



FROM BI TO BIG DATA



OR, THERE AND BACK AGAIN



FRANCESCO MUCIO
@mucio
[linkedin.com/in/mucio](https://www.linkedin.com/in/mucio)

01-05-2019



ZALANDO AT A GLANCE

~ **5.4** billion EUR
revenue 2018

> **250**
million

visits
per
month

> **300.000**
product choices

> **15.000**
employees in
Europe

> **79%**
of visits via
mobile devices

> **26**
million
active customers

~ **2.000**
brands

17
countries

ZALANDO

2008 ZALANDO IS BORN

2009 FIRST TV AD

2010

- FIRST PRIVATE LABEL
- START SHOP DEVELOPMENT IN HOUSE

2011

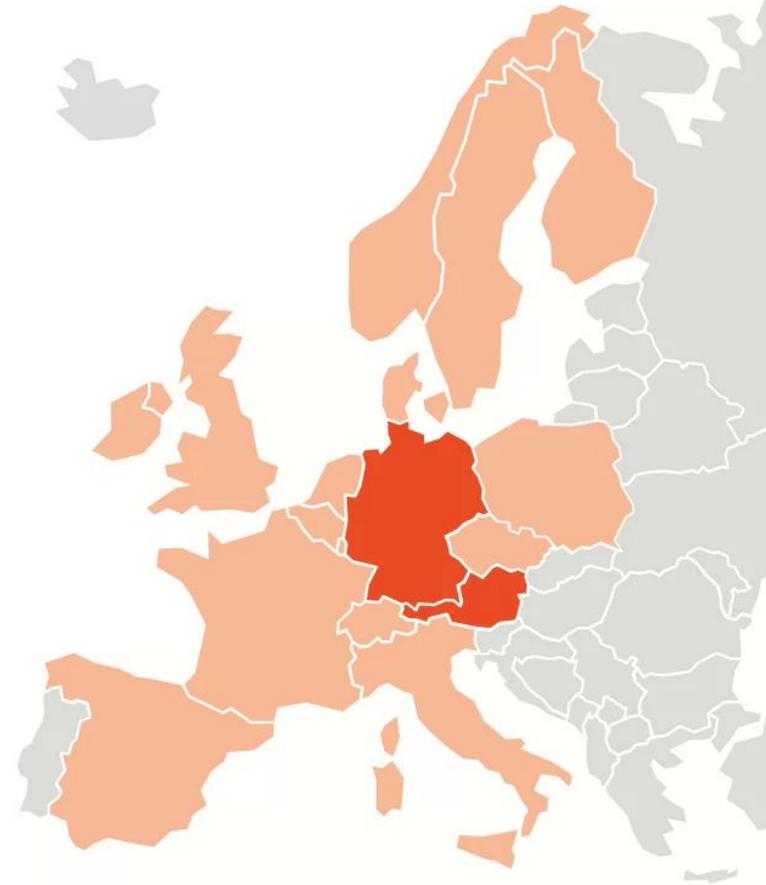
- SELF OPERATED FULFILLMENT CENTER
- PARTNER PROGRAM

2012 BREAK EVEN IN CORE REGIONS

2014 STOCK MARKET

2015

- FIRST WAREHOUSE ABROAD
