BigData-Systeme mit Apache Cassandra

Taking databases to a next level

im Juni 2014

WidasConcepts Unternehmensberatung GmbH • Maybachstraße 2 • 71299 Wimsheim • http://www.widas.de


CURSED
ORACLE®

1. Wenn relationale Datenbanken an ihre Grenzen kommen

   „Meine MySQL Datenbank funktioniert doch!“

2. Was Cassandra anders macht

   „Wo sind meine Joins?“

3. Der Datastax Cassandra Treiber

   „Das ist ja wie JDBC!“

Wenn relationale Datenbanken an ihre Grenzen kommen

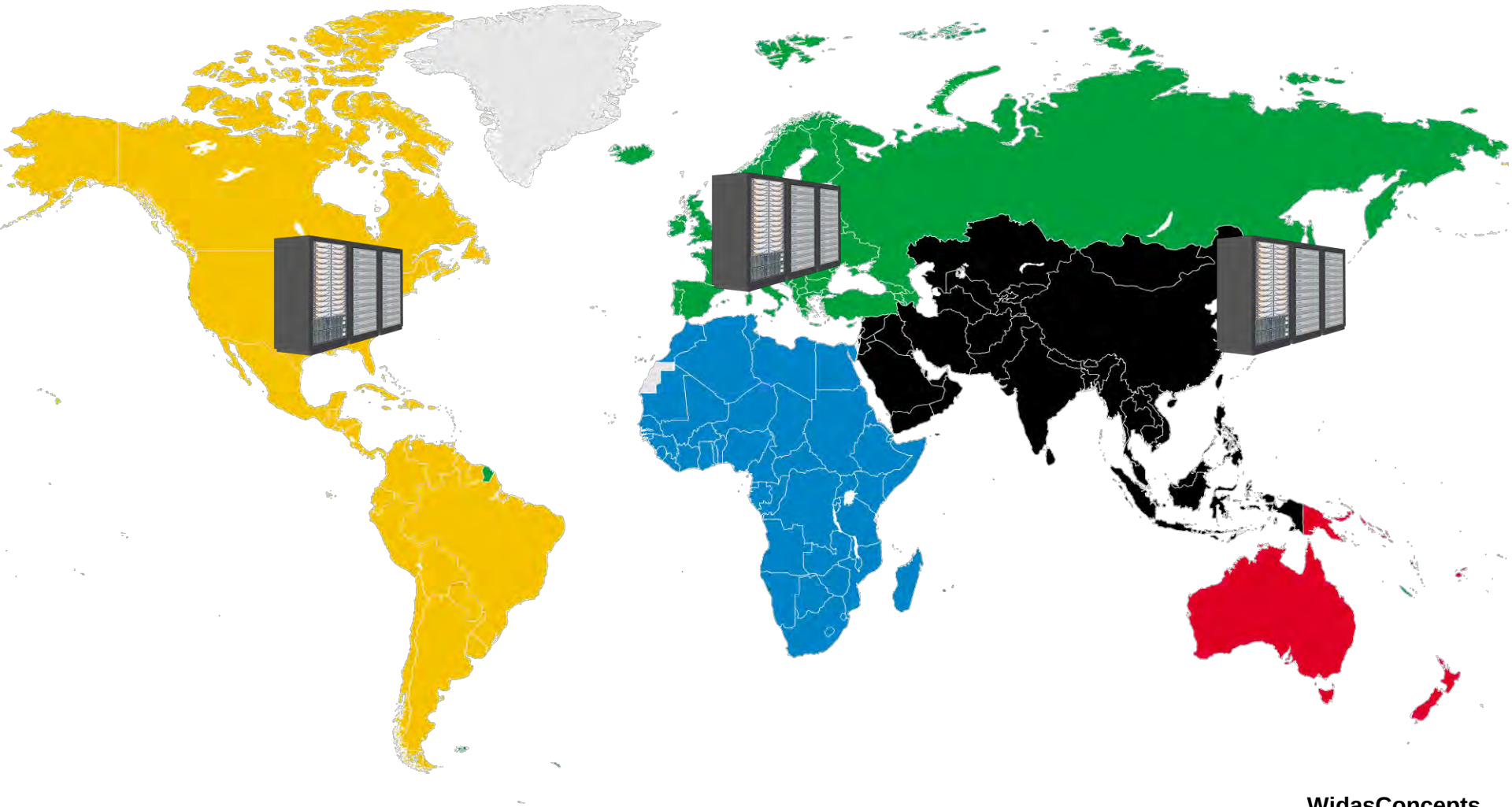
„Oracle SQL verhindert den weltweiten Datenaustausch“

„Wir benötigen regelmäßige Downtimes für Schema Änderungen“

„Relationale Datenbanken sind nicht ausgelegt auf unsere Datenmengen“

Kosteneffizienz?

