Tyler Guggenberger
Data Engineer

NHL Summer Analytics Challenge:

Implementing an Enterprise Data Warehouse

Proposed Service Offering -

Assist with the development and implementation of an Enterprise Data Warehouse for the Generic Hockey Club.

It should be noted that while most of the analysis below is directed towards the Hockey Operations Department, in practice, the EDW would encompass all facets of the organization.

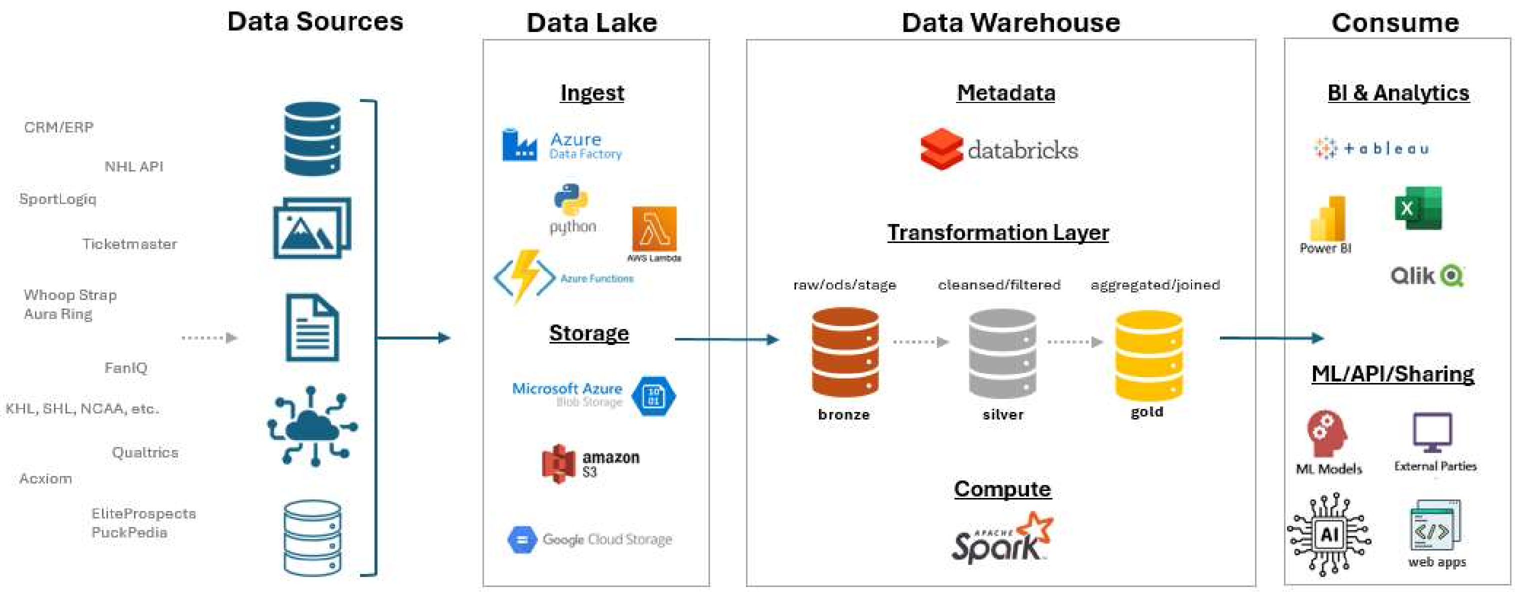
Value Add -

Investing in an EDW will enable more informed decision making, increased automation, sharper insights and greater overall capabilities; conversely, the absence of a well-architected data warehouse will result in issues around scalability, consistency, access, integrity and siloing. A robust, clean and systematic Data Warehouse is a critical foundation for successful Analytics, Data Science and Business Intelligence.

In the professional sports industry, where parity is tight and competitive edges are slim, I believe having a market-leading EDW can be a differentiator for the Generic Hockey Club.

High Level Architecture -

Below is a sample dataflow that I've designed to give us a reference point on what this could look like. Given the broad number of tools and products available to Data Engineers, this is by no means a comprehensive list and can be easily adjusted to accommodate existing systems and processes. Notes on each phase are found below.



Data Sources:

- Both Internal and External – includes APIs, FTP, SQL/NoSQL DBs, Cloud Storage Locations, On Prem Storage Locations, Email and any other means of data transfer.

