



CodeTalk 2018

REST DB Links Access Databases in the Cloud using ORDS, REST & JSON

Robert Marz

Co-sponsored by:



Robert Marz

Client

Senior Technical Architect
with database centric view of the world

its-people

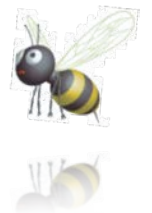
Portfolio Manager Database Technologies
Blog Editor

DOAG

Active Member Database Community
in charge of Cloud topics



ORACLE
ACE



@RobbieDatabee



blog.its-people.de



Robert.Marz
@its-people.de

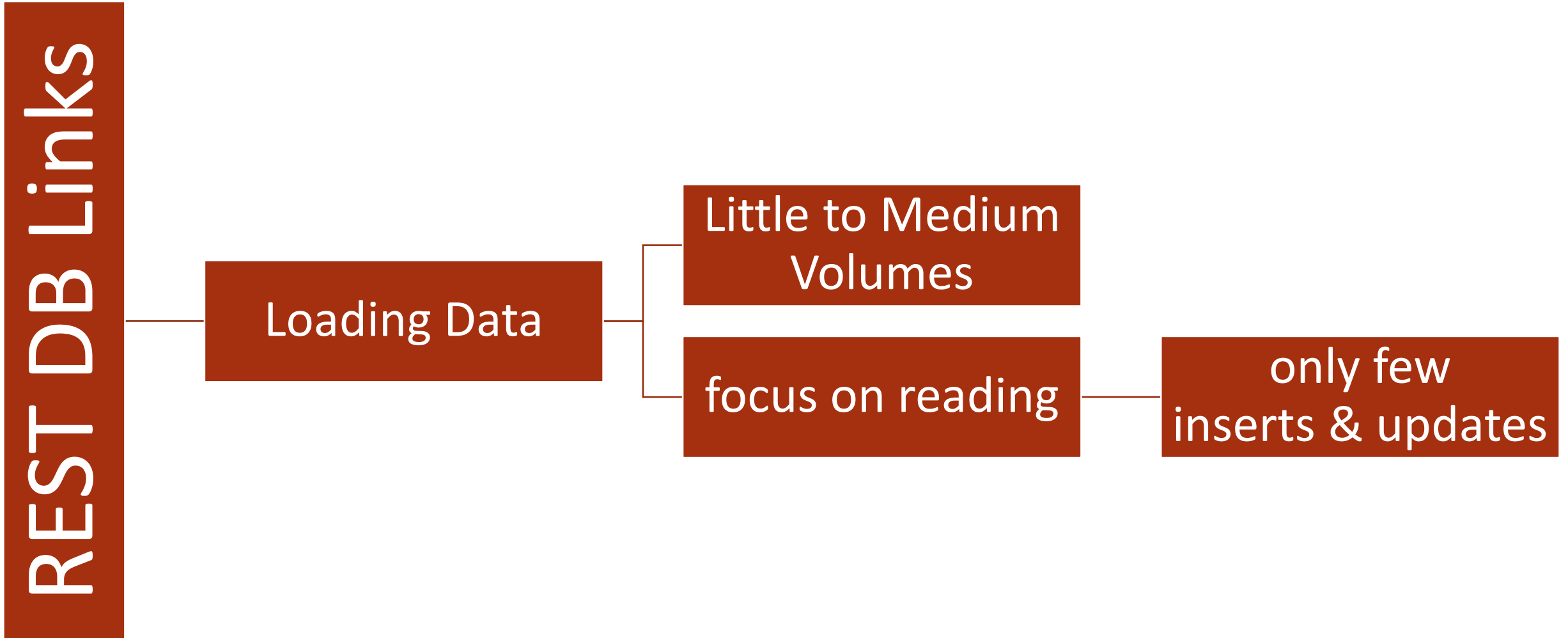
Database Links

Are trapped
inside the Oracle
world

Only work within
local or private
networks:
SQLNet

Objects are
treated as local

Database Links 2.0 – Use Case



Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

← http(s) requests
JSON documents →

4) JSON_TABLE Operator



ORDS



Remote Database

6) Generate all the things

1) REST enable



What is REST?

