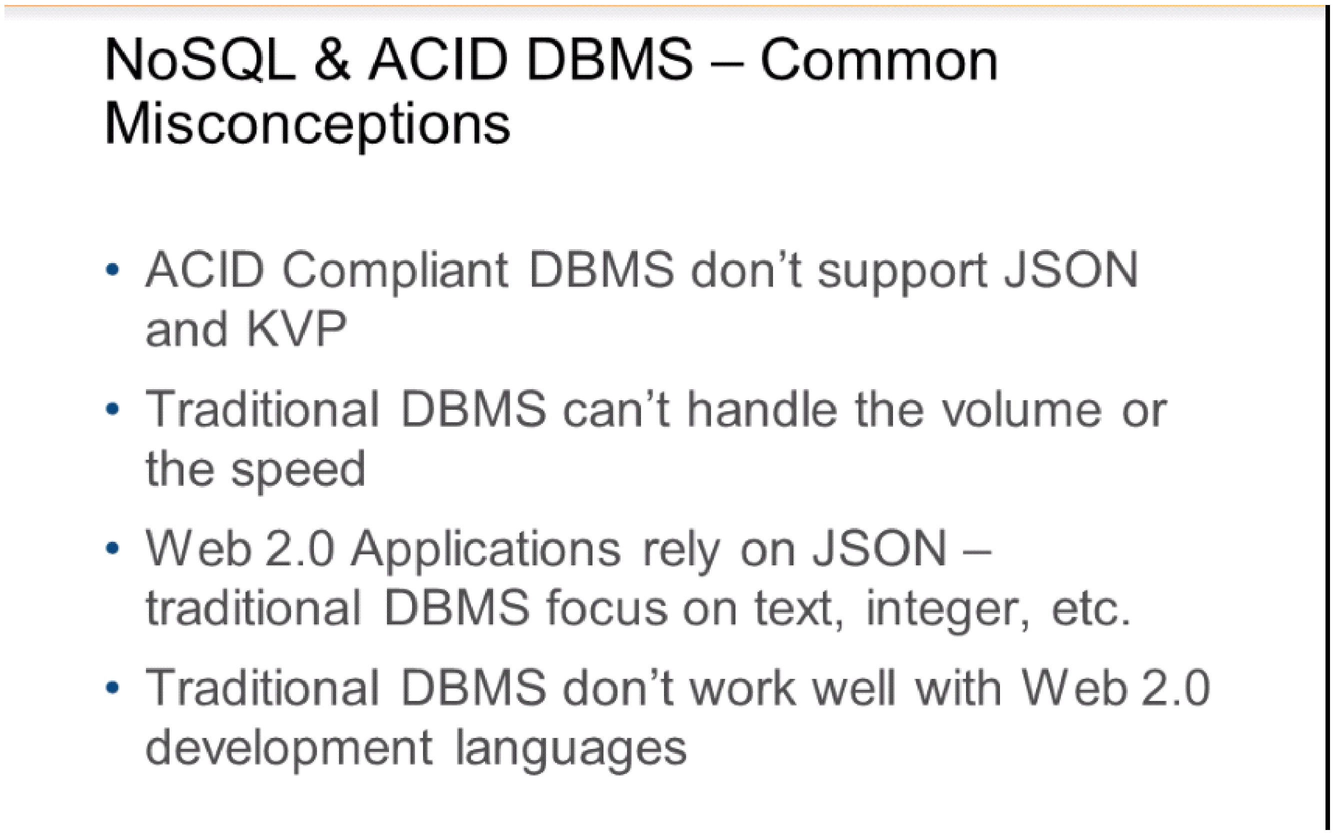
Enterprise DB

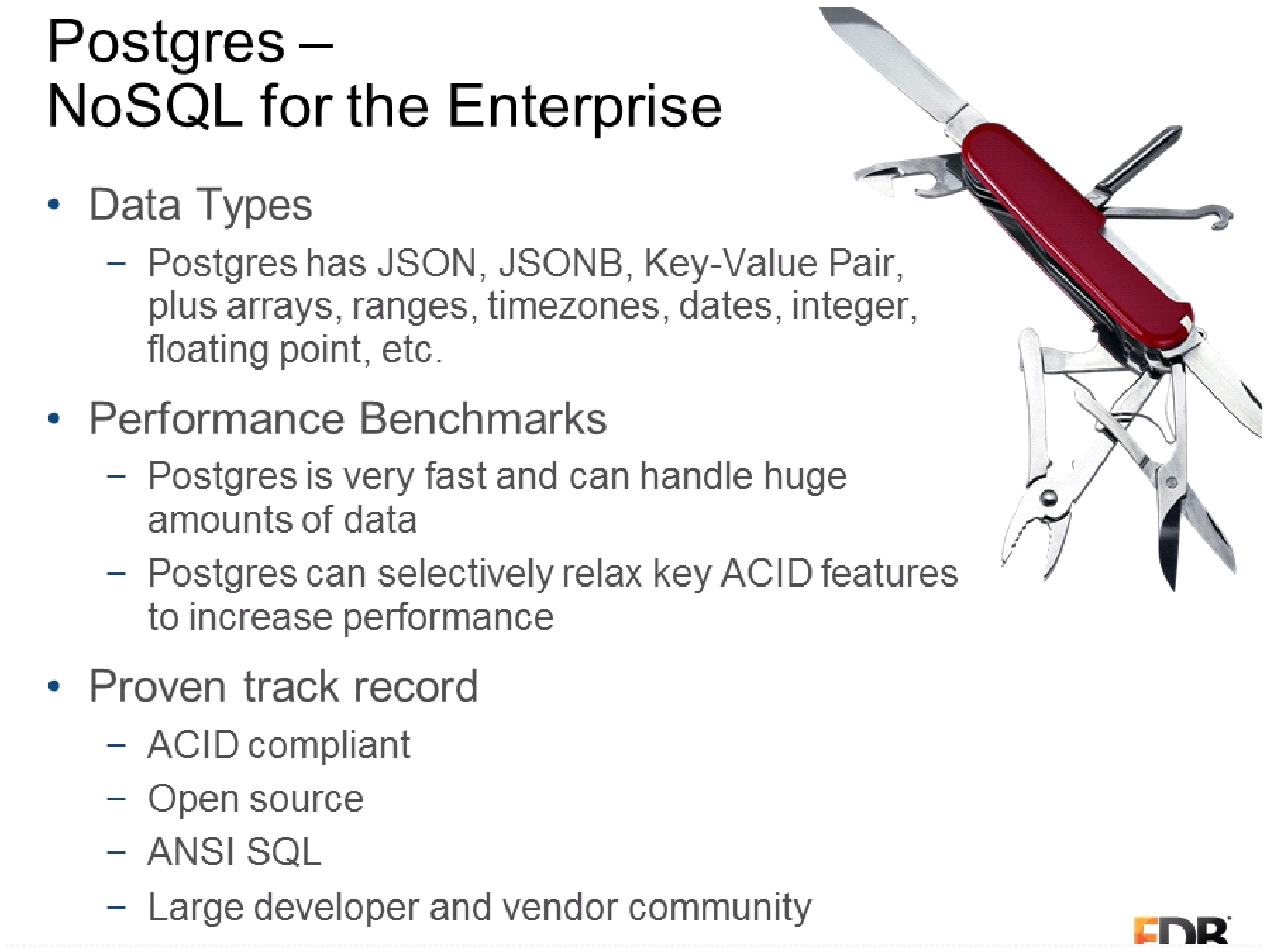
Thursday, June 15, 2017 1:42 PM



NoSQL & ACID DBMS – Common Misconceptions

- ACID Compliant DBMS don't support JSON and KVP
- Traditional DBMS can't handle the volume or the speed
- Web 2.0 Applications rely on JSON – traditional DBMS focus on text, integer, etc.
- Traditional DBMS don't work well with Web 2.0 development languages

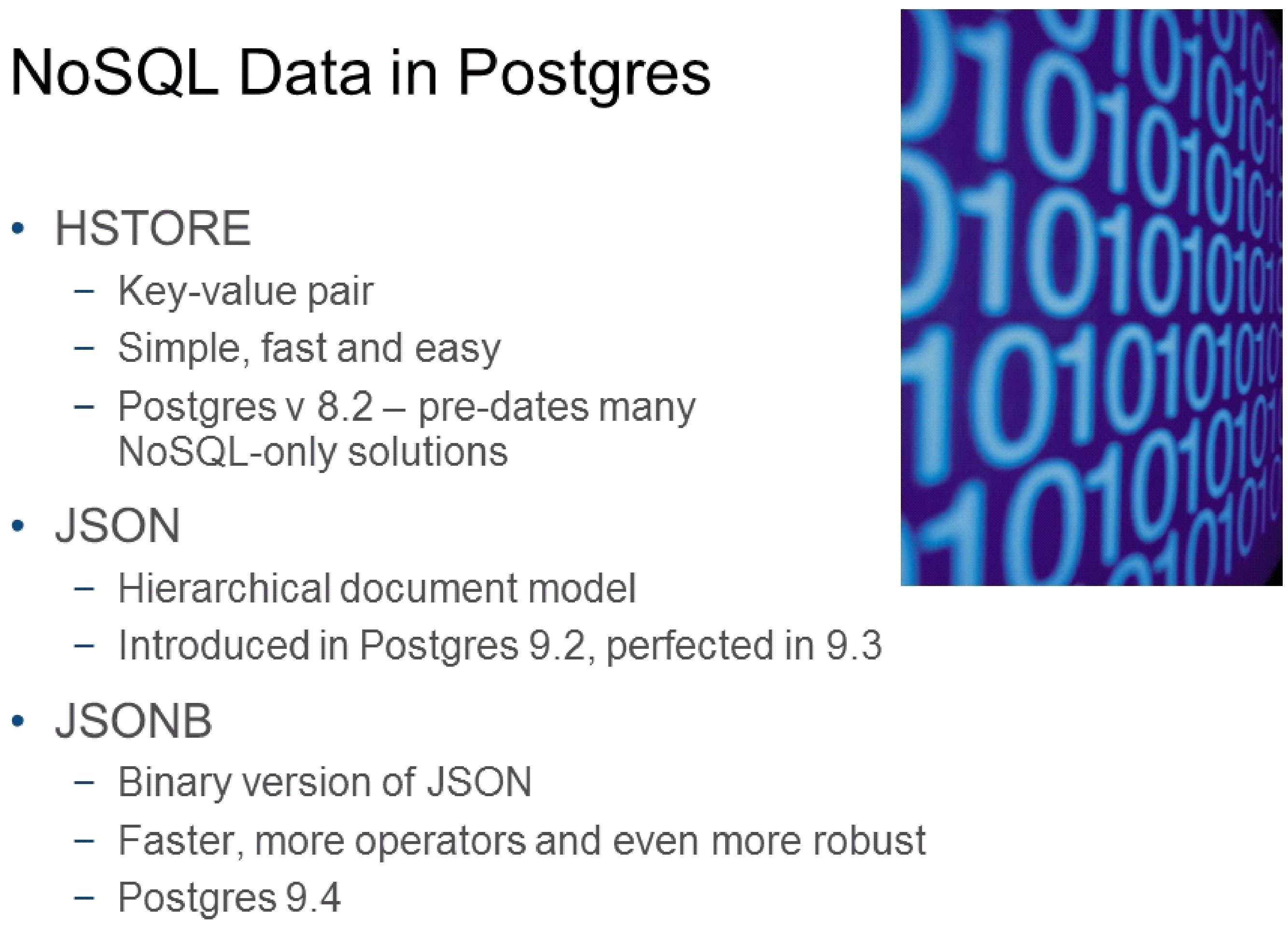
8/18/2017 3:14 PM - Screen Clipping



Postgres – NoSQL for the Enterprise

- Data Types
  - Postgres has JSON, JSONB, Key-Value Pair, plus arrays, ranges, timezones, dates, integer, floating point, etc.
- Performance Benchmarks
  - Postgres is very fast and can handle huge amounts of data
  - Postgres can selectively relax key ACID features to increase performance
- Proven track record
  - ACID compliant
  - Open source
  - ANSI SQL
  - Large developer and vendor community

