goto; copenhagen

GOTO Copenhagen 2019
Conference Nov. 18 - 20

The Hidden Costs of Poor Database Access

Denis Rosa

Developer Advocate at Couchbase

@deniswsrosa

Follow us @gotocph

WE ARE GETTING PAID TO BUILD

“GOOD ENOUGH” SOFTWARE

Performance

Vs

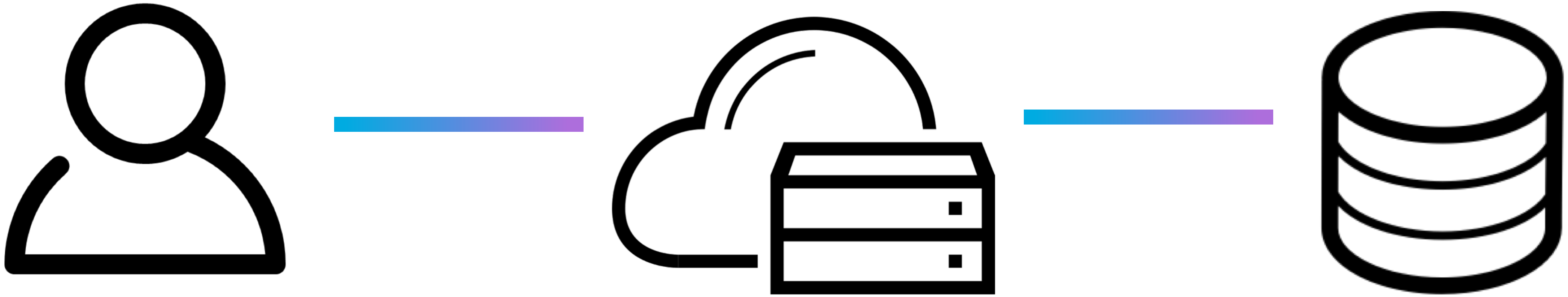
Productivity

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

Performance

Vs

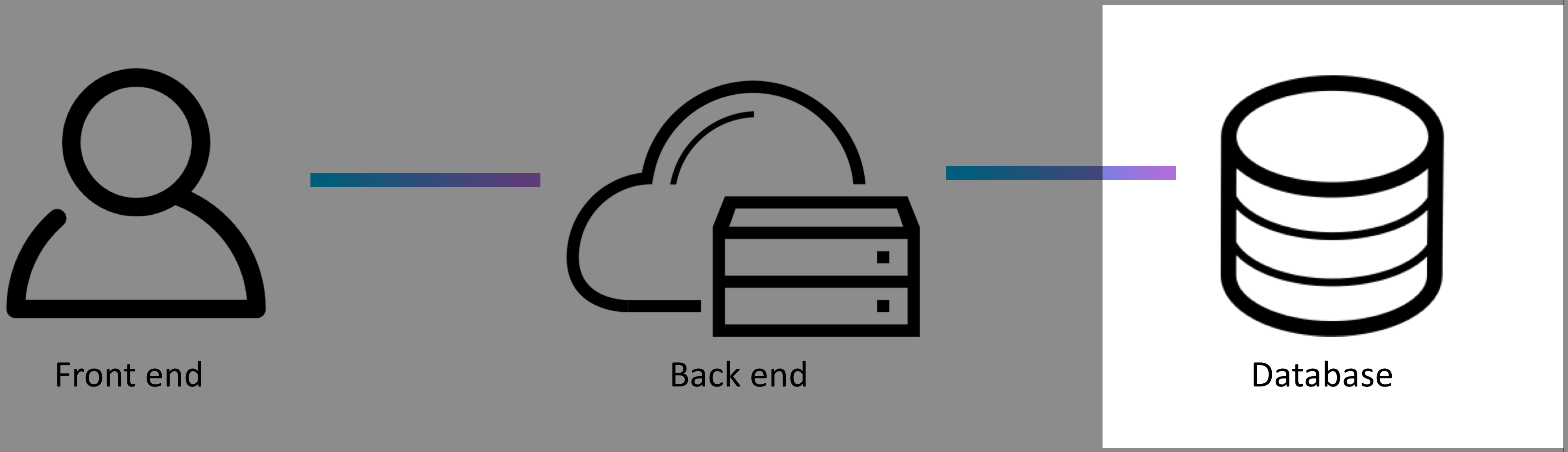
...



Front end

Back end

Database


Front end
Back end
Database

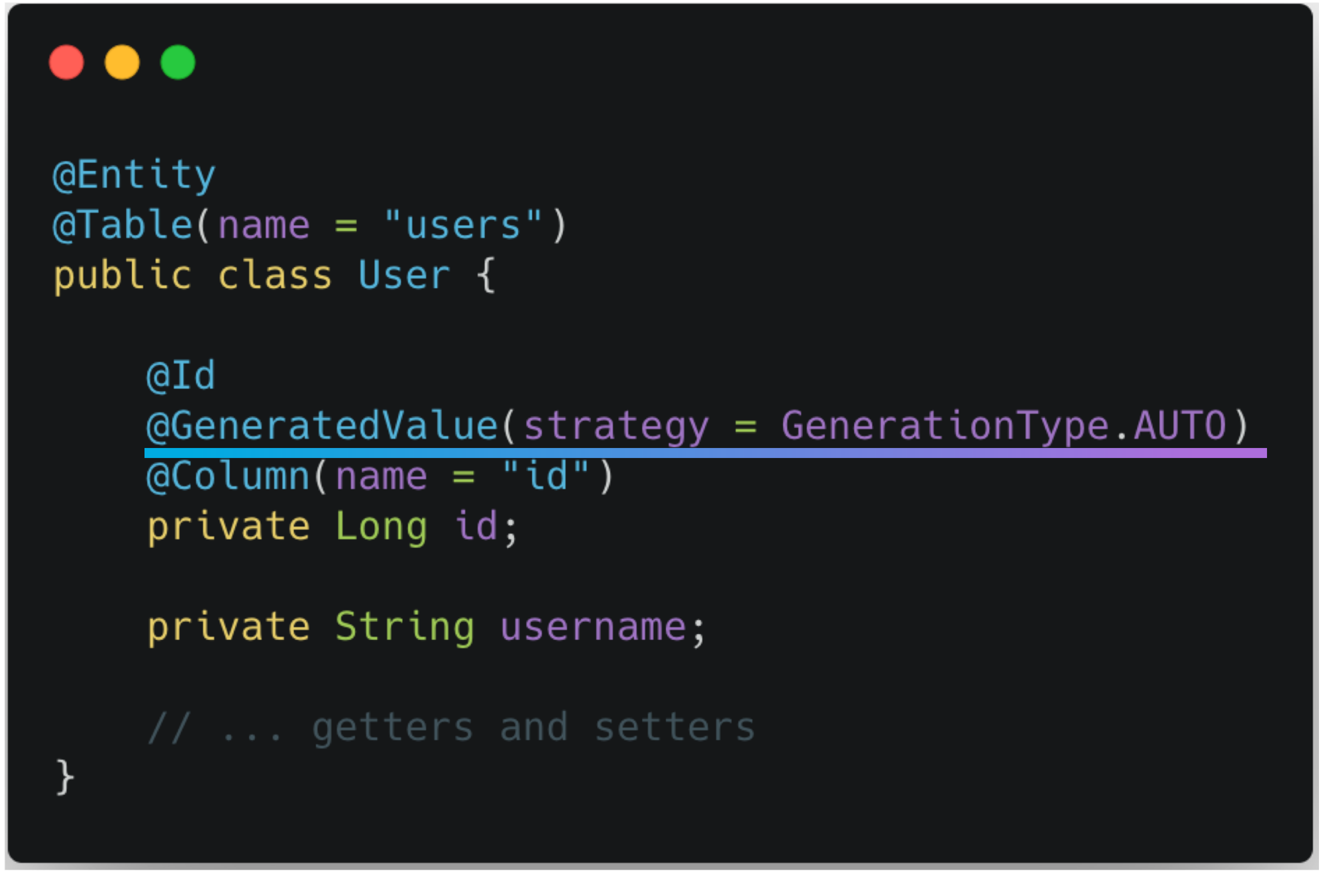
DATABASES ARE THE BOTTLENECK

OF MOST APPLICATIONS

HTTP 1.1 GET /myapp/john/profile

SELECT * **FROM** USERS
WHERE username = “john”

~3ms



```
@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;

    private String username;

    // ... getters and setters
}
```

Couchbase

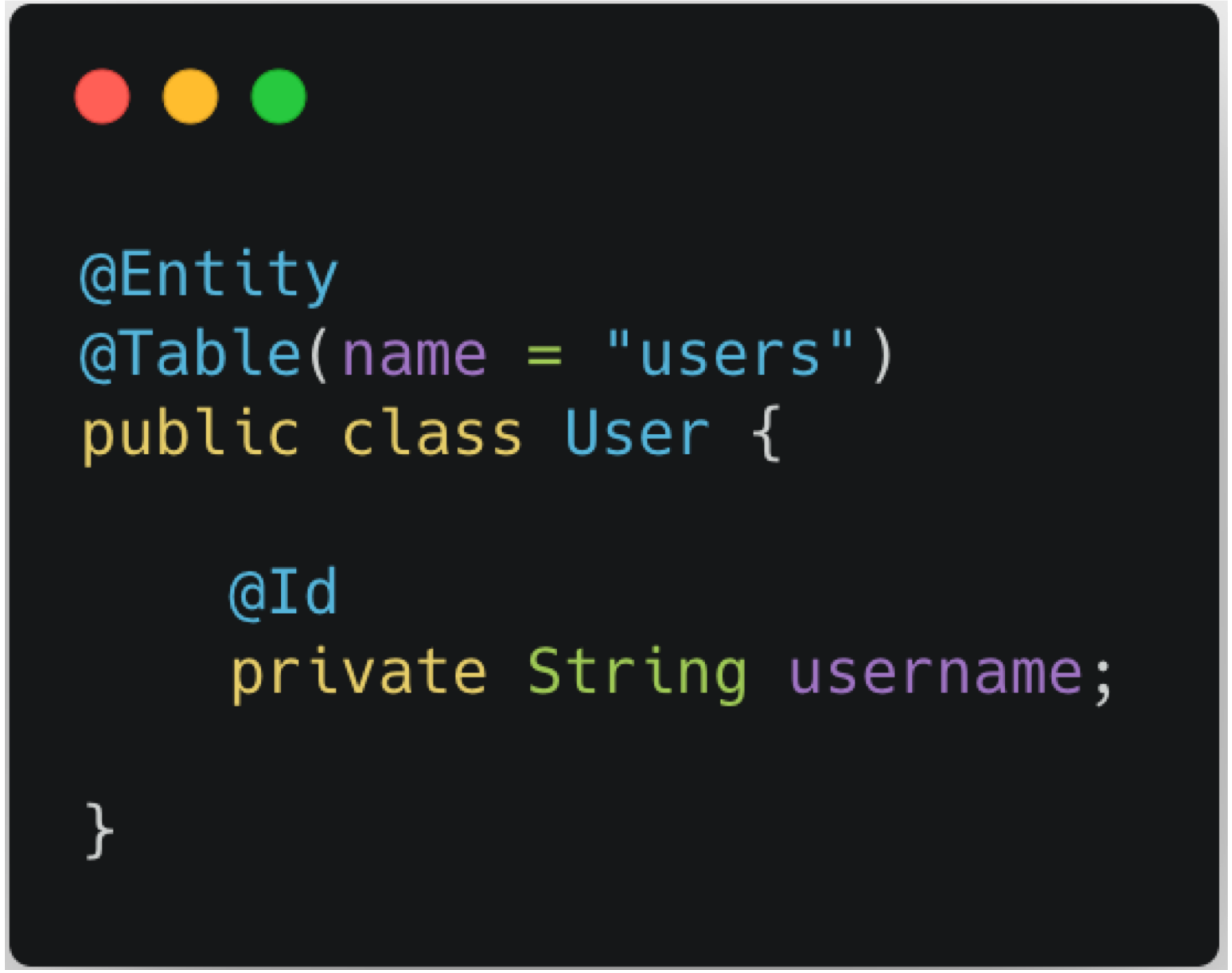
START A REVOLUTION

THE ONLY THING FASTER AND CHEAPER THAN LOADING DATA BY ID

IS NOT LOADING ANY DATA

SELECT * **FROM** USERS
WHERE username = “john”

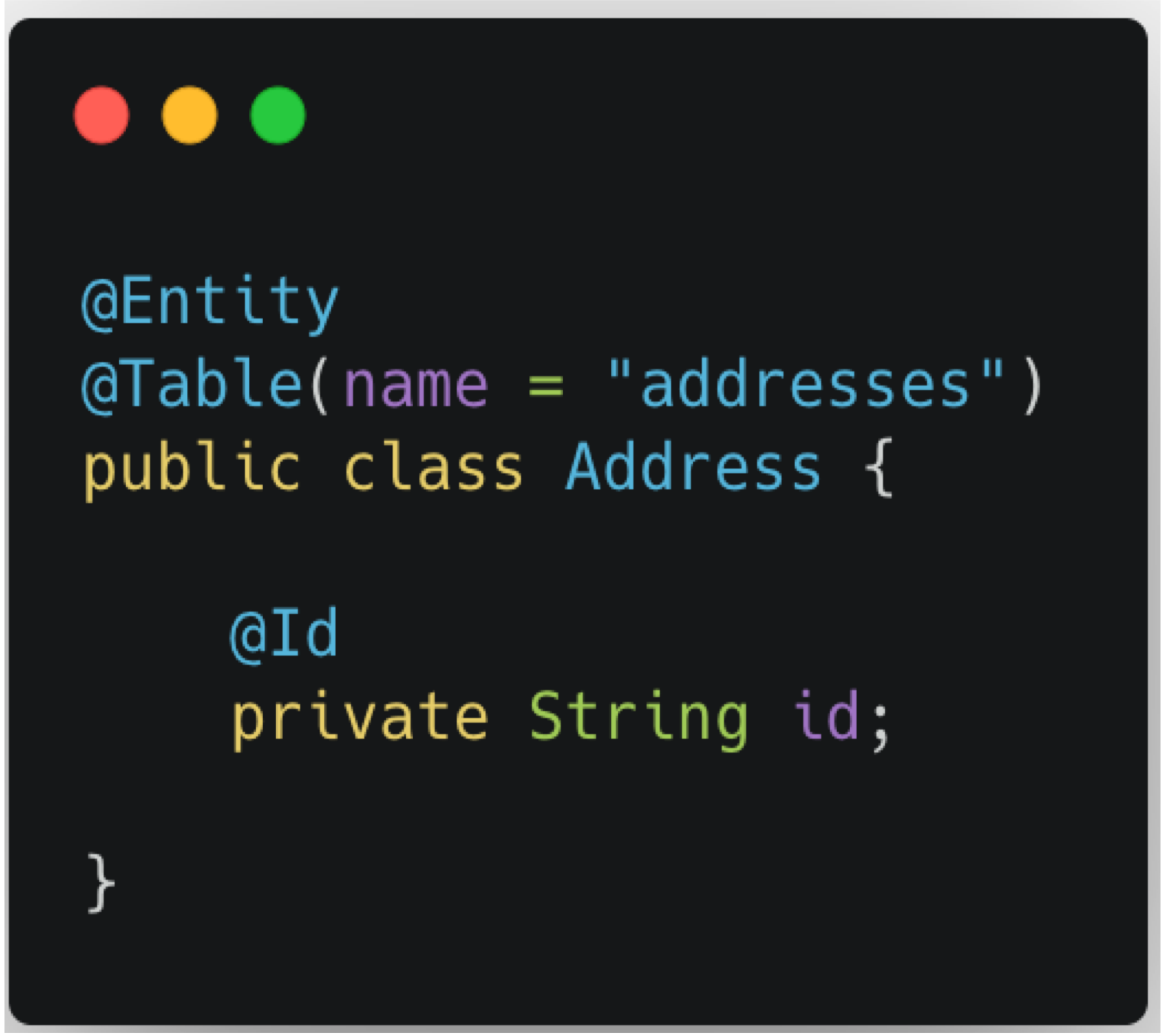
~µs



```
@Entity
@Table(name = "users")
public class User {

    @Id
    private String username;

}
```

