data lab

Finding the spark as a SQL data warehouse developer

Comparison

Thank you sponsors!



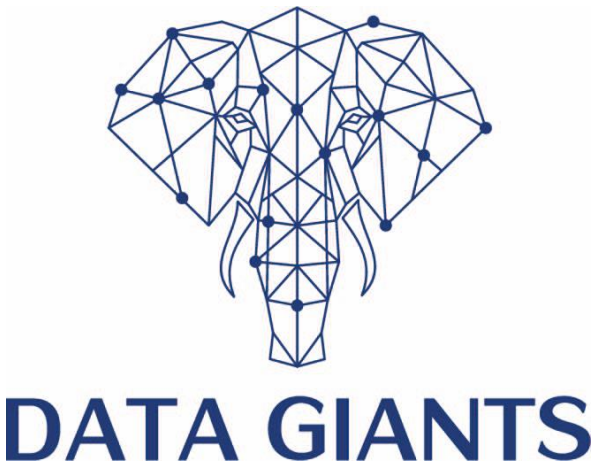
























Who am I

Ásgeir Gunnarsson

Data Platform MVP

Partner, Data Platform Architect @ data lab

20+ years in BI
Co-organizer of Power BI Next Step, Data Platform Next Step
Co-author of Pro Microsoft Power BI Administration

E-mail: asgeir@thesmalldatalab.com
Bluesky: @bidgeir
LinkedIn: linkedin.com/in/asgeirgun



We're The Small Data Lab – a modern Danish data consultancy helping companies access and create real value from data across the entire organisation.

We work with businesses that want more than just dashboards. Our focus is on delivering insight, clear governance, and data platforms that evolve with both people and technology – on site or remotely.

WHY THIS SESSION

- Introduce Spark to SQL people
- Show the possibilities
- Show why it´s not as hard/different/strange as one might think
- Demonstrate that even old dogs can learn new tricks

OLD DOG



North Insights

AGENDA

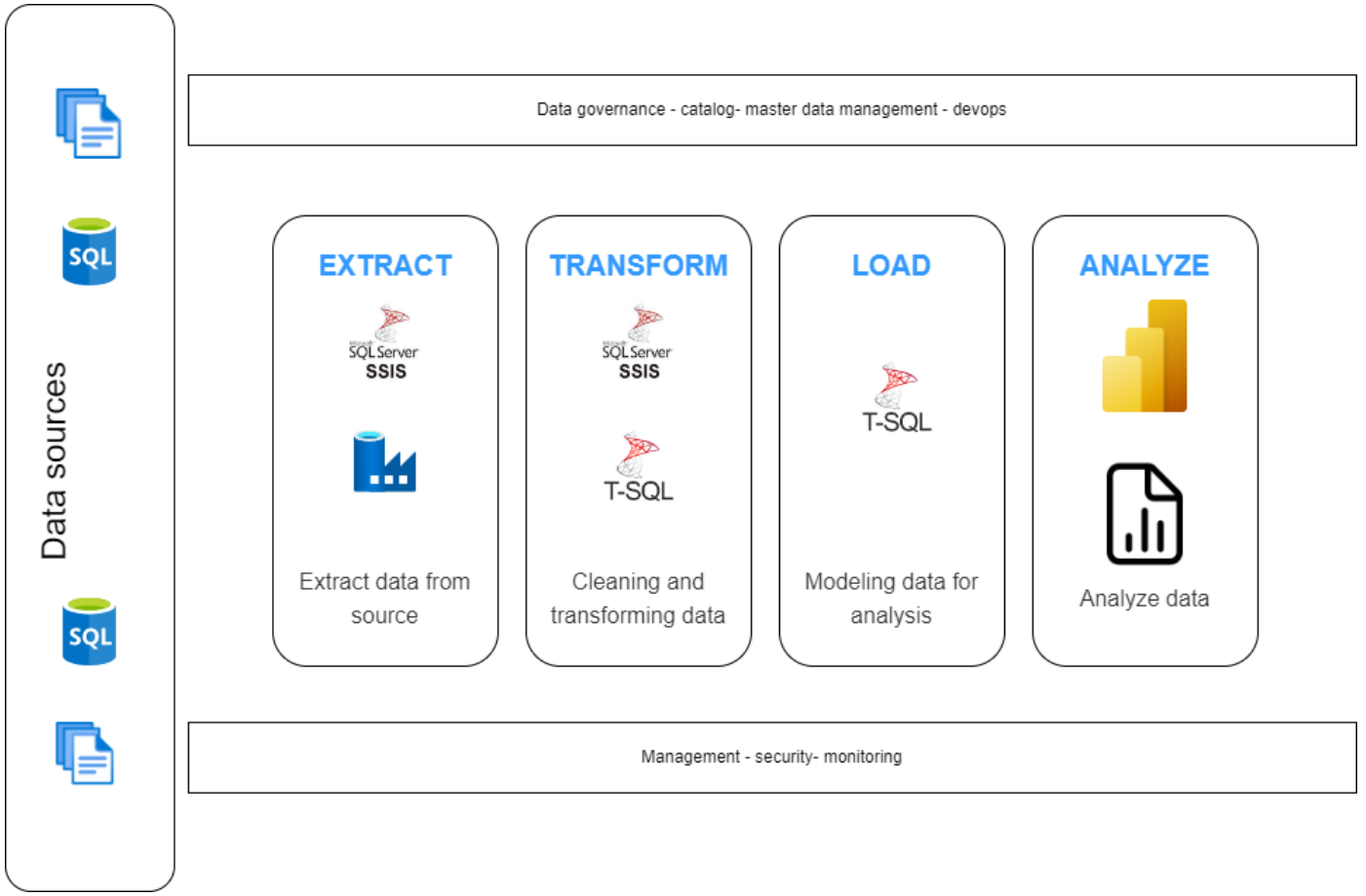
- Set the scene
- What is Spark and Delta
- Why should you care
- Advantages of using Spark compared to SQL
- Why is there about Spark that is less fun