„Oracle SQL verhindert den weltweiten Datenaustausch“



„Wir benötigen regelmäßige Downtimes für Schema Änderungen“

user_id	name	first_name
4711	Skywalker	Luke
4712	Solo	Han
4713	Chewbacca	null

user_id	name	first_name	homeworld
4711	Skywalker	Luke	Polis Massa
4712	Solo	Han	null
4713	Chewbacca	null	Kashyyyk

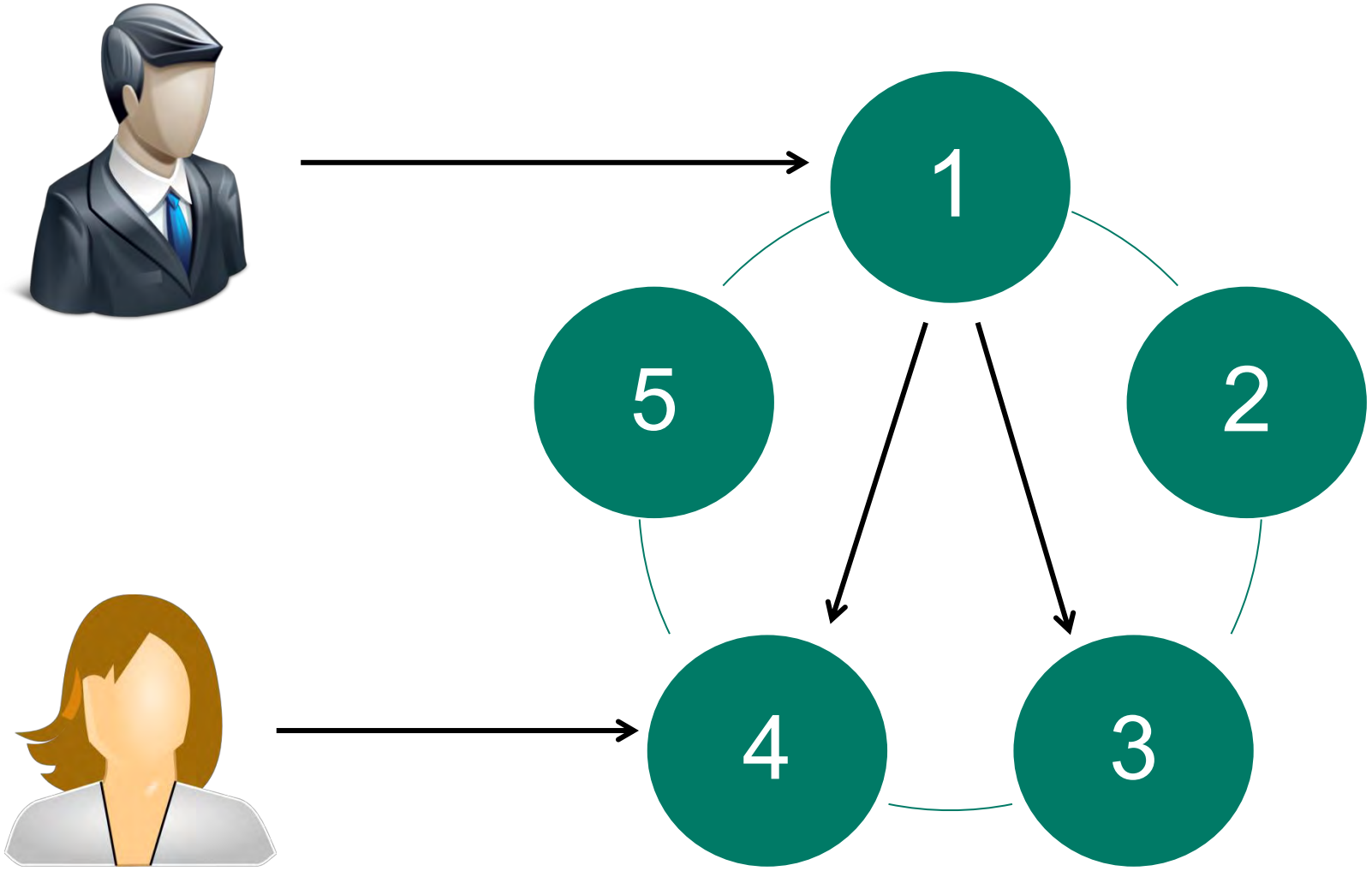
„Wir benötigen regelmäßige Downtimes für Schema Änderungen“