SELECT * FROM ADDRESS
WHERE
username = “userid::address”

~µs



```
@Entity
@Table(name = "addresses")
public class Address {

    @Id
    private String id;

}
```

SELECT *

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

You are not getting the most out of indexes

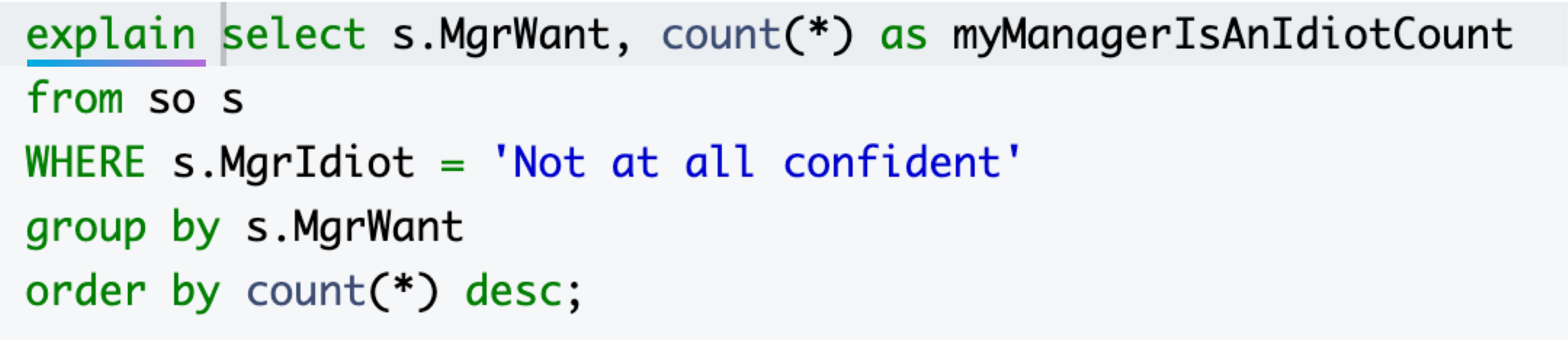
```
SELECT * FROM USERS
    WHERE active = true
```

Cover and Partial Indexes

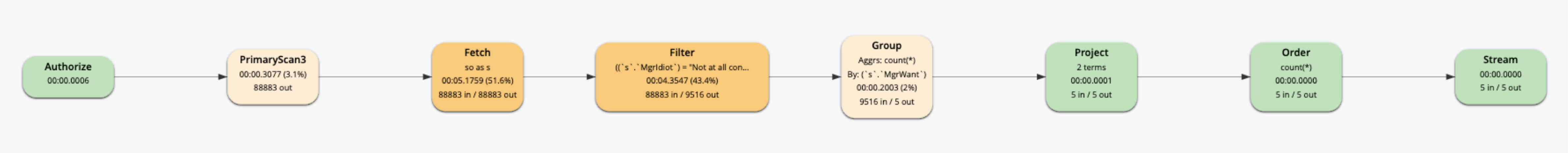
```
CREATE INDEX `user_index`
        ON USERS (username, name, lang)
        WHERE active = true


SELECT username, name, lang  FROM USERS
        WHERE active = true AND pro_user = true
```

Prepared Statements

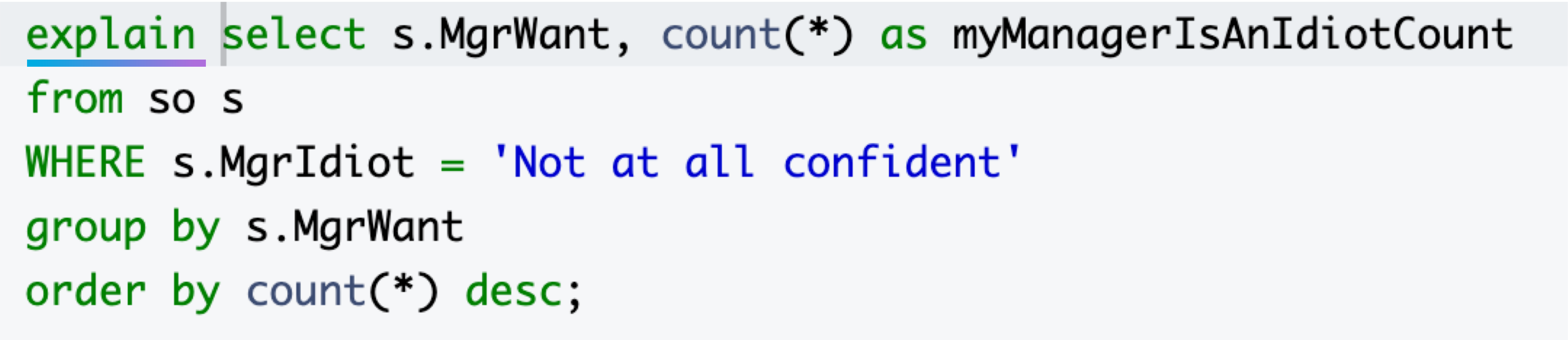
Query Plan



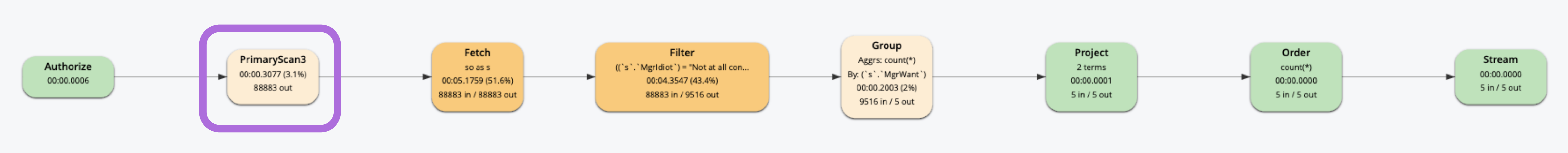
```
explain select s.MgrWant, count(*) as myManagerIsAnIdiotCount
from so s
WHERE s.MgrIdiot = 'Not at all confident'
group by s.MgrWant
order by count(*) desc;
```



Query Plan

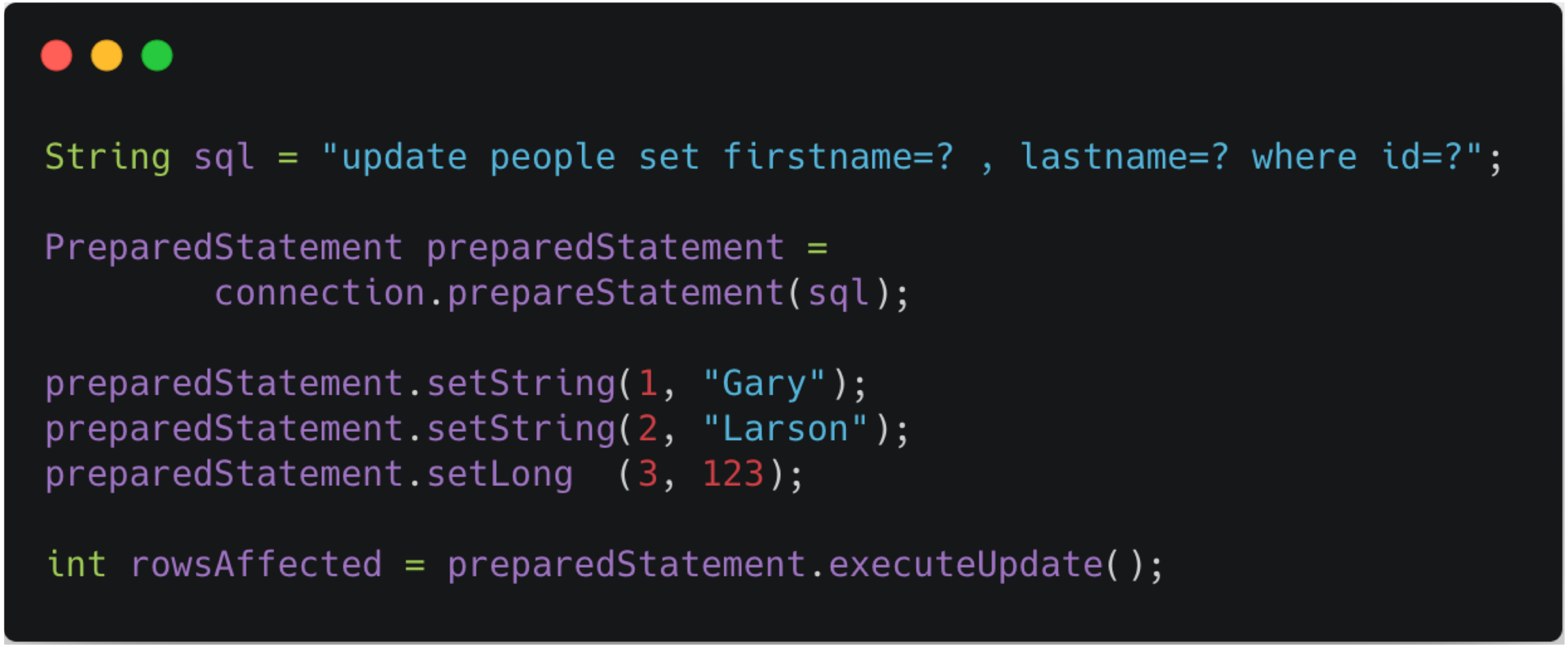


```
explain select s.MgrWant, count(*) as myManagerIsAnIdiotCount
from so s
WHERE s.MgrIdiot = 'Not at all confident'
group by s.MgrWant
order by count(*) desc;
```



~µs

Prepared Statements



```
String sql = "update people set firstname=? , lastname=? where id=?";

PreparedStatement preparedStatement =
        connection.prepareStatement(sql);

preparedStatement.setString(1, "Gary");
preparedStatement.setString(2, "Larson");
preparedStatement.setLong  (3, 123);

int rowsAffected = preparedStatement.executeUpdate();
```