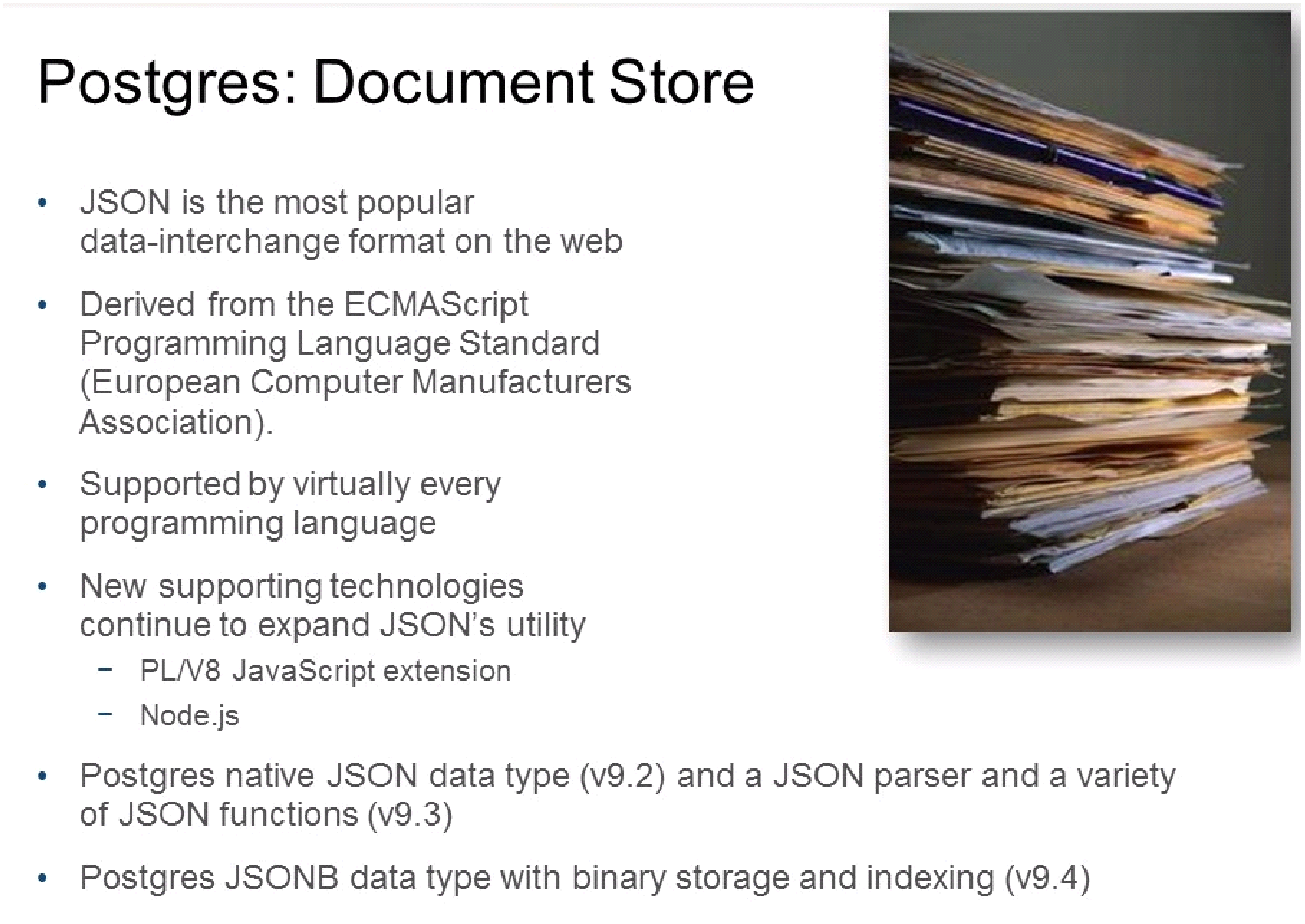
8/18/2017 3:17 PM - Screen Clipping



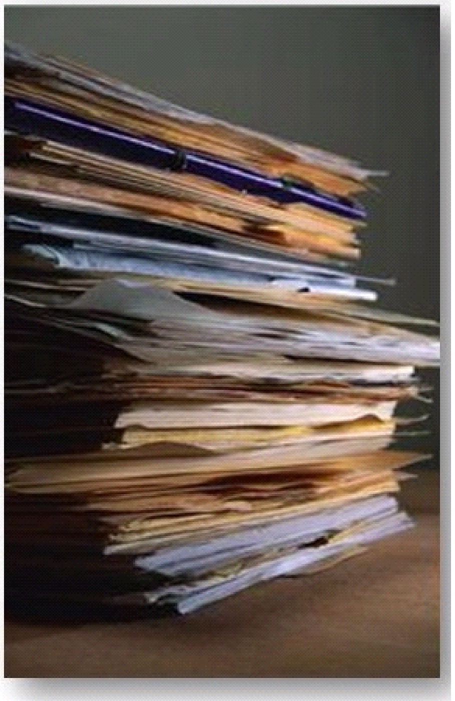
NoSQL Data in Postgres

- HSTORE
  - Key-value pair
  - Simple, fast and easy
  - Postgres v 8.2 – pre-dates many NoSQL-only solutions
- JSON
  - Hierarchical document model
  - Introduced in Postgres 9.2, perfected in 9.3
- JSONB
  - Binary version of JSON
  - Faster, more operators and even more robust
  - Postgres 9.4

8/18/2017 3:18 PM - Screen Clipping

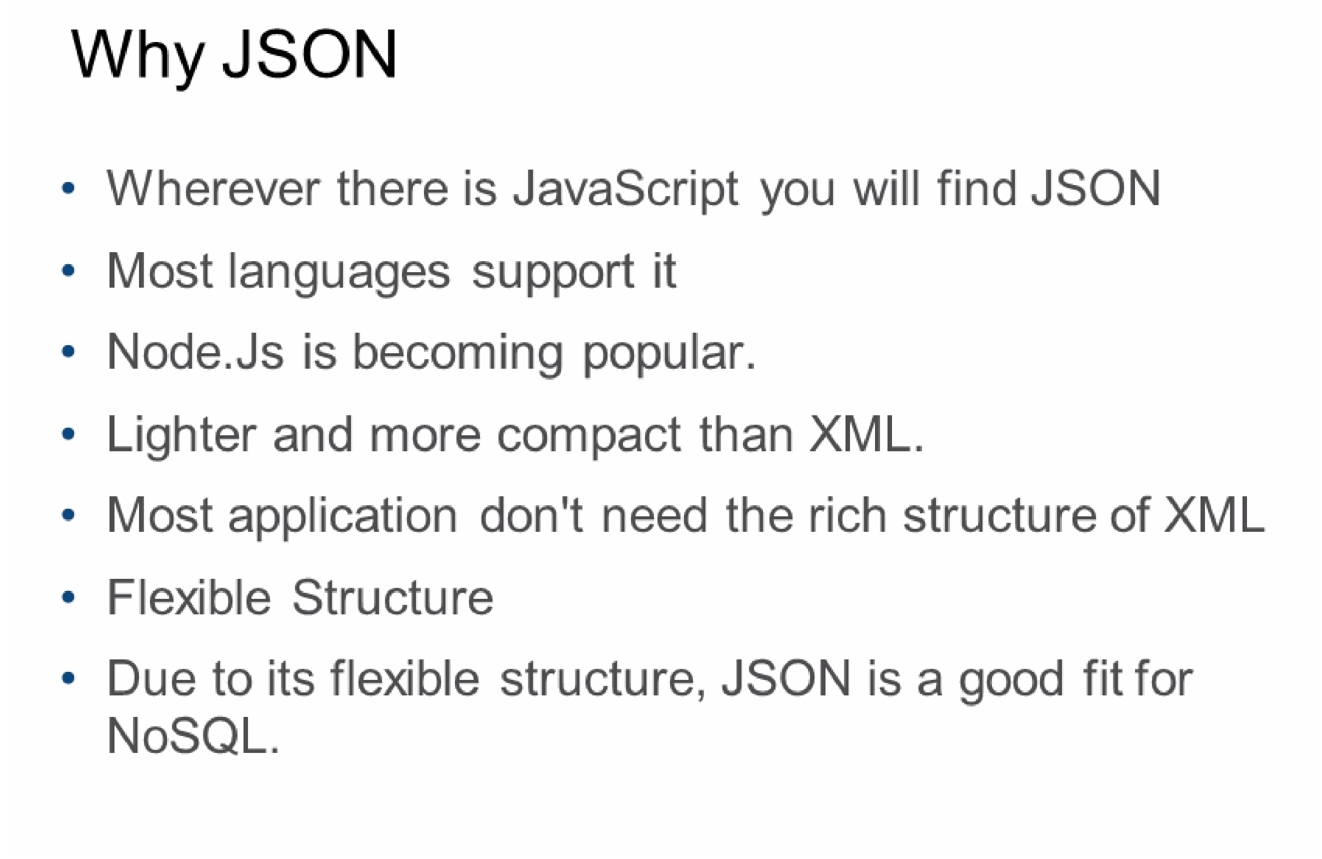


Postgres: Document Store



- JSON is the most popular data-interchange format on the web
- Derived from the ECMAScript Programming Language Standard (European Computer Manufacturers Association).
- Supported by virtually every programming language
- New supporting technologies continue to expand JSON's utility
  - PL/V8 JavaScript extension
  - Node.js
- Postgres native JSON data type (v9.2) and a JSON parser and a variety of JSON functions (v9.3)
- Postgres JSONB data type with binary storage and indexing (v9.4)

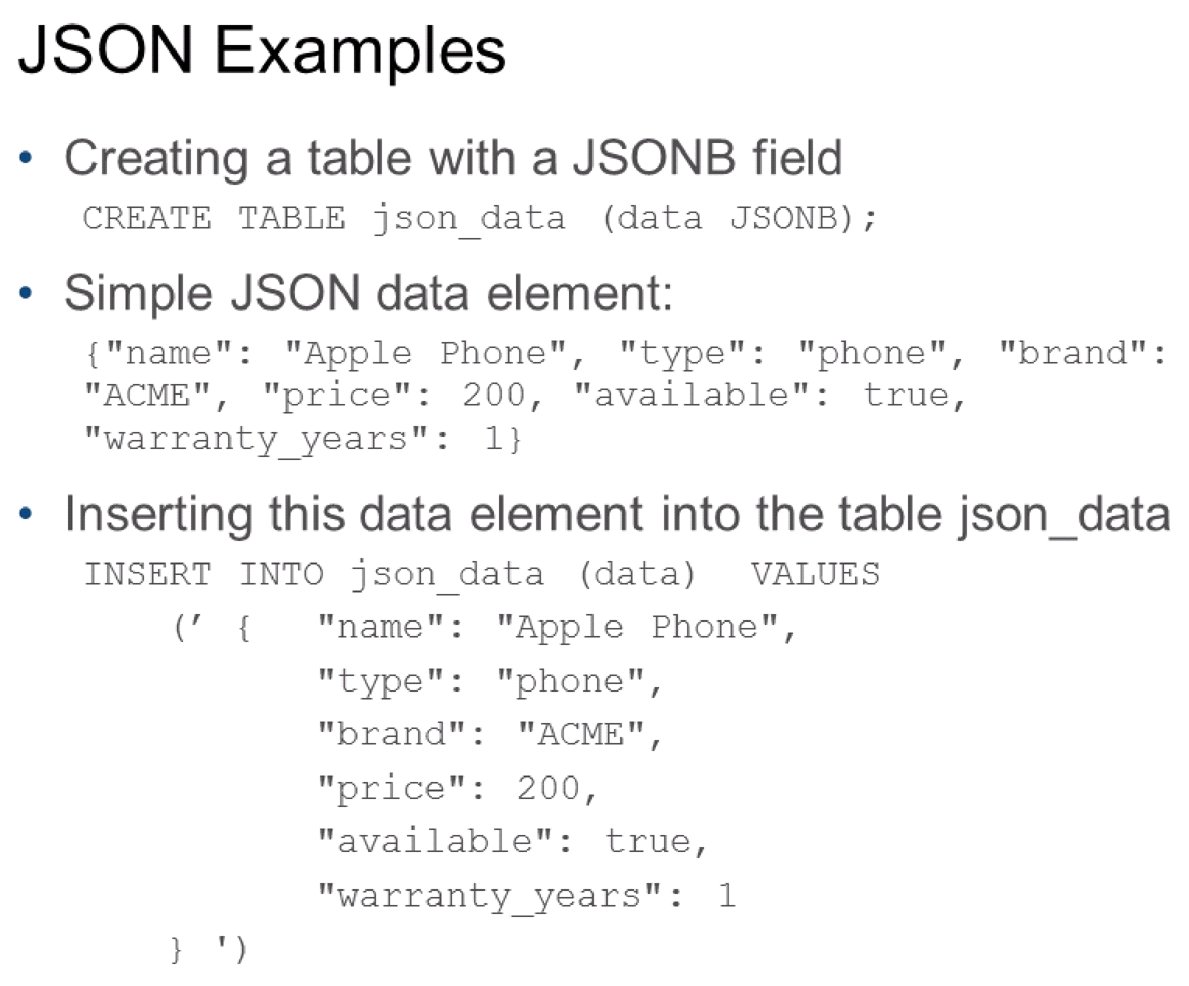
8/18/2017 3:19 PM - Screen Clipping



Why JSON

- Wherever there is JavaScript you will find JSON
- Most languages support it
- Node.Js is becoming popular.
- Lighter and more compact than XML.
- Most application don't need the rich structure of XML
- Flexible Structure
- Due to its flexible structure, JSON is a good fit for NoSQL.