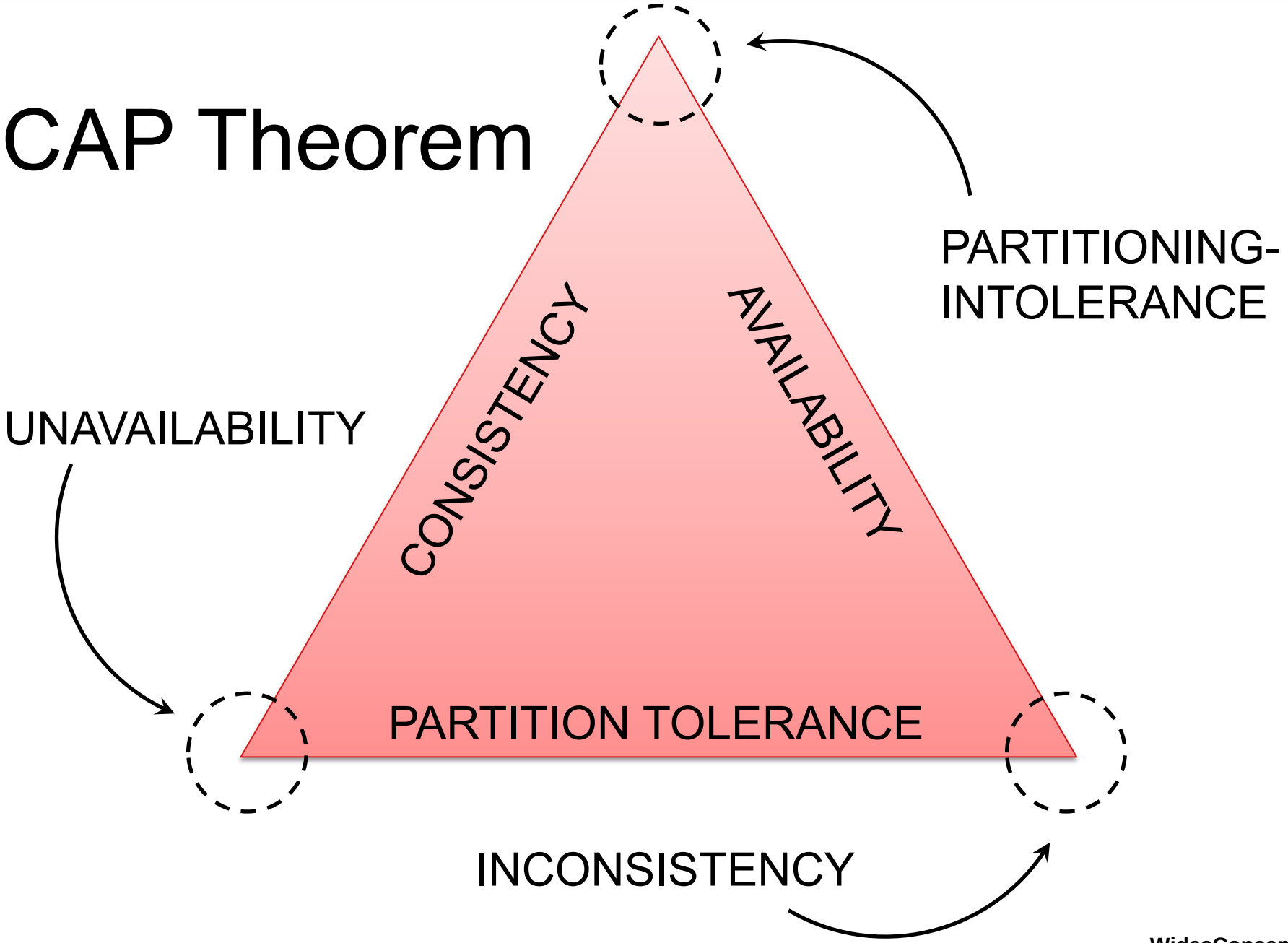
KEY	COLUMNS		
4711	„name“ : „Skywalker“	„first_name“ : „Luke“	„homeworld“ : „Polis Massa“
4712	„name“ : „Solo“	„first_name“ : „Han“	
4713	„name“ : „Chewbacca“	„homeworld“ : „Kashyyyk“	