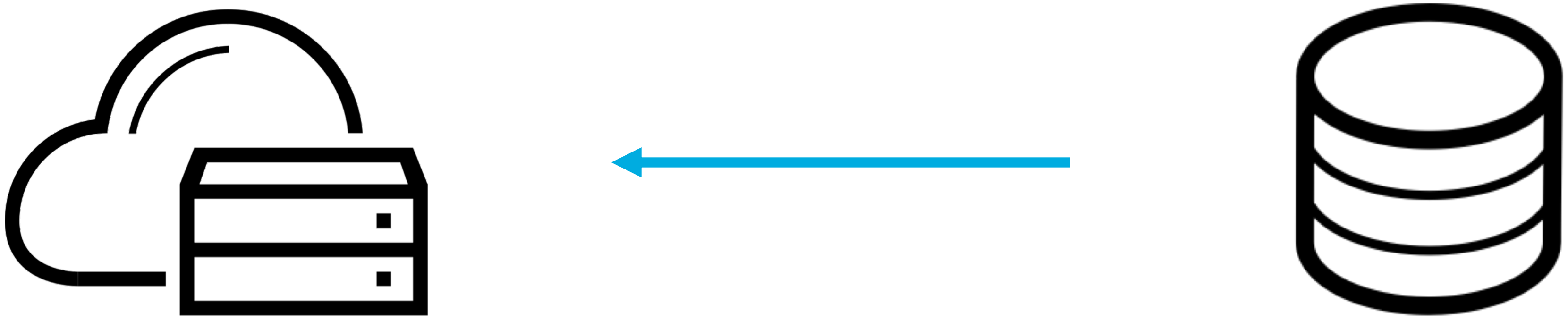
Blocking vs Non-Blocking Calls

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

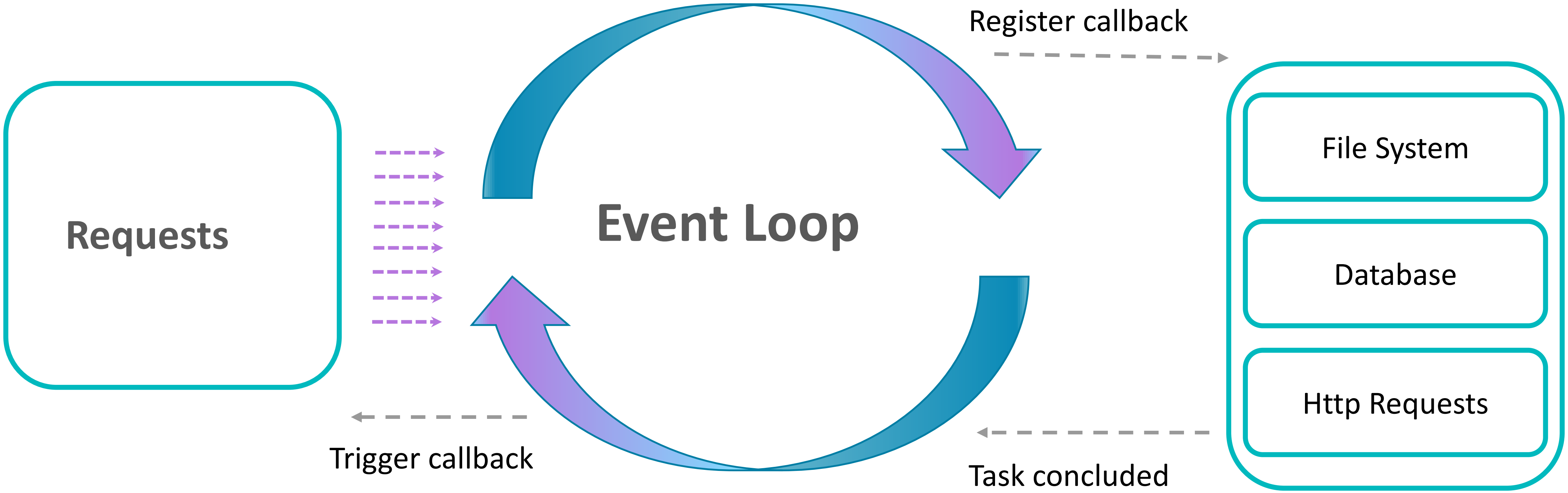
The Blocking approach

The Blocking approach

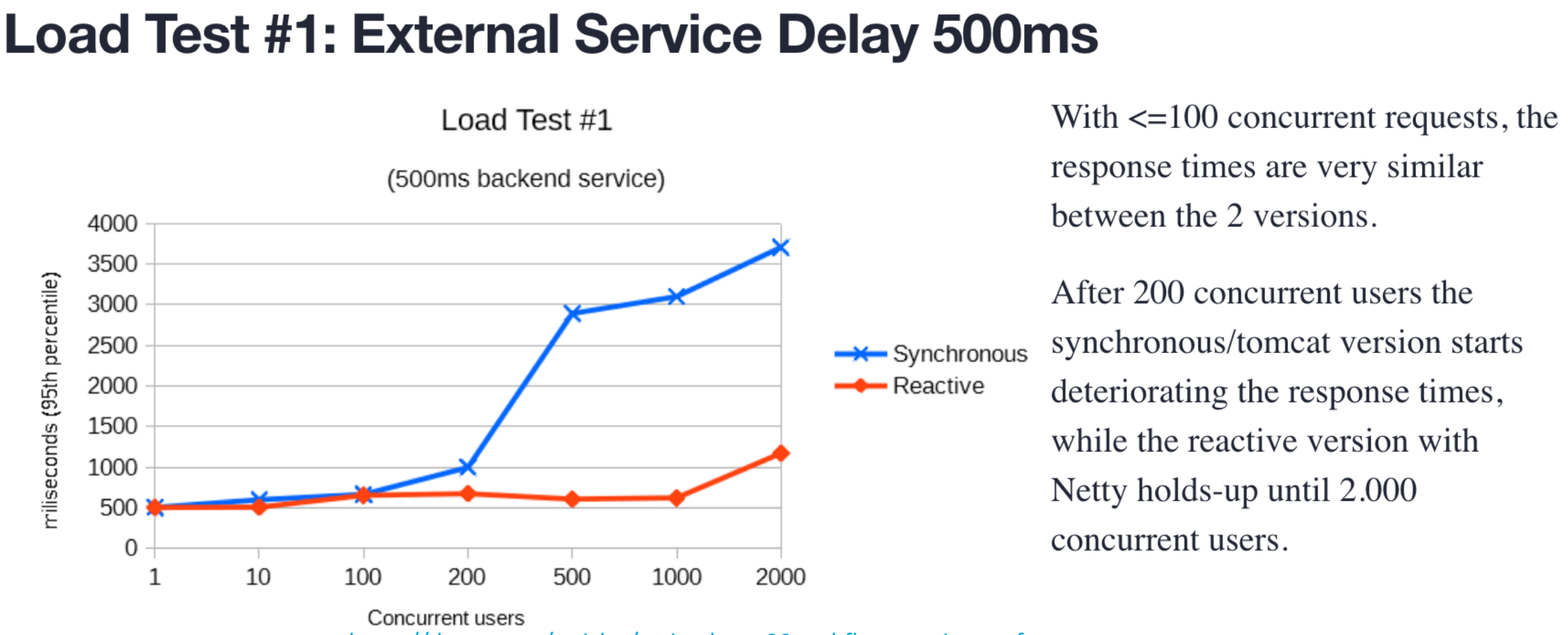
The Blocking approach



The Non-Blocking Approach



Blocking vs Non-Blocking



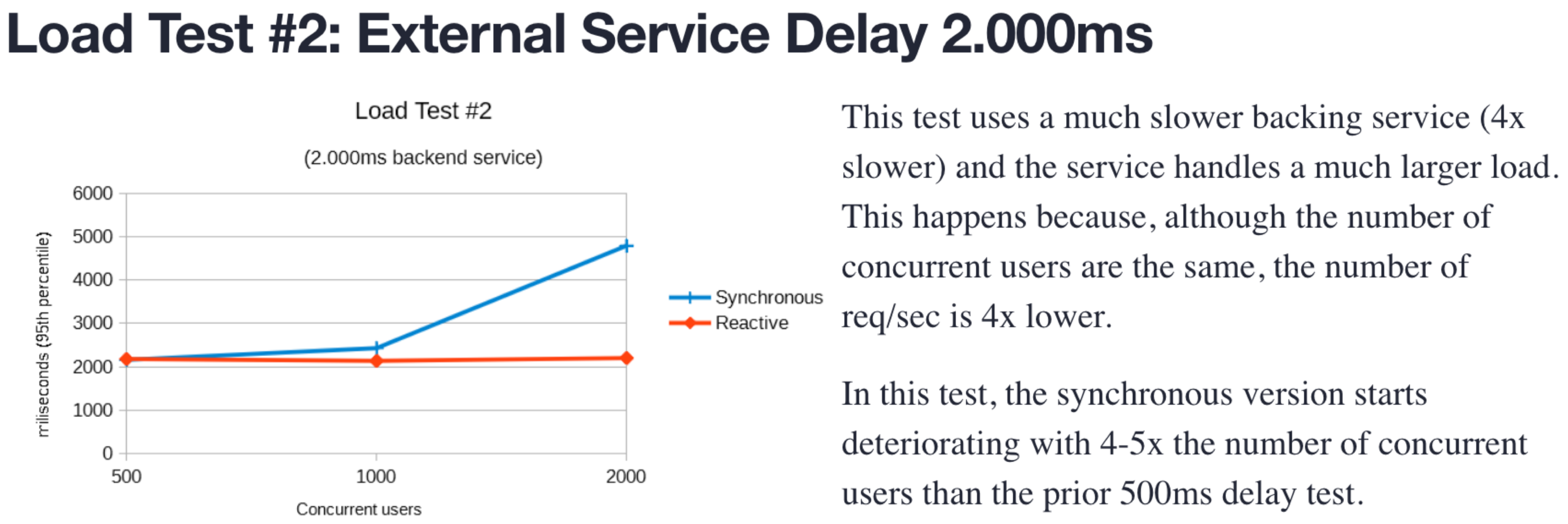
Load Test #1: External Service Delay 500ms

With <=100 concurrent requests, the response times are very similar between the 2 versions.

After 200 concurrent users the synchronous/tomcat version starts deteriorating the response times, while the reactive version with Netty holds-up until 2.000 concurrent users.

https://dzone.com/articles/spring-boot-20-webflux-reactive-performance-test

Blocking vs Non-Blocking



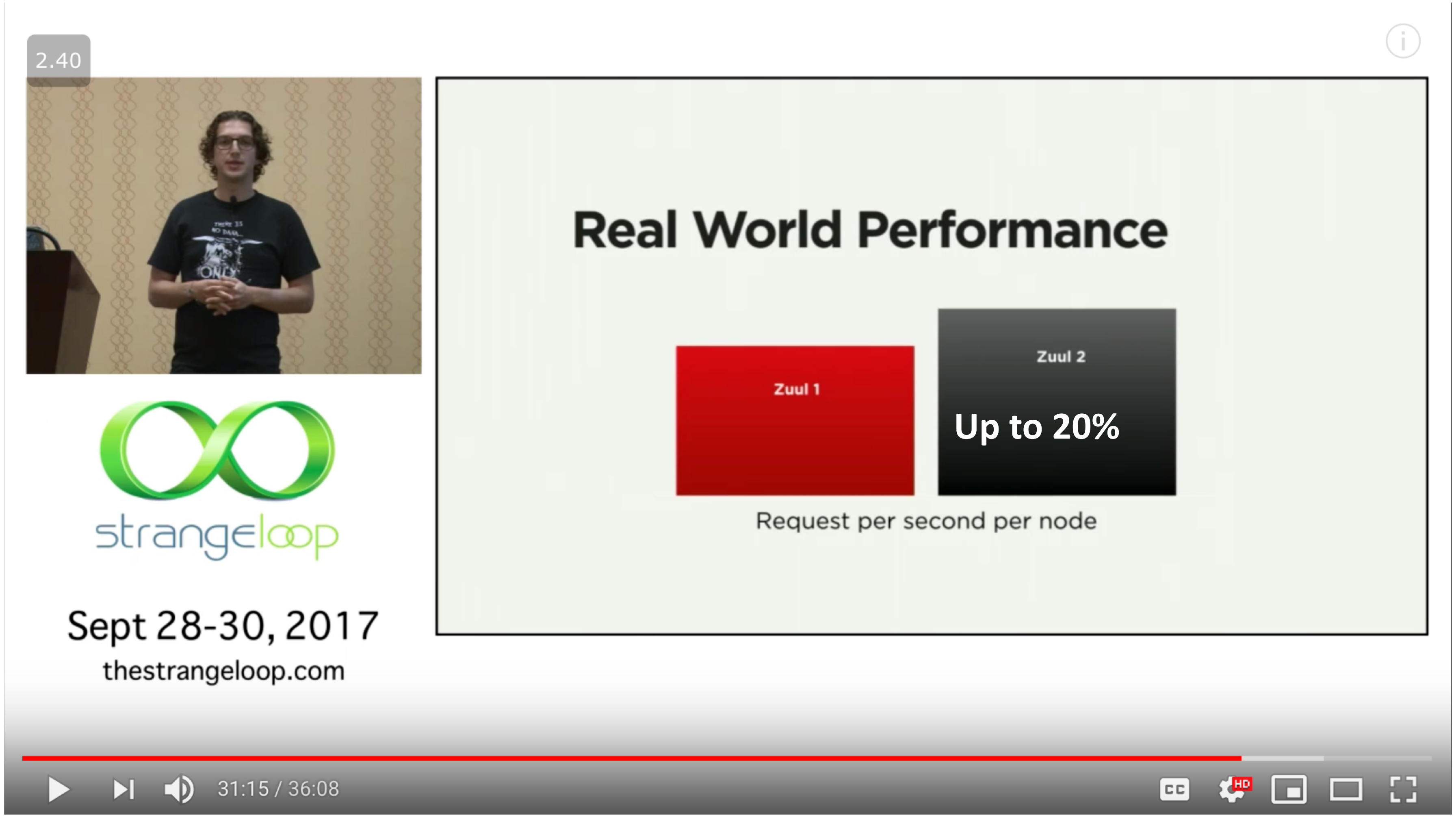
Load Test #2: External Service Delay 2.000ms

This test uses a much slower backing service (4x slower) and the service handles a much larger load. This happens because, although the number of concurrent users are the same, the number of req/sec is 4x lower.

In this test, the synchronous version starts deteriorating with 4-5x the number of concurrent users than the prior 500ms delay test.

https://dzone.com/articles/spring-boot-20-webflux-reactive-performance-test

Zuul’s Journey to Non-Blocking



Real World Performance

Up to 20%

Request per second per node

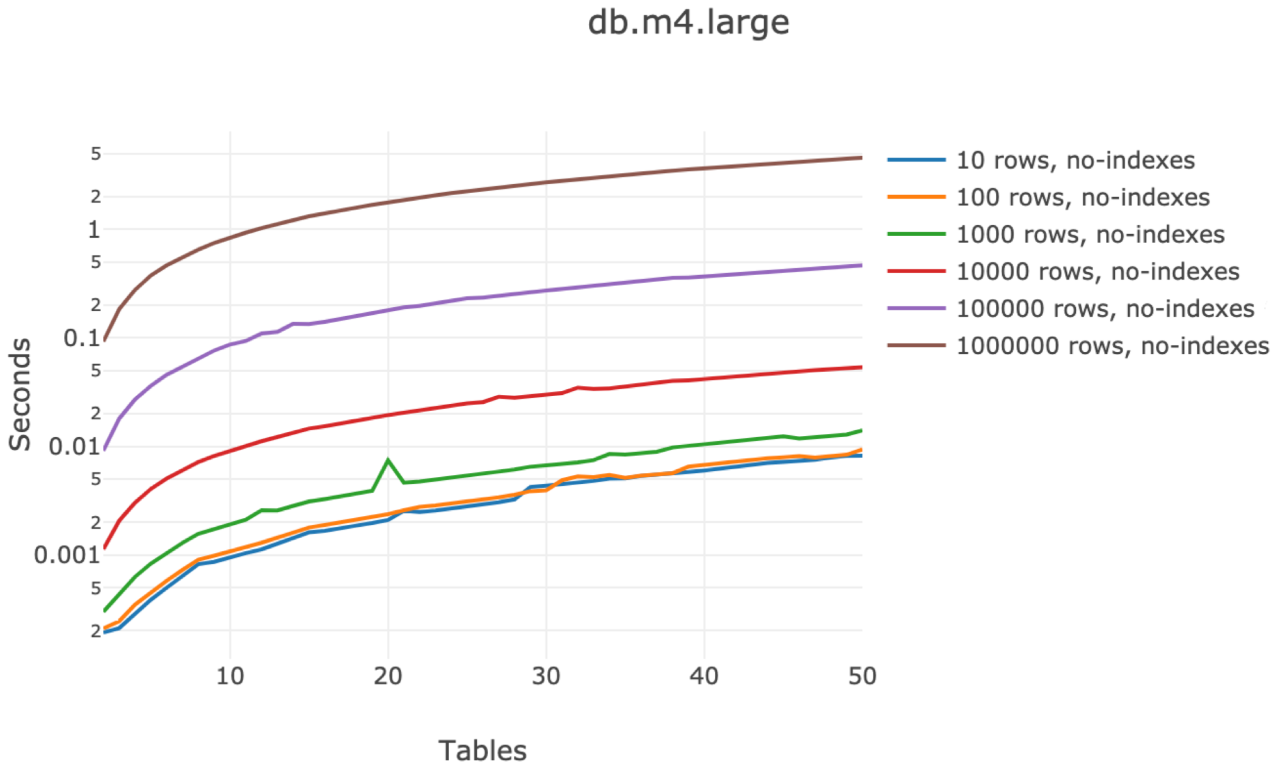
Sept 28-30, 2017
thestrangeloop.com

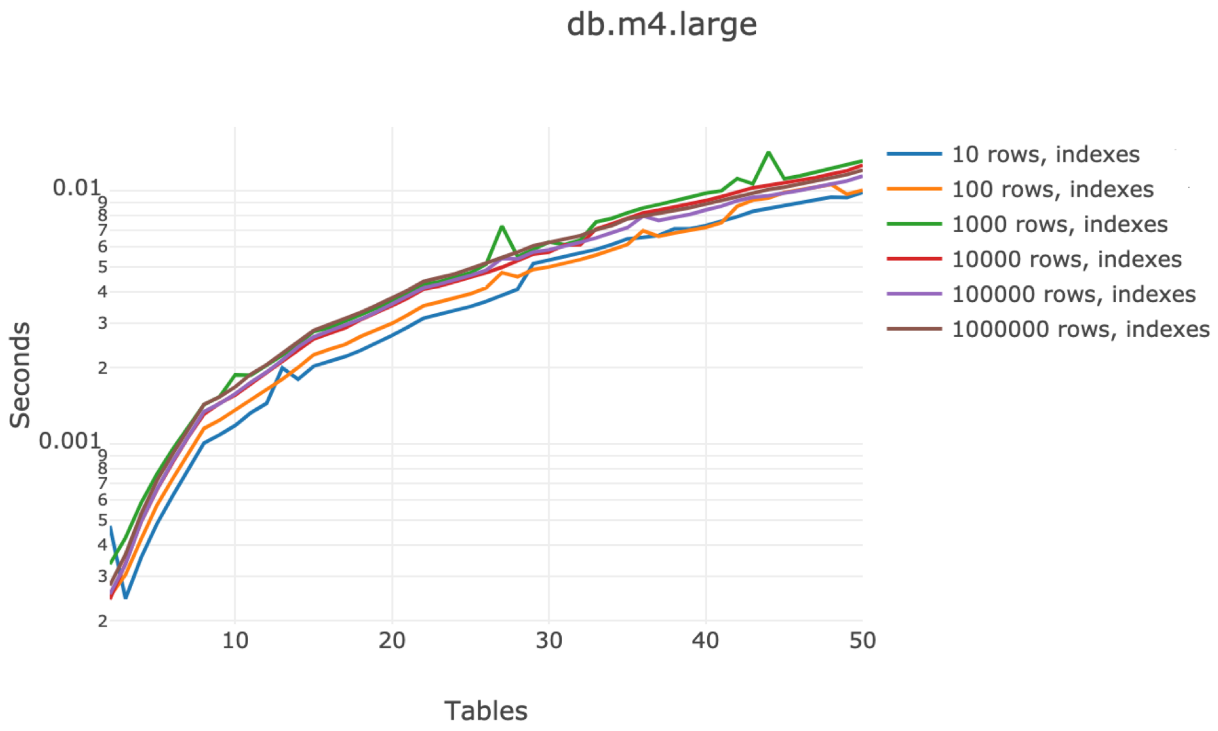
https://www.youtube.com/watch?v=2oXqbLhMS_A

JOINS

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

JOINS





https://www.brianlikespostgres.com/cost-of-a-join.html

JOINS

N of Joins	Avg Time (ms)*
2 to 10	0.176
10 to 20	0.209
20 to 30	0.246
30 to 40	0.263
40 to 50	0.322