8/18/2017 3:20 PM - Screen Clipping



JSON Examples

- Creating a table with a JSONB field

```
CREATE TABLE json_data (data JSONB);
```

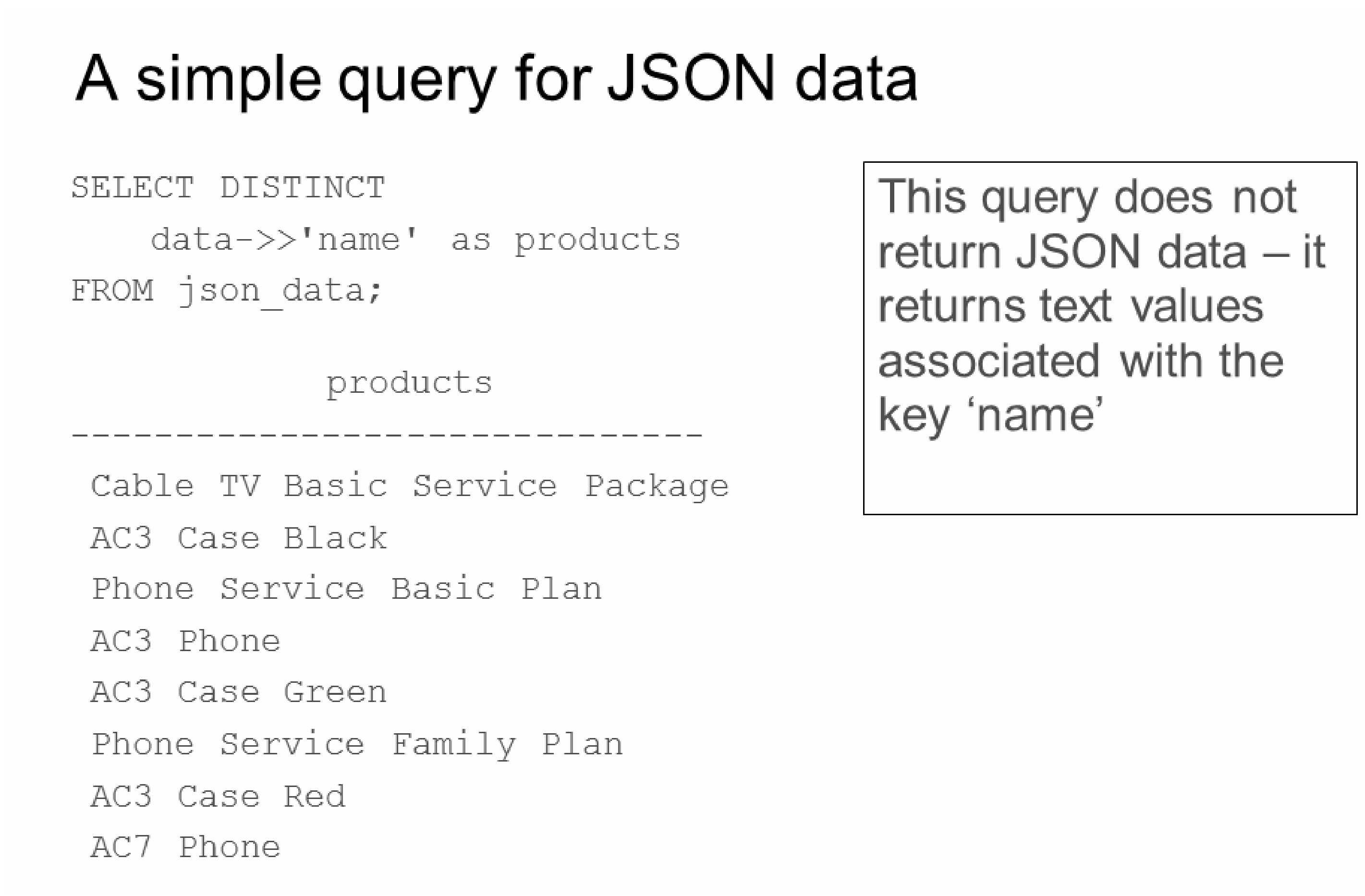
- Simple JSON data element:

```
{"name": "Apple Phone", "type": "phone", "brand":
"ACME", "price": 200, "available": true,
"warranty_years": 1}
```

- Inserting this data element into the table json_data

```
INSERT INTO json_data (data)  VALUES
    (’ {   "name": "Apple Phone",
           "type": "phone",
           "brand": "ACME",
           "price": 200,
           "available": true,
           "warranty_years": 1
    } ')
```

8/18/2017 3:21 PM - Screen Clipping

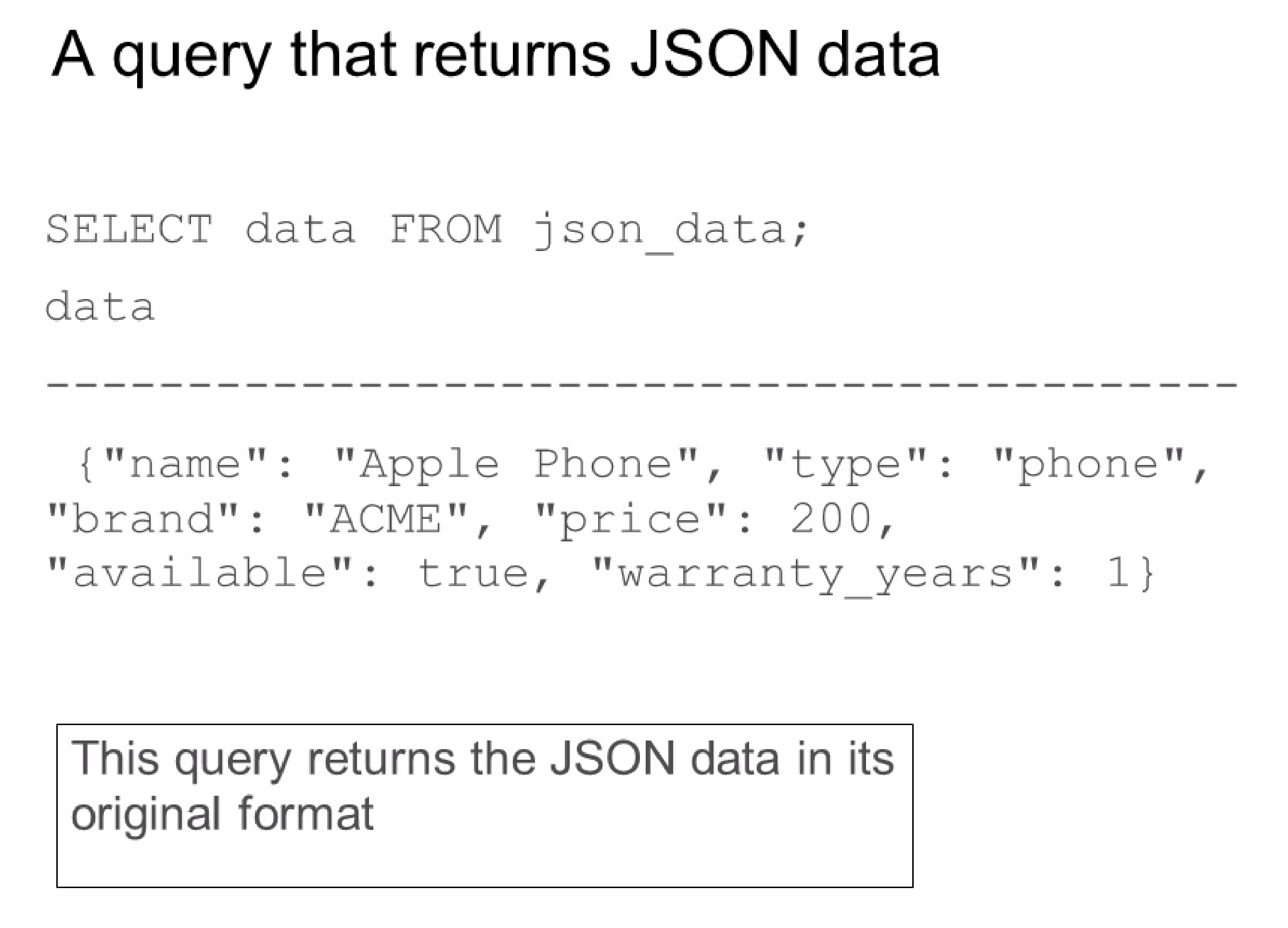


A simple query for JSON data

```
SELECT DISTINCT
   data->>'name' as products
FROM json_data;

            products
------------------------------
 Cable TV Basic Service Package
 AC3 Case Black
 Phone Service Basic Plan
 AC3 Phone
 AC3 Case Green
 Phone Service Family Plan
 AC3 Case Red
 AC7 Phone
```

This query does not return JSON data – it returns text values associated with the key ‘name’

8/18/2017 3:22 PM - Screen Clipping

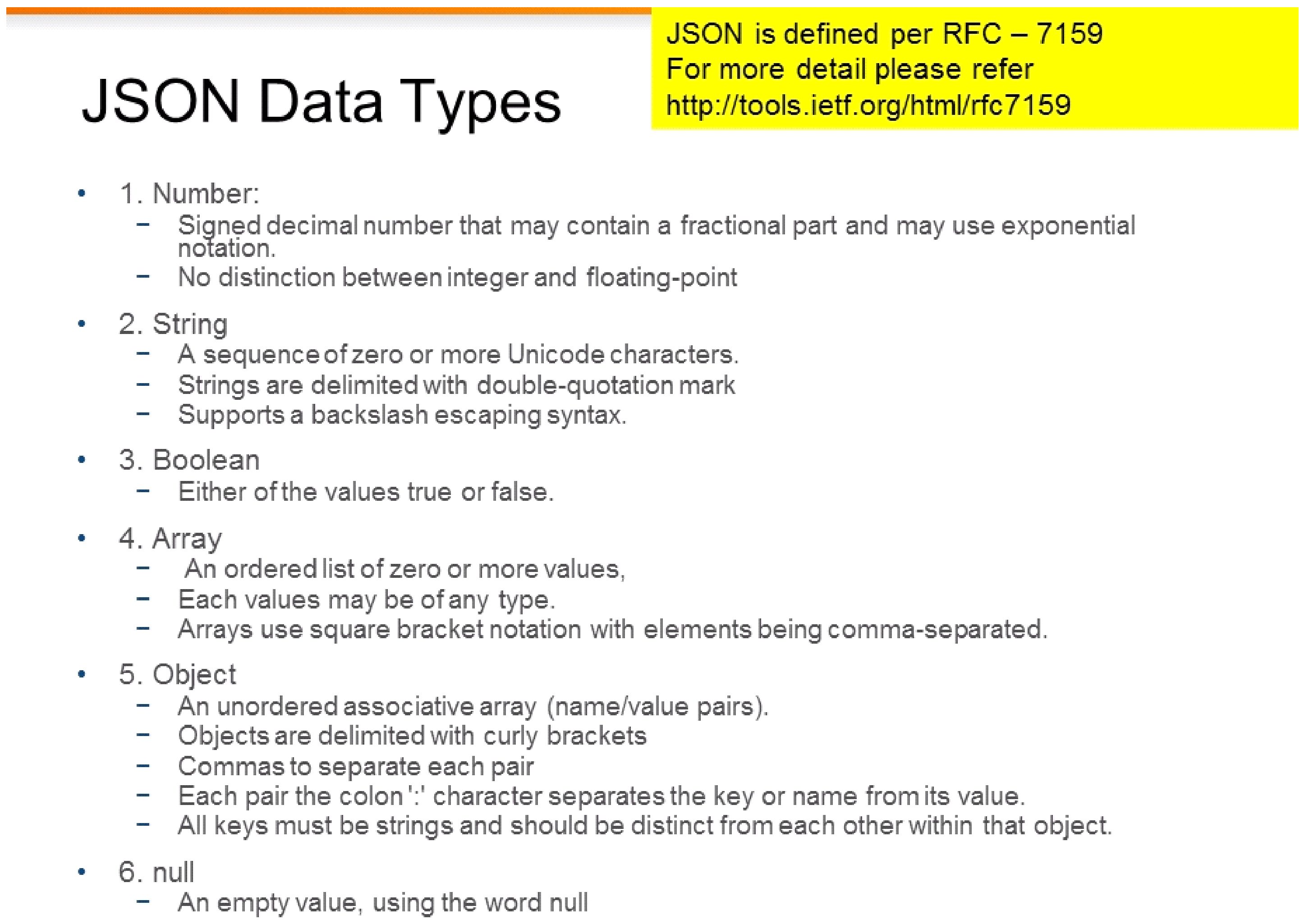


A query that returns JSON data

```
SELECT data FROM json_data;
data
-----------------------------------------
 {"name": "Apple Phone", "type": "phone",
"brand": "ACME", "price": 200,
"available": true, "warranty_years": 1}
```

This query returns the JSON data in its original format

8/18/2017 3:23 PM - Screen Clipping

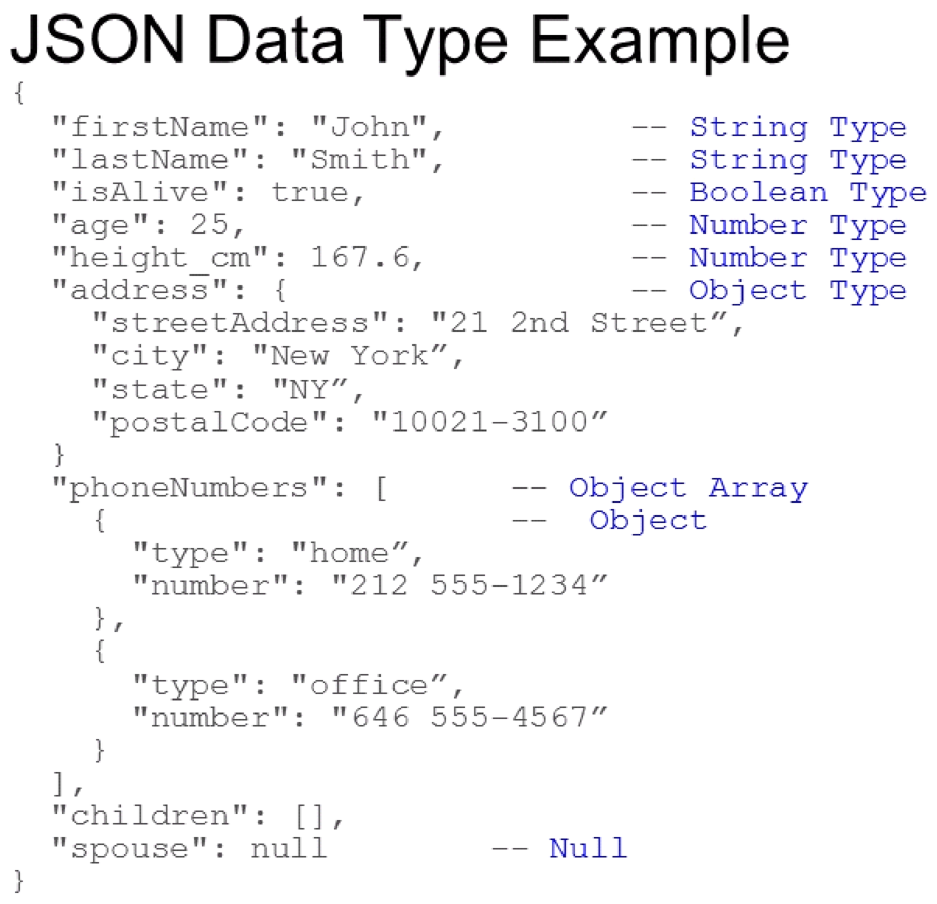


JSON is defined per RFC – 7159
For more detail please refer
http://tools.ietf.org/html/rfc7159

