NOSQL in Postgres

Play with MongoDB

品名：萧少聪(铁庵) Scott Siu
产地：中国 广东 中山

Postgres中国用户会 创始人之一
阿里云 RDS for PG/PPAS产品经理

EnterpriseDB认证数据库专家
RedHat RHCA认证架构师

Who am I

微信扫我(WeChat)

PostgreSQL

提　纲

- JSON的背景
- 要NoSQL也要ACID
- Postgres: JSON in SQL
- Play with MongoDB

JSON的背景

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。 这些特性使JSON成为理想的数据交换语言。

JSON 已经是 JavaScript 标准的一部分。目前，主流的浏览器对 JSON 支持都非常完善。应用 JSON，我们可以从 ~~XML~~ 的解析中摆脱出来，对那些应用 Ajax 的 Web 2.0 网站来说，JSON 确实是目前最灵活的轻量级方案。

JSON的 背景

XML

```
<book>
    <type>textbook</type>
    <pages>256</pages>
    <title>Programming Pearls 2nd Edition</title>
    <description>The first edition of Programming Pearls was one
of the most influential books I read early in my career...</
description>
    <rating>4.5</rating>
    <coverType>paperback</coverType>
    <genre>Computer Science</genre>
    <author>Jon Bentley</author>
    <publisher>Addison-Wesley Professional</publisher>
    <copyright>1999</copyright>
</book>
```

JSON的 背景

JSON

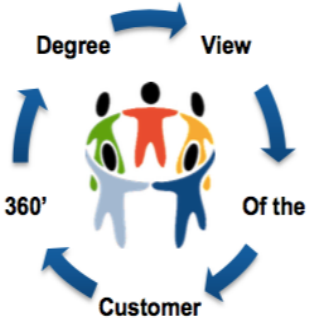

```
{
 "book": {
    "type": "textbook",
    "pages": "256",
    "title": "Programming Pearls 2nd Edition",
    "description": "The first edition of Programming Pearls was one of the most influential books I read early in my career...",
    "rating": "4.5",
    "coverType": "paperback",
    "genre": "Computer Science",
    "author": "Jon Bentley",
    "publisher": "Addison-Wesley Professional",
    "copyright": "1999"
    }
 }
```