* Tables with 1M rows each

JOINS

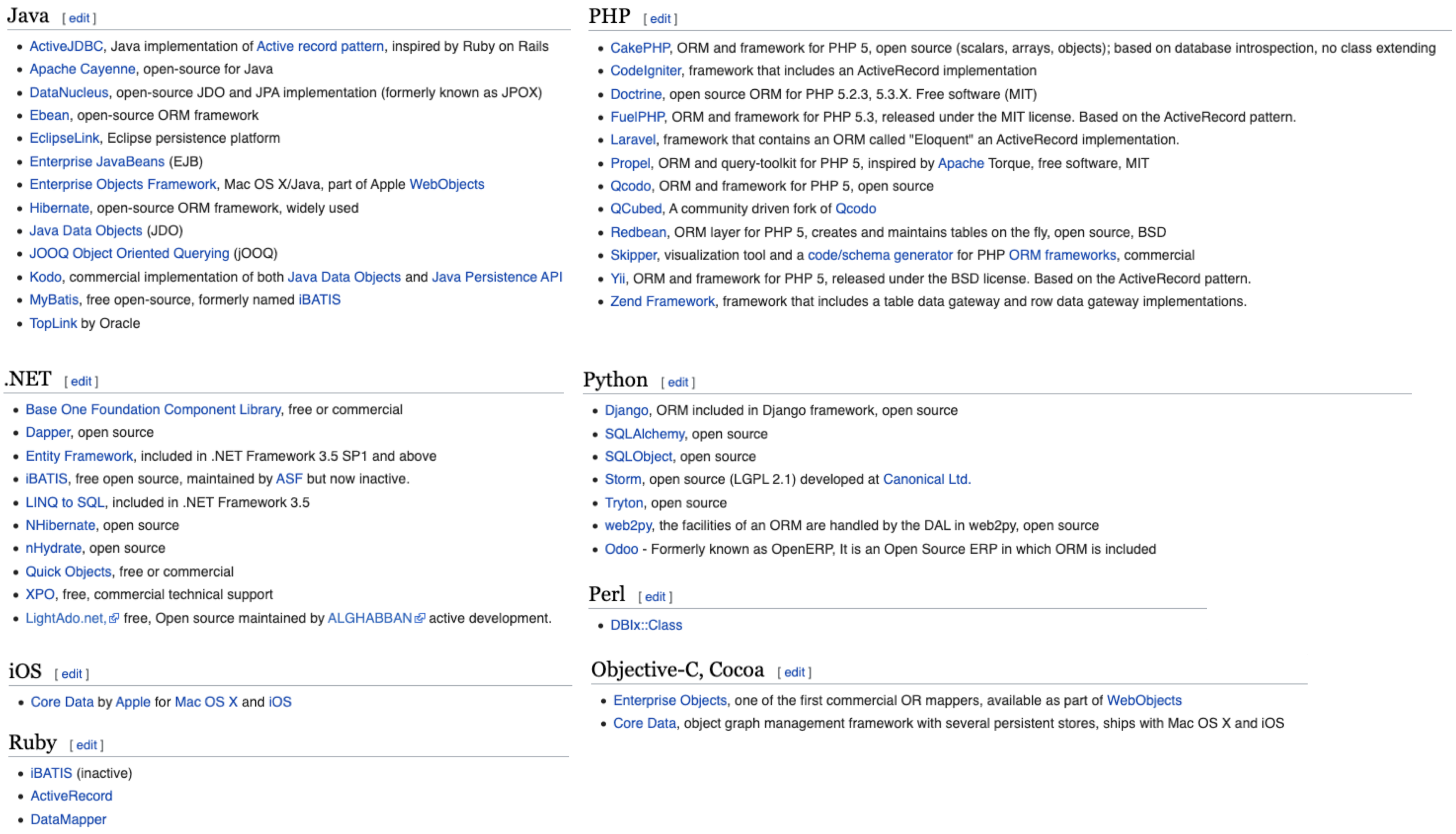
Apart from special scenarios, **with a correctly set up**, Joins are cheap and denormalization offers no benefits for RDBMS.

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

I want to be productive!

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

ORMs



Java [edit]

- ActiveJDBC, Java implementation of Active record pattern, inspired by Ruby on Rails
- Apache Cayenne, open-source for Java
- DataNucleus, open-source JDO and JPA implementation (formerly known as JPOX)
- Ebean, open-source ORM framework
- EclipseLink, Eclipse persistence platform
- Enterprise JavaBeans (EJB)
- Enterprise Objects Framework, Mac OS X/Java, part of Apple WebObjects
- Hibernate, open-source ORM framework, widely used
- Java Data Objects (JDO)
- JOOQ Object Oriented Querying (jOOQ)
- Kodo, commercial implementation of both Java Data Objects and Java Persistence API
- MyBatis, free open-source, formerly named iBATIS
- TopLink by Oracle

.NET [edit]

- Base One Foundation Component Library, free or commercial
- Dapper, open source
- Entity Framework, included in .NET Framework 3.5 SP1 and above
- iBATIS, free open source, maintained by ASF but now inactive.
- LINQ to SQL, included in .NET Framework 3.5
- NHibernate, open source
- nHydrate, open source
- Quick Objects, free or commercial
- XPO, free, commercial technical support
- LightAdo.net, free, Open source maintained by ALGHABBAN active development.

iOS [edit]

- Core Data by Apple for Mac OS X and iOS

Ruby [edit]

- iBATIS (inactive)
- ActiveRecord
- DataMapper

PHP [edit]

- CakePHP, ORM and framework for PHP 5, open source (scalars, arrays, objects); based on database introspection, no class extending
- CodeIgniter, framework that includes an ActiveRecord implementation
- Doctrine, open source ORM for PHP 5.2.3, 5.3.X. Free software (MIT)
- FuelPHP, ORM and framework for PHP 5.3, released under the MIT license. Based on the ActiveRecord pattern.
- Laravel, framework that contains an ORM called "Eloquent" an ActiveRecord implementation.
- Propel, ORM and query-toolkit for PHP 5, inspired by Apache Torque, free software, MIT
- Qcodo, ORM and framework for PHP 5, open source
- QCubed, A community driven fork of Qcodo
- Redbean, ORM layer for PHP 5, creates and maintains tables on the fly, open source, BSD
- Skipper, visualization tool and a code/schema generator for PHP ORM frameworks, commercial
- Yii, ORM and framework for PHP 5, released under the BSD license. Based on the ActiveRecord pattern.
- Zend Framework, framework that includes a table data gateway and row data gateway implementations.

Python [edit]

- Django, ORM included in Django framework, open source
- SQLAlchemy, open source
- SQLObject, open source
- Storm, open source (LGPL 2.1) developed at Canonical Ltd.
- Tryton, open source
- web2py, the facilities of an ORM are handled by the DAL in web2py, open source
- Odoo - Formerly known as OpenERP, It is an Open Source ERP in which ORM is included

Perl [edit]

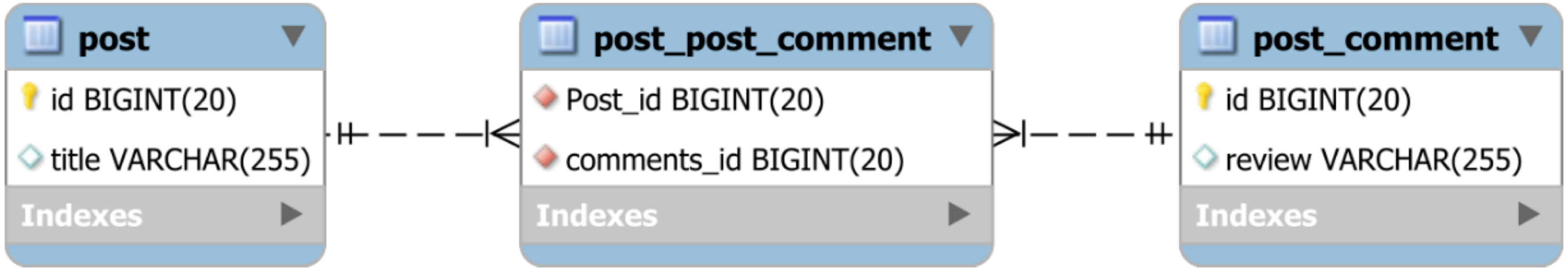
- DBIx::Class

Objective-C, Cocoa [edit]

- Enterprise Objects, one of the first commercial OR mappers, available as part of WebObjects
- Core Data, object graph management framework with several persistent stores, ships with Mac OS X and iOS

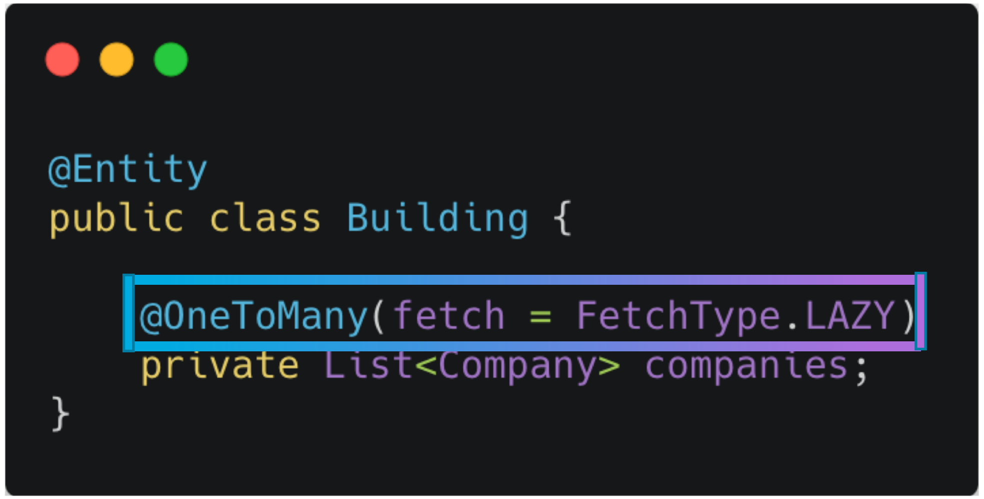
ORMs

Wrong mappings can generate **unnecessary Junction Tables**, and eager relationships can result in **unnecessary data** being loaded.

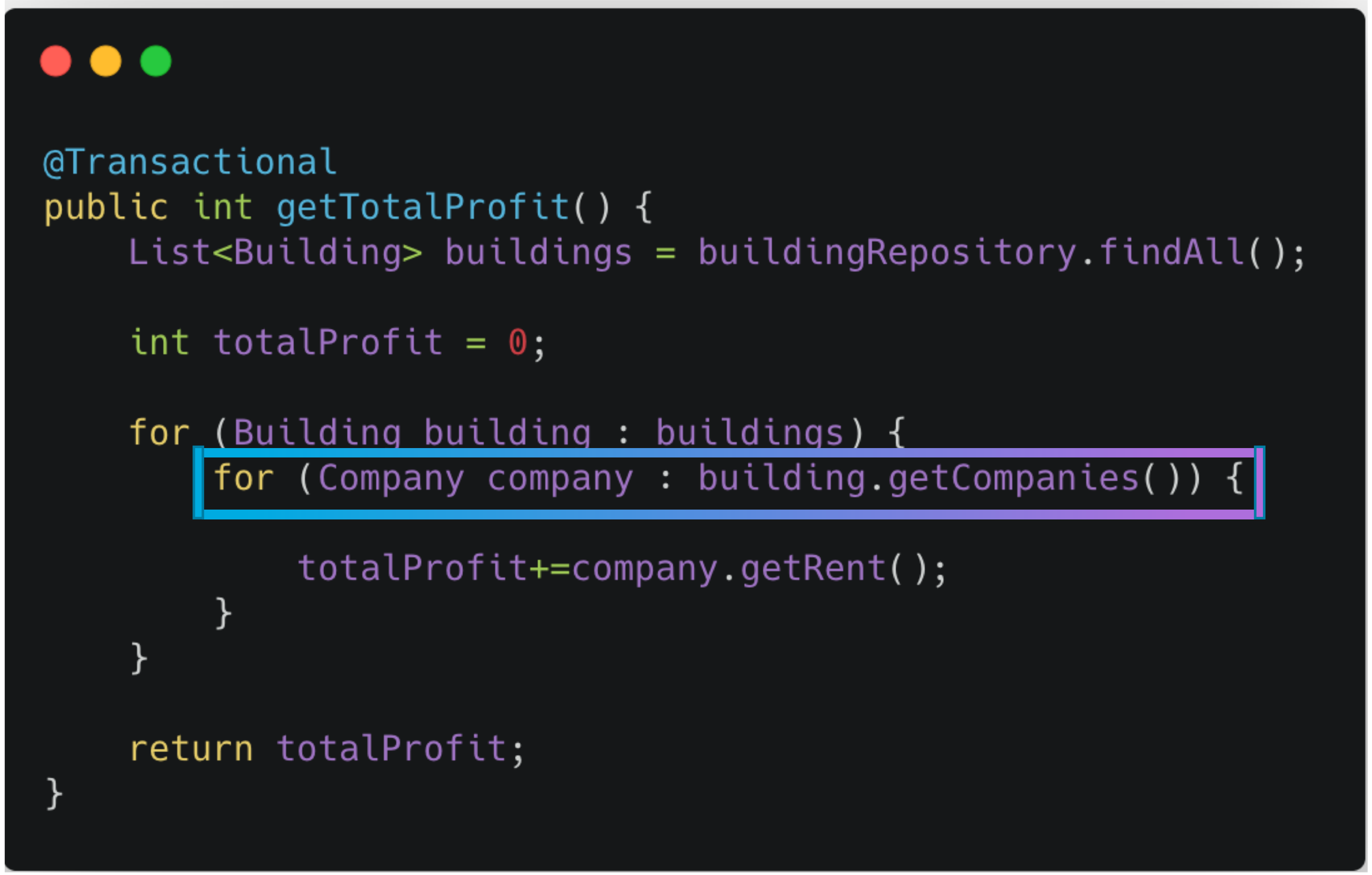


Source: High Performance Java Persistence, pg 195

N+1 Problem

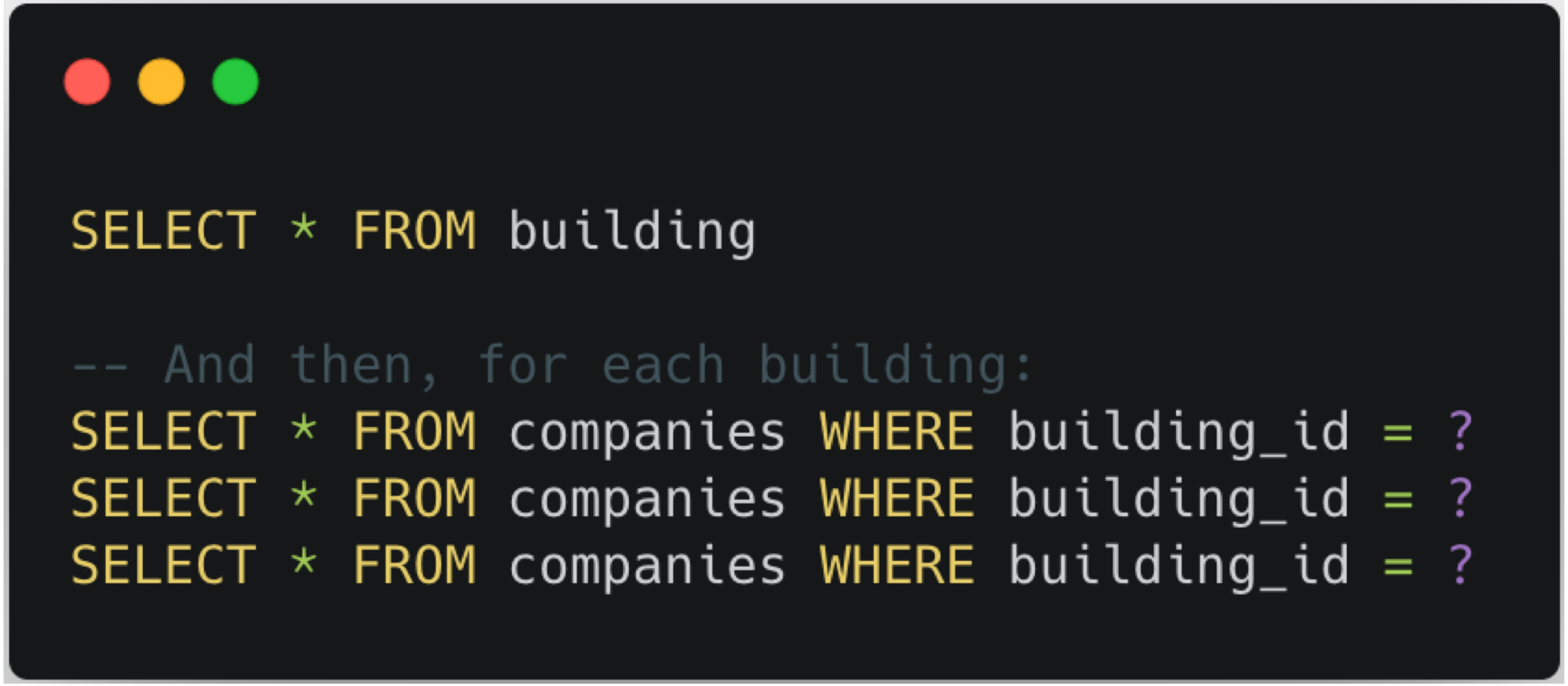


```
@Entity
public class Building {

    @OneToMany(fetch = FetchType.LAZY)
    private List<Company> companies;
}
```