JSON Data Types

- 1. Number:
  - Signed decimal number that may contain a fractional part and may use exponential notation.
  - No distinction between integer and floating-point
- 2. String
  - A sequence of zero or more Unicode characters.
  - Strings are delimited with double-quotation mark
  - Supports a backslash escaping syntax.
- 3. Boolean
  - Either of the values true or false.
- 4. Array
  - An ordered list of zero or more values,
  - Each values may be of any type.
  - Arrays use square bracket notation with elements being comma-separated.
- 5. Object
  - An unordered associative array (name/value pairs).
  - Objects are delimited with curly brackets
  - Commas to separate each pair
  - Each pair the colon ':' character separates the key or name from its value.
  - All keys must be strings and should be distinct from each other within that object.
- 6. null
  - An empty value, using the word null

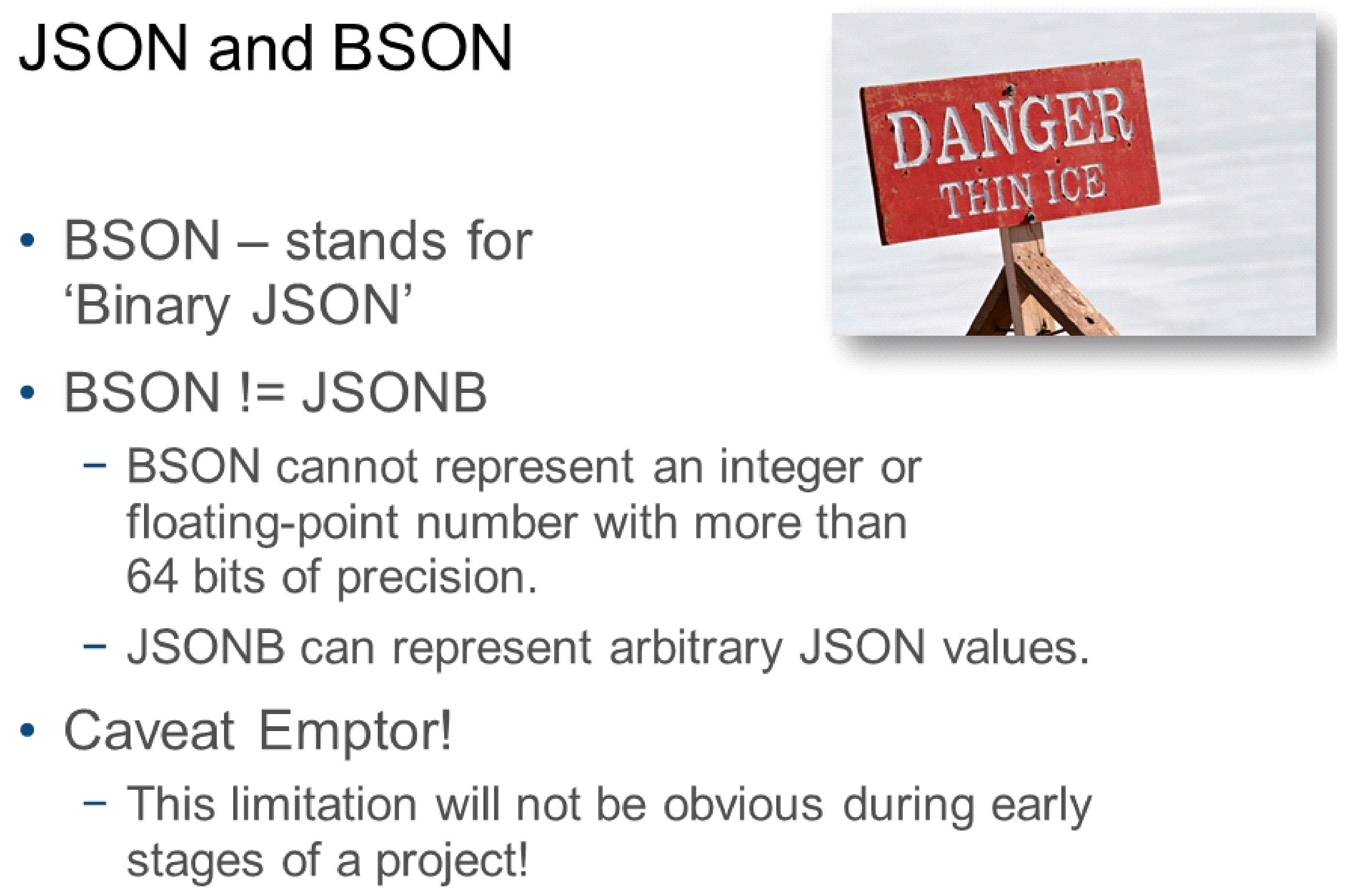
8/18/2017 3:24 PM - Screen Clipping



JSON Data Type Example

```
{
  "firstName": "John",          -- String Type
  "lastName": "Smith",          -- String Type
  "isAlive": true,              -- Boolean Type
  "age": 25,                    -- Number Type
  "height_cm": 167.6,           -- Number Type
  "address": {                  -- Object Type
    "streetAddress": "21 2nd Street”,
    "city": "New York”,
    "state": "NY”,
    "postalCode": "10021-3100”
  }
  "phoneNumbers": [       -- Object Array
    {                     --  Object
      "type": "home”,
      "number": "212 555-1234”
    },
    {
      "type": "office”,
      "number": "646 555-4567”
    }
  ],
  "children": [],
  "spouse": null        -- Null
}
```

8/18/2017 3:25 PM - Screen Clipping

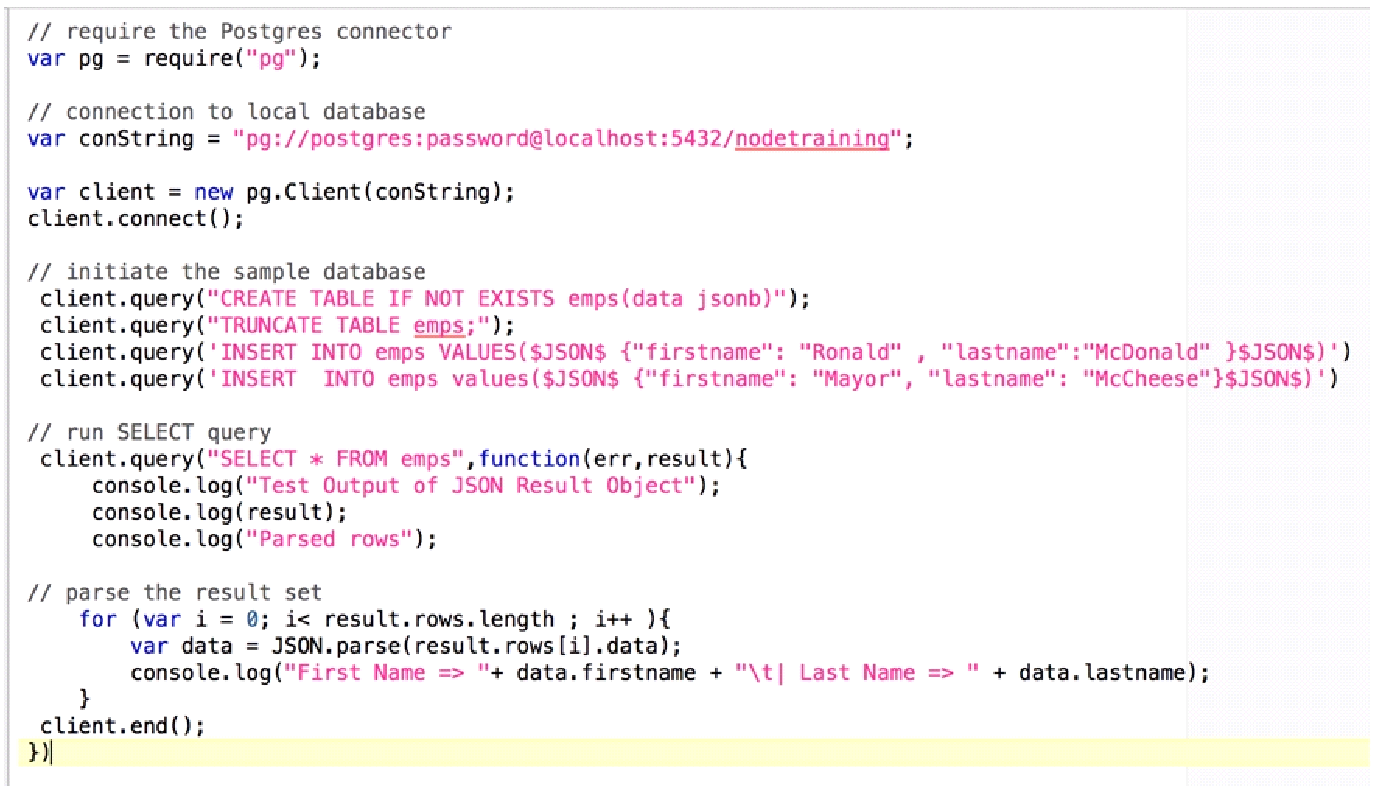


JSON and BSON

- BSON – stands for ‘Binary JSON’
- BSON != JSONB
  - BSON cannot represent an integer or floating-point number with more than 64 bits of precision.
  - JSONB can represent arbitrary JSON values.
- Caveat Emptor!
  - This limitation will not be obvious during early stages of a project!

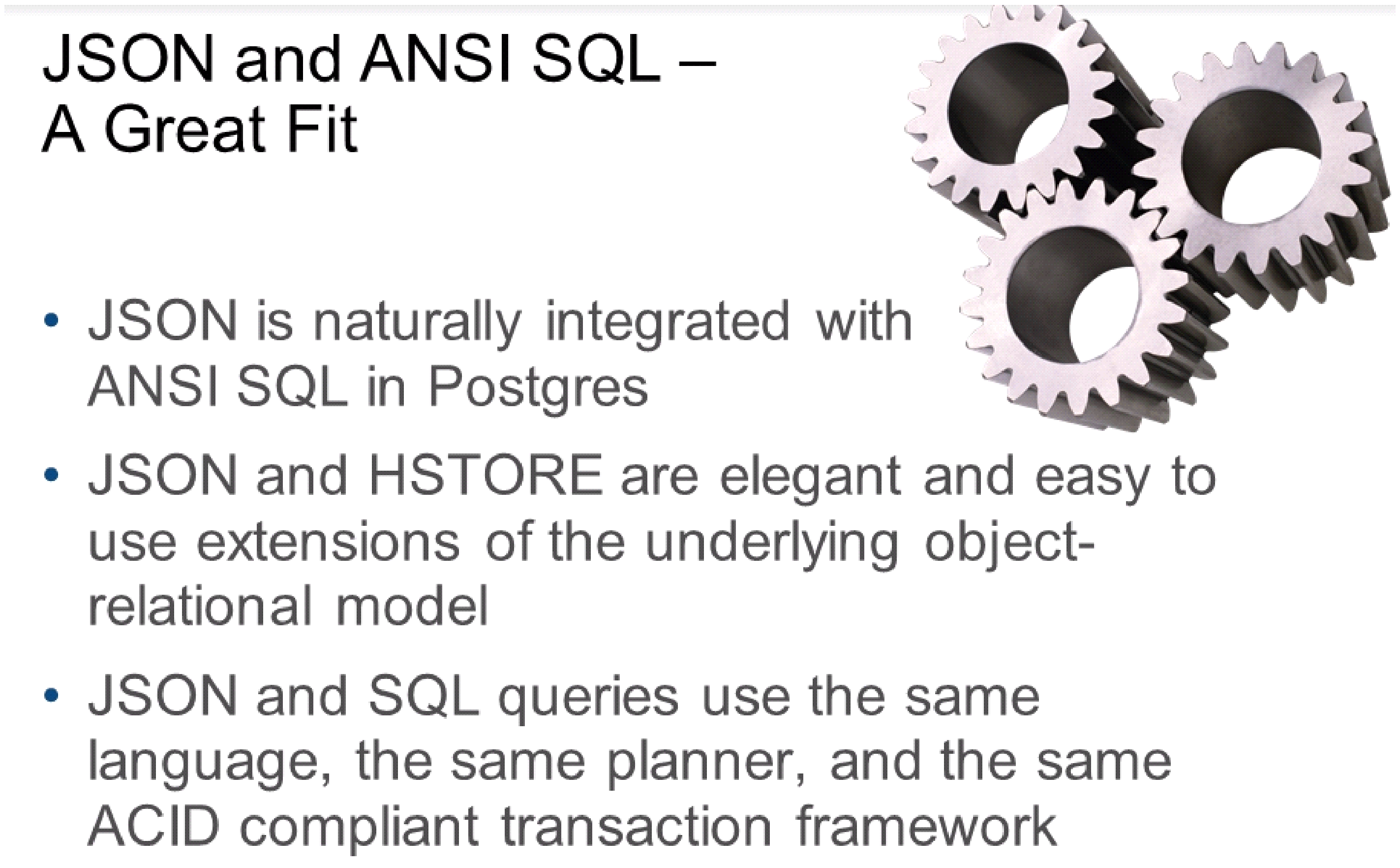
8/18/2017 3:27 PM - Screen Clipping

JSONB and Node.js - Easy as π



```
// require the Postgres connector
var pg = require("pg");

// connection to local database
var conString = "pg://postgres:password@localhost:5432/nodetraining";

var client = new pg.Client(conString);
client.connect();

// initiate the sample database
 client.query("CREATE TABLE IF NOT EXISTS emps(data jsonb)");
 client.query("TRUNCATE TABLE emps;");
 client.query('INSERT INTO emps VALUES($JSON$ {"firstname": "Ronald" , "lastname":"McDonald" }$JSON$)')
 client.query('INSERT  INTO emps values($JSON$ {"firstname": "Mayor", "lastname": "McCheese"}$JSON$)')

// run SELECT query
 client.query("SELECT * FROM emps",function(err,result){
     console.log("Test Output of JSON Result Object");
     console.log(result);
     console.log("Parsed rows");

// parse the result set
    for (var i = 0; i< result.rows.length ; i++ ){
        var data = JSON.parse(result.rows[i].data);
        console.log("First Name => "+ data.firstname + "\t| Last Name => " + data.lastname);
    }
 client.end();
})
```