„Kosteneffizienz?“

Was Cassandra anders macht

4711	„name“ : „Skywalker“	„first_name“ : „Luke“	„homeworld“ : „Polis Massa“




CAP Theorem
PARTITIONING-
INTOLERANCE
UNAVAILABILITY
CONSISTENCY
AVAILABILITY
PARTITION TOLERANCE
INCONSISTENCY

ACID

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

BASE

- Basic Availability
- Soft-state
- Eventual consistency

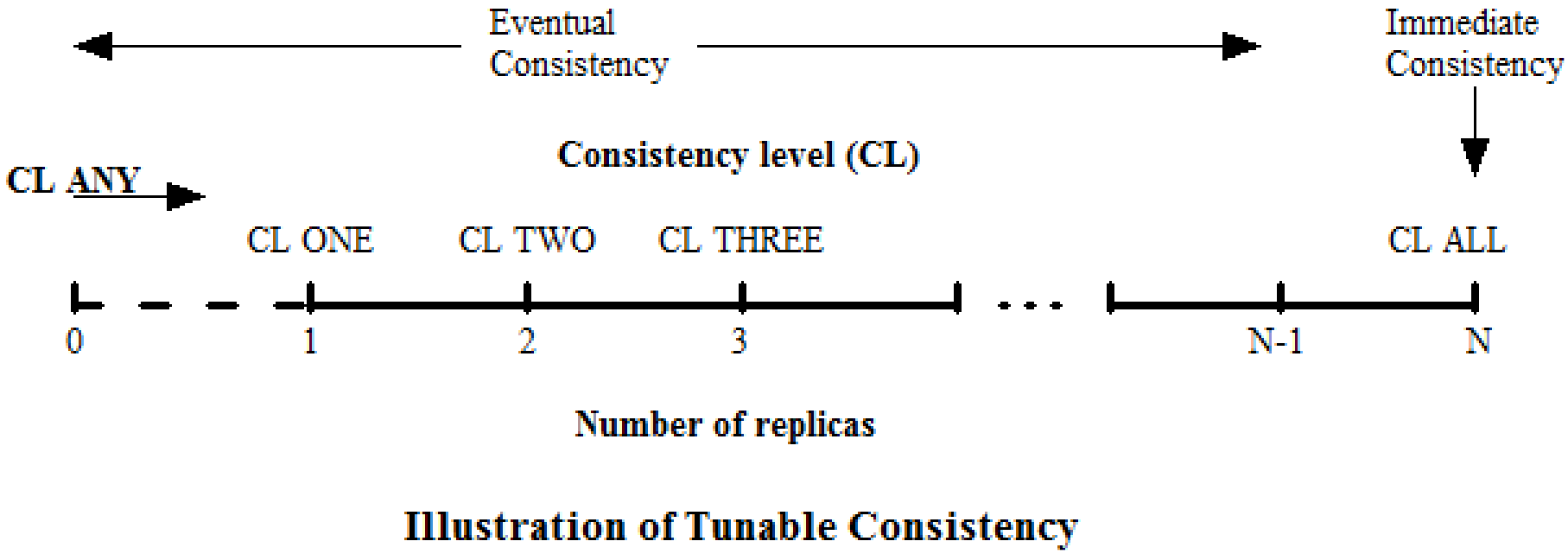
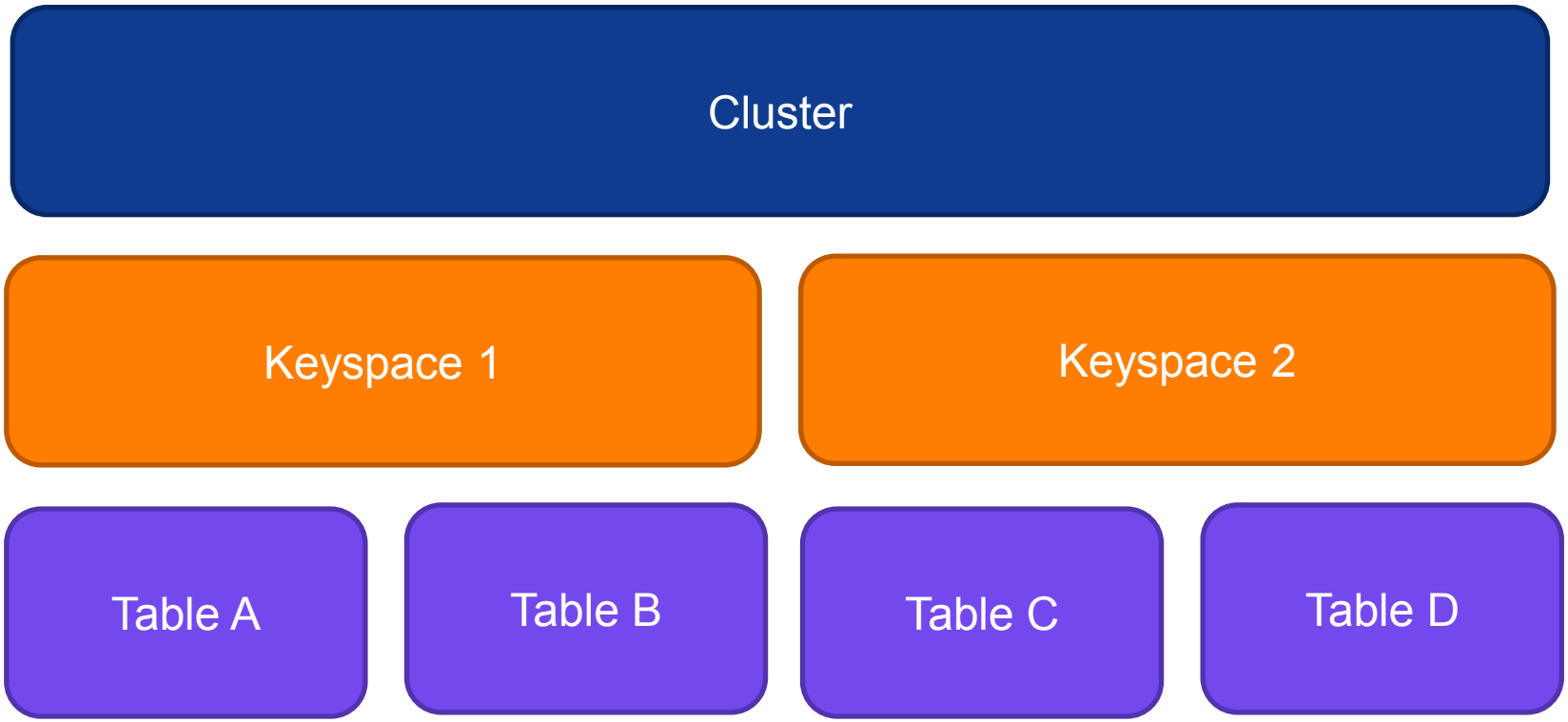