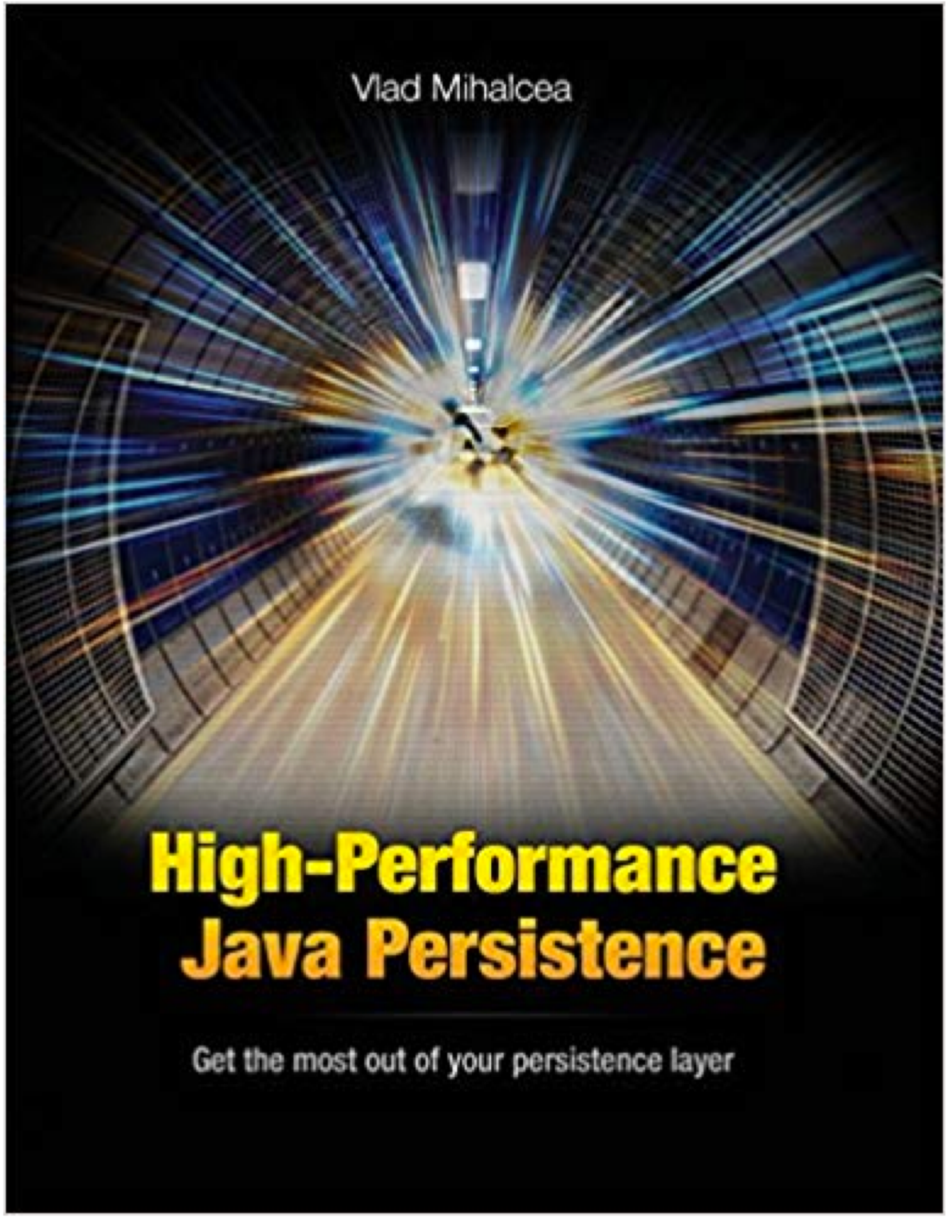


```
@Transactional
public int getTotalProfit() {
    List<Building> buildings = buildingRepository.findAll();

    int totalProfit = 0;

    for (Building building : buildings) {
        for (Company company : building.getCompanies()) {

            totalProfit+=company.getRent();
        }
    }

    return totalProfit;
}
```

N+1 Problem



```
SELECT * FROM building

-- And then, for each building:
SELECT * FROM companies WHERE building_id = ?
SELECT * FROM companies WHERE building_id = ?
SELECT * FROM companies WHERE building_id = ?
```



Vlad Mihalcea
High-Performance
Java Persistence
Get the most out of your persistence layer

WAIT A MINUTE…

Impedance Mismatch

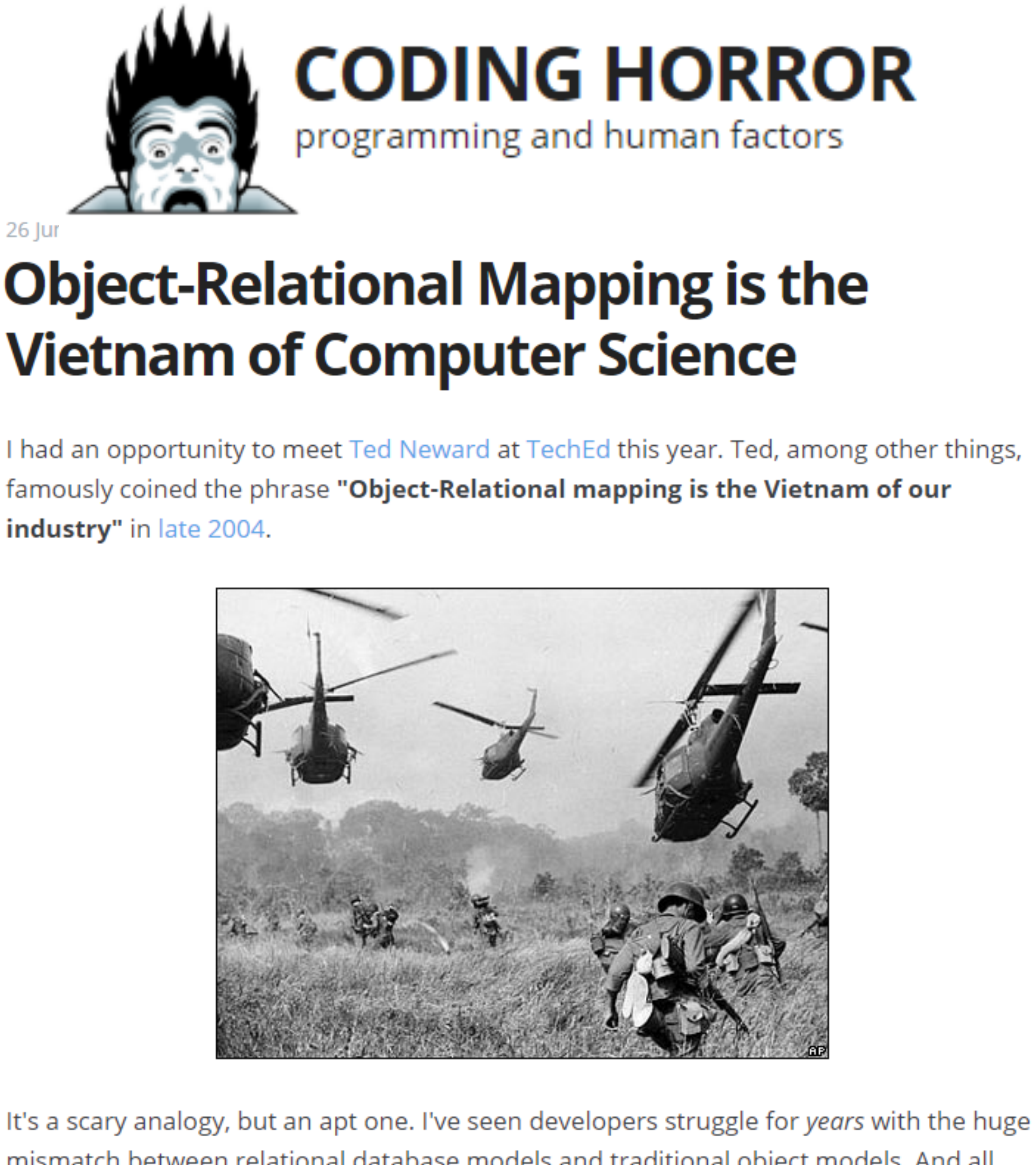
ShoppingCart

ID	Username	DateCreated
1	deniswsrosa	2019-06-13
2	mgroves	2019-06-14
.	.	.
.	.	.

ShoppingCartItems

CartID	Item	Price	Qty
1	hat	12.99	1
1	socks	11.99	1
2	t-shirt	15.99	1
.	.	.	.
.	.	.	.

```
public class ShoppingCart {
    int id;
    String username;
    List<Items> items;
}
```



CODING HORROR

programming and human factors

26 Jur

Object-Relational Mapping is the Vietnam of Computer Science

I had an opportunity to meet Ted Neward at TechEd this year. Ted, among other things, famously coined the phrase **"Object-Relational mapping is the Vietnam of our industry"** in late 2004.

It's a scary analogy, but an apt one. I've seen developers struggle for *years* with the huge mismatch between relational database models and traditional object models. And all

Hierarchical Data

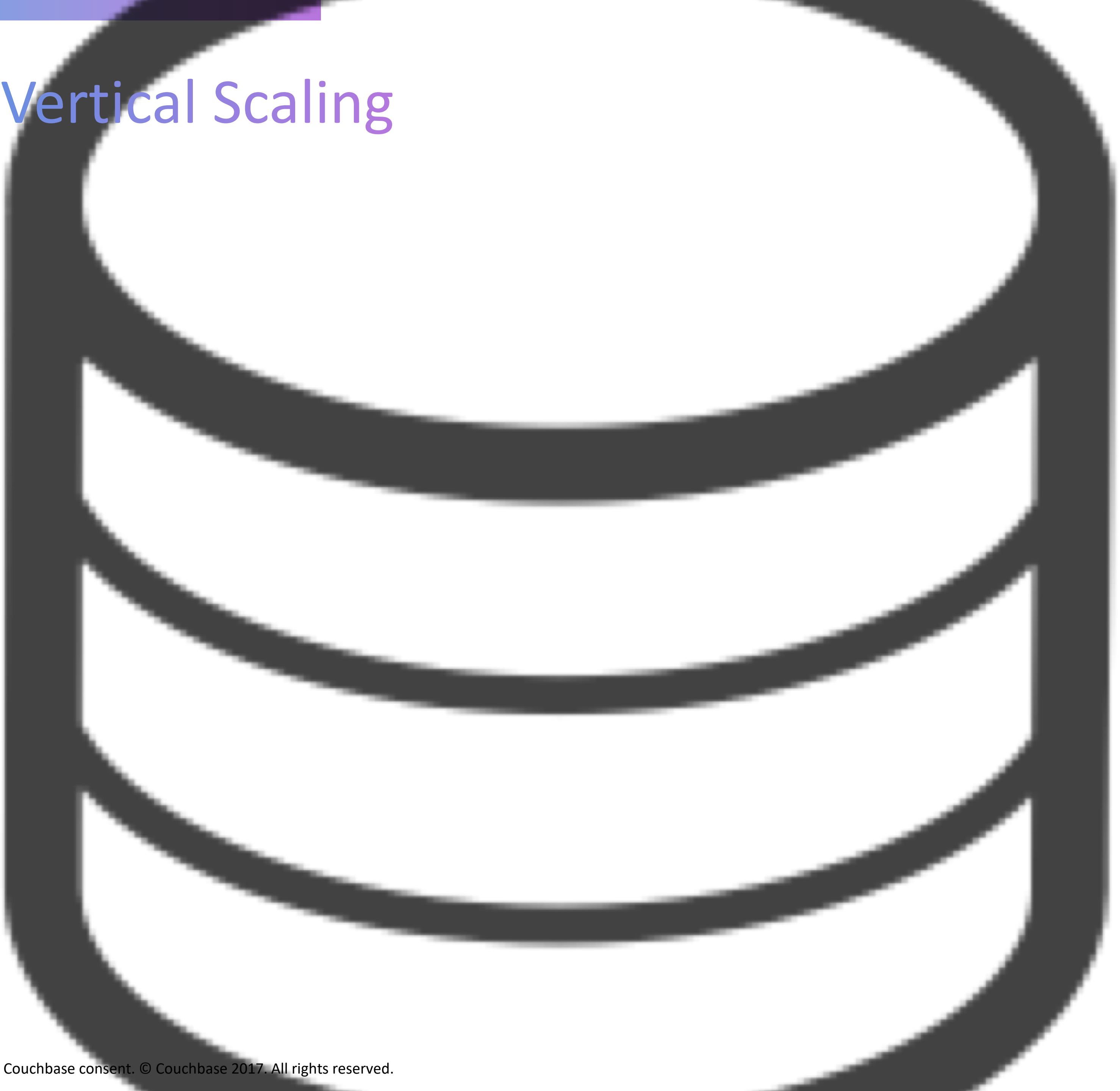
JSON | XML | YAML | …

Confidential and Proprietary. Do not distribute without Couchbase consent. © Couchbase 2017. All rights reserved.