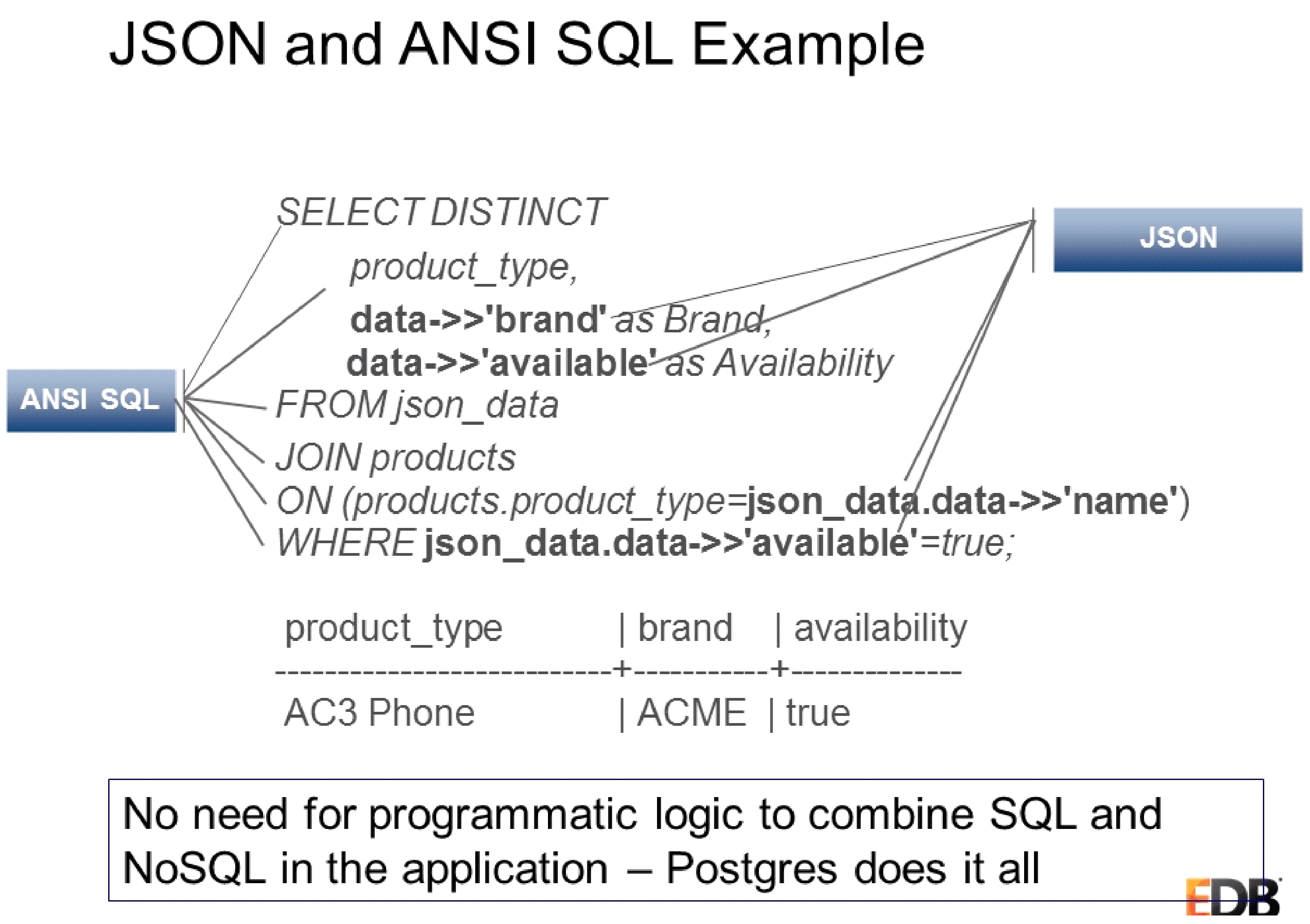
8/18/2017 3:29 PM - Screen Clipping



JSON and ANSI SQL – A Great Fit

- JSON is naturally integrated with ANSI SQL in Postgres
- JSON and HSTORE are elegant and easy to use extensions of the underlying object-relational model
- JSON and SQL queries use the same language, the same planner, and the same ACID compliant transaction framework

8/18/2017 3:30 PM - Screen Clipping


ANSI SQL
JSON

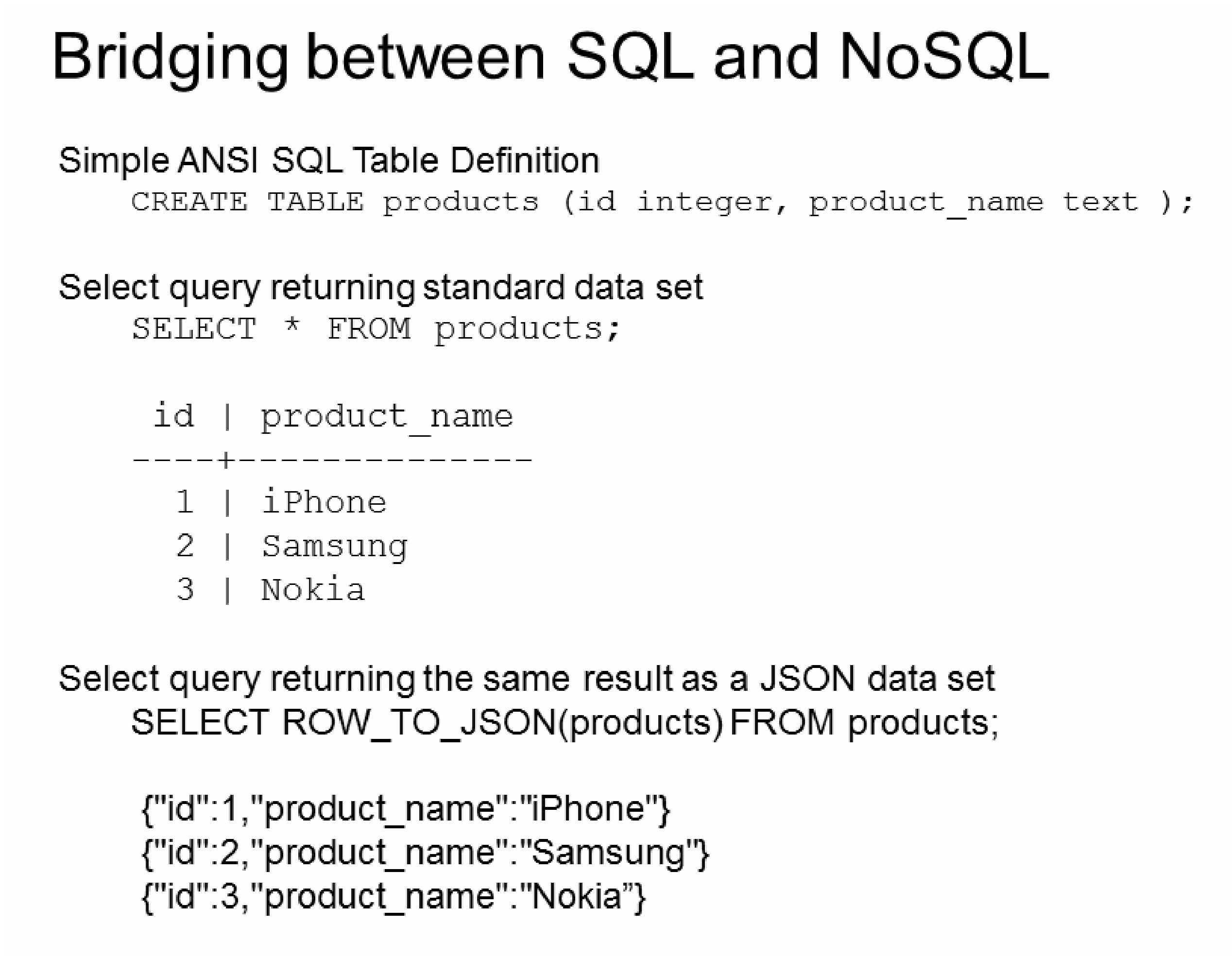
JSON and ANSI SQL Example

```
SELECT DISTINCT
    product_type,
    data->>'brand' as Brand,
    data->>'available' as Availability
FROM json_data
JOIN products
ON (products.product_type=json_data.data->>'name')
WHERE json_data.data->>'available'=true;
```

```
 product_type          | brand    | availability
-----------------------+----------+-------------
 AC3 Phone             | ACME     | true
```

No need for programmatic logic to combine SQL and NoSQL in the application – Postgres does it all

8/18/2017 3:31 PM - Screen Clipping



Bridging between SQL and NoSQL

Simple ANSI SQL Table Definition

```
CREATE TABLE products (id integer, product_name text );
```

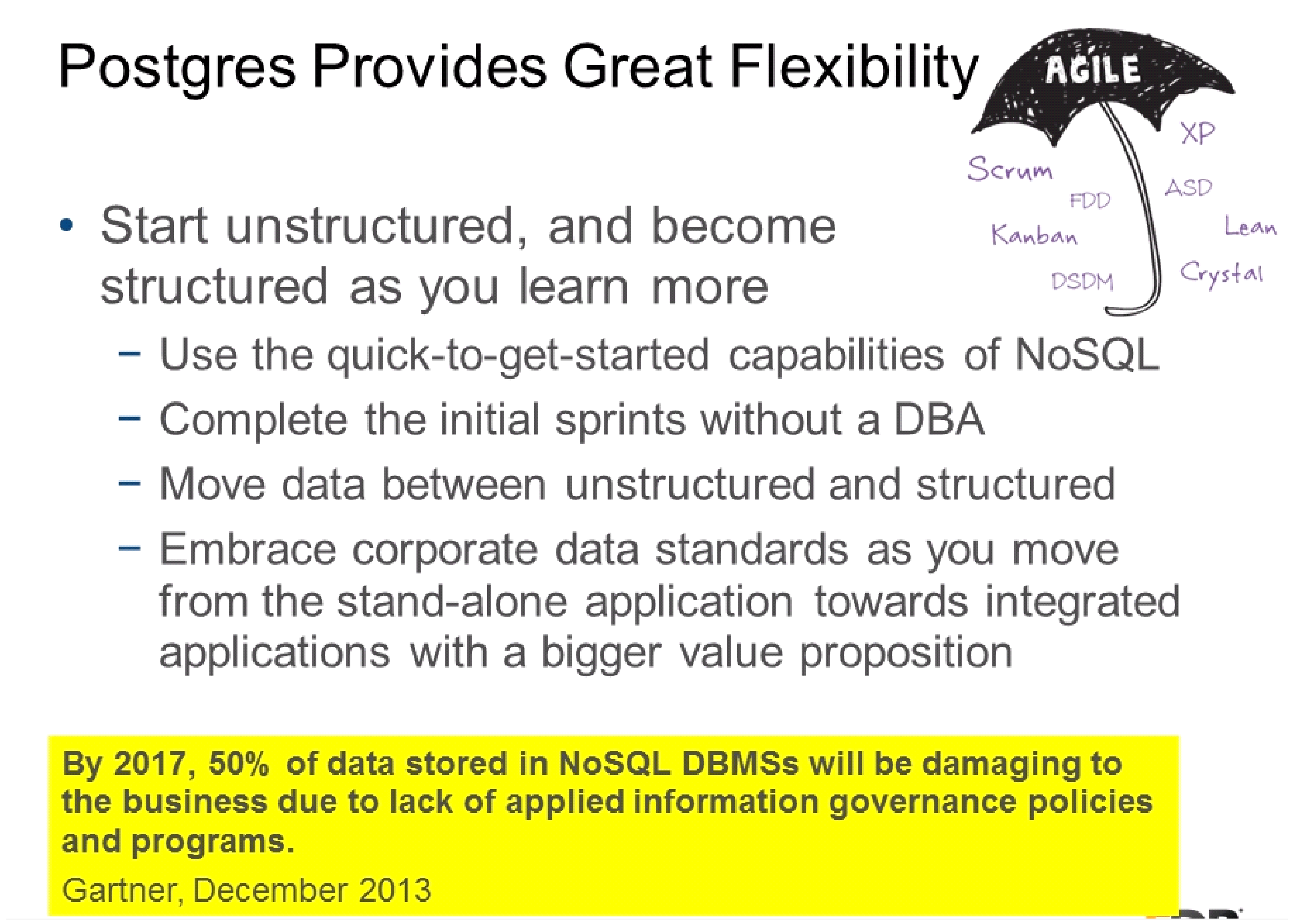
Select query returning standard data set

```
SELECT * FROM products;

 id | product_name
----+--------------
  1 | iPhone
  2 | Samsung
  3 | Nokia
```