- TECH HUBS IN DUBLIN AND HELSINKI



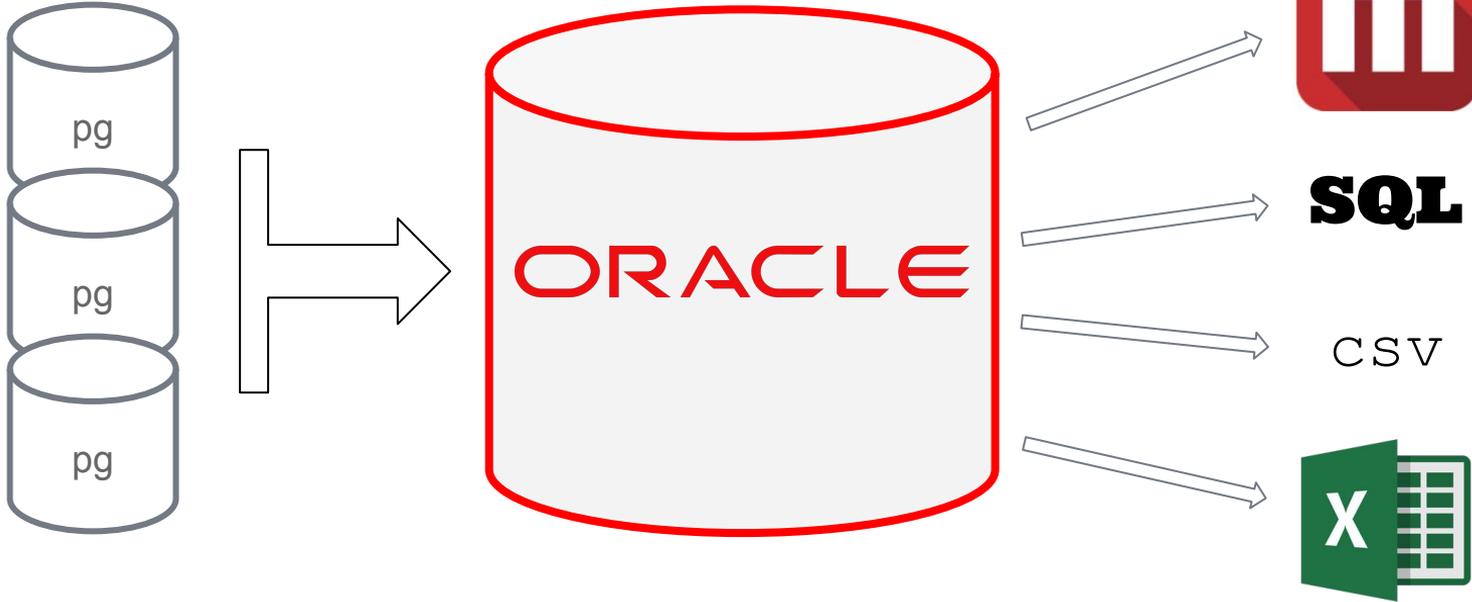
SHORT BI STORY
AT ZALANDO

OLD KENT
ROAD

A#60



THE OLD CLASSIC BI SETUP



OUR TARGET



GOOD REASONS TO HAVE A DATA LAKE



DEMOCRATIZE DATA

SPLIT STORAGE FROM COMPUTE





TIPS

START WITH A USE CASE

TO DO

1. WAKE UP

2. COFFEE

3. THE REST...



START SIMPLE



START PROPERLY

METADATA METADATA METADATA



The image features a dark, textured background covered with various colorful, fuzzy letters. The letters are in shades of red, yellow, green, purple, blue, and pink. Some letters are in focus, while others are blurred in the background. In the bottom right corner, there is a white text overlay that reads "THINGS I SHOULD HAVE LEARNED BY NOW".

