databricks

Data Privacy Patterns

Agenda

Data Privacy Patterns

Lesson Name	Lesson Name
Lecture: Store Data Securely	Lecture: Deleting Data in the Lakehouse
ADE 3.1 – Follow Along Demo – PII Lookup Table	ADE 3.4 – Follow Along Demo – Processing Records from CDF
ADE 3.2 – Follow Along Demo – Pseudonymized ETL	ADE 3.5L – Propagating Changes with CDF Lab
ADE 3.3 – Follow Along Demo – Deidentified PII Access	ADE 3.6 – Follow Along Demo: Propagating Deletes with CDF
Lecture: Streaming Data and CDF	

Store Data Securely

Learning Objectives

By the end of this lesson, you should be able to:

1. Identify common use cases for data privacy and describe optimal approaches to meet compliance regulations
2. Secure and handle PII/sensitive data in data pipelines
3. Create Dynamic views to perform data masking and control access to rows and columns

Regulatory Compliance

- EU = GDPR (General Data Protection Regulation)
- US = CCPA (California Consumer Privacy Act)
- Simplified Compliance Requirements
  - Inform customers what personal information is collected
  - Delete, update, or export personal information as requested
  - Process request in a timely fashion (30 days)

Regulatory Compliance

How Lakehouse Simplifies Compliance

- Reduce copies of your PII
- Find personal information quickly
- Reliably change, delete, or export data
- Built-in data skipping optimizations (**Z-order**) and housekeeping of obsolete/deleted data (**VACUUM**)
- Use transaction logs for auditing

Manage Access to PII

- Control access to storage locations with cloud permissions
- Limit human access to raw data
- Pseudonymize records on ingestion
- Use table ACLs to manage user permissions
- Configure dynamic views for data redaction
- Remove identifying details from demographic views

PII Data Security

Two main data modeling approaches to meet compliance requirements

Pseudonymization

- Protects data at record level
- Re-identification is possible
- Pseudonymised data is still considered PII

Name	John Doe
B_Date	14/04/1987

Name	**User-321**
B_Date	14/04/1987

Anonymization

- Protects entire dataset
- Irreversibly altered
- Non-linkable to original data
- Multiple anonymization methods might be used

Name	John Doe
B_Date	14/04/1987

Name	**********
Age	**20-30**

Pseudonymization

Overview of the approach

- Switches original data point with pseudonym for later **re-identification**
- Only authorized users will have access to keys/hash/table for re-identification
- Protects datasets on **record level** for machine learning
- A pseudonym is still considered to be personal data according to the GDPR
- Two main pseudonymization methods: **hashing** and **tokenization**

Pseudonymization

Method: **Hashing**

- Apply SHA or other hash to all PII
- Add random string "salt" to values before hashing
- Databricks secrets can be leveraged for obfuscating salt value
- Leads to some increase in data size
- Some operations will be less efficient

ID	SSN	Salary_R
1	000-11-1111	53K
2	000-22-2222	68K
3	000-33-3333	90K
4	000-44-4444	72K

ID	SSN	Salary_R
1	1ffa0bf4002a968e7d8	53K
2	1d55ec7079cb0a6at0	68K
3	be85b326855e0e748	90K
4	da20058e59fe8d311f	72K

Pseudonymization

Method: **Tokenization**

- Converts all PII to keys
- Values are stored in a secure lookup table
- Slow to write, but fast to read
- De-identified data stored in fewer bytes

ID	SSN	Salary_R
1	000-11-1111	53K
2	000-22-2222	68K
3	000-33-3333	90K
4	000-44-4444	72K

SSN	SSN_Token
000-11-1111	1ffa0bf4002a968e7d8
000-22-2222	[illegible]ec7079cb0a6at0
000-33-3333	[illegible]b326855e0e748
000-44-4444	da20058e59fe8d311f

Token Vault

ID	SSN	Salary_R
1	1ffa0bf4002a968e7d8	53K
2	1d55ec7079cb0a6at0	68K
3	be85b326855e0e748	90K
4	da20058e59fe8d311f	72K