WHO ARE WE

- Been doing data warehousing in SQL Server or Azure SQL for some years
- Comfortable writing SQL
- Do our work in stored procedures
- Bit set in our ways
- Do we want to / need to learn new technologies when there is plenty of SQL work out there?

TYPICAL DATAWAREHOUSE IN SQL

- Data loaded in as is
- Data transformed via stored procedures or SSIS
- Data loaded into the Datawarehouse (schema or database)
- Data mart exposed via views or physical tables



North Insights

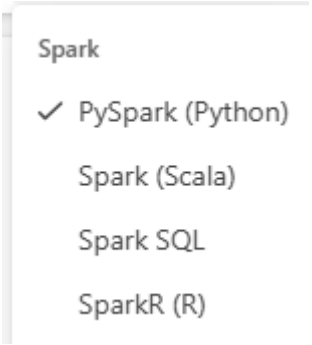
SO WHAT ABOUT SPARK?

WHAT IS SPARK

- Memory-based Massively Parallel Distribution technology
- Built to address weaknesses in Hadoop (all CPU and disk)
- Built by students, researchers and faculty at Berkely University
- Donated to the Apache foundation – open source
- Biggest contributor is Databricks (the some of the inventors of Spark)

WHY WOULD YOU LEARN SPARK

- Flexible technology
- Capable of massive parallel processing
- Separation of compute and storage
- Can use multiple languages (including SQL)
  - Some of them are full programming languages
- Works on Delta file format



DELTA FILE FORMAT

- Foundation of the Data Lakehouse
- Traditionally data lakes have had an issue as they are not ACID compliant
  - A: Atomicity (transactions succeed or fail as a whole)
  - C: Consistency (data committed so there is not many versions of the data)
  - I: Isolation (prevents conflicts between statements)
  - D: Durability (committed changes are permanent)
- Delta fixes that by adding transaction log on top of the Parquet files
- Data lakes can therefore be as reliable as databases