Data Lake:

- Ingestion to Data Lake is streamlined to accommodate the various data sources. Cloud Solution Providers offer products such as Azure Data Factory, AWS Lamba Functions or Glue jobs, Apache Kafka, etc., while traditional On-Prem solutions may utilize SQL Server Integration Services. Regardless, a systematic approach is essential to keeping it a Data Lake and not a "data swamp."

- Cloud Storage offers a cost-effective means to house our data. The data lake contains structured, semi-structured, and unstructured data, so file formats will include Parquet, JSON, Avro, CSV, ORC, delimited TXT, etc. Schema will be defined and/or inferred in the Data Warehouse layer.

Data Warehouse:

- I've depicted this using Databricks, where I've adopted their multi-hop or 'medallion' architecture. Databricks catalogs the Metadata and lays over the Data Lake, building Delta Tables. This can be thought of as ELT vs. ETL, since all transformations are handled in the Data Warehouse.

- Data is landed into a 'bronze' layer that can be regarded as our raw or ODS tables – essential for restart/recovery purposes. The 'silver' layer is our cleansed, filtered and transformed data. And finally, the 'gold' layer is our aggregated and joined data – you may think of these as our business level data marts.

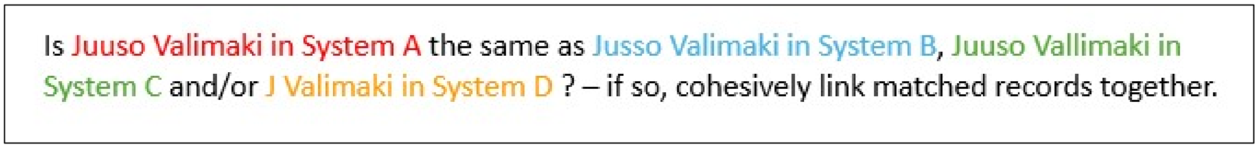
- All compute is handled via Apache Spark and file formats have been standardized into Delta Table format.

Consume:

- This phase represents the "customer" or reporting layer. It can be regarded as a one-stop shop for Analysts, Data Scientists, Applications, Dashboards, ML Models, Generative AI, APIs, Secure Data Sharing, etc.

Example Use Case – Player Lineage and Master Data Management (MDM)

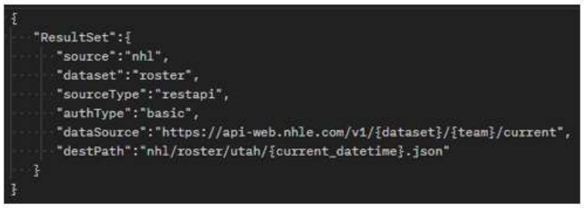
Problem Statement: Player data is derived from several different source systems, but there is no universal identifier on which to key. As a result, duplicate player records exist, data is inconsistent, joins across datasets are difficult and overall player reporting lacks trust and transparency.



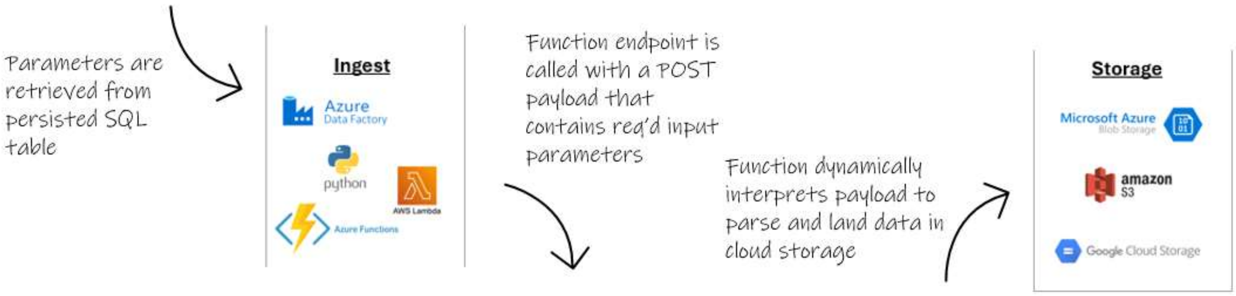
Is Juuso Valimaki in System A the same as Jusso Valimaki in System B, Juuso Vallimaki in System C and/or J Valimaki in System D ? – if so, cohesively link matched records together.

Solution Overview: Deploy scalable and reusable code to fetch data, ingest to our EDW, standardize the player record(s) into a 'Master' Person table to serve as our 'golden identifier'. Example snippets are listed below.