Anonymization

Overview of the approach

- Protects **entire dataset** (tables, databases or entire data catalogues) mostly for Business Intelligence
- Personal data is **irreversibly altered** in such a way that a data subject can no longer be identified directly or indirectly
- Usually a combination of more than one technique used in real-world scenarios
- Two main anonymization methods: **data suppression** and **generalization**

Anonymization Methods

Method: **Data Suppression**

- Exclude columns with PII from views
- Remove rows where demographic groups are too small
- Use dynamic access controls to provide conditional access to full data

Source Table		
ID	SSN	Salary_R
1	000-11-1111	53K
2	000-22-2222	68K
3	000-33-3333	90K

View with no PII	
ID	Salary_R
1	53K
2	68K
3	90K

Anonymization Methods

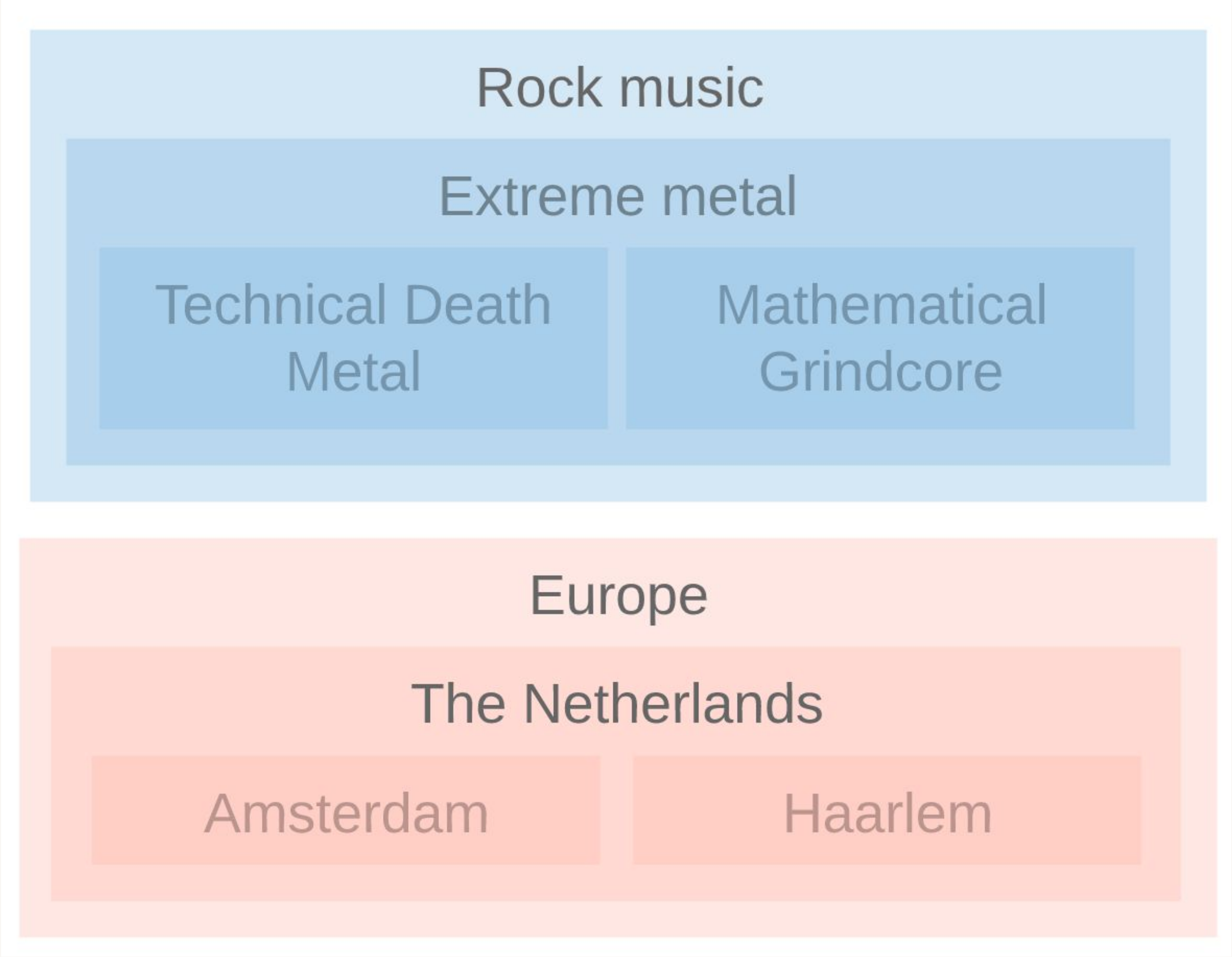
Method: **Generalization**

- Categorical generalization
- Binning
- Truncating IP addresses
- Rounding

Anonymization Methods

Method: Generalization → **Categorical Generalization**

- Removes precision from data
- Move from specific categories to more general
- Retain level of specificity that still provides insight without revealing identity



Anonymization Methods

Method: Generalization → **Binning**

- Identify meaningful divisions in data and group on boundaries
- Allows access to demographic groups without being able to identify individual PII
- Can use domain expertise to identify groups of interest

ID	Department	BirthDate
1	IT	28/09/1997