REST	Representational state transfer	
	programming paradigm	distributed systems Web services.
RESTful Applications	implements 6 constraints most important:	Uniform Interface (API via URIs) Stateless Cacheable
Implementation	Transport protocol	http(s)
	content	JSON Documents

Oracle REST Data Services ORDS (1/2)

Java

Evolved from APEX Listener

Deploy in Application Server

- Tomcat
- Glassfish (deprecated)
- WebLogic

Standalone mode

- Brings own http-server
- Supported for production use



Links

Installation

- Install ORDS in less than 5 Minutes by Colm Divilly (@cdivilly):
<http://blog.cdivilly.com/2015/03/11/install-ords-3.0.0/>

Official Homepage

- <http://www.oracle.com/technetwork/developer-tools/rest-data-services/overview/index.html>

Documentation

- https://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm

Video

- Oracle REST Data Services by Oracle Database Development Tools:
<https://www.youtube.com/watch?v=8XlbFRm-c6w>

Oracle REST Data Services ORDS (2/2)

REST

- Restful Services: Implement full
- SODA: Simple Oracle Document Access

Provides

AutoRest

- Rest enable tables
 - very easy
 - some limitations

Schemas

PL/SQL
Package

ORDS_ME
TADATA

ORDS_PUB
LIC_USER

ORDS

OAUTH

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

← http(s) requests
JSON documents →

4) JSON_TABLE Operator



ORDS



Remote Database

6) Generate all the things

1) REST enable



Preparing the Server:

Step 1 ORDS enable Schema (simple)

```
begin
  ords.enable_schema;
  commit;  -- This commit is important!
end;
/
```

Preparing the Server: Step 1 ORDS enable Schema (with Options)

```
BEGIN
  ORDS.ENABLE_SCHEMA
    ( p_schema => 'RESTDBLINKSPROV',
      p_url_mapping_type => 'BASE_PATH',
      p_url_mapping_pattern => 'rdbl',
      p_auto_rest_auth => false );
  commit;  -- This commit is important!
END;
/
```

Schema Part of URL

Preparing the Server:

Step 2 ORDS enable Object

```
begin
  ords.enable_object (
    P_ENABLED          => true,
    P_SCHEMA           => 'RESTDBLINKSPROV',
    P_OBJECT            => 'STOCKTICKER',
    P_OBJECT_TYPE       => 'TABLE',
    P_OBJECT_ALIAS      => 'tab-StockTicker',
    P_AUTO_REST_AUTH   => false
  );
  commit;  -- This commit is important, too!
end;
/
```

Case Sensitive ←

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

← http(s) requests
JSON documents →

4) JSON_TABLE Operator



ORDS



Remote Database

6) Generate all the things

1) REST enable



Anatomy of a ORDS AutoREST URL

ORDS Base

Table

Parameter

http://192.168.56.101:8080/ords/rdbl/Tab-StockTicker/?offset=55&limit=100
/ORCL,2016-01-08

Protocol

Host:Port

Schema

PKCol1,PKCol2
or Primary Key

HTTP Method	ORDS AutoREST Action
GET	Retrieve Data – Single Row or Rowset
PUT	Insert or Modify Row
POST	Bulk Insert csv-data
DELETE	Delete Row

What is JSON?

JSON

Java Script
Object Notation

```
{
  "name": "STOCKTICKER",
  "primaryKey": [
    "symbol",
    "tstamp"
  ],
  "members": [
    {
      "name": "symbol",
      "type": "VARCHAR2"
    },
    {
      "name": "tstamp",
      "type": "DATE"
    },
    {
      "name": "price",
      "type": "NUMBER"
    }
  ],
  "links": [
    {
      "rel": "collection",
      "href": "http://192.168.56.101:8080/o",
      "mediaType": "application/json"
    },
    {
      "rel": "canonical",
      "href": "http://192.168.56.101:8080/ords/rmougprov/metadata-catalog/tab-StockTicker/"
    },
    {
      "rel": "describes",
      "href": "http://192.168.56.101:8080/ords/rmougprov/tab-StockTicker/"
    }
  ]
}
```

Think of XML with
<Tags> replaced by
brackets

Schemaless

“Key”：“Value” Pairs

{ } – Groupings

[] - Arrays

no constraints for your implementation

Developers hell when dealing with
documents not produced by your code

Interpreting the ORDS AutoREST Responses (1)

```
1 ▾ {
2 ▾   "items": [
3 ▾     {
4       "symbol": "TDC",
5       "id1": 56,
6       "tstamp": "2017-05-06T23:27:00Z",
7       "price": 20.625,
8 ▾     "links": [
9 ▾       {
10        "rel": "self",
11        "href": "http://127.0.0.1:8080/ords/rdb1/Tab-StockTicker/56"
12      }
13     ]
14   },
15 ▾ {
16     "symbol": "ORCL",
17     "id1": 57,
18     "tstamp": "2017-05-06T23:28:00Z",
19     "price": 42,
```