**THINGS
I SHOULD HAVE LEARNED
BY NOW**

**SCHEMA ON WRITE.
SCHEMA ON READ.**

INSERT INTO <TABLE-NAME>





**SCHEMA
ON WRITE**

CENTRAL TELEVISION



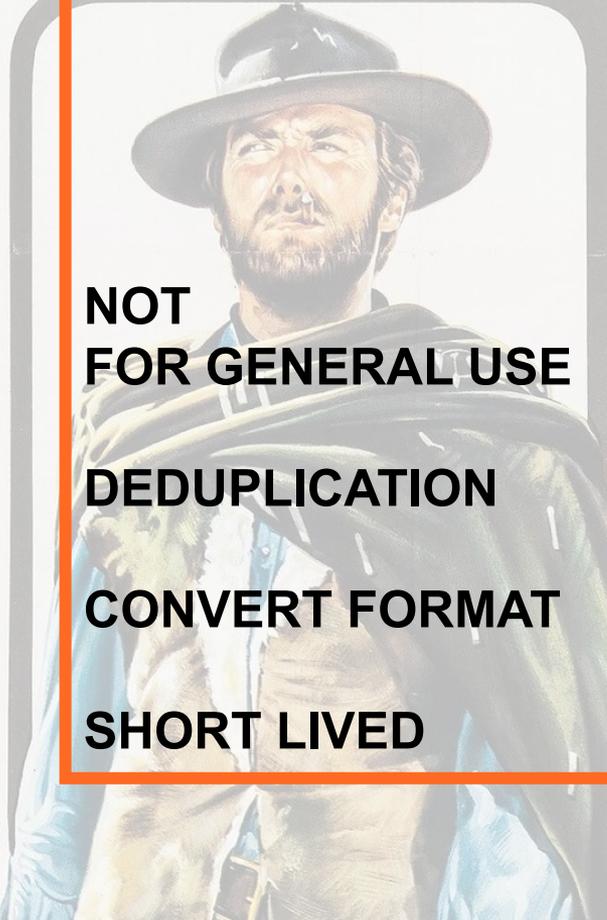
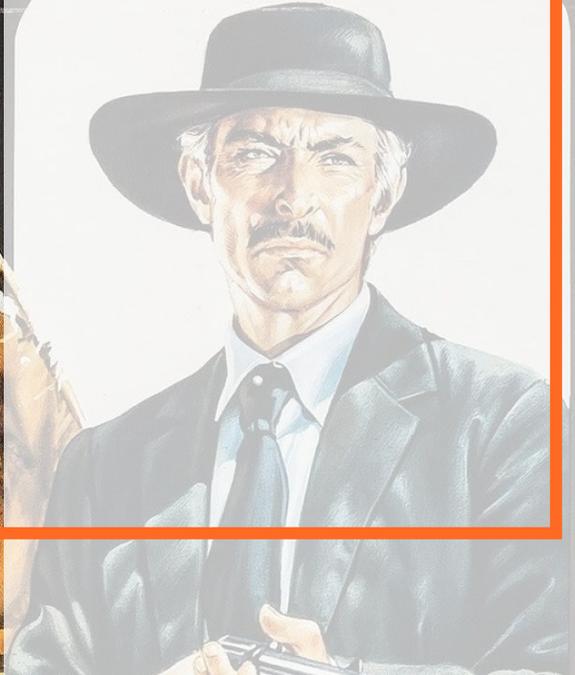
THE BRONZE
THE SILVER
THE GOLD

**NOT
FOR GENERAL USE**

DEDUPLICATION

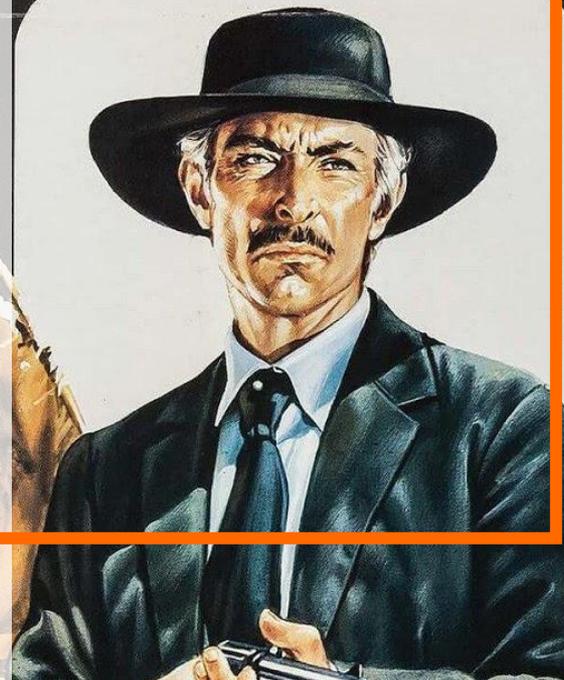
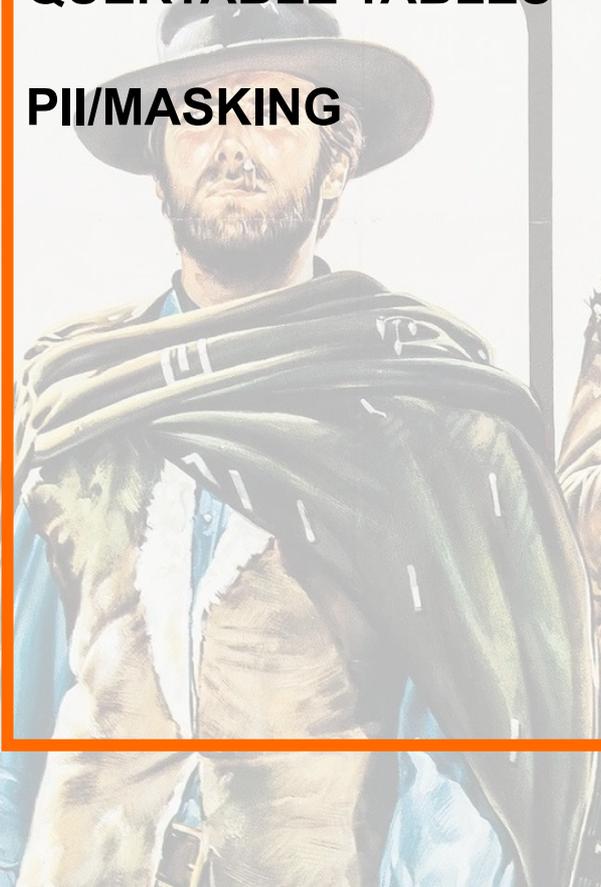
CONVERT FORMAT