2	Sales	13/02/1976
3	Marketing	02/04/1985
4	Engineering	19/12/2002

ID	Department	Age_Range
1	IT	20-30
2	Sales	40-50
3	Marketing	30-40
4	Engineering	20-30

Anonymization Methods

Method: Generalization → **Truncating IP addresses**

IP addresses need special anonymization rules;

- Rounding IP address to /24 CIDR
- Replace last byte with 0
- Generalizes IP geolocation to city or neighbourhood level

ID	IP	IP_Truncated
1	10.130.176.215	10.130.176.**0/24**
2	10.5.56.45	10.5.56.**0/24**
3	10.208.126.183	10.208.126.**0/24**
4	10.106.62.87	10.106.62.**0/24**

Anonymization Methods

Method: Generalization → **Rounding**

- Apply generalized rounding rules to all number data, based on required precision for analytics
- Example:
  - Integers are rounded to multiples of 5
  - Values less than 2.5 are rounded to 0 or omitted from reports
  - Consider suppressing outliers

ID	Department	Age_Range	Salary
1	IT	20-30	1245.4
2	Sales	40-50	1300
3	Marketing	30-40	1134

ID	Department	Age_Range	Salary_R
1	IT	20-30	1200
2	Sales	40-50	1300
3	Marketing	30-40	1100

Knowledge Check

Which of the following terms refers to irreversibly altering personal data in such a way that a data subject can no longer be identified directly or indirectly?

Select one response

A. Tokenization
B. Pseudonymization
C. Anonymization
D. Binning

Which of the following is a regulatory compliance program specifically for Europe?

Select one response

A. HIPAA
B. PCI-DSS
C. GDPR
D. CCPA

Which of the following are examples of generalization?