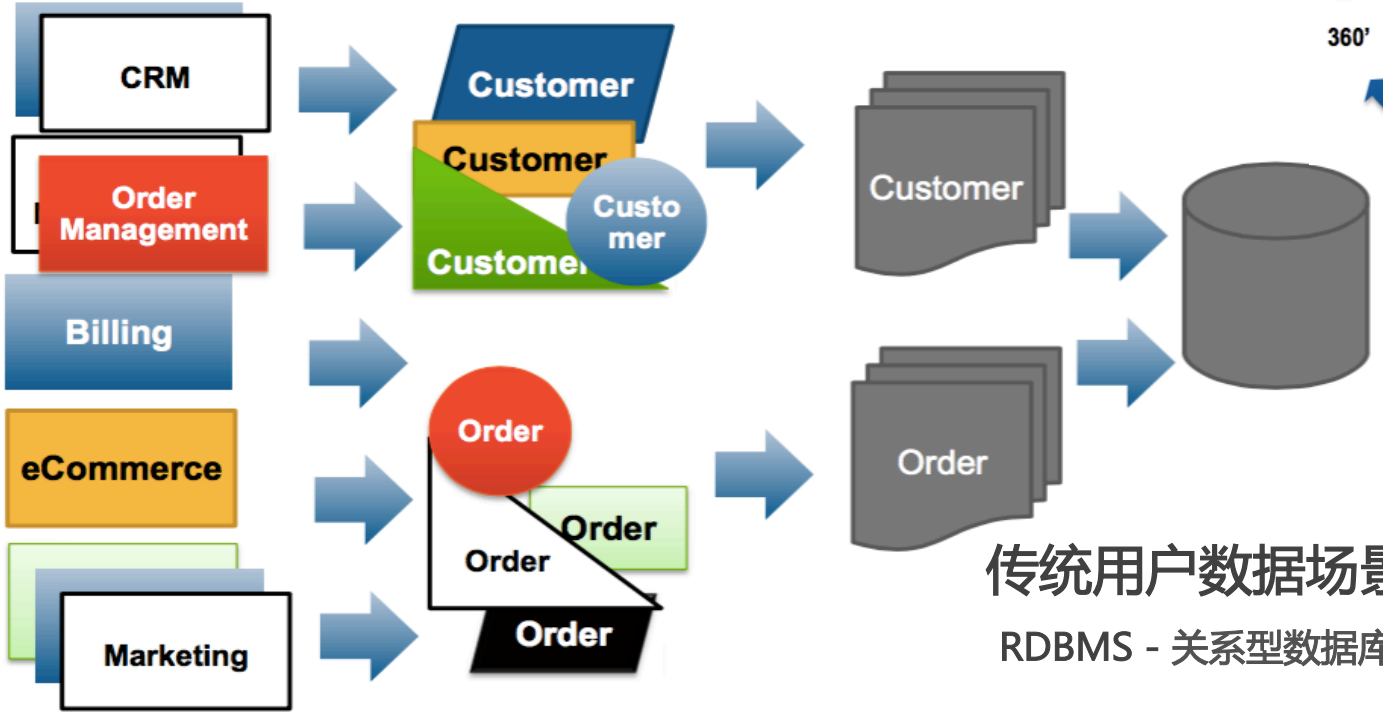
JSON的背景

使用上面的 ~~XML~~ 和 JSON 文件分别运行解析测试10,000,000次。结果并不令人惊讶，解析和转换 JSON 成一个Java对象的速度比 ~~XML~~ 解析速度提高了30%,占用空间少30%。这些结果似乎和多数开发社区对两种格式的看法一样。

所以，换用 JSON 处理数据在性能上可以有不小的提升，而且还会减少空间的占用。

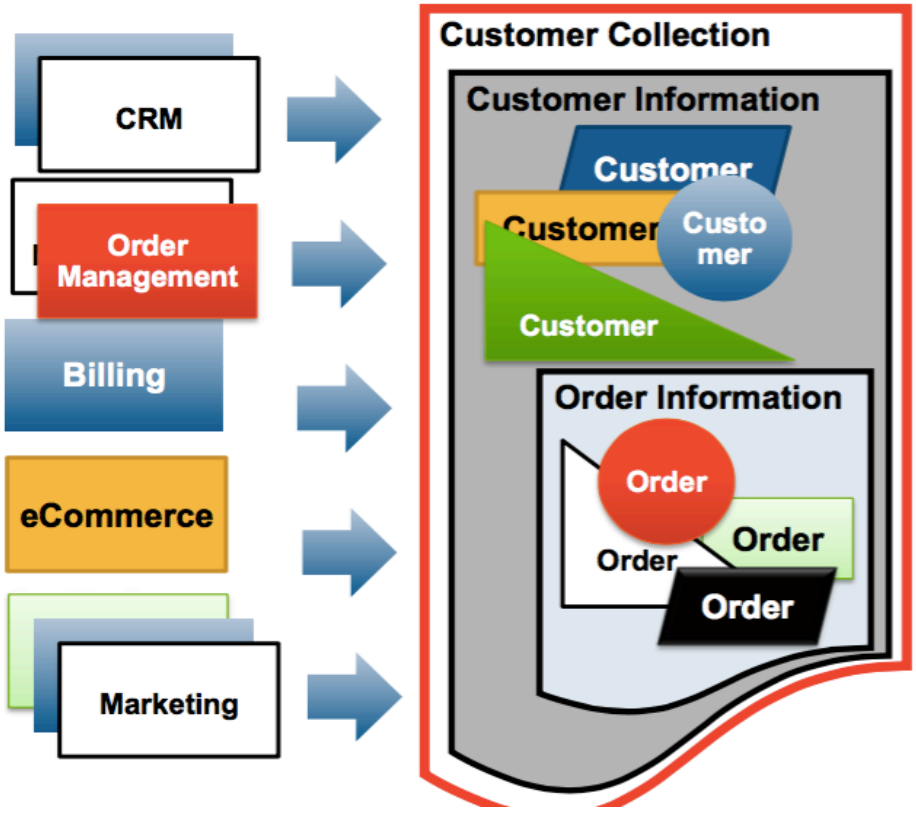
要NoSQL与要ACID

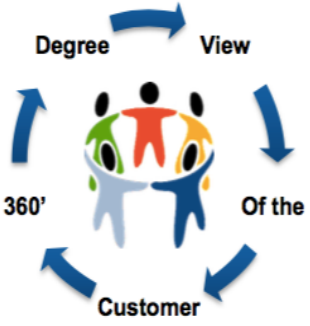




PostgreSQL

要NoSQL与要ACID



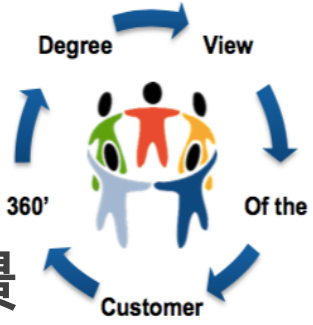


- 灵活存储各种格式
- 应用定义数据类型
- 大表，大字段
- 可拆可合

移动互联网新数据场景

NoSQL – 非关系型数据库

要NoSQL与要ACID



传统用户数据场景

- 强一致性
- 分段进行
- 规范统一

移动互联网数据场景

- 灵活构建
- 整体存储
- 分段展现

左边Insert，右边save

左边Select，右边find

码农就是996

终生无缘007

Postgres: JSON in SQL

- HSTORE
  - Key-value pair
  - Simple, fast and easy
  - Postgres v 8.2
  - pre-dates many NoSQL-only solutions
  - Ideal for flat data structures that are sparsely populated
- 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