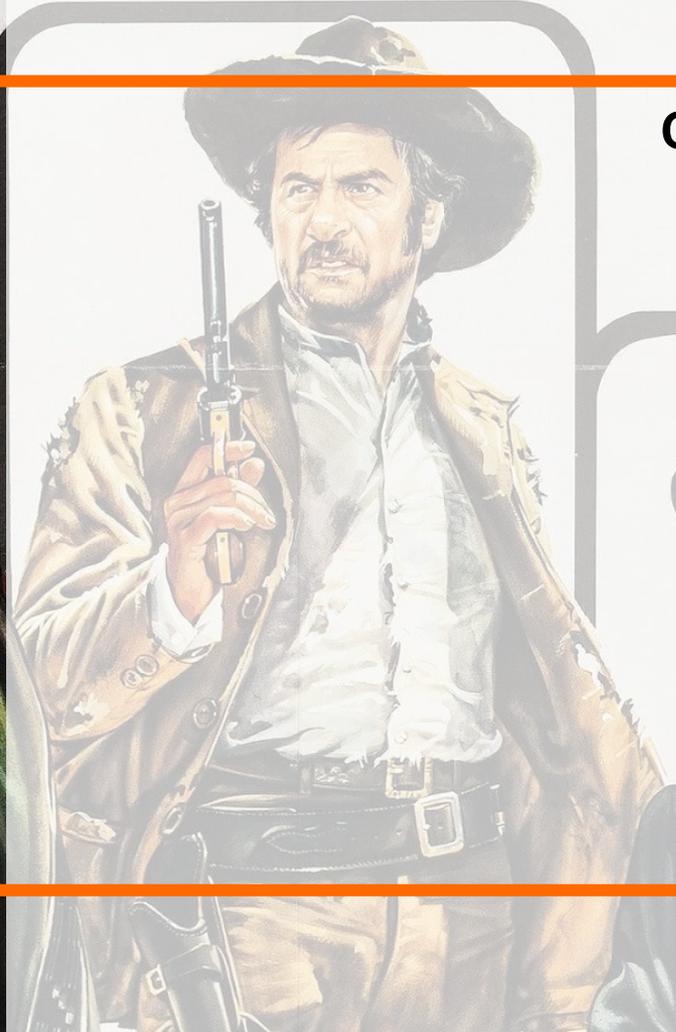
SHORT LIVED



QUERYABLE TABLES

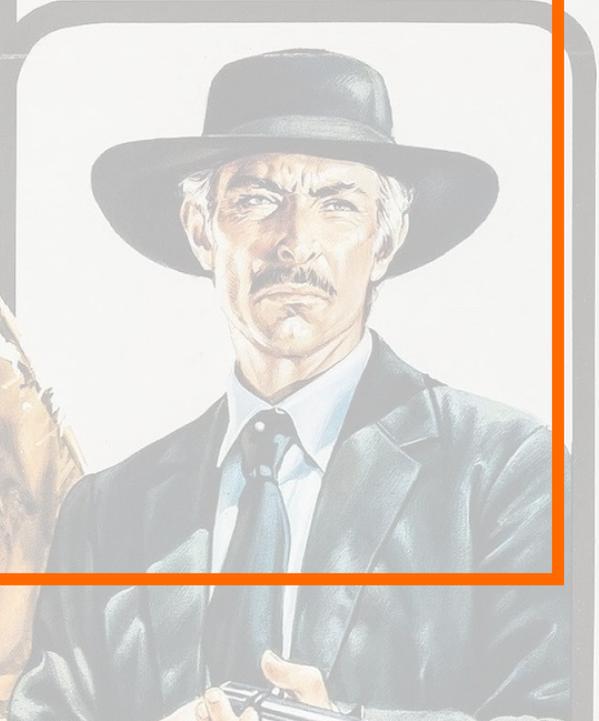
PII/MASKING





CURATED TABLES

AGGREGATES



BRONZE

**NOT FOR GENERAL USE
DEDUPLICATION
CONVERT FORMAT
SHORT LIVED**

SILVER

**QUERYABLE TABLES
PII/MASKING**

GOLD

**CURATED TABLES
AGGREGATES**

BRONZE
SOURCE DATA

SILVER
DATA WAREHOUSE

GOLD
DATA MART

PRESTO AND SPARK

SQL

FAST SQL

SQL ON EVERYTHING

presto

DATA PIPELINES

MACHINE LEARNING

STREAMING

APACHE

SPARK

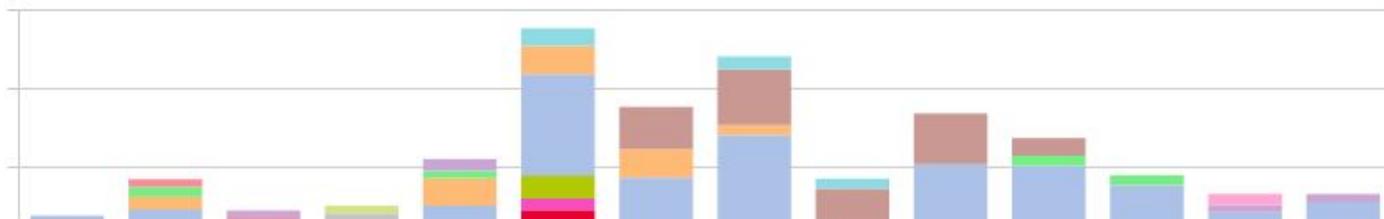
SPARK: HOW DOES IT LOOK LIKE?

```

%sql
with t1 as ( select date_trunc('WEEK', timestamp) dt,
                case when substr(clustername, 1, 4) = 'job-' then 'job-XXXX' else clustername end cn,
                sum(machinehours) machinehours
            from delta.`/user/hive/warehouse/billing.db/databricks_usage`
            group by date_trunc('WEEK', timestamp),
                case when substr(clustername, 1, 4) = 'job-' then 'job-XXXX' else clustername end
        ),
    t2 as ( select *,
                rank() over (partition by dt order by machinehours desc) rnk
            from t1 )
select to_date(dt),
    case when rnk <= 10 then cn else 'other' end clustername,
    sum(machinehours) * 0.05 cost
from t2
where dt > date_add(current_date, -105)
group by dt,
    case when rnk <= 10 then cn else 'other' end
order by dt asc

```

▶ (1) Spark Jobs



```
result_purchase = spark.sql(""SELECT DISTINCT  
customer.name AS name FROM purchase JOIN book ON  