Select two responses

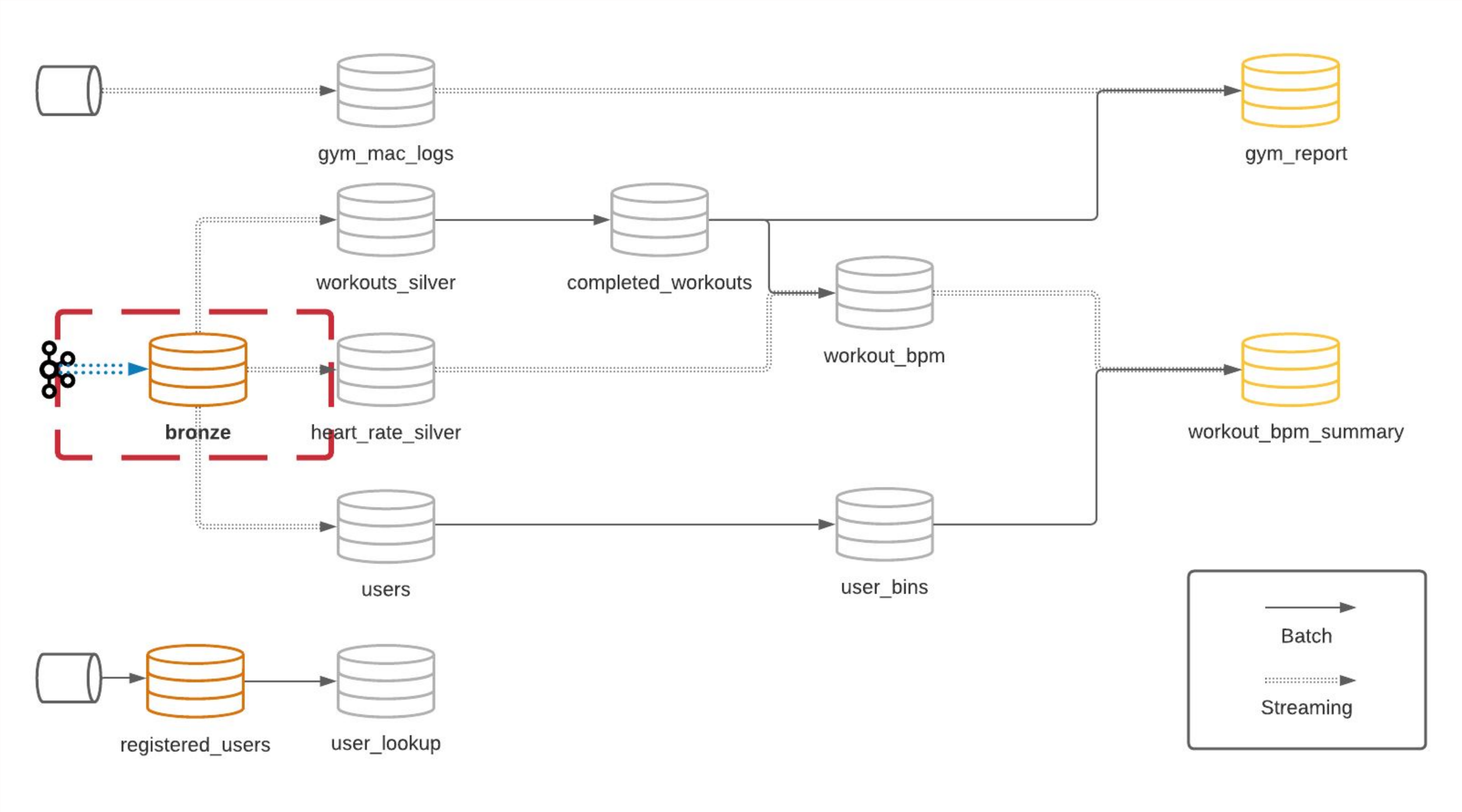
A. Hashing
B. Truncating IP addresses
C. Data suppression
D. Binning

Which of the following can be used to obscure personal information by outputting a string of randomized characters?

Select one response

A. Tokenization
B. Categorical generalization
C. Binning
D. Hashing

Demo: Pseudonymized ETL


gym_mac_logs
gym_report
workouts_silver
completed_workouts
workout_bpm
bronze
heart_rate_silver
workout_bpm_summary
users
user_bins
Batch
Streaming
registered_users
user_lookup

Demo: De-identified PII Access

Streaming Data and CDF

Learning Objectives

By the end of this lesson, you should be able to:

1. Explain how CDF is enabled and how it works
2. Discuss why ingesting from CDF can be beneficial for streaming data
3. Articulate multiple strategies for using Streaming data to create CDF
4. Discuss how CDF addresses past difficulties propagating updates and deletes

Streaming Data and Data Changes

Updates and Deletes in streaming data

- In Structured Streaming, a data stream is treated as a table that is being **continuously appended**. Structured Streaming expects to work with data sources that are **append only**.
- Changes in existing data (updates and deletes) breaks this expectation!
- We need a **deduplication logic** to identify updated and deleted records.

- **Note:** Delta transaction logs tracks files than rows. Updating a single row will point to a new version of the file.