Postgres: JSON in SQL

- *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} ')
```

Postgres: JSON in SQL

- *支持嵌套、数组:*

```
{ “full name” : “John Joseph Carl Salinger” , “names” :
    [
    {"type": "firstname", “value” : ” John” },
    { “type” : “middlename” , “value” : “Joseph” },
    { “type” : “middlename” , “value” : “Carl” },
    { “type” : “lastname” , “value” : “Salinger” }
    ]
}
```

Postgres: JSON in SQL


```
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
```

注意：这里返回的不是JSON格式的数据，而是返回普通的字符串类型数据，传统应用可以直接读取进行展示

Postgres: JSON in SQL


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

注意：这里返回的是JSON格式的字符串类型，可以直接通过如Node.js等进行直接读取，解析并展示到页面上

Postgres: JSON in SQL

- 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.

Postgres: JSON in SQL

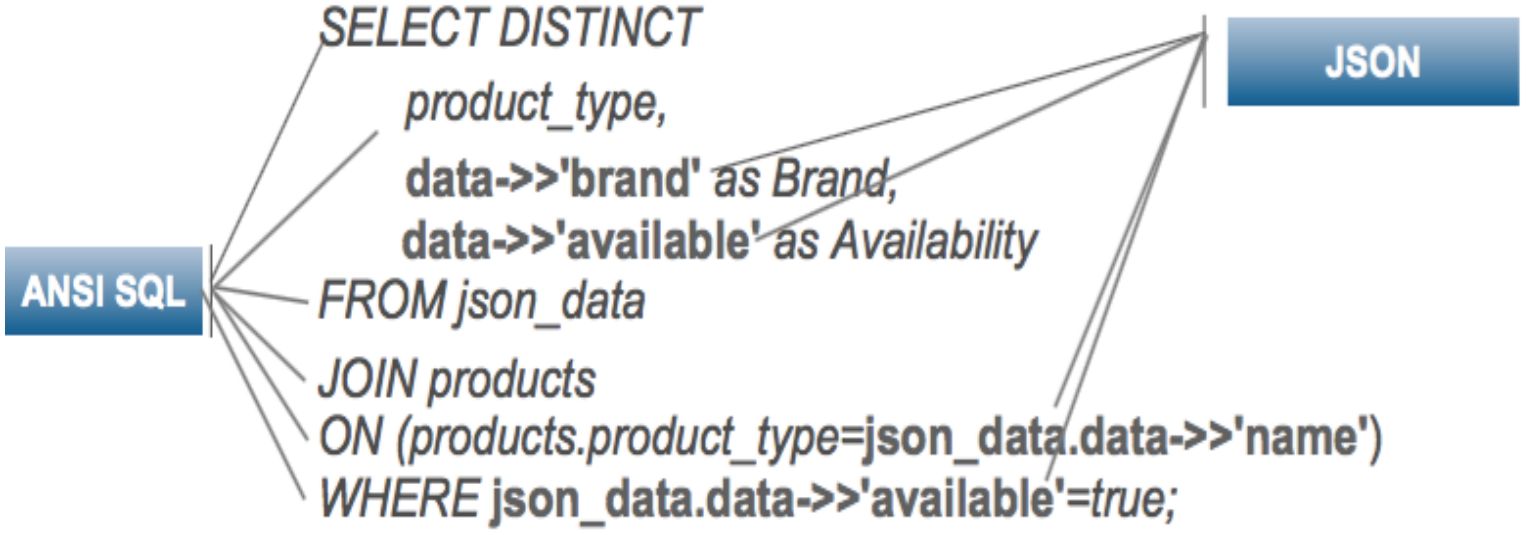
- 4. Array
  - An ordered list of zero or more values,
  - Each values may be of any type.
  - Arrays use square bracket notation with elements being comma-separated.
- 5. Object
  - An unordered associative array (name/value pairs).
  - Objects are delimited with curly brackets
  - Commas to separate 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

Postgres: JSON in SQL


```
{ "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
    { "type": "home”, "number": "212 555-1234” },    // Object
    { "type": "office”, "number": "646 555-4567” } ],
"children": [],
"spouse": null
}
```

Postgres: JSON in SQL