Fetch: This step represents landing data into the Data Lake. Here, there is a function that can be called via a POST API request, which dynamically executes certain code blocks based on the inbound 'source' type (FTP, API, Cloud Storage, etc.) – in short, we have one re-usable fetch process to retrieve player-related data from all different sources and land in our cloud storage account.



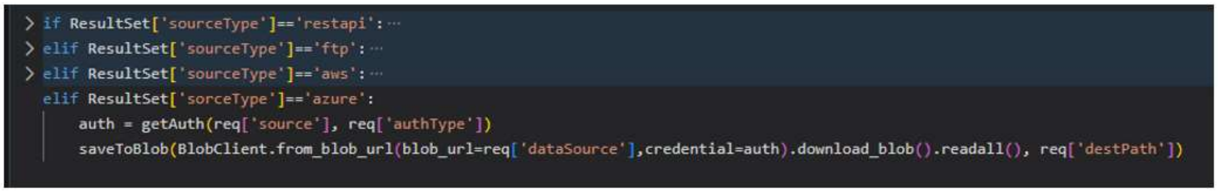
```
{
  "ResultSet":{
    "source":"nhl",
    "dataset":"roster",
    "sourceType":"restapi",
    "authType":"basic",
    "dataSource":"https://api-web.nhle.com/v1/{dataset}/{team}/current",
    "destPath":"nhl/roster/utah/{current_datetime}.json"
  }
}
```



Parameters are retrieved from persisted SQL table

Function endpoint is called with a POST payload that contains req'd input parameters

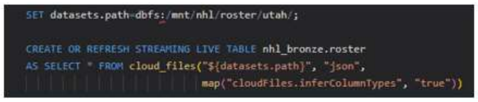
Function dynamically interprets payload to parse and land data in cloud storage



```
> if ResultSet['sourceType']=='restapi':···
> elif ResultSet['sourceType']=='ftp':···
> elif ResultSet['sourceType']=='aws':···
  elif ResultSet['sorceType']=='azure':
      auth = getAuth(req['source'], req['authType'])
      saveToBlob(BlobClient.from_blob_url(blob_url=req['dataSource'],credential=auth).download_blob().readall(), req['destPath'])
```

Transform: This step represents the ETL from Cloud Storage to our Data Warehouse. Here, Player Data is normalized and loaded to bronze tables. At this stage, player-related data is continuously arriving to many different source tables with various Source IDs and standard data cleansing is performed (bronze to silver).

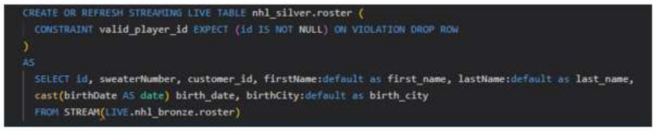
Bronze table(s) load from Cloud Storage



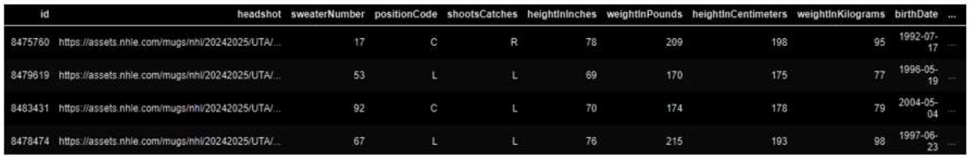
```
SET datasets.path=dbfs:/mnt/nhl/roster/utah/;

CREATE OR REFRESH STREAMING LIVE TABLE nhl_bronze.roster
AS SELECT * FROM cloud_files("${datasets.path}", "json",
                             map("cloudFiles.inferColumnTypes", "true"))
```