Solution 1: Ignore Changes

Prevent re-processing by ignoring deletes, updates and overwrites

Ignore Deletes

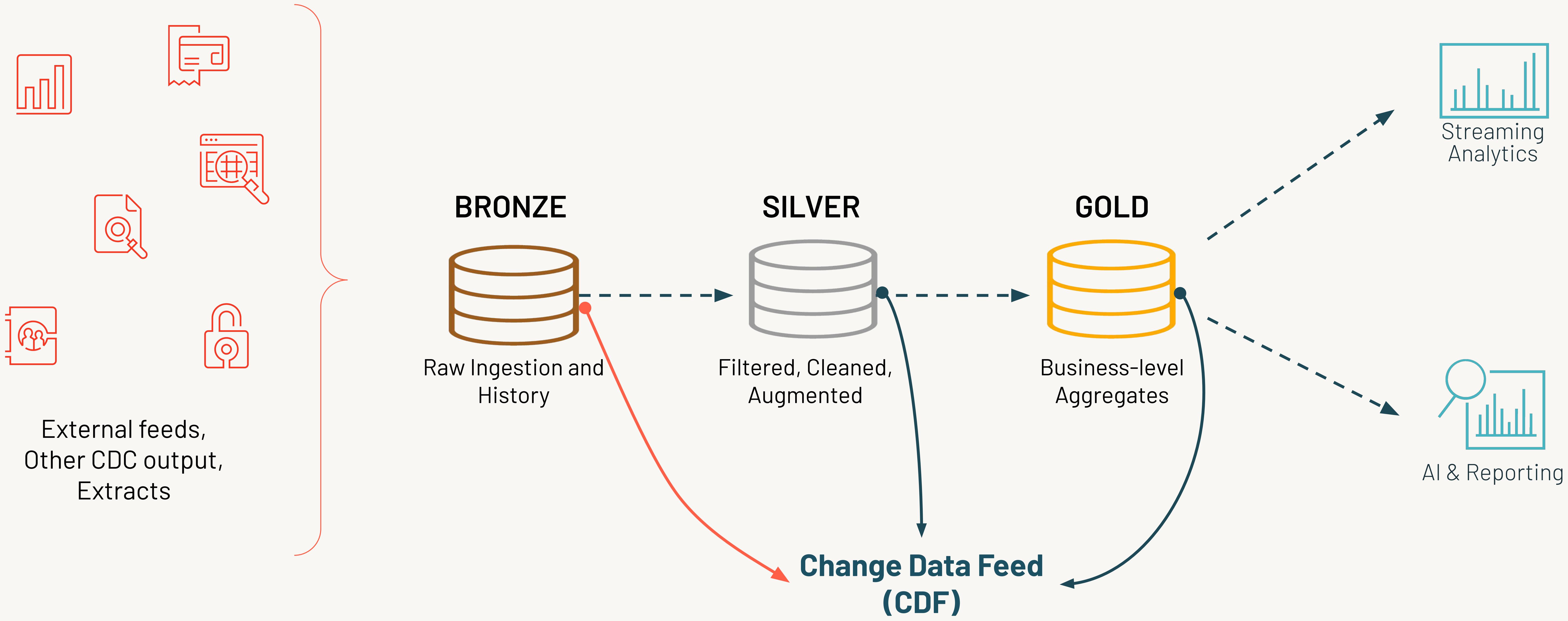
- Ignore transactions that delete data at partition boundaries
- No new data files are written with full partition removal
- `spark.readStream.format("delta")`
  `.option("ignoreDeletes", "true")`

Ignore Changes

- Allows stream to be executed against Delta table with upstream changes
- Must implement logic to avoid processing duplicate records
- Subsumes ignoreDeletes
- `spark.readStream.format("delta")`
  `.option("ignoreChanges", "true")`

Solution 2: Change Data Feeds (CDF)

Propage incremental changes to downstream tables



What Delta Change Data Feed Does for You

Benefits and use cases of CDF

Improve ETL Pipelines

Process less data during ETL to increase efficiency of your pipelines by processing only row-level changes

Unify batch and streaming

Common change format for batch and streaming updates, appends, and deletes

BI on your data lake

Incrementally update the data supporting your BI tool of choice

Meet regulatory needs

Full history available of changes made to the data, including deleted information

How Does Delta CDF Work?