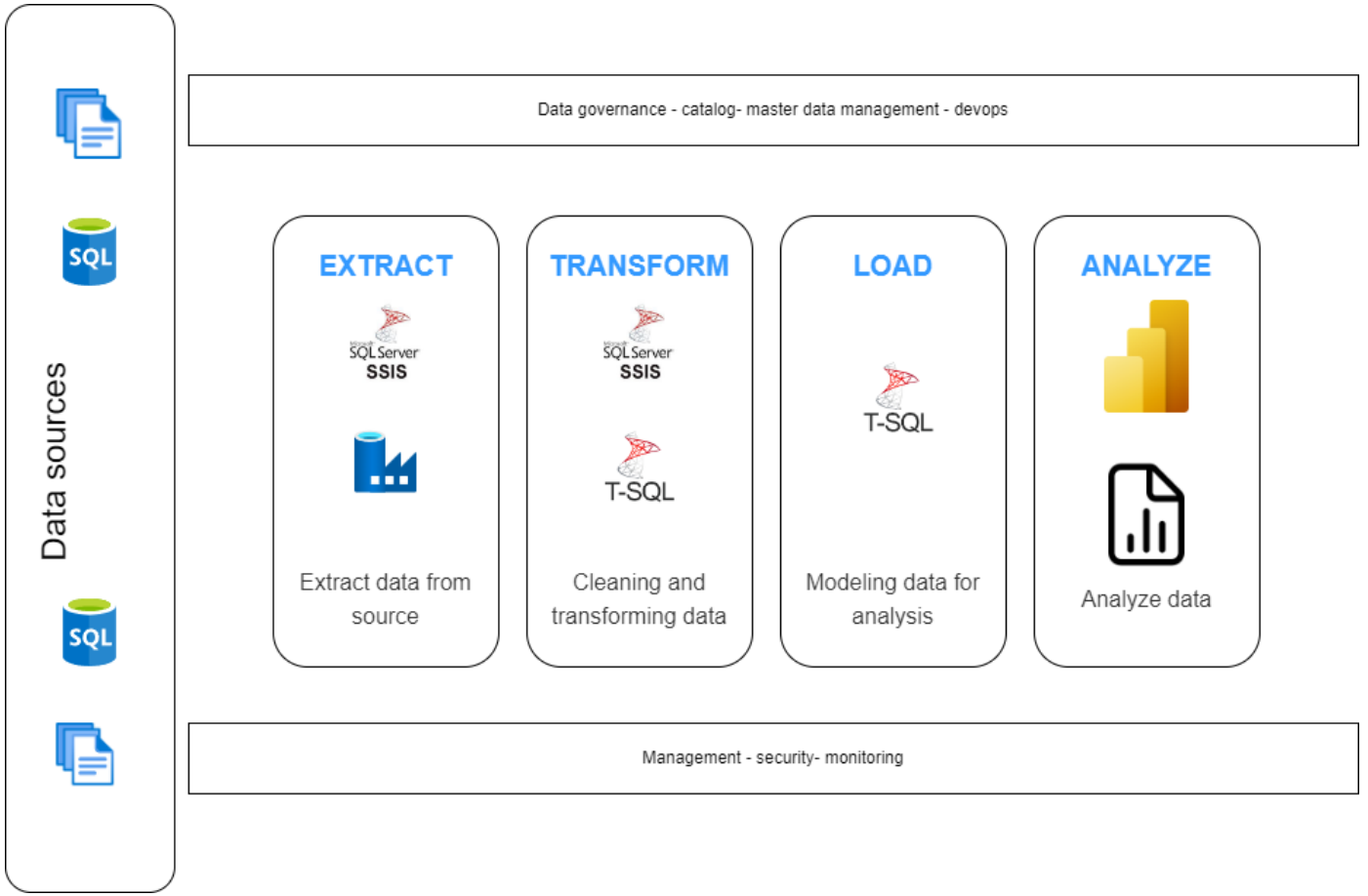
WHAT DO I NEED TO LEARN TO USE SPARK

- The transition is easier than you think
- Lot of SQL skills can be re-used
- The patterns are very similar
- The tools are fairly similar but there is an upside
- Benefit if you know Python
- Need to learn Spark basics