File formats are normalized / standardized

Silver tables read and cascade from bronze to cleansed data



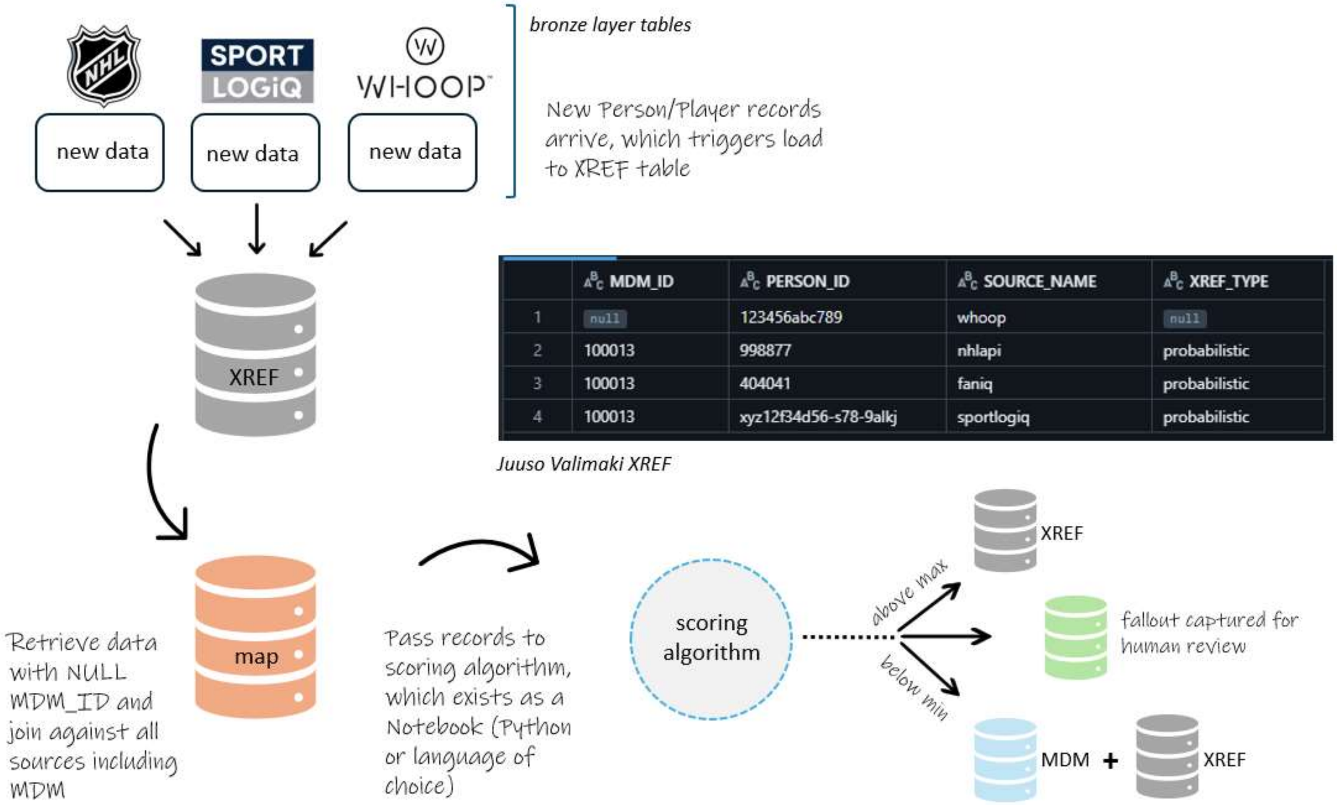
```
CREATE OR REFRESH STREAMING LIVE TABLE nhl_silver.roster (
  CONSTRAINT valid_player_id EXPECT (id IS NOT NULL) ON VIOLATION DROP ROW
)
AS
  SELECT id, sweaterNumber, customer_id, firstName:default as first_name, lastName:default as last_name,
  cast(birthDate AS date) birth_date, birthCity:default as birth_city
  FROM STREAM(LIVE.nhl_bronze.roster)
```



id	headshot	sweaterNumber	positionCode	shootsCatches	heightInInches	weightInPounds	heightInCentimeters	weightInKilograms	birthDate	...
8475760	https://assets.nhle.com/mugs/nhl/20242025/UTA/...	17	C	R	78	209	198	95	1992-07-17	...
8479619	https://assets.nhle.com/mugs/nhl/20242025/UTA/...	53	L	L	69	170	175	77	1996-05-19	...
8483431	https://assets.nhle.com/mugs/nhl/20242025/UTA/...	92	C	L	70	174	178	79	2004-05-04	...
8478474	https://assets.nhle.com/mugs/nhl/20242025/UTA/...	67	L	L	76	215	193	98	1997-06-23	...

Model: This demonstrates the transformation from Silver to Gold (or cleansed to modeled data mart). When new Player data arrives and does not exist in our prebuilt XREF table, it gets inserted with a NULL MDM_ID. These new cleansed records are then passed into a scoring algorithm that attributes a probabilistic weight for record matching purposes.