Sample CDF data schema

Original Table (v1)

	PK	B
	A1	B1
	A2	B2
	A3	B3

Change data (Merged as v2)

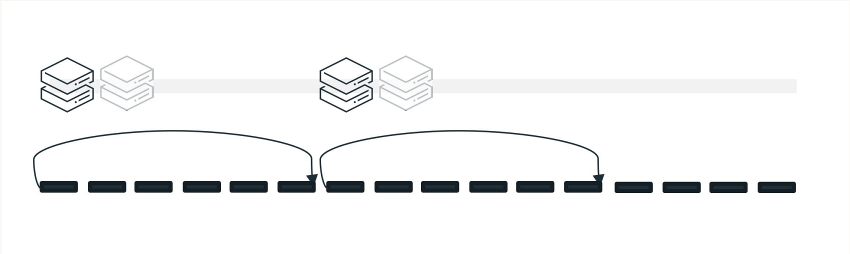
	PK	B
	A1	B1
	A2	Z2
	A3	B3
	A4	B4

Change Data Feed Output

PK	B	Change Type	Time	Version
A2	B2	**Preimage**	12:00:00	2
A2	Z2	**Postimage**	12:00:00	2
A3	B3	Delete	12:00:00	2
A4	B4	Insert	12:00:00	2

A1 record did not receive an update or delete.
So it will not be output by CDF.

Consuming the Delta CDF



Stream - micro batches

12:00

A2	B2	Preimage	12:00:00	2
A2	Z2	Postimage	12:00:00	2
A3	B3	Delete	12:00:00	2
A4	B4	Insert	12:00:00	2

12:08

A5	B5	Insert	12:08:0 0	3

12:10:10

A6	B6	Insert	12:09:00	4
A6	B6	Preimage	12:10:05	5
A6	Z6	Postimage	12:10:05	5

Stream-based Consumption

- Delta Change Feed is processed as each source commit completes
- Rapid source commits can result in multiple Delta versions being included in a single micro-batch

12:00

A5	B5	Insert	12:08:00	3
A6	B6	Insert	12:09:00	4

12:10:0 0

A6	B6	Preimage	12:10:05	5
A6	Z6	Postimag e	12:10:05	5

12:20

Batch - every 10 mins

Batch Consumption

- Batches are constructed based in time-bound windows which may contain multiple Delta versions

CDF Configuration

Important notes for CDF configuration

- CDF is **not** enabled by default. It can be enabled;
  - At table level: `ALTER TABLE myDeltaTable SET TBLPROPERTIES (delta.enableChangeDataFeed = true)`
  - For all new tables: `set spark.databricks.delta.properties.defaults.enableChangeDataFeed = true;`
- Change feed can be read by;
  - Version
  - Timestamp

Demo: Processing Records from CDF

Lab: Propagating Changes with CDF

Learning Objectives

By the end of this lesson, you should be able to:

- Learn how to activate Change Data Feed (CDF) for a specific table.
- Understanding of how to set up a streaming read operation to access and work with the change data generated by CDF.
- Demonstrate how to use SQL's DELETE statement to remove specific records from a table based on criteria and then verify the success of those deletions.
- Ensure data consistency by propagating delete operations from one table to related tables when records are deleted in one table.

Demo

CDF, Data Processing, Deletions, Consistency

- Enable CDF for a specific table using SQL's ALTER TABLE command with the **delta.enableChangeDataFeed** property set to **true**, allowing us to track changes made to the table.
- Configure a streaming read operation on the table with CDF enabled, using options like **'format("delta")'** and enabling continuous monitoring and processing of change data.
- Use SQL's DELETE statement to selectively remove records from the table by specifying the column name and value, aiding data management and cleanup.
- Establish data integrity by creating a temporary view with marked-for-deletion record and execute delete operations on related tables using **MERGE INTO** statements.