RDBMS - Vertical Scaling

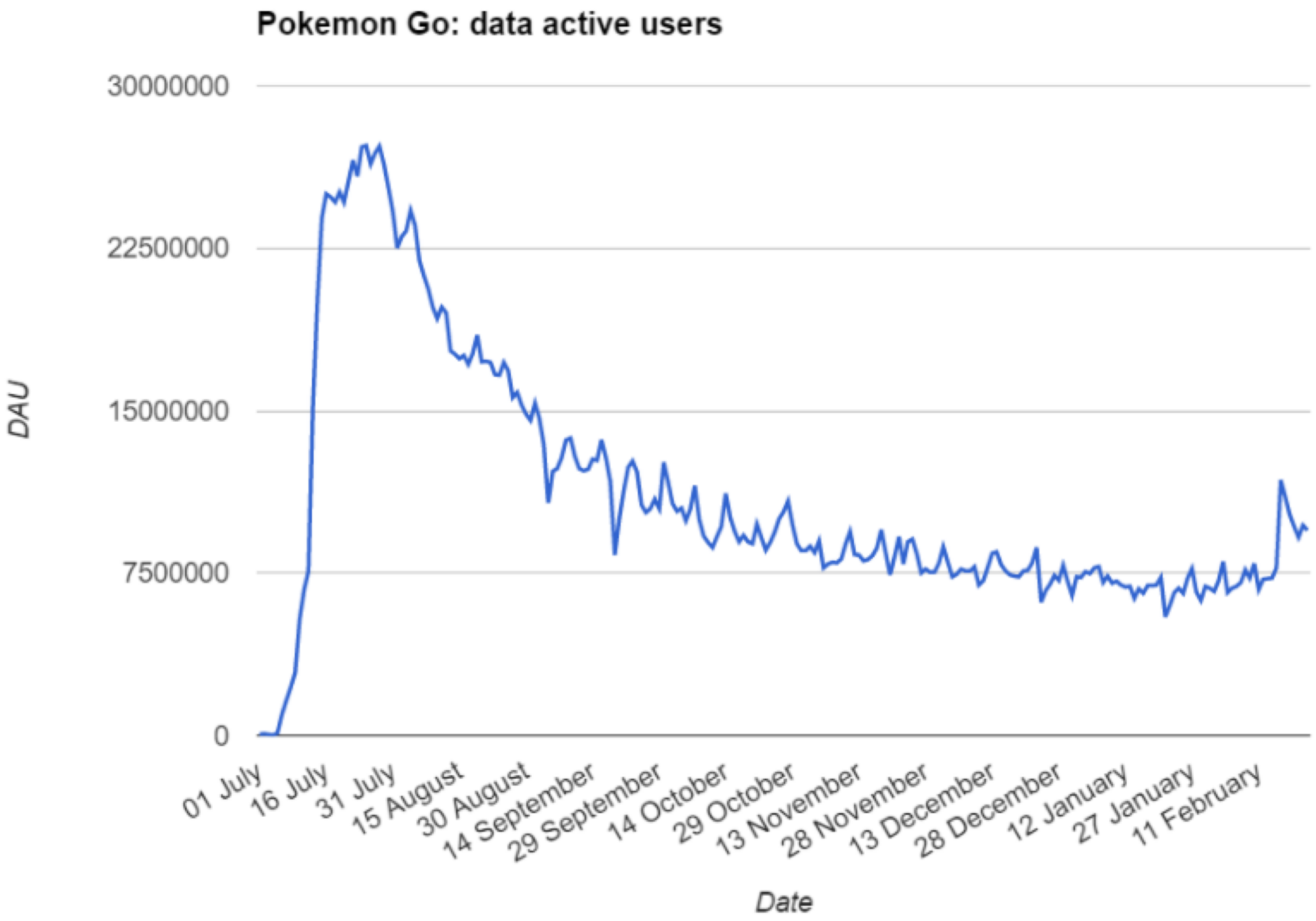
RDBMS - Vertical Scaling

RDBMS - Vertical Scaling



Data Growth

POKéMON GO
™


Pokemon Go: data active users
DAU
30000000
22500000
15000000
7500000
0
01 July
16 July
31 July
15 August
30 August
14 September
29 September
14 October
29 October
13 November
28 November
13 December
28 December
12 January
27 January
11 February
Date

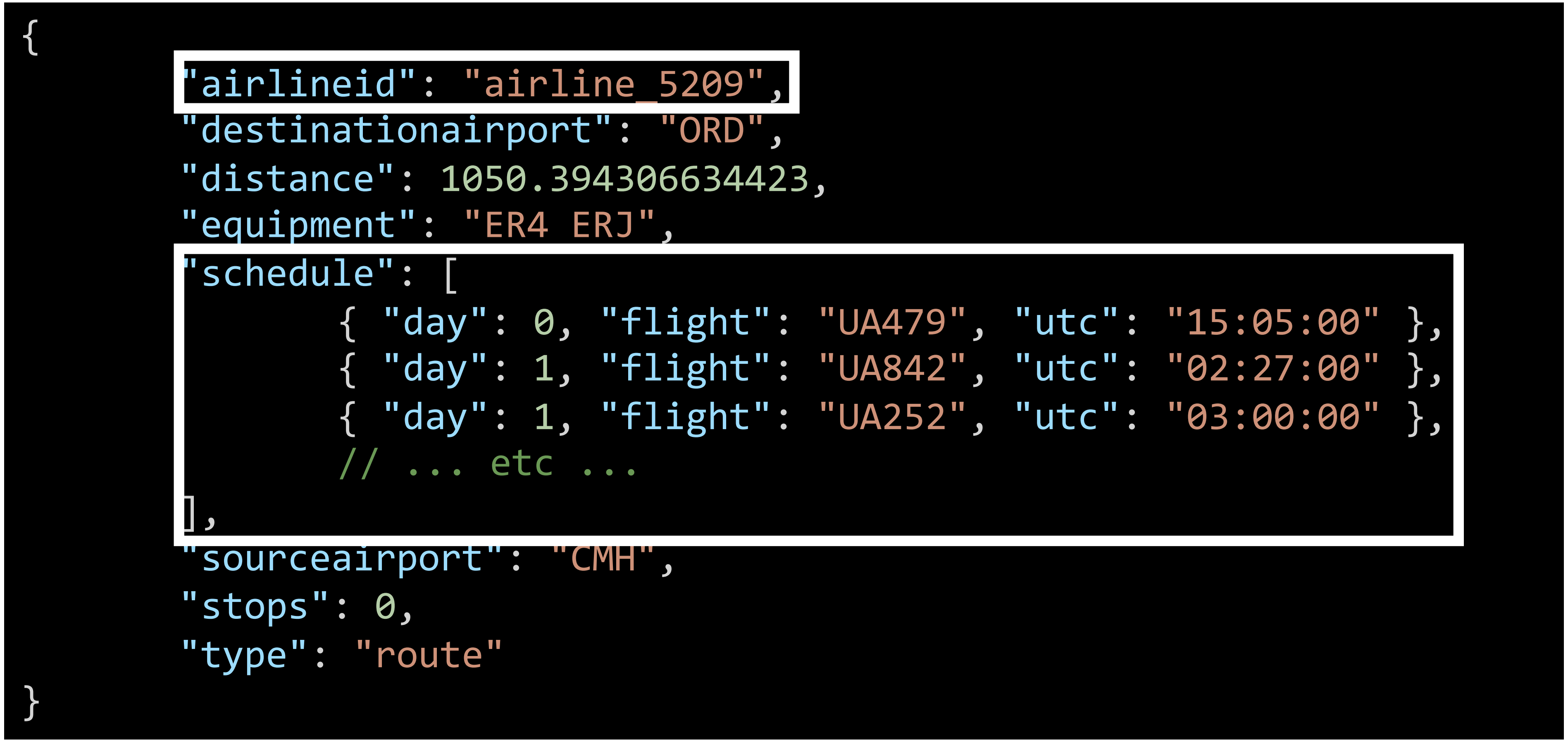
JSON

document key: **route_55758**

```
{
        "airlineid": "airline_5209",
        "destinationairport": "ORD",
        "distance": 1050.394306634423,
        "equipment": "ER4 ERJ",
        "schedule": [
                { "day": 0, "flight": "UA479", "utc": "15:05:00" },
                { "day": 1, "flight": "UA842", "utc": "02:27:00" },
                { "day": 1, "flight": "UA252", "utc": "03:00:00" },
                // ... etc ...
        ],
        "sourceairport": "CMH",
        "stops": 0,
        "type": "route"
}
```

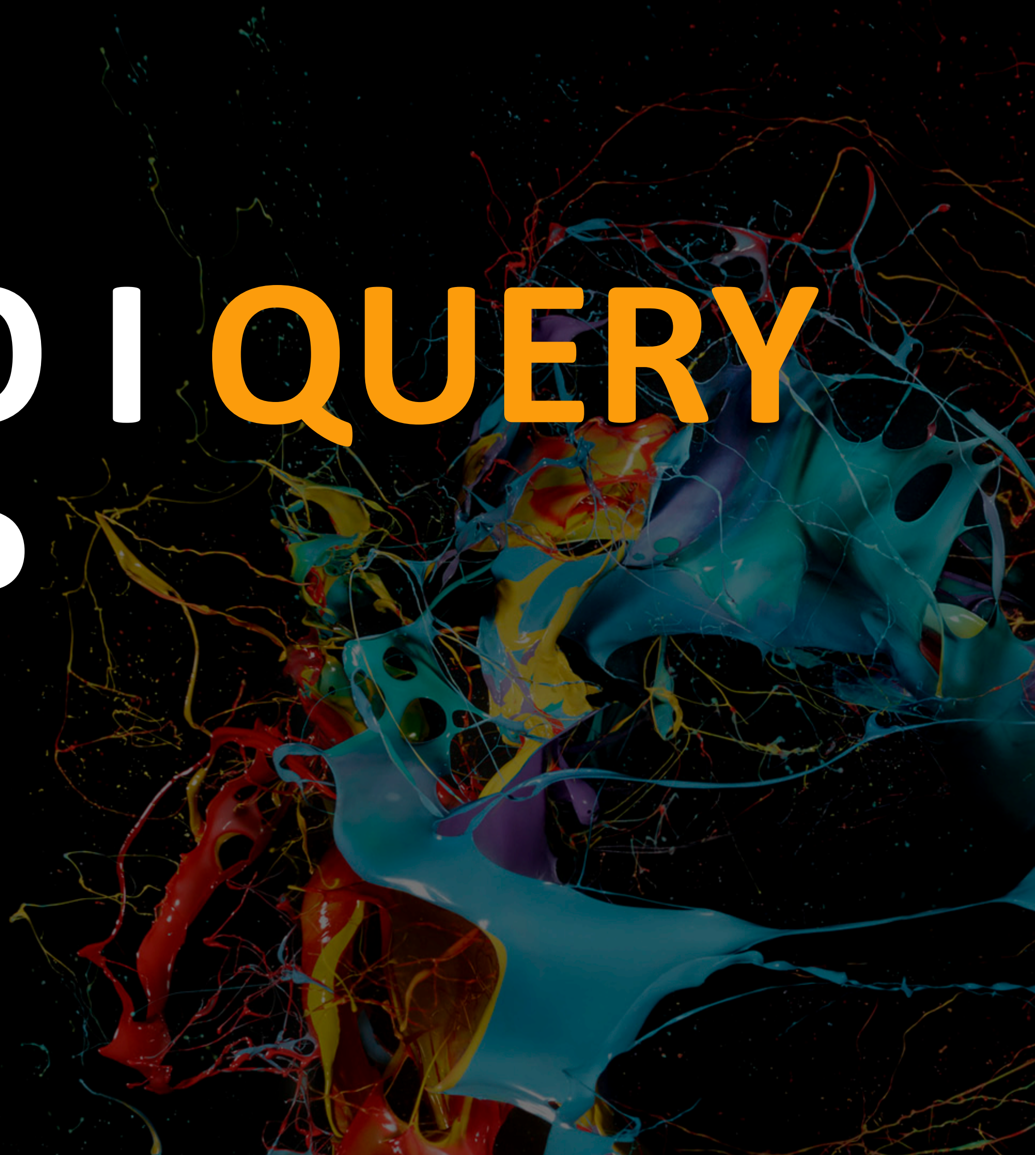
JSON

document key: **route_55758**



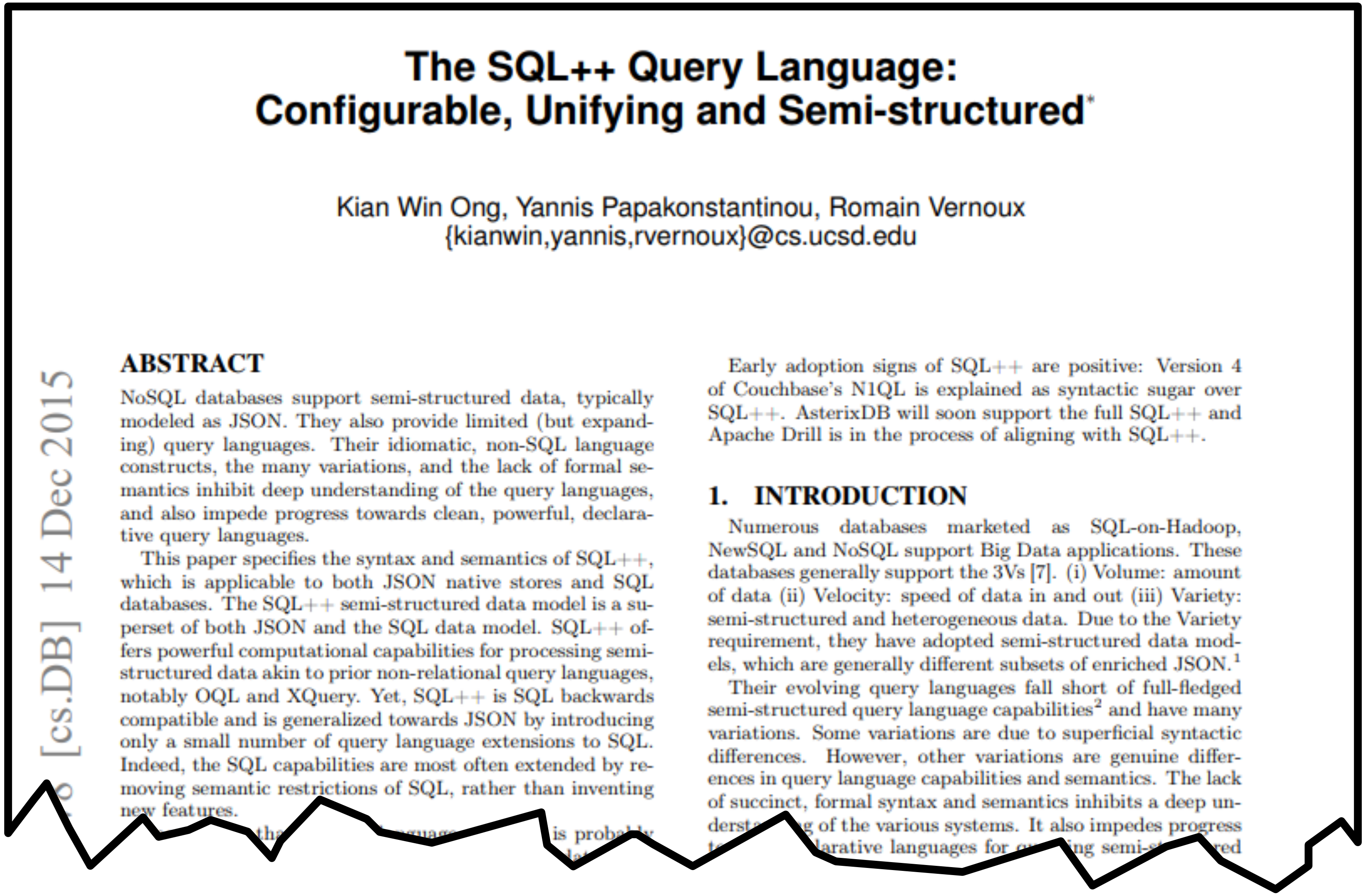
```
{
        "airlineid": "airline_5209",
        "destinationairport": "ORD",
        "distance": 1050.394306634423,
        "equipment": "ER4 ERJ",
        "schedule": [
                { "day": 0, "flight": "UA479", "utc": "15:05:00" },
                { "day": 1, "flight": "UA842", "utc": "02:27:00" },
                { "day": 1, "flight": "UA252", "utc": "03:00:00" },
                // ... etc ...
        ],
        "sourceairport": "CMH",
        "stops": 0,
        "type": "route"
}
```

BUT… HOW DO I QUERY THAT?



SQL
JAQL,
HLQL, CQL
HiveQL, HLQL
Map/Reduce
This is brilliant.
SQL
JAQL,
HLQL, CQL
HiveQL, HLQL
Map/Reduce
But I like this.

UC San Diego



The SQL++ Query Language: Configurable, Unifying and Semi-structured*

Kian Win Ong, Yannis Papakonstantinou, Romain Vernoux
{kianwin,yannis,rvernoux}@cs.ucsd.edu

ABSTRACT

NoSQL databases support semi-structured data, typically modeled as JSON. They also provide limited (but expanding) query languages. Their idiomatic, non-SQL language constructs, the many variations, and the lack of formal semantics inhibit deep understanding of the query languages, and also impede progress towards clean, powerful, declarative query languages.