Illustration of Tunable Consistency

„CQL is exactly like SQL“

(except where it is not)


Cluster
Keyspace 1
Keyspace 2
Table A
Table B
Table C
Table D

Keyspaces

```
CREATE KEYSPACE starwars
     WITH replication = {
          'class': 'SimpleStrategy',
          'replication_factor': 3
     }
```

Tabellen

CQL

```
CREATE TABLE user (
      user_id uuid,
      name text,
      first_name text,
      homeworld text,
      PRIMARY KEY (user_id));
```

SQL

```
CREATE TABLE user (
      user_id int,
      name varchar(255),
      first_name varchar(255),
      homeworld varchar(255),
      PRIMARY KEY (user_id));
```

INSERT INTO

SQL

```
INSERT INTO user (
      user_id, name, first_name)
VALUES (
      4712, ‘Solo’, ‘Han’);
```

CQL

```
INSERT INTO user (
      user_id, name, first_name)
VALUES (
      4712, ‘Solo’, ‘Han’);
```

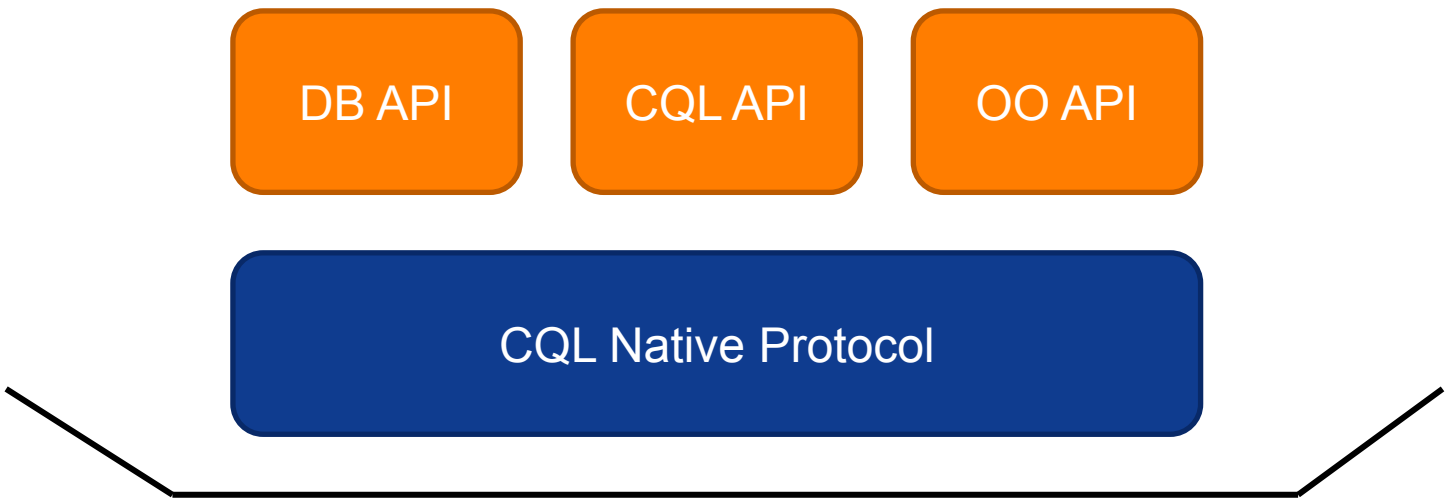
SQL

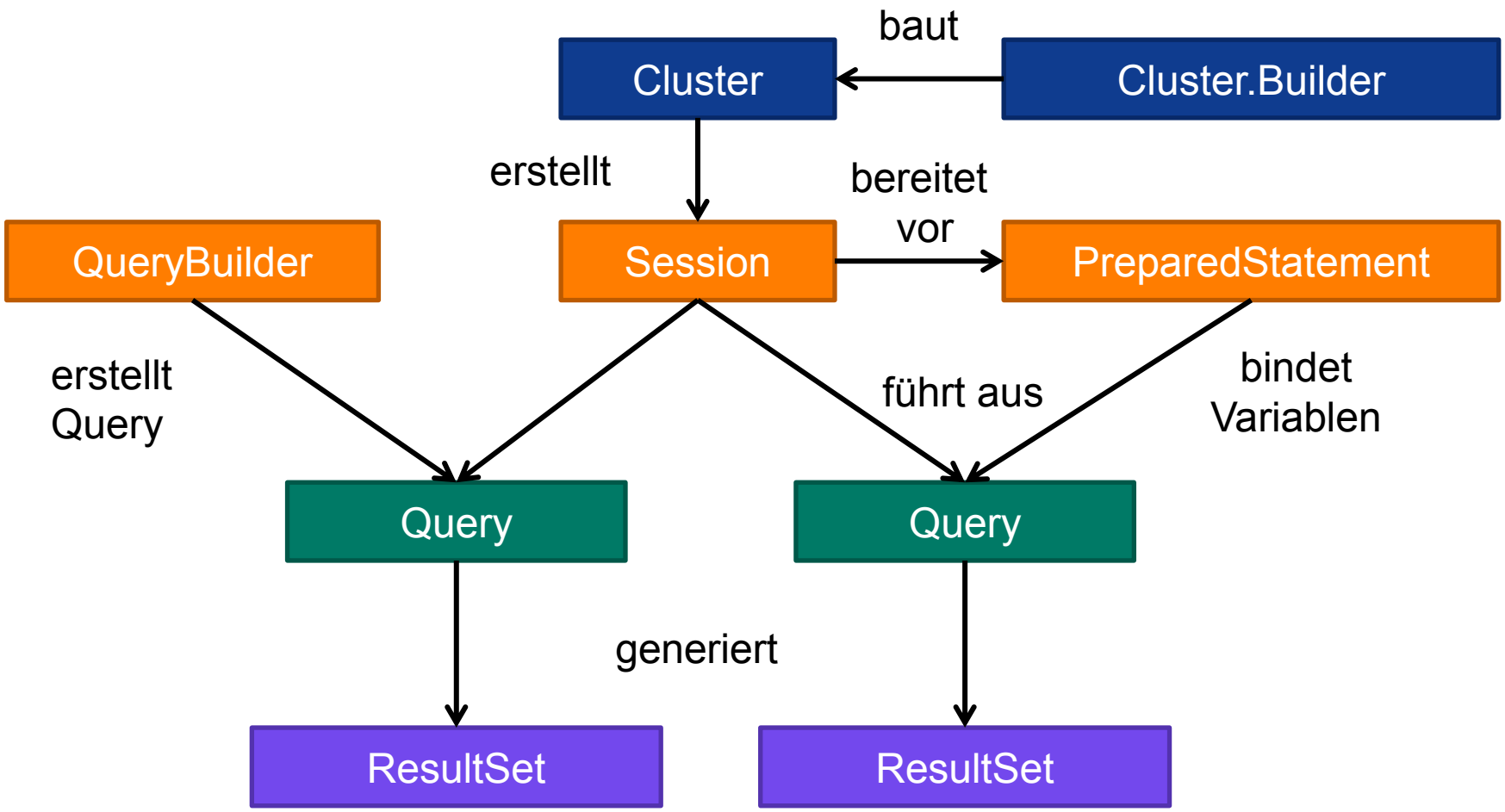
SELECT

```
SELECT name, homeplanet
FROM user
WHERE user_id = 4712;
```