RE-CAP

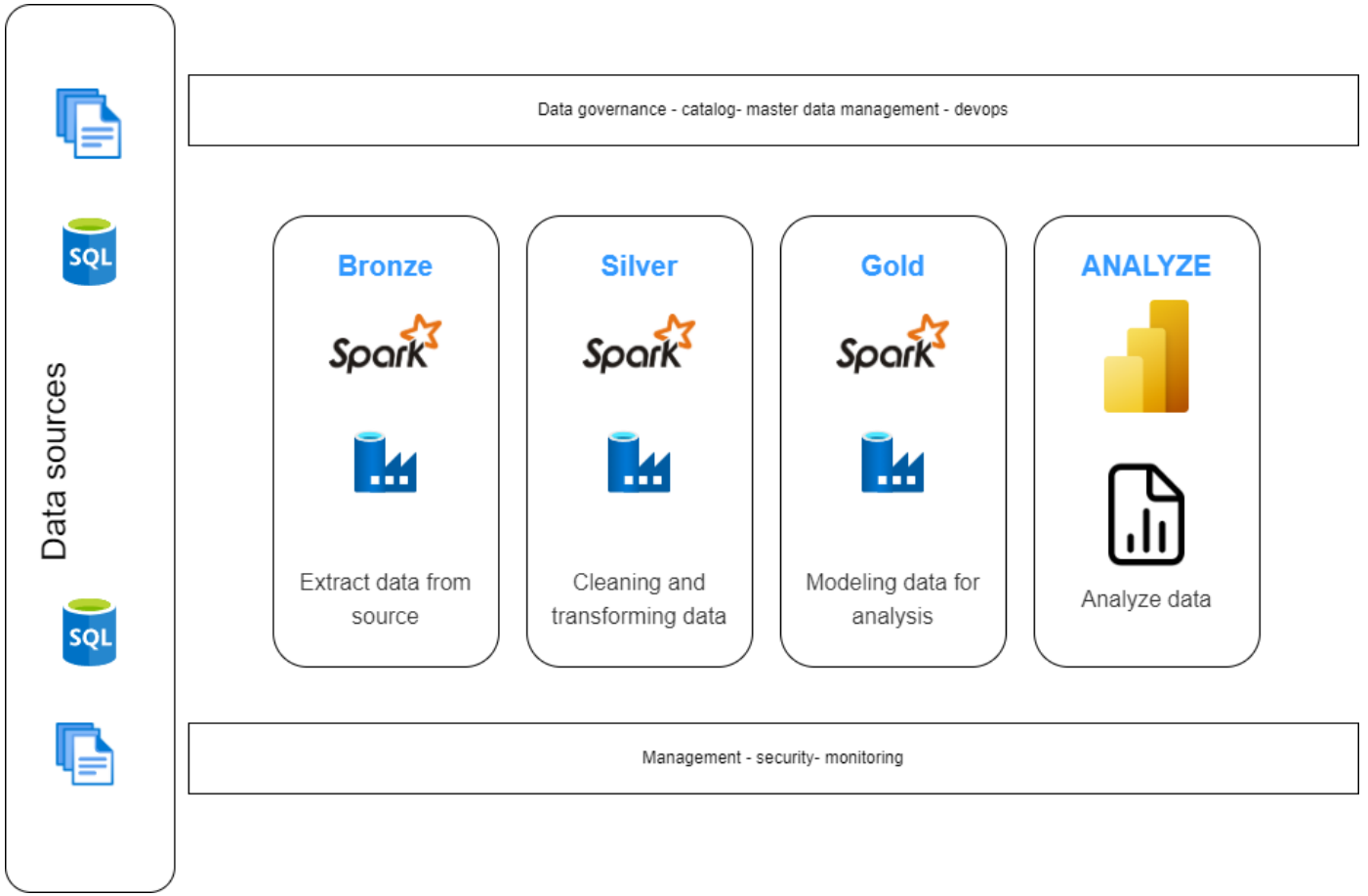
TYPICAL DATAWAREHOUSE IN SQL

- Data loaded in as is
- Data transformed via stored procedures or SSIS
- Data loaded into the Datawarehouse (schema or database)
- Data mart exposed via views or physical tables



DATA LAKEHOUSE

- Data loaded in as is (Bronze)
- Data transformed via Spark (Silver)
- Data loaded into the Lakehouse via Spark (Gold)
- Data mart exposed via SQL endpoints or delta tables