purchase.isbn = book.isbn JOIN customer ON customer.cid  
= purchase.cid WHERE customer.name != 'Harry Smith' AND  
purchase.isbn IN (SELECT purchase.isbn FROM customer  
JOIN purchase ON customer.cid = purchase.cid WHERE  
customer.name = 'Harry Smith')""")
```

```
select distinct customer.name name
from purchase
join book
    on purchase.isbn = book.isbn
join customer
    on customer.cid = purchase.cid
where customer.name != 'Harry Smith'
    and purchase.isbn in ( select purchase.isbn
                            from customer
                            join purchase
                                on customer.cid = purchase.cid
                            where customer.name = 'Harry Smith' )
```

```
val temp = customer.join(purchase, customer("cid")==purchase("cid") )
    .where(customer("name")==="Harry Smith")
    .select(purchase("isbn").as("purchase_isbn"))

val result = purchase.join(book, purchase("isbn")==book("isbn"))
    .join(customer, customer("cid")==purchase("cid"))
    .where(customer("name") != "Harry Smith")
    .join(temp, purchase("isbn")==temp("purchase_isbn"))
    .select(customer("name").as("NAME")).distinct()
```

SPARK SQL, BUT LEARN USING SCALA

**YOU NEED A PROGRAMMING LANGUAGE
FASTER THAN PYTHON
MAKES YOU BETTER AT SPARK**



STRUCTURED STREAMING AND CONTINUOUS APPLICATIONS



STRUCTURED STREAMING

COLUMN1	COLUMN2	COLUMN3

COLUMN1	COLUMN2	COLUMN3

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
1	XYZ	12:00

```
val joinedStream = stream1
    .join(stream2, $"id" === $"id")
```