CQL

```
SELECT name, homeplanet
FROM user
WHERE user_id = 4712;
```


DB API
CQL API
OO API
CQL Native Protocol


baut
Cluster
Cluster.Builder
erstellt
bereitet vor
QueryBuilder
Session
PreparedStatement
erstellt Query
führt aus
bindet Variablen
Query
Query
generiert
ResultSet
ResultSet

Simple Cassandra Client

user_id	name	first_name	homeworld
4711	Skywalker	Luke	Polis Massa
4712	Solo	Han	null
4713	Chewbacca	null	Kashyyyk

maven

```
<dependency>
  <groupId>com.datastax.cassandra</groupId>
  <artifactId>
     cassandra-driver-core</artifactId>
  <version>2.0.1</version>
</dependency>
```

1. Aufbau der Verbindung

```
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.Host;
import com.datastax.driver.core.Metadata;

public class SimpleClient {
      // ...
}
```

```
private Cluster cluster;
```

```
public void connect(String node) {
      // ...
}
```

```
public void close() {
      // ...
}
```

```
public static void main(String[] args) {
      // ...
}
```

```
public void connect(String node) {
    cluster = Cluster.builder()
            .addContactPoint(node)
            .build();
}
```

```
public void close() {
    cluster.close();
}
```

```
public static void main(String[] args) {
    SimpleClient client = new SimpleClient();
    client.connect("127.0.0.1");
    client.close();
}
```

2. Absetzen von Abfragen

```
private Session session;
```

```
public void connect(String node) {
    cluster = Cluster.builder()
            .addContactPoint(node)
            .build();
    // NEU
    session = cluster.connect();
}
```

```
public void createSchema() {}
```

```
public void loadData() {}
```

```
public void queryData() {}
```

```
public void createSchema() {
  session.execute(
    "CREATE KEYSPACE sw WITH replication "
    + “= {'class':'SimpleStrategy', "
    + "'replication_factor':3};"
  );

  session.execute(
    "CREATE TABLE sw.user (" +
    "user_id uuid PRIMARY KEY," +
    "name text," +
    "first_name text," +
    "homeworld text);"
  );
}
```

```
public void loadData() {
  session.execute(
    "INSERT INTO sw.user (user_id, name, " +
      "first_name, homeworld) " +
    "VALUES (" +
      "756716f7-2e54-4715-9f00-91dcbea6cf50," +
      "'Skywalker'," +
      "'Luke'," +
      "'Polis Massa')";
  );
}
```

```
public void queryData() {
  ResultSet results = session.execute(
    "SELECT * FROM sw.user " +
    "WHERE id = " +
    "756716f7-2e54-4715-9f00-91dcbea6cf50;");

  for (Row row : results) {
    System.out.println(
      row.getString("first_name") + " | " +
      row.getString("name") + " | " +
      row.getString("homeworld")
    );
  }
}
```