Above threshold records are linked to existing MDM record. Below threshold records determined as new are inserted. All others can be reviewed manually.



bronze layer tables

new data

new data

new data

New Person/Player records arrive, which triggers load to XREF table

XREF

	MDM_ID	PERSON_ID	SOURCE_NAME	XREF_TYPE
1	null	123456abc789	whoop	null
2	100013	998877	nhlapi	probabilistic
3	100013	404041	faniq	probabilistic
4	100013	xyz12f34d56-s78-9alkj	sportlogiq	probabilistic

Juuso Valimaki XREF

map

Retrieve data with NULL MDM_ID and join against all sources including MDM

Pass records to scoring algorithm, which exists as a Notebook (Python or language of choice)

scoring algorithm

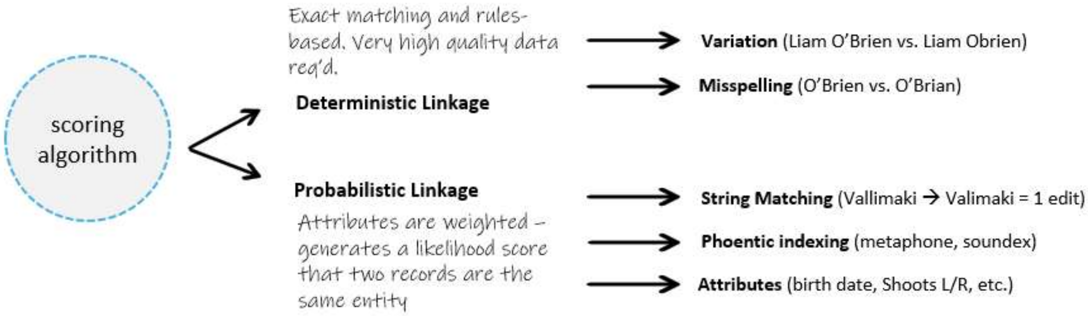
above max

below min

XREF

fallout captured for human review

MDM + XREF



scoring algorithm

Exact matching and rules-based. Very high quality data req'd.

Deterministic Linkage

Variation (Liam O'Brien vs. Liam Obrien)

Misspelling (O'Brien vs. O'Brian)

Probabilistic Linkage

Attributes are weighted – generates a likelihood score that two records are the same entity

String Matching (Vallimaki → Valimaki = 1 edit)

Phoentic indexing (metaphone, soundex)

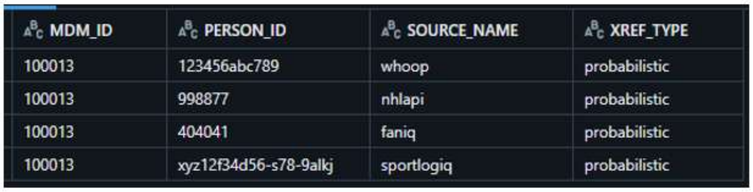
Attributes (birth date, Shoots L/R, etc.)

Output/Usability: And finally, we have our Master Person table. One unique record per person that bridges between sources and aggregates data into one object. This unified object ensures consistency and reliability across applications, making cross-source analysis seamless.

XREF table updated with appropriate MDM_ID for previously rogue record

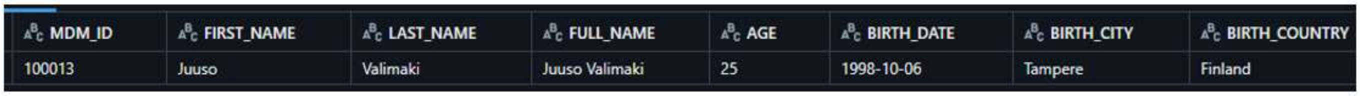
Unified master record bridges systems and offers a reliable key on which join

XREF Table



MDM_ID	PERSON_ID	SOURCE_NAME	XREF_TYPE
100013	123456abc789	whoop	probabilistic
100013	998877	nhlapi	probabilistic
100013	404041	faniq	probabilistic
100013	xyz12f34d56-s78-9alkj	sportlogiq	probabilistic

MDM Table



MDM_ID	FIRST_NAME	LAST_NAME	FULL_NAME	AGE	BIRTH_DATE	BIRTH_CITY	BIRTH_COUNTRY
100013	Juuso	Valimaki	Juuso Valimaki	25	1998-10-06	Tampere	Finland

Closing Remarks

The use-case outlined above is just one, specific application of how an EDW can help solve a business problem. In practice, this EDW would see your organization ingest, govern and intimately understand everything from merchandise and concession data to on-ice player tracking, player biometrics, CBA information, amateur draft models, etc.

That is to say, the applications of an EDW far-exceed the scope of what I've outlined here.

And lastly, I'd just note that while I understand this submission deviates a bit from the prompt, I wholeheartedly believe that this is a critical building block to achieve what your club is seeking with the essence of the challenge.

Best of luck this season and thank you for your consideration!