works (for dummies) by Martin Gorner
- BigQuery Java API
- BigQuery Query Reference

Image Credits

- Clouds by Paul VanDerWerf Creative Commons 2.0
- Google BigQuery Datacenter images by Connie Zhou
  http://www.google.com/about/datacenters/gallery/#/tech
- Let us solve the big data problem… The Marketoonist
- Sheriff Badge, Runner on starting blocks, 17.5 speed sign, Photodune
- Tree Architecture & Performance Comparisons with MapReduce, Google
- MelbJVM Logo by Noel Richards www.nsquaredstudio.com.au

Thank you

Kon Soulianidis
Principal Engineer Gradleware
Organiser MelbJVM
kon at outrospective dot org
http://melbjvm.com
http://gradleware.com
http://github.com/neversleepz