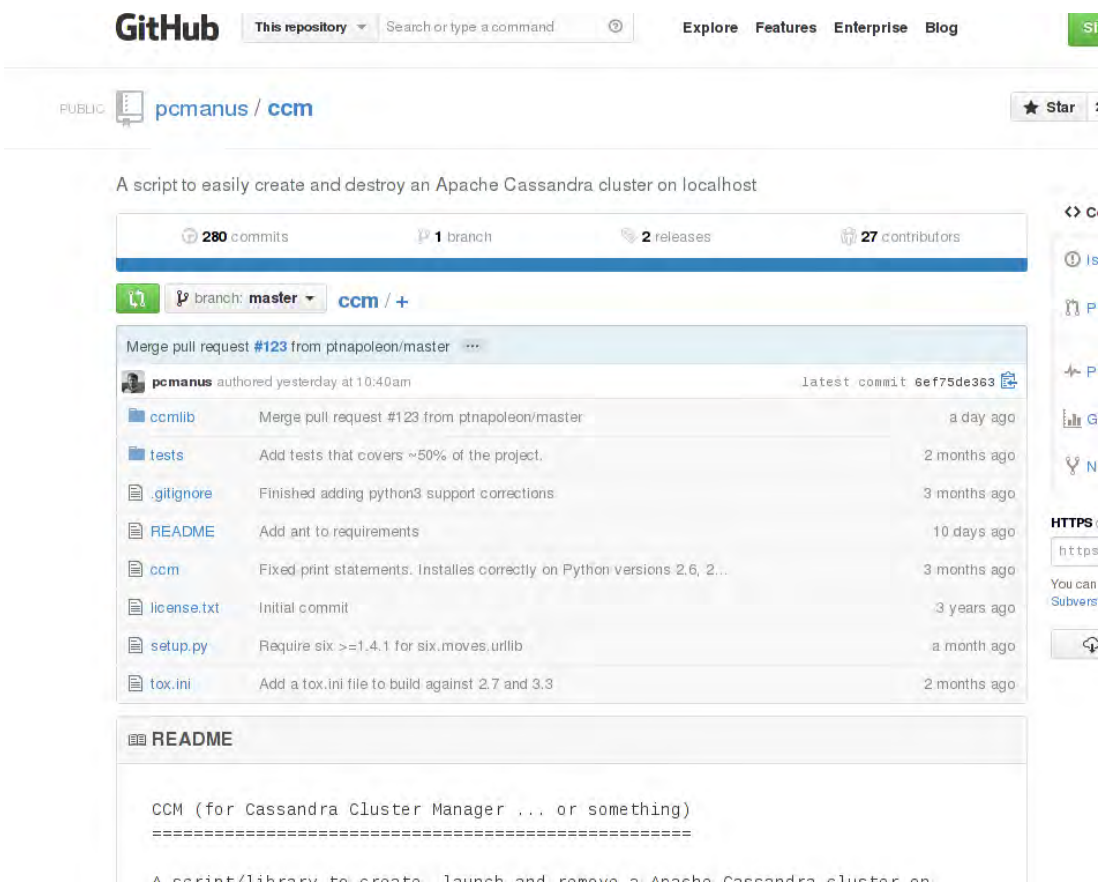
3. Simple Client

```
public static void main(String[] args) {
  SimpleClient client = new SimpleClient();
  client.connect("127.0.0.1");

  client.createSchema();
  client.loadData();
  client.queryData();

  client.close();
}
```

CCM – Cassandra Cluster Manager



https://github.com/pcmanus/ccm

QueryBuilder

```
public List<Row> getRows() {
    Statement statement = QueryBuilder
        .select()
        .all()
        .from("sw", "user")
        .where(eq("user_id", "xyz");
    return getSession()
        .execute(statement)
        .all();
}
```

Philipp Stussak, Consultant

Telefon: +49 (7044) 95103 – 100

Mail: philipp.stussak@widas.de

widas concepts

IT Consulting

www.widas.de