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
1	XYZ	12:00

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
...
1	XYZ	13:00

```
val stream1W = stream1.withWatermark("TS", "2 hours")
val stream2W = stream2.withWatermark("TS", "2 hours")

val joinedStreamW = stream1W
    .join(stream2W, $"id" === $"id")
```

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
...
1	XYZ	15:00

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
...
1	XYZ	13:00

ID	COL1	COL2	TS
1	ABC		12:00
1		XYZ	15:00

ID	COL1	TS
1	ABC	12:00

ID	COL2	TS
...
1	XYZ	13:00

ID	COL1	COL2	TS

1	ID	COL1	COL2	TS
1	1	ABC	XYZ	15:00

ID	COL1	TS
1	ABC2	16:00

ID	COL2	TS
...
1	XYZ2	17:00

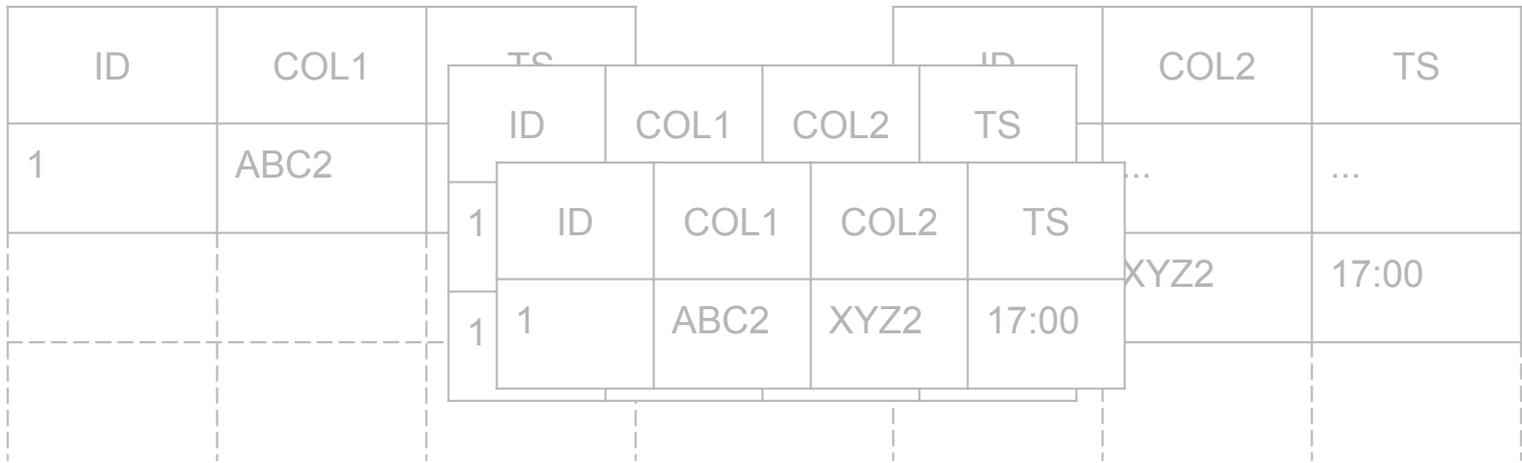
ID	COL1	COL2	TS
1	ABC	XYZ	15:00

ID	COL1	TS	ID	COL2	TS
1	ABC2		
				XYZ2	17:00

ID	COL1	COL2	TS
1	ABC	XYZ	15:00

ID	COL1	TS	ID	COL1	COL2	TS	COL2	TS
1	ABC2					
		1	ID	COL1	COL2	TS		
		1	1	ABC2	XYZ2	17:00	XYZ2	17:00

ID	COL1	COL2	TS
1	ABC	XYZ	15:00



ID	COL1	COL2	TS
1	ABC2	XYZ2	17:00

SURROGATE KEYS IN A DISTRIBUTED WORLD

CUSTOMER_ID	CUST_NUMBER	FIRST_NAME	LAST_NAME
	1234	MICKEY	MOUSE

CUSTOMER_ID	CUST_NUMBER	FIRST_NAME	LAST_NAME
1	1234	MICKEY	MOUSE

CUSTOMER_ID	CUST_NUMBER	FIRST_NAME	LAST_NAME
1	1234	Mickey	Mouse
2	2345	Minnie	Mouse
3	666	George G.	Goof