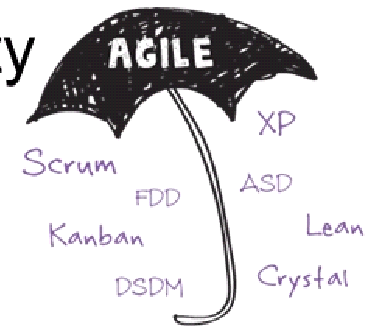
Select query returning the same result as a JSON data set

```
SELECT ROW_TO_JSON(products) FROM products;

{"id":1,"product_name":"iPhone"}
{"id":2,"product_name":"Samsung"}
{"id":3,"product_name":"Nokia”}
```

8/18/2017 3:32 PM - Screen Clipping



Postgres Provides Great Flexibility

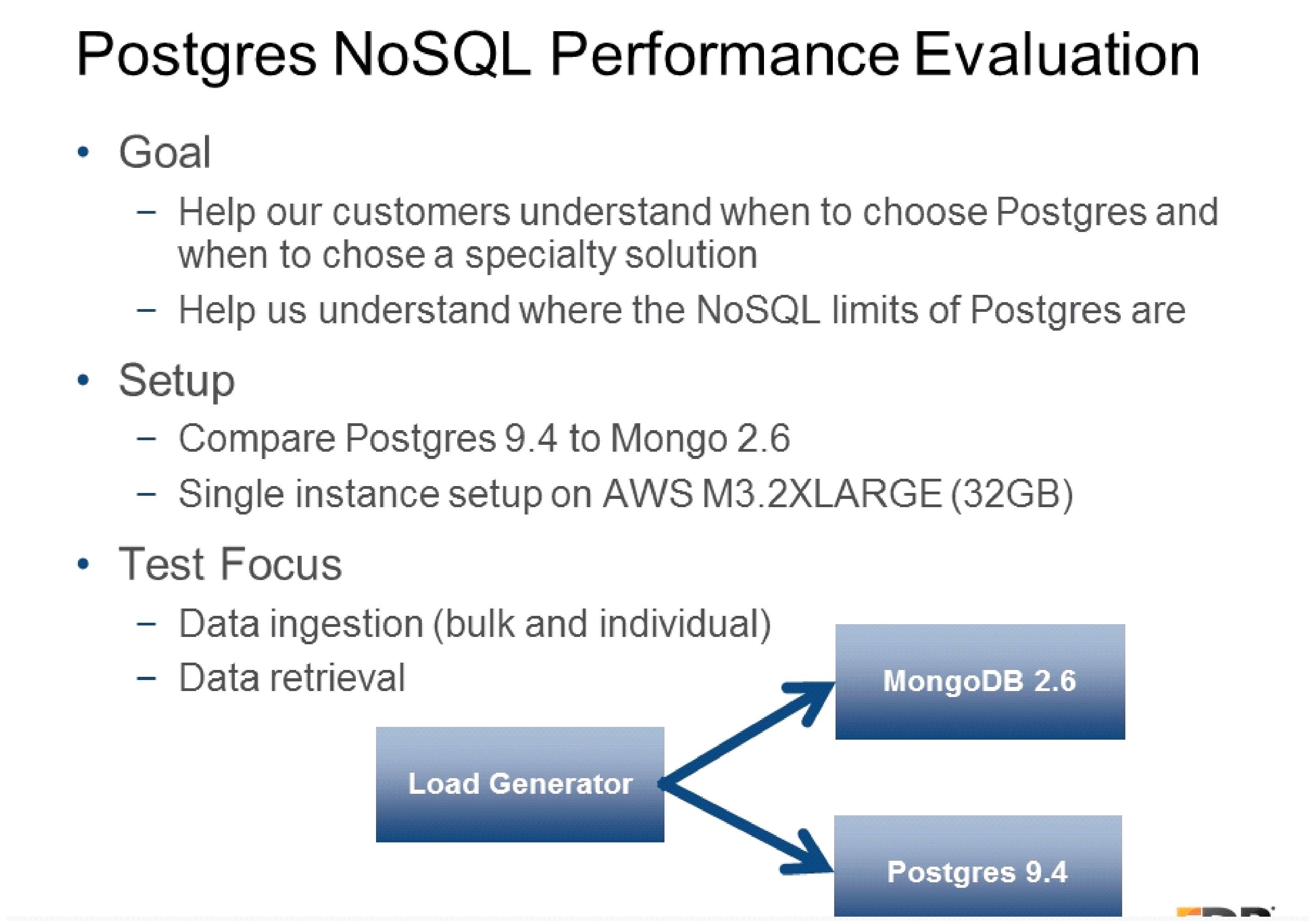


- Start unstructured, and become structured as you learn more
  - Use the quick-to-get-started capabilities of NoSQL
  - Complete the initial sprints without a DBA
  - Move data between unstructured and structured
  - Embrace corporate data standards as you move from the stand-alone application towards integrated applications with a bigger value proposition

By 2017, 50% of data stored in NoSQL DBMSs will be damaging to the business due to lack of applied information governance policies and programs.

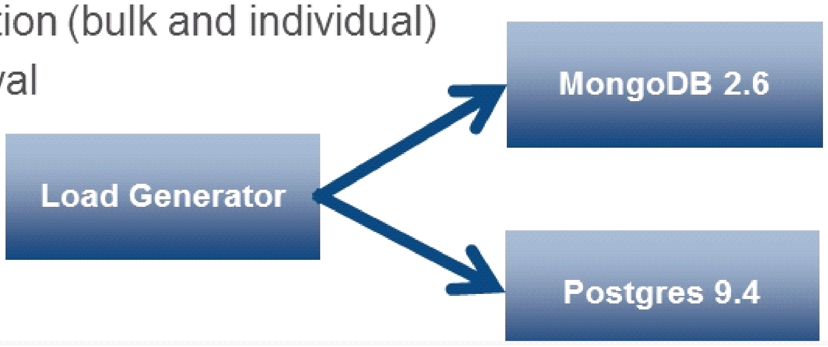
Gartner, December 2013

8/18/2017 3:34 PM - Screen Clipping



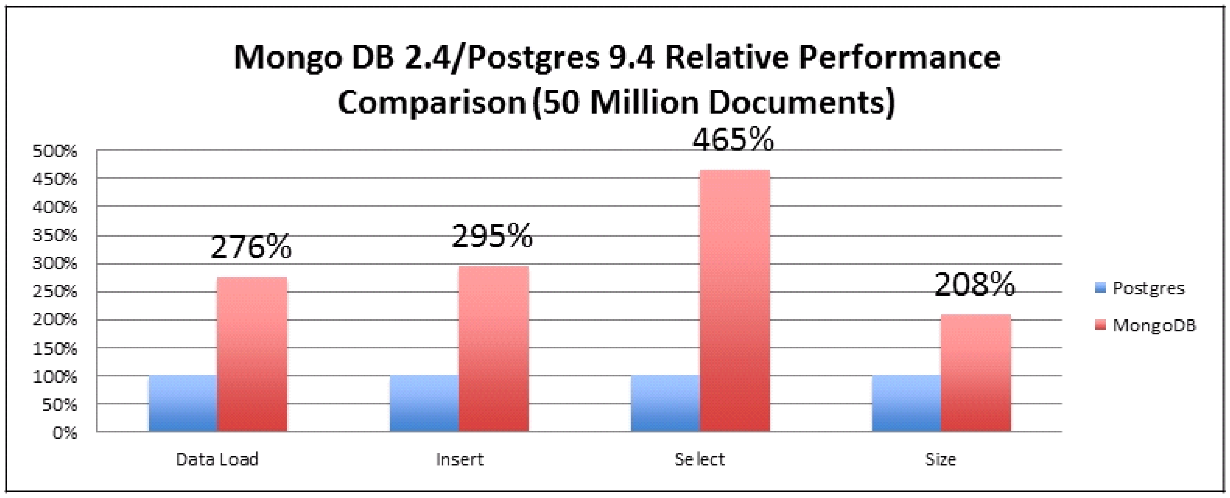
Postgres NoSQL Performance Evaluation

- Goal
  - Help our customers understand when to choose Postgres and when to chose a specialty solution
  - Help us understand where the NoSQL limits of Postgres are
- Setup
  - Compare Postgres 9.4 to Mongo 2.6
  - Single instance setup on AWS M3.2XLARGE (32GB)
- Test Focus
  - Data ingestion (bulk and individual)
  - Data retrieval



8/18/2017 3:36 PM - Screen Clipping

NoSQL Performance Evaluation



	Postgres	MongoDB
Data Load (s)	4,732	13,046
Insert (s)	29,236	86,253
Select (s)	594	2,763
Size (GB)	69	145

Correction to earlier versions:

MongoDB console does not allow for INSERT of documents > 4K. This lead to truncation of the MongoDB size by approx. 25% of all records in the benchmark.

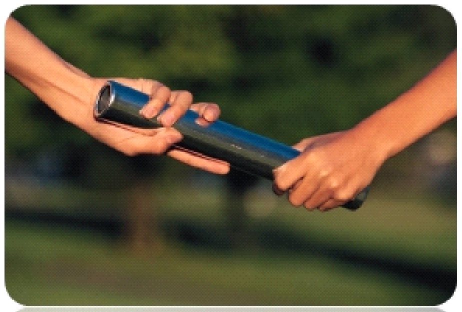
8/18/2017 5:15 PM - Screen Clipping

Postgres NoSQL Performance Evaluation

- Tests confirm that Postgres can handle many NoSQL workloads
- EDB is making the test scripts publically available
- EDB encourages community participation to better define where Postgres should be used and where specialty solutions are appropriate
- Download the source at https://github.com/EnterpriseDB/pg_nosql_benchmark

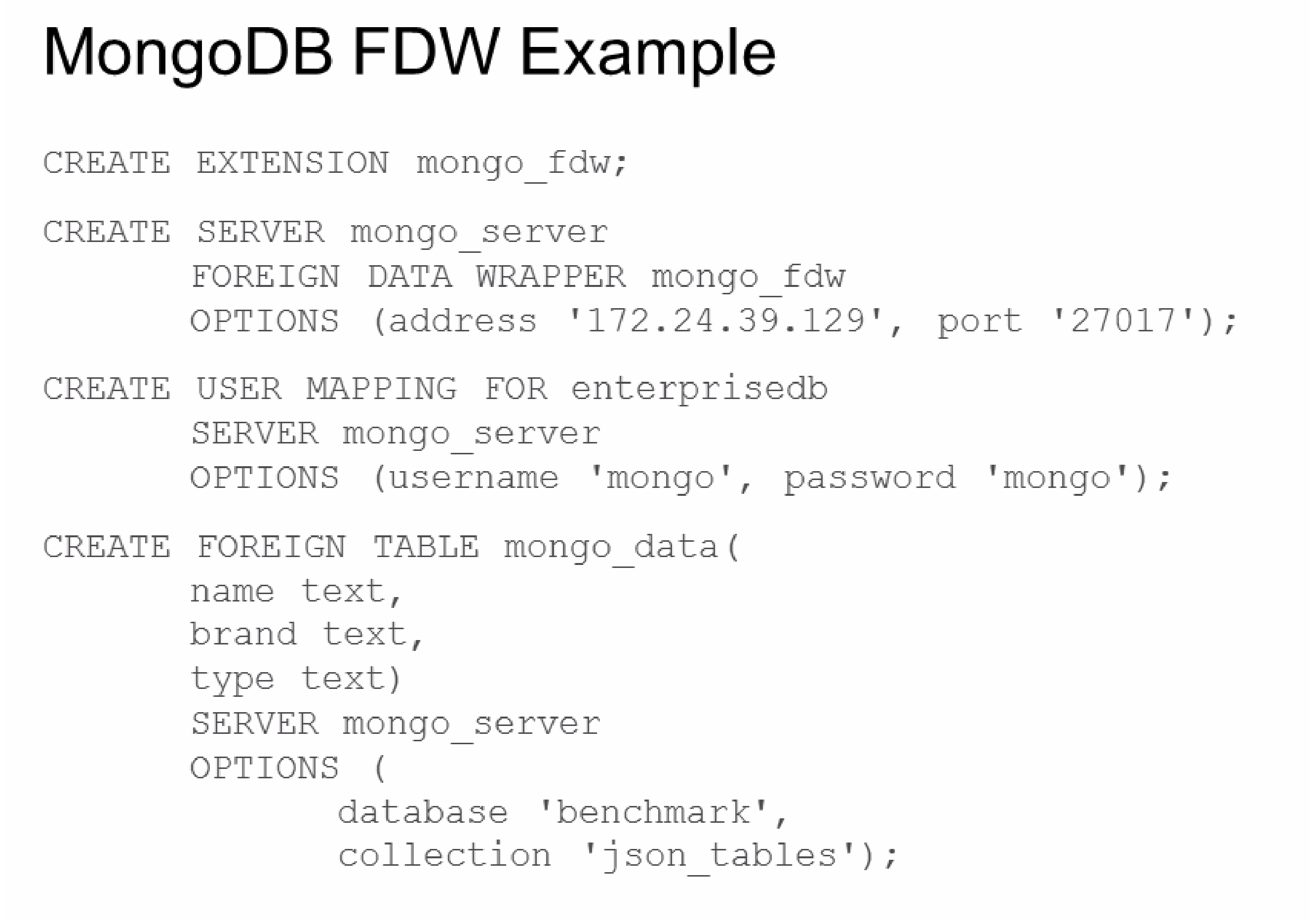
8/18/2017 5:16 PM - Screen Clipping

Foreign Data Wrappers – Co-Existence Platform



- FDW implements SQL/MED ("SQL Management of External Data")
- PostgreSQL 9.1 - read-only support
- PostgreSQL 9.3 – read/write support
- FDW
  - Makes data on other servers (or services) look like tables in Postgres
  - available for databases (MongoDB, MySQL, Oracle, …), files, services (Twitter, …)
- MongoDB FDW: https://github.com/EnterpriseDB