Interpreting the ORDS AutoREST Responses (2)

```
1 ▾ {
2 ▶   "items": [↔],
28   "hasMore": true,
29   "limit": 2,
30   "offset": 55,
31   "count": 2,
32 ▾   "links": [
33 ▶     {↔},
37 ▶     {↔},
41 ▶     {↔},
45 ▶     {↔},
49 ▾     {
50         "rel": "next",
51         "href": "http://127.0.0.1:8080/ords/rdb1/Tab-StockTicker/?offset=57&limit=2"
52     },
53 ▶     {↔}
57   ]
58 }
```

ORDS and JSON revisited

The JSON produced by ORDS is NOT schemaless

Oracle has defined a new Media Type

`application/vnd.oracle.resource+json`

Whitepaper

<http://www.oracle.com/webfolder/technetwork/tutorials/appdevinfo/New%20REST%20Media%20Type.pdf>

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

http(s) requests
JSON documents



ORDS



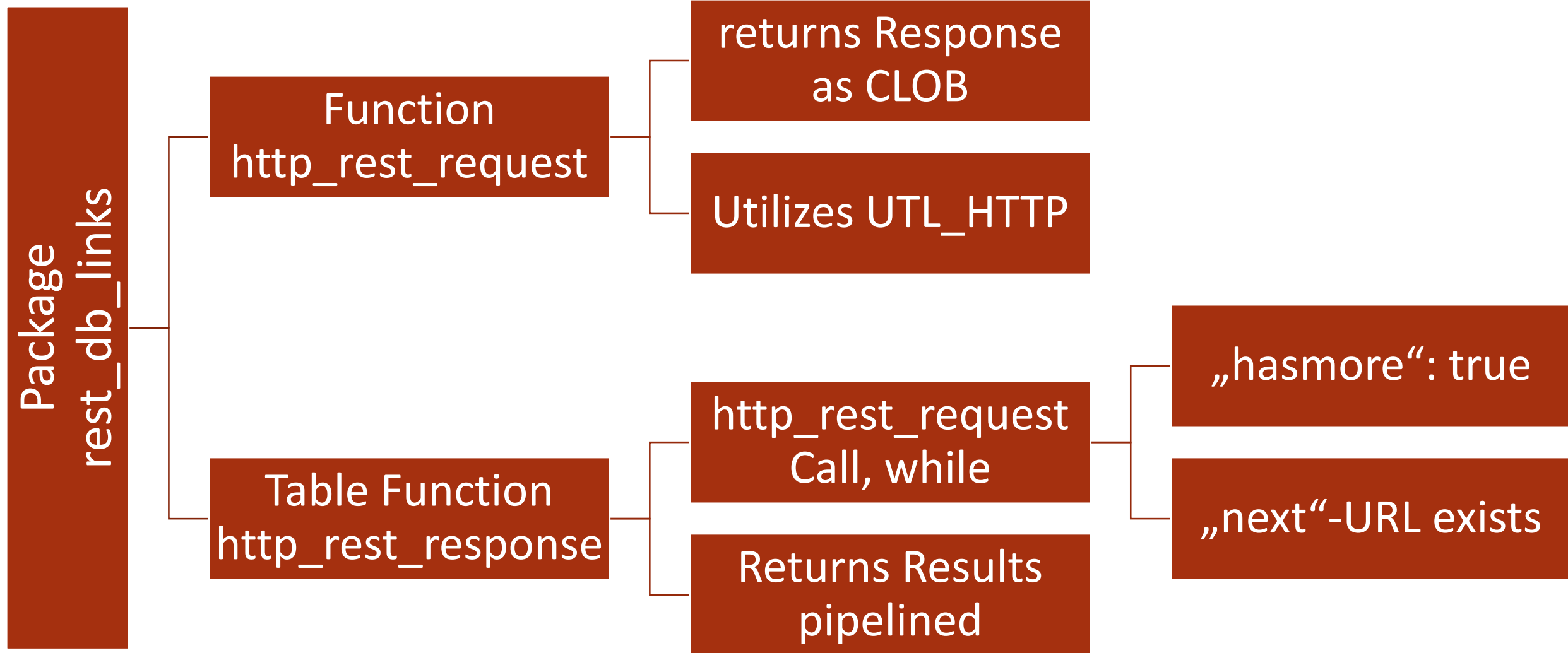
Remote Database

4) JSON_TABLE Operator

6) Generate all the things

1) REST enable

Building the Client – Step 1: Fetch all the data





Building the Client – UTL_HTTP needs ACL (1/2)

```
begin
  DBMS_NETWORK_ACL_ADMIN.create_acl (
    acl          => 'local_rest_acl_file.xml',
    description  => 'Grant Access to REST
Services on Host 127.0.0.1',
    principal    => upper('restdblinksCONS'),
    is_grant     => TRUE,
    privilege    => 'connect',
    start_date   => SYSTIMESTAMP,
    end_date     => NULL);
end;
/
```

Building the Client – UTL_HTTP needs ACL (2/2)

```
begin
  DBMS_NETWORK_ACL_ADMIN.assign_acl (
    acl      => 'local_rest_acl_file.xml',
    host     => '127.0.0.1',
    lower_port => 8080,
    upper_port => NULL);
end;
/

commit;
```

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

← http(s) requests
JSON documents →

4) JSON_TABLE Operator



ORDS



Remote Database

6) Generate all the things

1) REST enable



Building the Client – JSON_TABLE Operator (1/2)

JSON_TABLE

Lives inside the SQL-From-Clause

Produces **Rows** and **Columns**

Accepts CLOBs with JSON data

Included in SQL:2016 Standard

Building the Client – JSON_TABLE Operator (2/2)

The JSON Document

```
select wert
  from json_table( '["Eins", "Zwei", "Drei",
                    "Vier", "Fünf", "Sechs"]'
                  , '$[*]'