CUSTOMER_ID	CUST_NUMBER	FIRST_NAME	LAST_NAME
1	1234	Mickey	Mouse
2	2345	Minnie	Mouse
3	666	George G.	Goof



Amazon Lambda



DynamoDB



DynamoDB

RELIABLE PERFORMANCE

HIGHLY AVAILABLE

FULLY MANAGED

SERVERLESS



Amazon Lambda

SERVERLESS COMPUTE

CONTINUOUS SCALING



Amazon Lambda

```
client = boto3.client('dynamodb')
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('customer')

try:
    sk = get_sk(event['cust_number'])
except KeyError:
    sk = new_sk(event['cust_number'])

return {
    'statusCode': 200,
    'body': json.dumps({'customer_id': str(sk)})
}
```

???

Rate today's session

Cyberconflict: A new era of war, sabotage, and fear See passes & pricing

David Sanger (The New York Times)
9:55am-10:10am Wednesday, March 27, 2019
Location: Ballroom
Secondary topics: Security and Privacy

Rate This Session

We're living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you're often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we're uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

David Sanger
The New York Times

David E. Sanger is the national security correspondent for the *New York Times* as well as a national security and political contributor for CNN and a frequent guest on *CBS This Morning*, *Face the Nation*, and many PBS shows.

Session page on conference website

✓ Attending Notes Remove

Cyberconflict: A new era of war, sabotage, and fear

9:55 AM - 10:10 AM, Wed, Mar 27, 2019

Speakers

 David Sanger
National Security Correspondent
The New York Times

📍 Ballroom

Keynotes

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

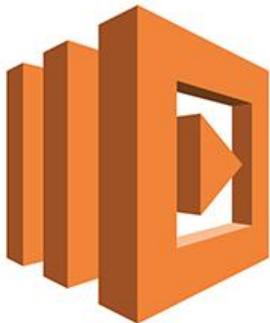
SESSION EVALUATION

O'Reilly Events App

bit.ly/bi2bigdata



APPENDIX



Amazon Lambda

```
import json
import boto3

sk_name = 'customer_id'
nk_name = 'cust_number'
table_name = 'customer'

sk_name_max = sk_name + '_max'

def lambda_handler(event, context):

    def get_sk(nk):
        return table.get_item(Key={nk_name: nk})['Item'][sk_name]

    def get_max_sk(sk_name_max):
        return get_sk(sk_name_max)
```



Amazon Lambda

```
def new_sk(nk):
    sk_max = get_max_sk(sk_name_max)

    sk_nextval = -1

    # Getting the next available customer_id value
    while sk_nextval == -1:
        sk_max = sk_max + 1
        try:
            table.put_item(
                Item = {
                    nk_config: sk_name_max,
                    sk_name: sk_max
                },
                ConditionExpression = sk_name + "< :new_max",
                ExpressionAttributeValues = {
                    ':new_max': sk_max
                }
            )
            sk_nextval = sk_max
        except ClientError as e:
            if e.response["Error"]["Code"] == 'ConditionalCheckFailedException':
                pass
            else:
                raise
```



Amazon Lambda

```
# Writing the new couple (natural key, surrogate key)
#
# There will be an exception in case another process already stored
# the same natural key value
try:
    table.put_item(
        Item = {
            nk_name: nk,
            sk_name: sk_nextval
        },
        ConditionExpression = 'attribute_not_exists(' + nk_name + ')',
    )
except ClientError as e:
    if e.response['Error']['Code'] == 'ConditionalCheckFailedException':
        pass
    else:
        raise

return get_sk(nk)

client = boto3.client('dynamodb')
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table(table_name)

try:
    sk = get_sk(event[nk_name])
except KeyError:
    sk = new_sk(event[nk_name])

return {
    'statusCode': 200,
    'body': json.dumps({'sk_name': str(sk)})
}
```

DISCLAIMER

This presentation and its contents are strictly confidential. It may not, in whole or in part, be reproduced, redistributed, published or passed on to any other person by the recipient.

The information in this presentation has not been independently verified. No representation or warranty, express or implied, is made as to the accuracy or completeness of the presentation and the information contained herein and no reliance should be placed on such information. No responsibility is accepted for any liability for any loss howsoever arising, directly or indirectly, from this presentation or its contents.



 zalando



FRANCESCO MUCIO



@mucio

linkedin.com/in/mucio



francesco.mucio@zalando.com



28-03-2019