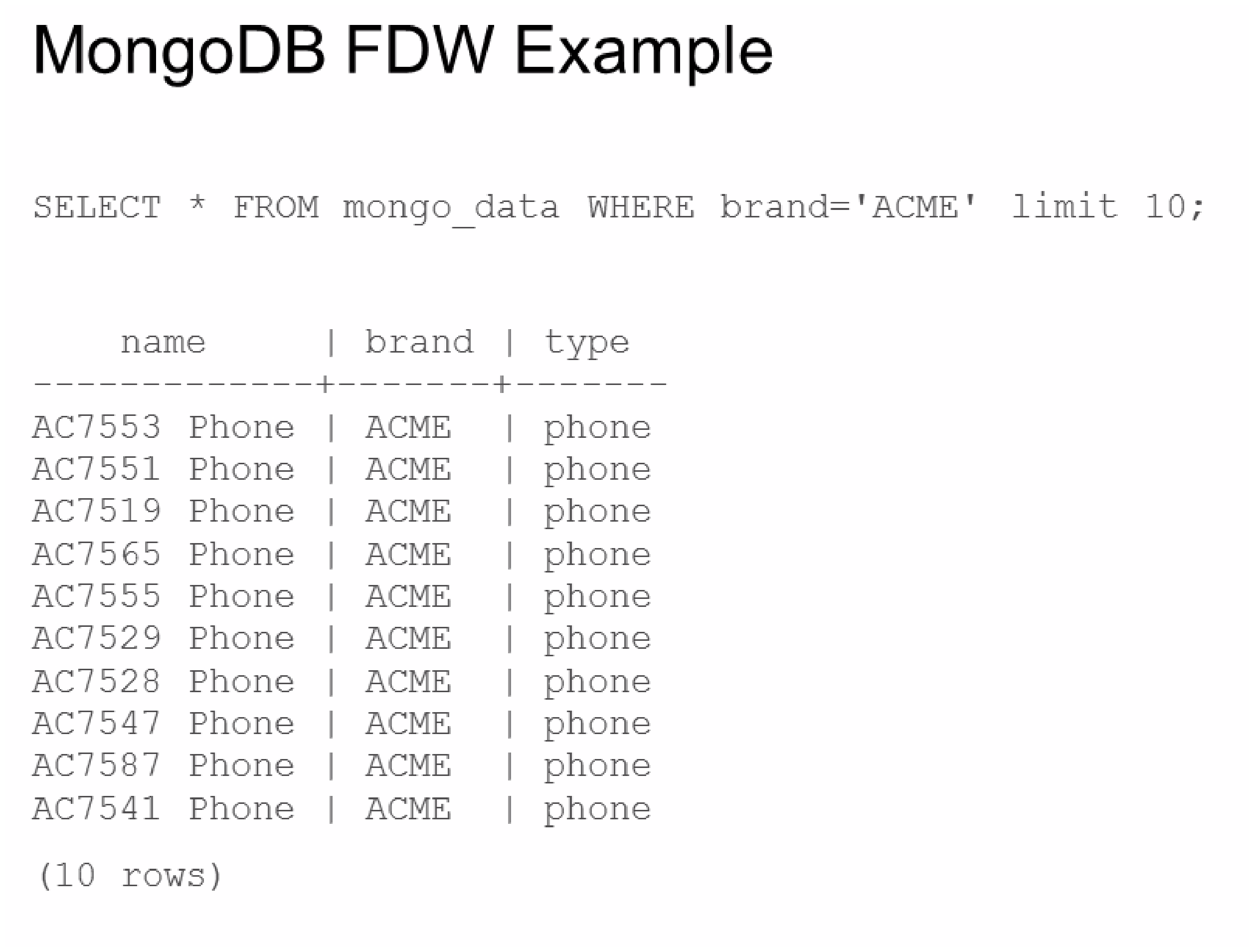
8/18/2017 5:17 PM - Screen Clipping



MongoDB FDW Example

```
CREATE EXTENSION mongo_fdw;

CREATE SERVER mongo_server
       FOREIGN DATA WRAPPER mongo_fdw
       OPTIONS (address '172.24.39.129', port '27017');

CREATE USER MAPPING FOR enterprisedb
       SERVER mongo_server
       OPTIONS (username 'mongo', password 'mongo');

CREATE FOREIGN TABLE mongo_data(
       name text,
       brand text,
       type text)
       SERVER mongo_server
       OPTIONS (
              database 'benchmark',
              collection 'json_tables');
```

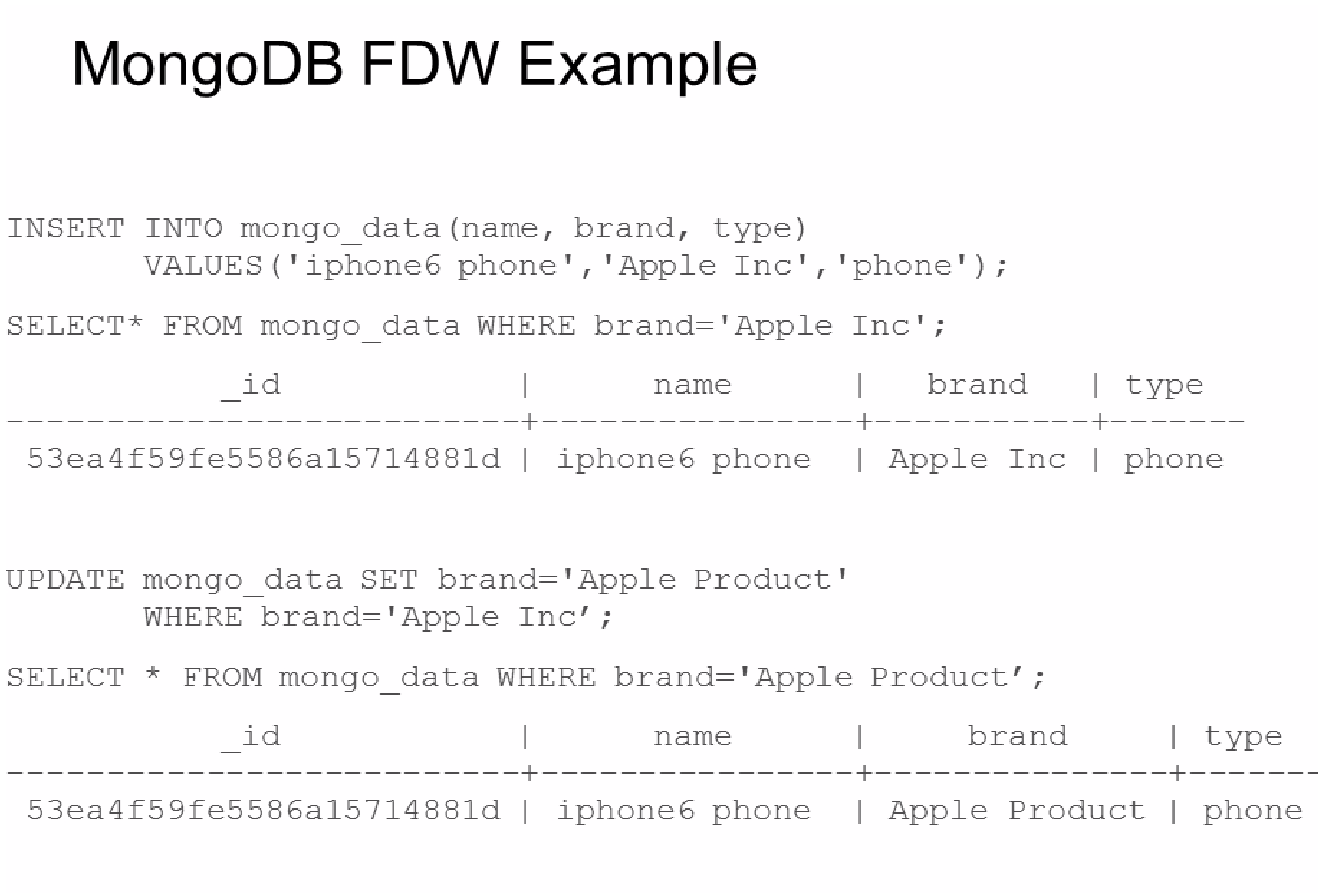
8/18/2017 5:19 PM - Screen Clipping



MongoDB FDW Example

```
SELECT * FROM mongo_data WHERE brand='ACME' limit 10;


    name     | brand | type
-------------+-------+-------
AC7553 Phone | ACME  | phone
AC7551 Phone | ACME  | phone
AC7519 Phone | ACME  | phone
AC7565 Phone | ACME  | phone
AC7555 Phone | ACME  | phone
AC7529 Phone | ACME  | phone
AC7528 Phone | ACME  | phone
AC7547 Phone | ACME  | phone
AC7587 Phone | ACME  | phone
AC7541 Phone | ACME  | phone
(10 rows)
```

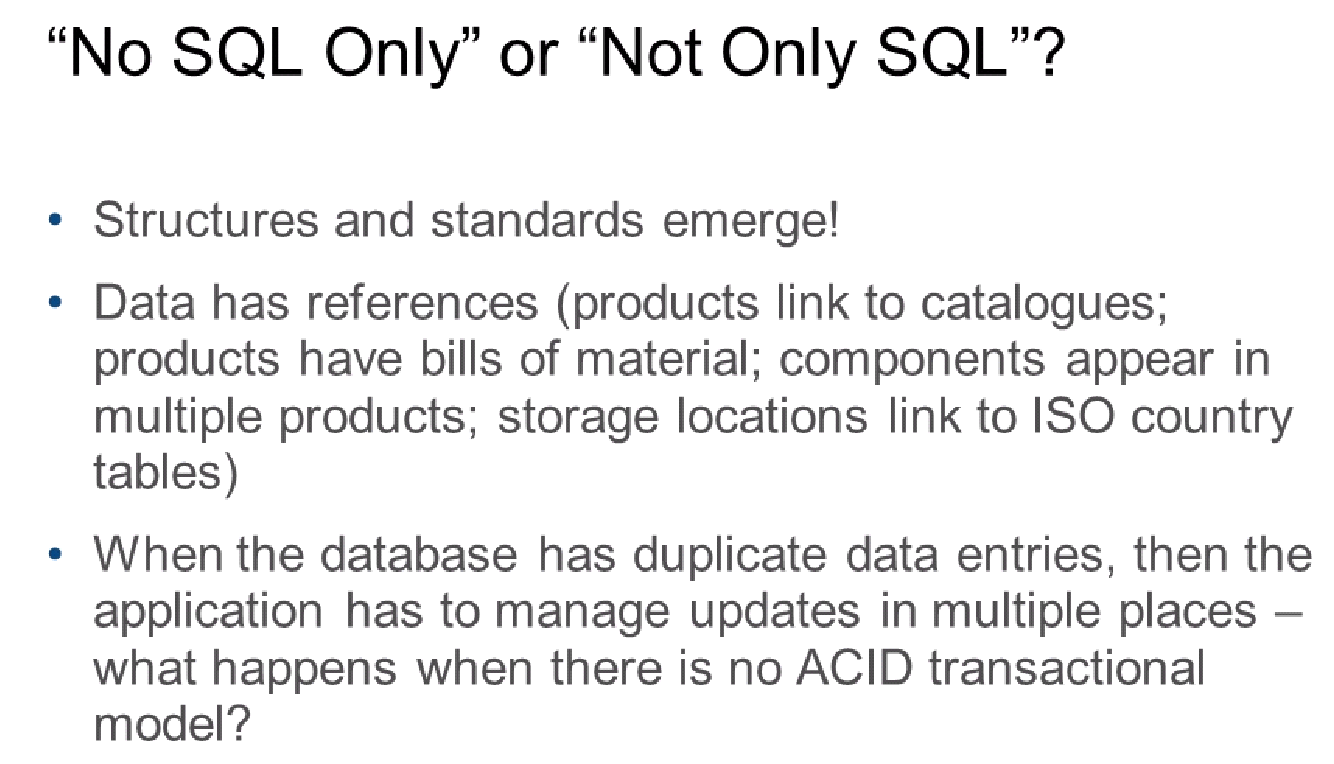
8/18/2017 5:20 PM - Screen Clipping



MongoDB FDW Example

```
INSERT INTO mongo_data(name, brand, type)
       VALUES('iphone6 phone','Apple Inc','phone');

SELECT* FROM mongo_data WHERE brand='Apple Inc';

          _id             |      name       |   brand   | type
--------------------------+-----------------+-----------+-------
 53ea4f59fe5586a15714881d | iphone6 phone   | Apple Inc | phone


UPDATE mongo_data SET brand='Apple Product'
       WHERE brand='Apple Inc';

SELECT * FROM mongo_data WHERE brand='Apple Product';

          _id             |      name       |     brand     | type
--------------------------+-----------------+---------------+-------
 53ea4f59fe5586a15714881d | iphone6 phone   | Apple Product | phone
```