This paper specifies the syntax and semantics of SQL++, which is applicable to both JSON native stores and SQL databases. The SQL++ semi-structured data model is a superset of both JSON and the SQL data model. SQL++ offers powerful computational capabilities for processing semi-structured data akin to prior non-relational query languages, notably OQL and XQuery. Yet, SQL++ is SQL backwards compatible and is generalized towards JSON by introducing only a small number of query language extensions to SQL. Indeed, the SQL capabilities are most often extended by removing semantic restrictions of SQL, rather than inventing new features.

tha languag is probably

Early adoption signs of SQL++ are positive: Version 4 of Couchbase's N1QL is explained as syntactic sugar over SQL++. AsterixDB will soon support the full SQL++ and Apache Drill is in the process of aligning with SQL++.

1. INTRODUCTION

Numerous databases marketed as SQL-on-Hadoop, NewSQL and NoSQL support Big Data applications. These databases generally support the 3Vs [7]. (i) Volume: amount of data (ii) Velocity: speed of data in and out (iii) Variety: semi-structured and heterogeneous data. Due to the Variety requirement, they have adopted semi-structured data models, which are generally different subsets of enriched JSON.[1]

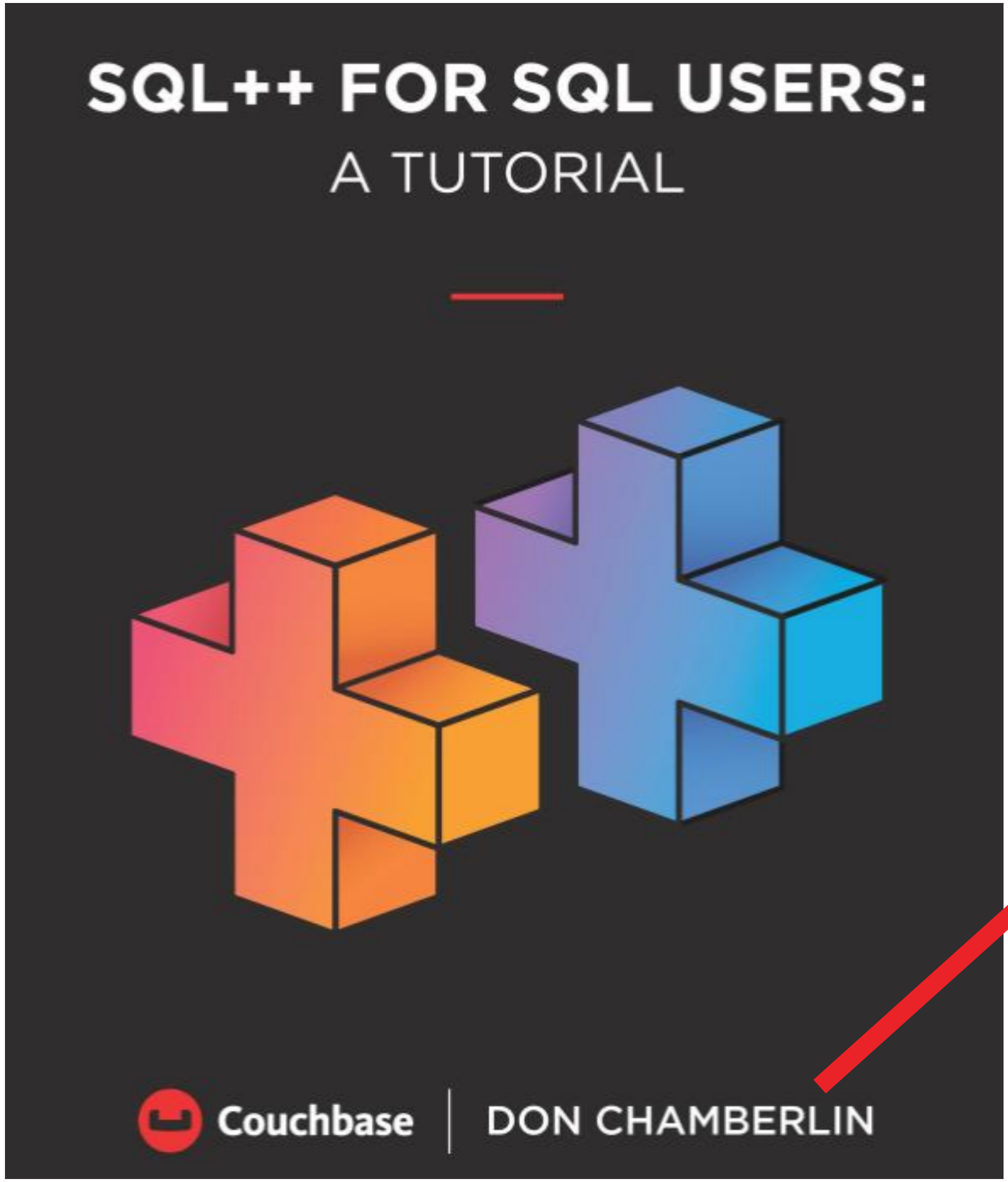
Their evolving query languages fall short of full-fledged semi-structured query language capabilities[2] and have many variations. Some variations are due to superficial syntactic differences. However, other variations are genuine differences in query language capabilities and semantics. The lack of succinct, formal syntax and semantics inhibits a deep understanding of the various systems. It also impedes progress larative languages for ing semi-st red

UCSD

- http://forward.ucsd.edu/sqlpp.html

The SQL++ Query Language

- https://arxiv.org/abs/1405.3631



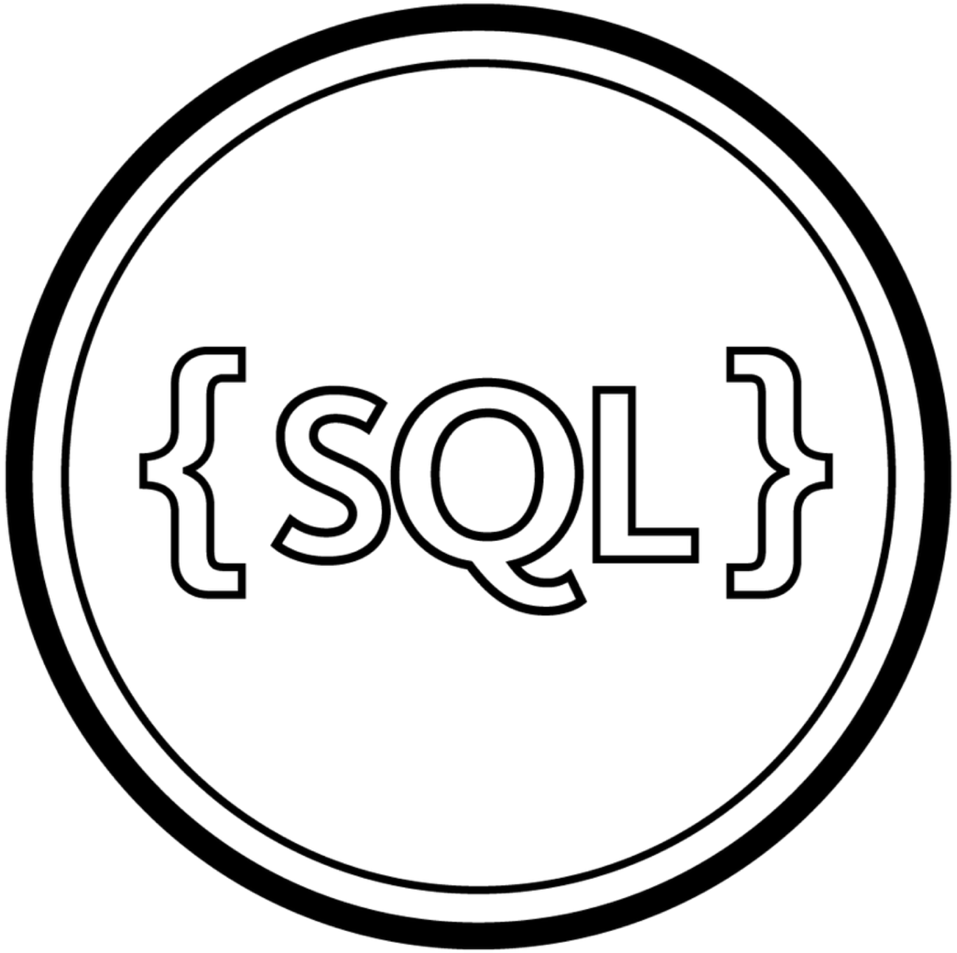
SQL++ FOR SQL USERS:

A TUTORIAL

Couchbase | DON CHAMBERLIN

Couchbase

N1QL = JSON + SQL



SQL EXAMPLE

mytable

ID	foo	bar	baz
1	matt	groves	qux
2	ali	groves	notqux
3	emma	groves	notqux

```
SELECT foo, bar
FROM mytable
WHERE baz = 'qux'
```

SQL++

mybucket

```
key: 1
{
    "foo" : "matt",
    "bar" : "groves",
    "baz" : "qux"
}

key: 2
{
    "foo" : "ali",
    "bar" : "groves",
    "baz" : "notqux"
}

key: 3
{
    "foo" : "emma",
    "bar" : "groves",
    "baz" : "notqux"
}
```

```
SELECT foo, bar
FROM mybucket
WHERE baz = 'qux'
```

myusers

```
key 1
{
    "name" : "matt",
    "address" : {
        "street" : "White Rd",
        "city" : "Grove City",
        "state" : "OH"
    }
}

key 2
{
    "name" : "emma",
    "address" : {
        "street" : "High St",
        "city" : "Columbus",
        "state" : "OH"
    }
}
```

THIS IS OLD NEWS THIS IS

```
SELECT address.city
FROM myusers
```

myusers

```
key 1
{
    "name" : "matt",
    "favoriteFoods" : [
        "pizza",
        "cheesecake",
        "donuts"
    ]
}

key 2
{
    "name" : "emma",
    "favoriteFoods" : [
        "donuts",
        "Lucky Charms",
        "chicken"
    ]
}
```

```
SELECT favoriteFoods[1]
FROM myusers
```

myusers

```
key 1
{
    "name" : "matt",
    "favoriteFoods" : [
        "pizza",
        "cheesecake",
        "donuts"
    ]
}

key 2
{
    "name" : "emma",
    "favoriteFoods" : [
        "donuts",
        "Lucky Charms",
        "chicken"
    ]
}
```

```
SELECT u.name
FROM myusers u
WHERE ANY f
  IN u.favoriteFoods
  SATISFIES f == 'pizza'
END;
```

myusers

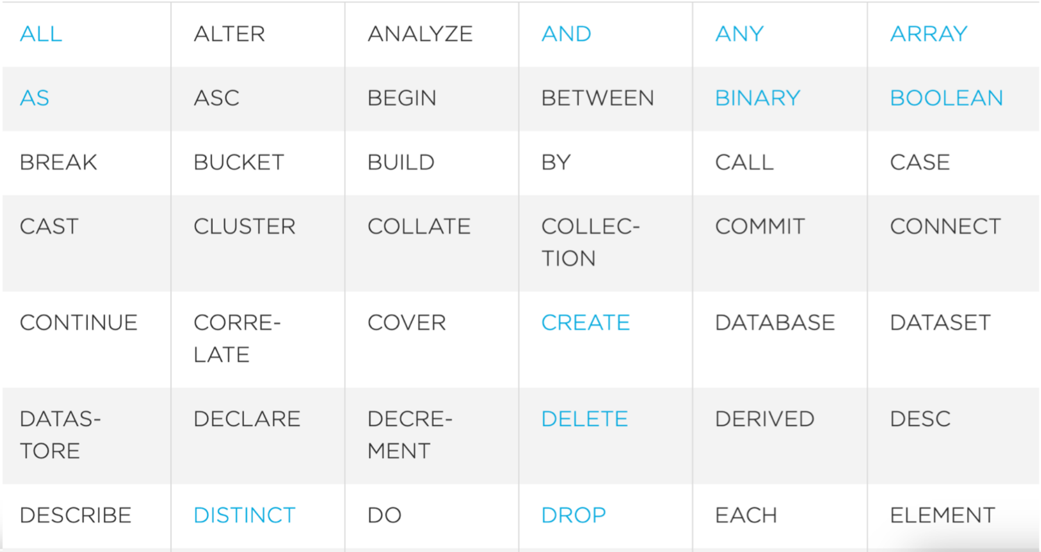
```
key 1
{
    "name" : "matt",
    "favoriteFoods" : [
        "pizza",
        "cheesecake",
        "donuts"
    ]
}
```

```
SELECT food, u.name
FROM myusers u
UNNEST u.favoriteFoods food;
```

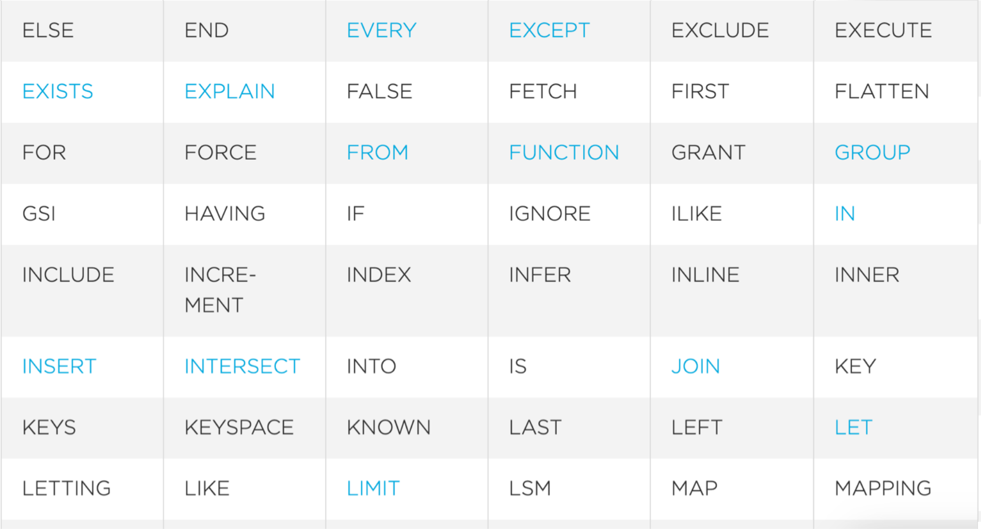
```
[
    {
        "food": "pizza",
        "name": "matt"
    },
    {
        "food": "cheesecake",
        "name": "matt"
    },
    {
        "food": "donuts",
        "name": "matt"
    }
]
```