```
CREATE TABLE json_data (data jsonb);
INSERT INTO json_data VALUES ('{"name":"xxx","brand":"aaa", "available":true}');
```



Postgres: JSON in SQL

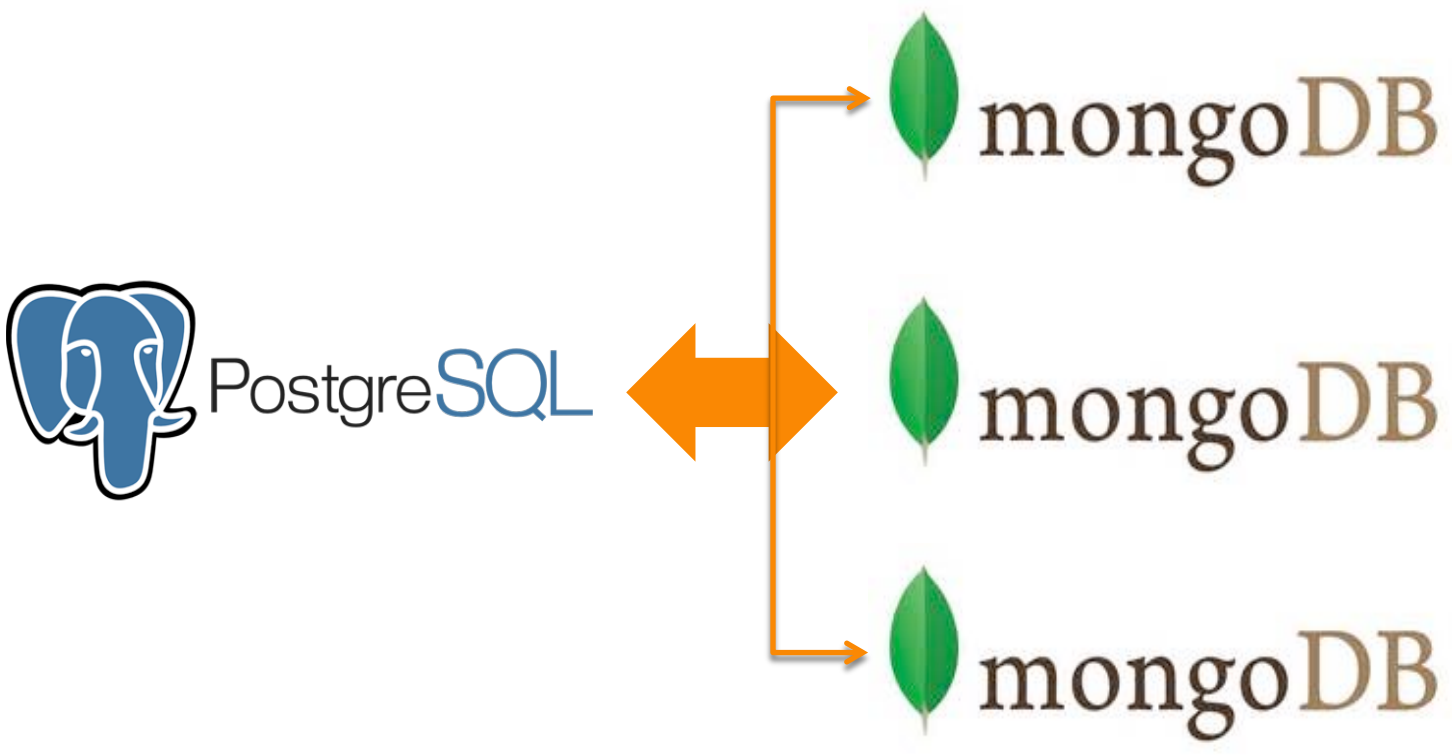
- JSONB 数据类型
  - Canonical representation
    - Whitespace and punctuation dissolved away
    - Only one value per object key is kept
    - Last insert wins
    - Key order determined by length, then bytewise comparison
  - Equality, containment and key/element presence tests
  - Smaller, faster GIN indexes
  - jsonb subdocument indexes
    - Use “get” operators to construct expression indexes on subdocument:
    - CREATE INDEX author_index ON books USING GIN ((jsondata -> 'authors'));
    - SELECT * FROM books WHERE jsondata -> 'authors' ? 'Carl Bernstein'

Postgres: JSON in SQL


```
// 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();
})
```

PostgreSQL

Play with MongoDB



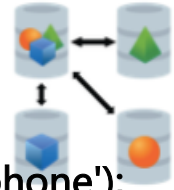
Play with MongoDB

```
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 postgresql
      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');
```

Play with MongoDB

```
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)
```

Play with MongoDB



```
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
```

PostgreSQL

- Not Only SQL(NOSQL) -

SQL + NoSQL
=
Not Only SQL

来，和我一起成为Postgres的新生代！

谢谢！

Q & A

微信扫我(WeChat)