8/18/2017 5:24 PM - Screen Clipping

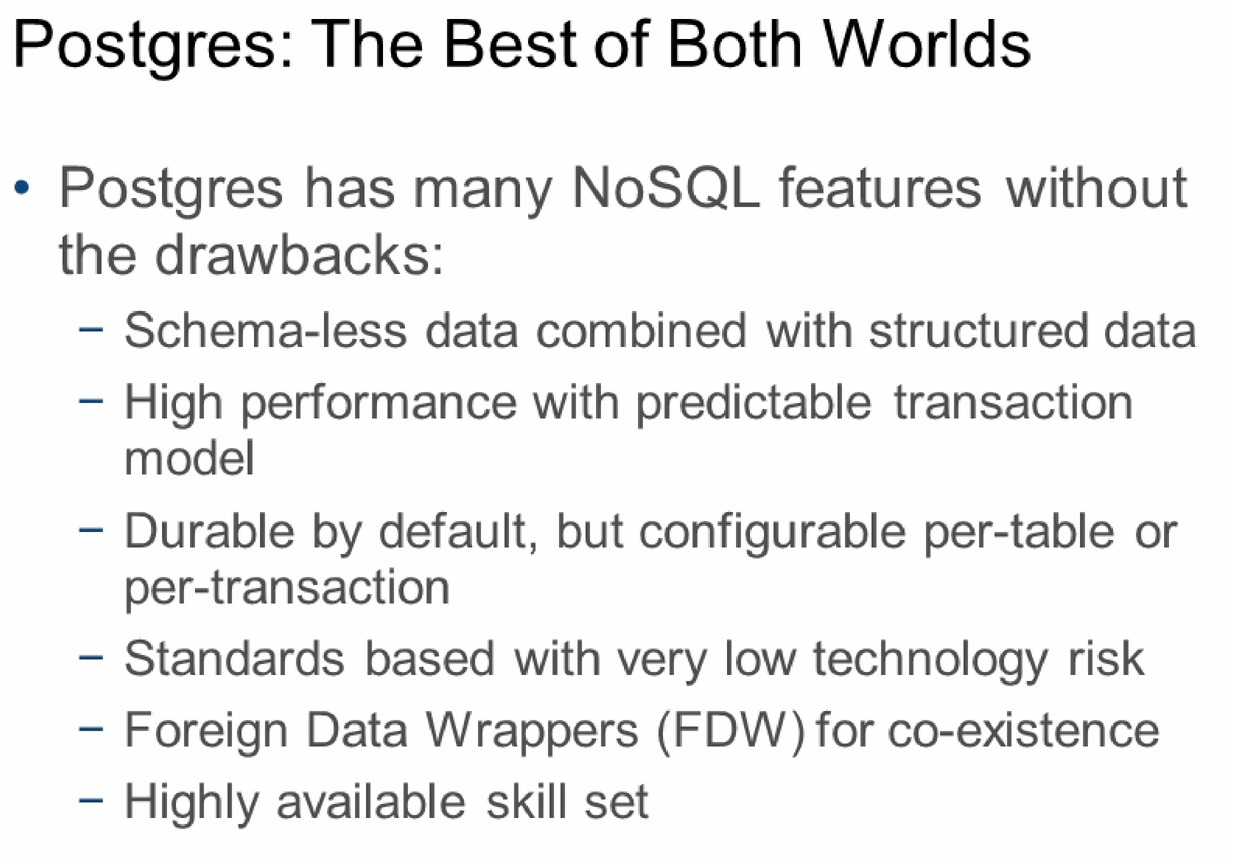


"No SQL Only" or "Not Only SQL"?

- Structures and standards emerge!
- Data has references (products link to catalogues; products have bills of material; components appear in multiple products; storage locations link to ISO country tables)
- When the database has duplicate data entries, then the application has to manage updates in multiple places – what happens when there is no ACID transactional model?

8/18/2017 5:26 PM - Screen Clipping

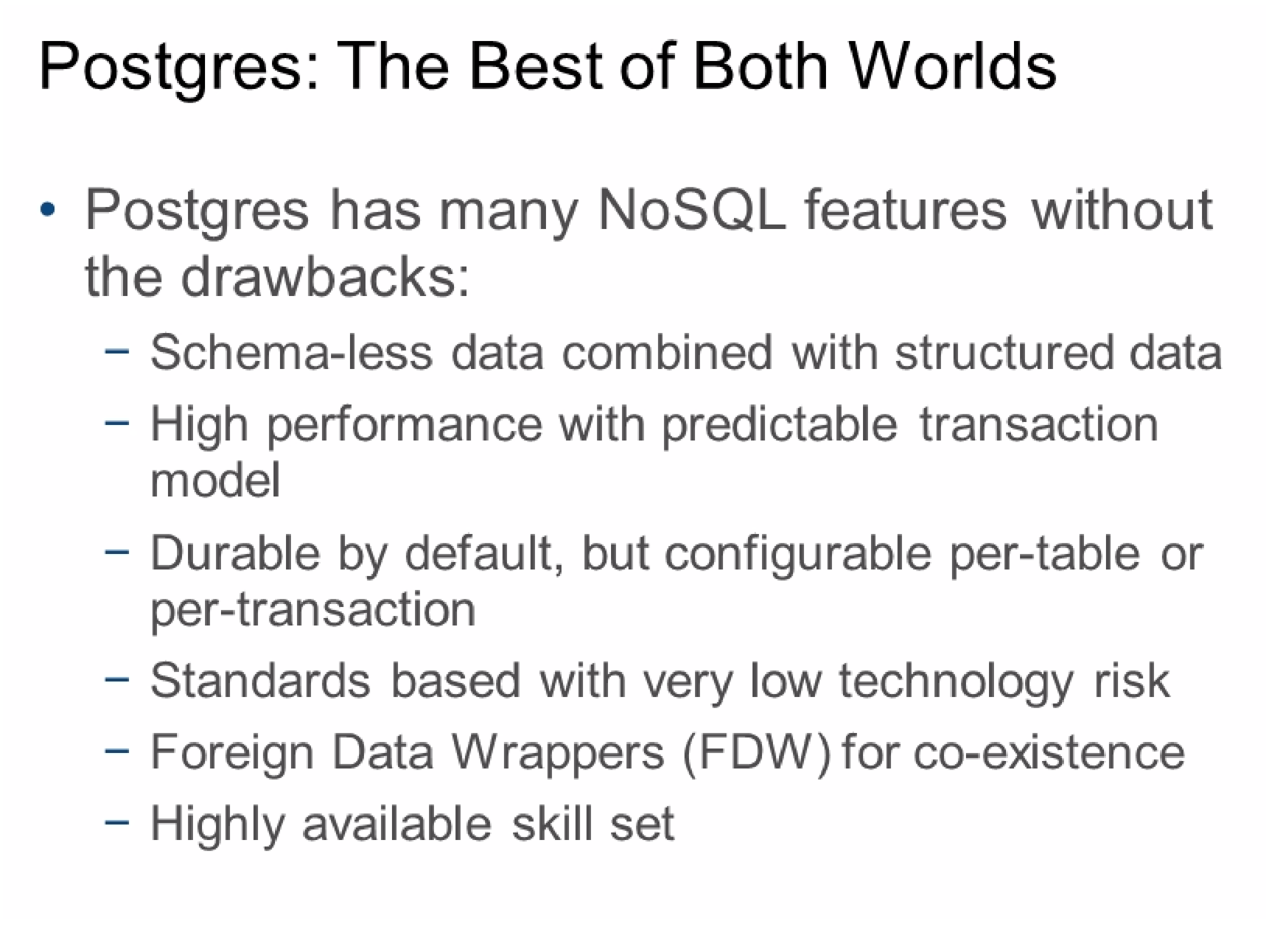
- NoSQL only databases are not ACID transactional. Which means, if you tell go and change this data element in 5 collections, there is no guarantee to check if all of your changes happen or none of your changes happen.
- There is no concept of atomicity in NoSQL only databases



Postgres: The Best of Both Worlds

- Postgres has many NoSQL features without the drawbacks:
  - Schema-less data combined with structured data
  - High performance with predictable transaction model
  - Durable by default, but configurable per-table or per-transaction
  - Standards based with very low technology risk
  - Foreign Data Wrappers (FDW) for co-existence
  - Highly available skill set

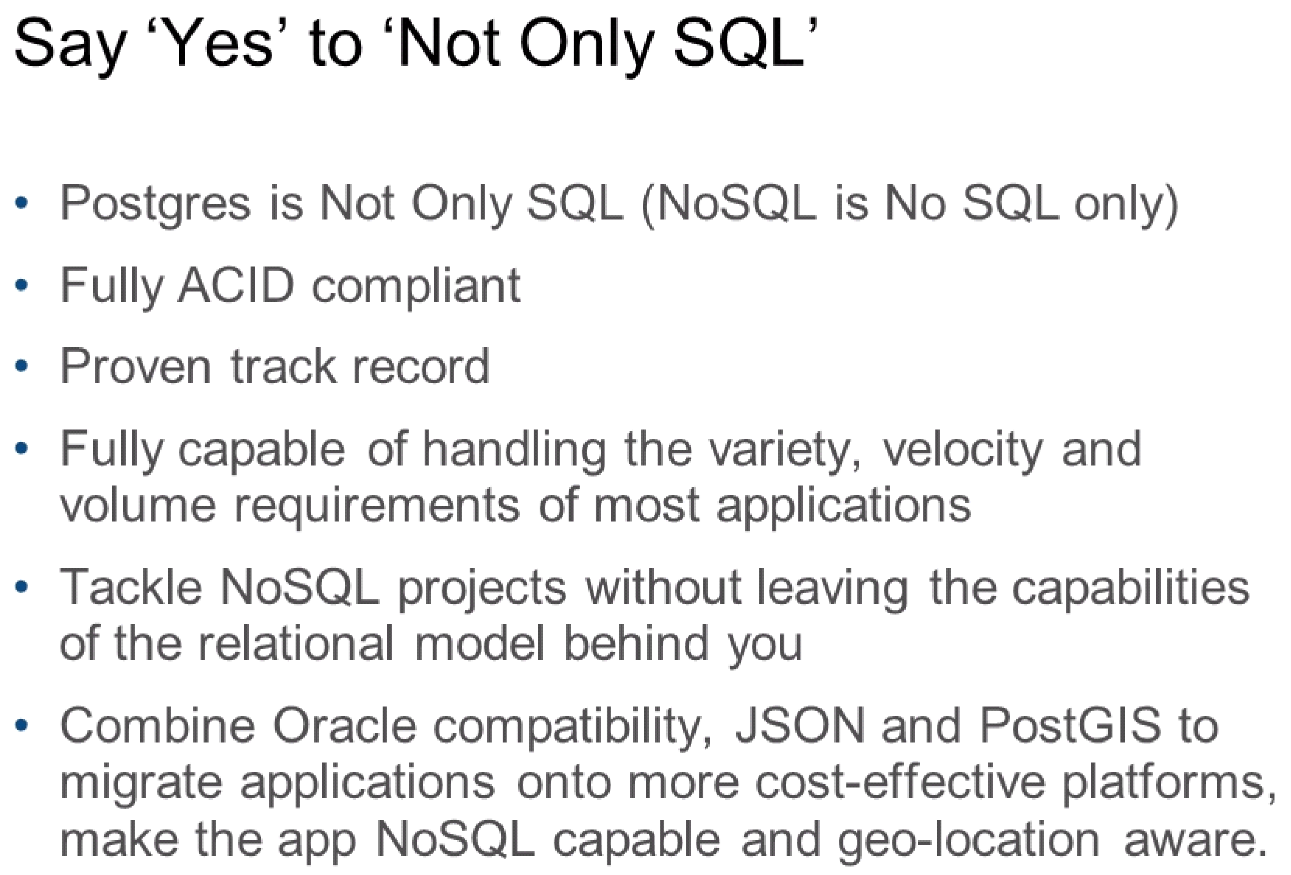
8/18/2017 5:33 PM - Screen Clipping



Postgres: The Best of Both Worlds

- Postgres has many NoSQL features without the drawbacks:
  - Schema-less data combined with structured data
  - High performance with predictable transaction model
  - Durable by default, but configurable per-table or per-transaction
  - Standards based with very low technology risk
  - Foreign Data Wrappers (FDW) for co-existence
  - Highly available skill set

8/18/2017 5:35 PM - Screen Clipping



Say ‘Yes’ to ‘Not Only SQL’

- Postgres is Not Only SQL (NoSQL is No SQL only)
- Fully ACID compliant
- Proven track record
- Fully capable of handling the variety, velocity and volume requirements of most applications
- Tackle NoSQL projects without leaving the capabilities of the relational model behind you
- Combine Oracle compatibility, JSON and PostGIS to migrate applications onto more cost-effective platforms, make the app NoSQL capable and geo-location aware.

8/18/2017 5:36 PM - Screen Clipping