TOOLS

Data warehouse in SQL

- SSMS / VS Code / Visual Studio
  - T-SQL

Data lakehouse in Spark

- Notebooks (Browser or VS Code)
  - SQL
  - Python
  - Scala
  - R

DEMO NOTEBOOK

TOOLS

- T-SQL can be complex to debug if it needs to be dynamic or you have a large script
- Spark has a lazy executor
  - You can write code in multiple cells and run them continuously or independently
  - Great for debugging
  - Beware that code might not be executed in chronological order
- Notebooks make it easy to comment and explain the code
- You can inject parameters into notebooks
- Can visualize data in notebooks

DEMO LAZY EXECUTOR

North Insights

IN GENERAL, SPARK SQL AND T-SQL ARE RELATIVELY SIMILAR, DATE QUERIES ARE THE MOST COMMON DIFFERENCE. WINDOW FUNCTIONS ARE THE SAME IN SPARK SQL. UNPIVOT AND PIVOT ARE DIFFERENT IN SPARK SQL TO T-SQL

HTTPS://PLAINLYRESULTS.COM/BLOG/SPARK-SQL-VS-T-SQL-DATE-QUERY-DIFFERENCES/#:~:TEXT=IN%20GENERAL%2C%20SPARK%20SQL%20AND,SPARK%20SQL%20TO%20T%2DSQL.

T-SQL QUERIES VS SPARK SQL QUERIES – JOURNEY WITH DATA STUFFS BY ARULMOUZHI

EXAMPLES OF BENEFITS

- Meta data driven architecture (source to landing/ODS)
  - Auto create if not exists (dfChannel.write.format("delta").mode("overwrite").saveAsTable("ProductCategoryChannel")
- Schema drift
  - Schema drift is built in - As simple as a parameter in creating/overwriting a table
  - (dfChannel.write.format("delta").mode("overwrite").option(
  - "mergeSchema", "true").saveAsTable("ProductCategoryChannel")
  - SET spark.databricks.delta.schema.autoMerge.enabled = TRUE
  - (dfChannel.write.format("delta").mode("overwrite").option(
  - “overwriteSchema", "true").saveAsTable("ProductCategoryChannel")
- Work with semi- and unstructured data
- Merge statements
- Modularization
- Packages for common data tasks such as data cleaning

MERGE STATEMENTS

- Can be made much simpler than t-sql
- Can add conditions to all merge types
- Often you don´t need to explicitly mention column names (think about dynamic code)

```
dmDeltaTable.alias('target') \
  .merge(sourceDF.alias('source'),
"source.AddressID = target.AddressID") \
  .whenMatchedUpdateAll() \
  .whenNotMatchedInsertAll() \
  .whenNotMatchedBySourceDelete() \
  .execute()
```

```
MERGE WITH SCHEMA EVOLUTION INTO target USING
source
  ON source.key = target.key
  WHEN MATCHED THEN UPDATE SET *
  WHEN NOT MATCHED THEN INSERT *
  WHEN NOT MATCHED BY SOURCE THEN DELETE
```

MERGE STATEMENTS

- Can be made much simpler than t-sql
- Can add conditions to all merge types
- Often you don´t need to explicitly mention column names (think about dynamic code)

```
dmDeltaTable.alias('target') \
                .merge(
                    df.alias('source'),
                    "tbl.row_sha2_PK = updates.row_sha2_PK"
            ) \
            .whenMatchedUpdateAll(\
                "tbl.DW_IsCurrent = 1 AND tbl.row_sha2_scd1 <> updates.row_sha2_scd1"
            )\
            .whenNotMatchedInsertAll()\
            .execute()
```

DEMO REAL LIFE MERGE STATEMENT

MODULARIZATION

- Easy to create functions for common tasks
- Functions can be in the same or another notebook
- Reference notebooks and use all their functions

PACKAGES

- Python and R have a lot of packages you can use
- A lot of common tasks have packages
  - Cleaning data
  - Calling REST APIs
- Possible to create your own packages

DEMO SEND MAIL PACKAGE

WHAT ABOUT DRAWBACKS

- Certain things are different
- No concept of identity column (well not yet in Fabric)
  - Hard to parallelize
- Different way to tune
- Different way to debug
- If you cannot use the starter pool you have to wait for the cluster to start up (3-5 minutes)
- If most of your data is “small” you might need to spend time on cluster configurations to not over allocate resources

QUESTIONS

Rate This session

Find the Spark as a SQL data warehouse developer