                  columns wert varchar2 path '$'
                  )
/
```

Produces rows

Produces columns

```
WERT
-----
Eins
Zwei
Drei
Vier
Fünf
Sechs

6 rows selected

Elapsed: 00:00:00.011
```

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

http(s) requests
JSON documents



ORDS



Remote Database

4) JSON_TABLE Operator

6) Generate all the things

1) REST enable



Building the Client – Producing Rows and Columns

```
select j.*
       , t.*
  from table(rest_db_links.http_rest_response('http://...') ) t
       , json_table( t.response, '$.items[*]'
                    columns
                    ( symbol  varchar2 path '$.symbol'
                    , tstamp  varchar2 path '$.tstamp'
                    , price   number   path '$.price'
                    , selfurl  varchar2 path '$.links[0].href'
                    )
              ) j
;
```

Building the Client – Casting Datatypes

Numbers

JSON numbers come in US Locale: Decimal Point and thousand separator is comma

Get varchar2 JSON columns, cast explicit

```
to_number( price
          , '999999999999999D99999999'
          , 'nls_numeric_characters=''.,''' ) price
```

Dates

JSON Dates are ISO 8601 Zulu (UTC) Time

```
cast( to_timestamp_tz
      ( to_char(
          to_date(tstamp, 'YYYY-MM-DD"T"HH24:MI:SS"Z"')
          , 'YYYY-MM-DD HH24:MI:SS "UTC"')
      , 'YYYY-MM-DD HH24:MI:SS TZR')
      at time zone sessiontimezone as date ) tstamp
```



Building the Client – View DML

create or replace trigger StockTicker_ORDS_IUD
instead of **insert** or **update** or **delete**
on StockTicker_ORDS
for each row

```
rest_db_links.http_rest_request( );
```

Operation	REST URL	HTTP-Method	Payload (JSON)
deleting	:old.selfurl	DELETE	empty
updating	:old.selfurl	PUT	all columns & values
inserting	ORDS-Table-URL '/' PKCol1 ';' PKCol2	PUT	all columns & values

Database Links 2.0 – Architecture

3) Package rest_db_link

5) The View

2) AutoREST URL



Local Database

← http(s) requests
JSON documents →

4) JSON_TABLE Operator



ORDS



Remote Database

6) Generate all the things

1) REST enable

The Generator

Use from sqlcl.