SQL++ is Backwards Compatible

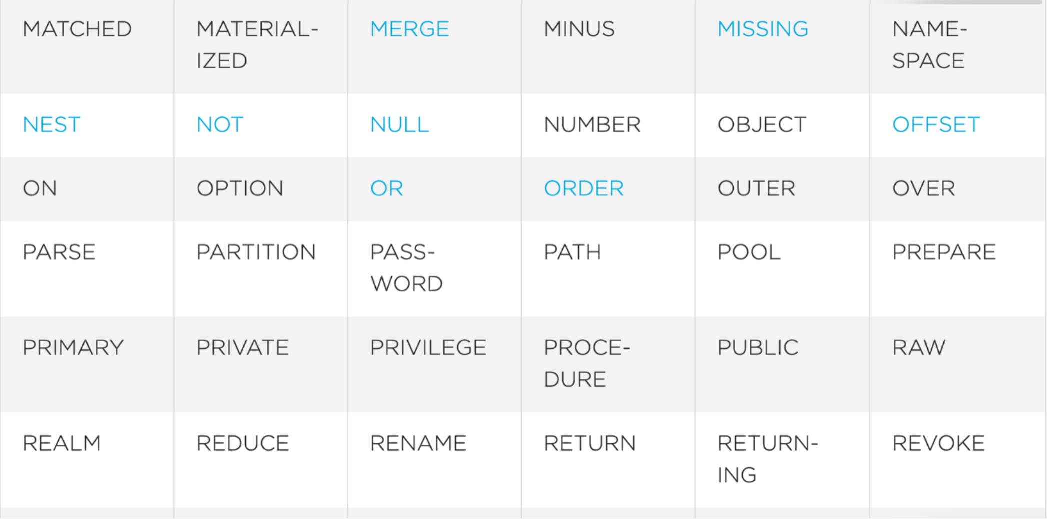
N1QL Keywords



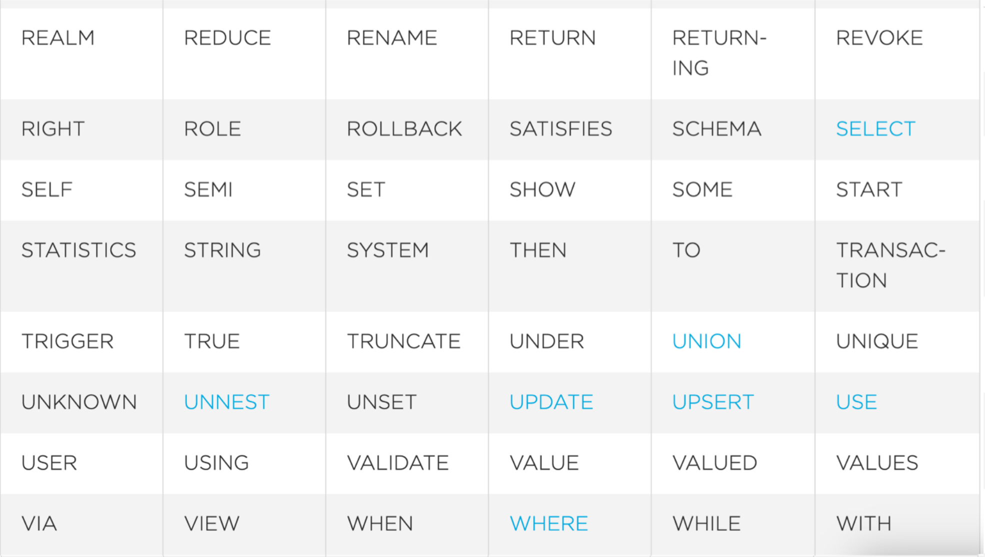
ALL	ALTER	ANALYZE	AND	ANY	ARRAY
AS	ASC	BEGIN	BETWEEN	BINARY	BOOLEAN
BREAK	BUCKET	BUILD	BY	CALL	CASE
CAST	CLUSTER	COLLATE	COLLECTION	COMMIT	CONNECT
CONTINUE	CORRELATE	COVER	CREATE	DATABASE	DATASET
DATASTORE	DECLARE	DECREMENT	DELETE	DERIVED	DESC
DESCRIBE	DISTINCT	DO	DROP	EACH	ELEMENT



ELSE	END	EVERY	EXCEPT	EXCLUDE	EXECUTE
EXISTS	EXPLAIN	FALSE	FETCH	FIRST	FLATTEN
FOR	FORCE	FROM	FUNCTION	GRANT	GROUP
GSI	HAVING	IF	IGNORE	ILIKE	IN
INCLUDE	INCREMENT	INDEX	INFER	INLINE	INNER
INSERT	INTERSECT	INTO	IS	JOIN	KEY
KEYS	KEYSPACE	KNOWN	LAST	LEFT	LET
LETTING	LIKE	LIMIT	LSM	MAP	MAPPING



MATCHED	MATERIALIZED	MERGE	MINUS	MISSING	NAMESPACE
NEST	NOT	NULL	NUMBER	OBJECT	OFFSET
ON	OPTION	OR	ORDER	OUTER	OVER
PARSE	PARTITION	PASSWORD	PATH	POOL	PREPARE
PRIMARY	PRIVATE	PRIVILEGE	PROCEDURE	PUBLIC	RAW
REALM	REDUCE	RENAME	RETURN	RETURNING	REVOKE



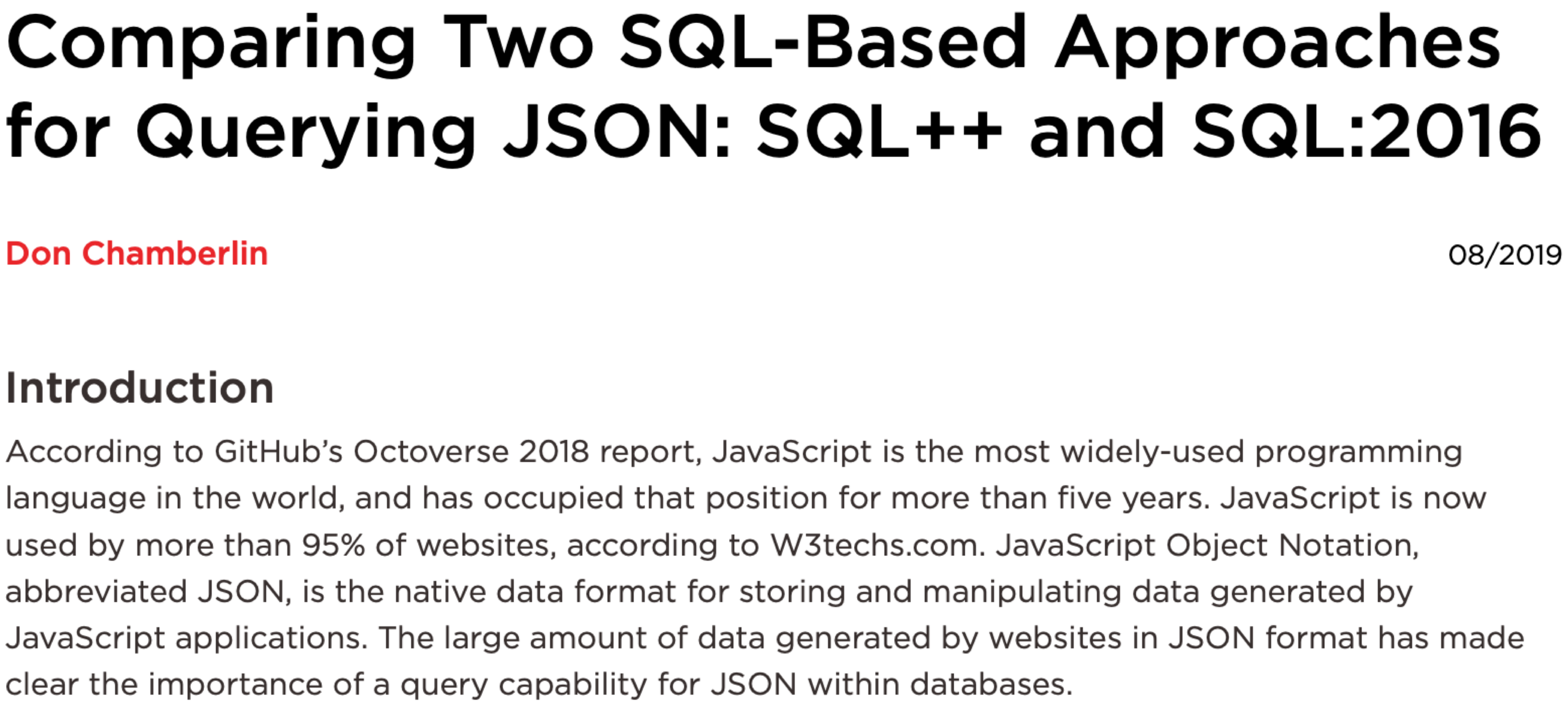
REALM	REDUCE	RENAME	RETURN	RETURNING	REVOKE
RIGHT	ROLE	ROLLBACK	SATISFIES	SCHEMA	SELECT
SELF	SEMI	SET	SHOW	SOME	START
STATISTICS	STRING	SYSTEM	THEN	TO	TRANSACTION
TRIGGER	TRUE	TRUNCATE	UNDER	UNION	UNIQUE
UNKNOWN	UNNEST	UNSET	UPDATE	UPSERT	USE
USER	USING	VALIDATE	VALUE	VALUED	VALUES
VIA	VIEW	WHEN	WHERE	WHILE	WITH

Other SQL++ Implementations

Couchbase

START A REVOLUTION

BUT SQL:2016 INTRODUCED JSON SUPPORT



Comparing Two SQL-Based Approaches for Querying JSON: SQL++ and SQL:2016

Don Chamberlin

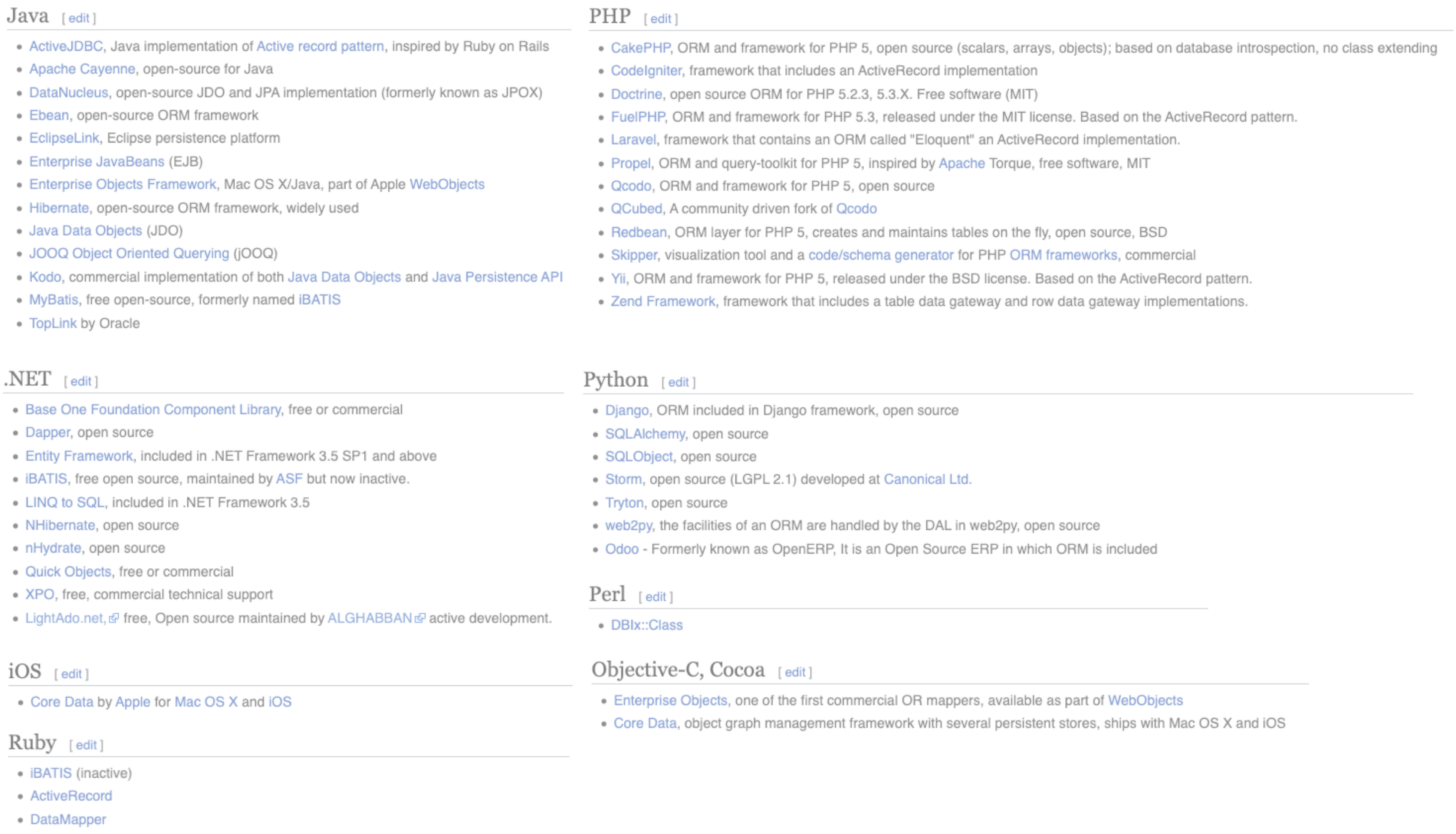
08/2019

Introduction

According to GitHub's Octoverse 2018 report, JavaScript is the most widely-used programming language in the world, and has occupied that position for more than five years. JavaScript is now used by more than 95% of websites, according to W3techs.com. JavaScript Object Notation, abbreviated JSON, is the native data format for storing and manipulating data generated by JavaScript applications. The large amount of data generated by websites in JSON format has made clear the importance of a query capability for JSON within databases.

Persistence Abstraction Layer (PAL)

(no pun intended)



Java [edit]

- ActiveJDBC, Java implementation of Active record pattern, inspired by Ruby on Rails
- Apache Cayenne, open-source for Java
- DataNucleus, open-source JDO and JPA implementation (formerly known as JPOX)
- Ebean, open-source ORM framework
- EclipseLink, Eclipse persistence platform
- Enterprise JavaBeans (EJB)
- Enterprise Objects Framework, Mac OS X/Java, part of Apple WebObjects
- Hibernate, open-source ORM framework, widely used
- Java Data Objects (JDO)
- JOOQ Object Oriented Querying (jOOQ)
- Kodo, commercial implementation of both Java Data Objects and Java Persistence API
- MyBatis, free open-source, formerly named iBATIS
- TopLink by Oracle

.NET [edit]

- Base One Foundation Component Library, free or commercial
- Dapper, open source
- Entity Framework, included in .NET Framework 3.5 SP1 and above
- iBATIS, free open source, maintained by ASF but now inactive.
- LINQ to SQL, included in .NET Framework 3.5
- NHibernate, open source
- nHydrate, open source
- Quick Objects, free or commercial
- XPO, free, commercial technical support
- LightAdo.net, free, Open source maintained by ALGHABBAN active development.

iOS [edit]

- Core Data by Apple for Mac OS X and iOS

Ruby [edit]

- iBATIS (inactive)
- ActiveRecord
- DataMapper

PHP [edit]

- CakePHP, ORM and framework for PHP 5, open source (scalars, arrays, objects); based on database introspection, no class extending
- CodeIgniter, framework that includes an ActiveRecord implementation
- Doctrine, open source ORM for PHP 5.2.3, 5.3.X. Free software (MIT)
- FuelPHP, ORM and framework for PHP 5.3, released under the MIT license. Based on the ActiveRecord pattern.
- Laravel, framework that contains an ORM called "Eloquent" an ActiveRecord implementation.
- Propel, ORM and query-toolkit for PHP 5, inspired by Apache Torque, free software, MIT
- Qcodo, ORM and framework for PHP 5, open source
- QCubed, A community driven fork of Qcodo
- Redbean, ORM layer for PHP 5, creates and maintains tables on the fly, open source, BSD
- Skipper, visualization tool and a code/schema generator for PHP ORM frameworks, commercial
- Yii, ORM and framework for PHP 5, released under the BSD license. Based on the ActiveRecord pattern.
- Zend Framework, framework that includes a table data gateway and row data gateway implementations.

Python [edit]

- Django, ORM included in Django framework, open source
- SQLAlchemy, open source
- SQLObject, open source
- Storm, open source (LGPL 2.1) developed at Canonical Ltd.
- Tryton, open source
- web2py, the facilities of an ORM are handled by the DAL in web2py, open source
- Odoo - Formerly known as OpenERP, It is an Open Source ERP in which ORM is included

Perl [edit]

- DBIx::Class

Objective-C, Cocoa [edit]

- Enterprise Objects, one of the first commercial OR mappers, available as part of WebObjects
- Core Data, object graph management framework with several persistent stores, ships with Mac OS X and iOS

Key Takeaways

- Performance vs Productivity
- ORMs are awesome, but don’t forget to double check it;
- The relational model needs to evolve, and you are part of this change.

goto; copenhagen

GOTO Copenhagen 2019
Conference Nov. 18 - 20

Please

Remember to rate this session

Thank you!

Follow us @gotocph

Thanks!

Follow me at @deniswsrosa