JavaScript based

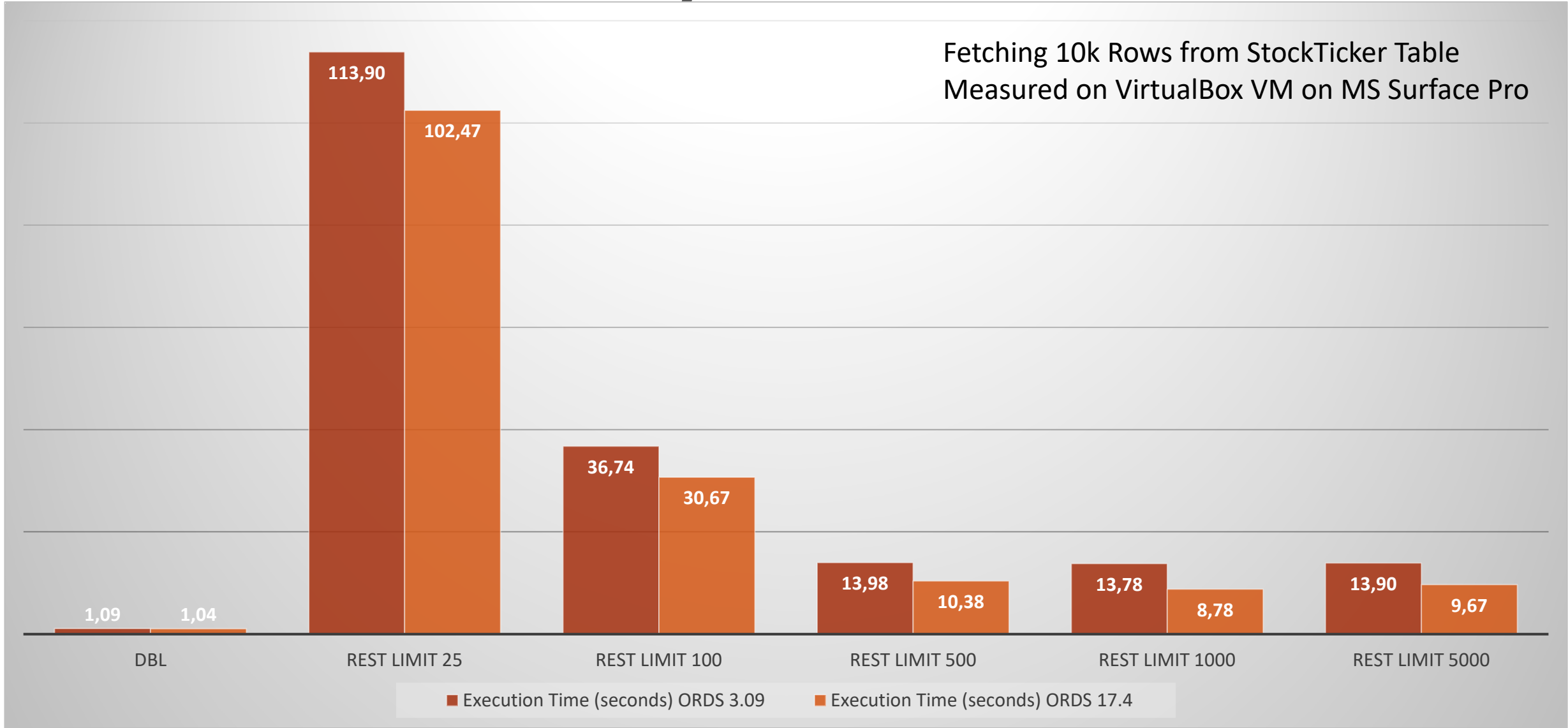
Parameter:

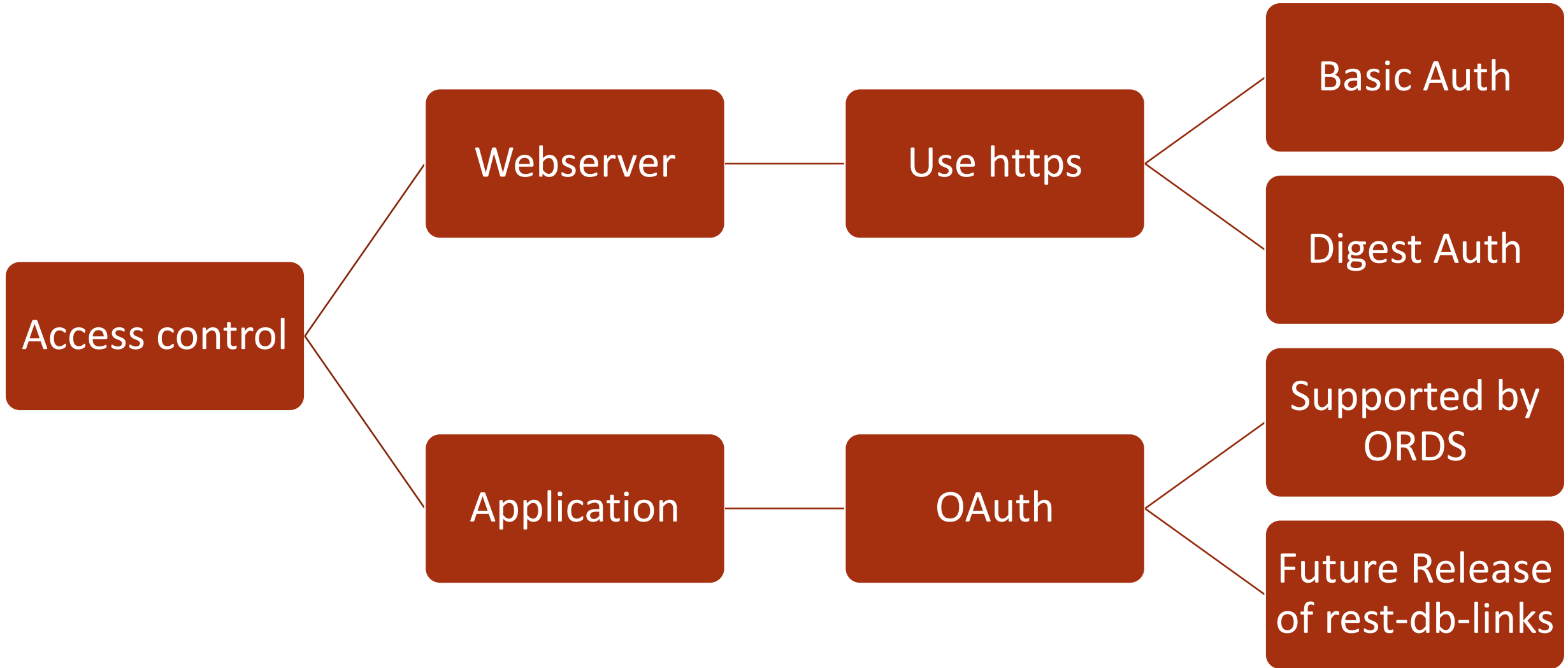
- View Name
- ORDS Metadata URL
- optional: Parameter for URL

```
script ../generator/generator  
    v_stock  
    http://localhost:8080/ords/rdbl/metadata-catalog/Tab-StockTicker/  
    ?limit=5  
-- call from single line
```

Performance Analysis

Fetching 10k Rows from StockTicker Table
Measured on VirtualBox VM on MS Surface Pro





DEMO Environment



Demo

OTN
Developer Day VM

- Virtual Box Appliance (March 2nd, 2017)
 - Oracle Linux 7
 - Oracle Database 12cR2 EE (12.2.0.1 with In-Memory Option)
 - Oracle Application Express 5.1
 - Oracle REST Data Services 3.0.9
 - <http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html>

Schemas

- restdblinksCons: Consumer: Local Database
- restdblinksProv: Provider: Remote Database (Data Source)

Client

- Browser: Google Chrome
- REST Client: Insomnia
- SQL Developer
- sqlcl

Limitations / Future enhancements

Limitations

simple data types only

- No Spatial
- No Object Types

Planned enhancements

OAuth

Complex data types

Integrate into
SQLDeveloper (oddgen)

How do you get it?

Conference website

Slides and scripts uploaded to

<http://collaborate.ioug.org>

GitHub

Latest version always here

You can help enhance the generator – I'm accepting pull requests

<https://github.com/its-people/rest-db-links>

notification

Twitter: @RobbieDatabee



Blog: <http://www.its-people.de/blog>



Conclusion

Only a little magic is needed.

Performance is worse, but acceptable in most cases.

With REST and JSON, you can reach out from your database to the Internet.

Database Links can be replaced by a modern Architecture.