Deleting Data in the Lakehouse

Learning Objectives

By the end of this lesson, you should be able to:

1. Explain the use case of CDF for removing data in downstream tables
2. Describe various methods of recording important data changes
3. Discuss how CDF can be leveraged to ensure deletes are committed fully

Data Deletion in the Lakehouse

Data deletion needs special attention!

- Companies need to handle data deletion requests carefully to main compliance with **privacy regulations** such as GDPR and CCPA.
- PII for users need to be effectively and efficiently handled in the Lakehouse, including deletion.
- These operations usually handled in pipelines separate from ETL pipelines.
- CDF data can be used to propagate deletion action to downstream table.

Recording Important Data Changes

Using commit messages

- Delta Lake supports **arbitrary commit messages** that will be recorded to the Delta transaction log and viewable in the table history. This can help with later **auditing**.
- Commit messages can be;
  - Set at global level
  - Can be specified as part of write operation. For example, data insertion can be labeled based on processing type; manual, automated.

Propagating Data Deletion with CDF

How can CDF be used for propagating deletes?

- Data deletion requests can be streamlined with automated triggers using Structured Streaming.
- CDF can be separately leveraged to **identify records** need to be deleted or modified in downstream tables.

Note: When Delta Lake's history and CDF features are implemented, deleted values are still present in older versions of the data.

- We can solve this by **deleting at a partition boundary**.

CDF Retention Policy

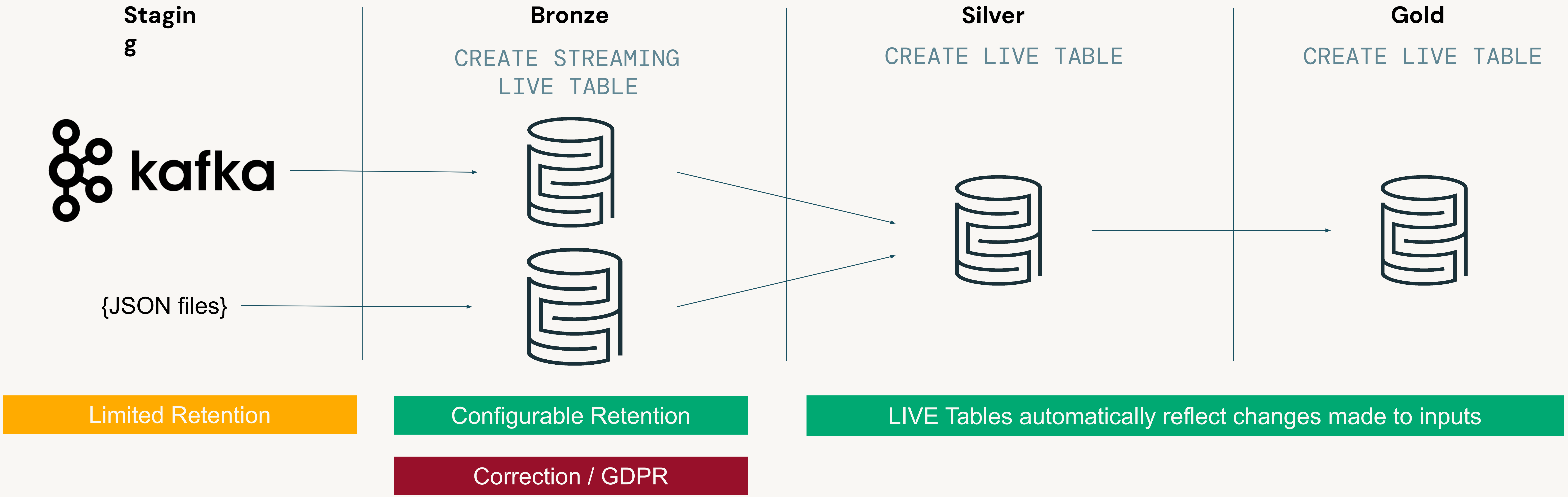
Important notes for CDF configuration

- File deletion will not actually occur until we VACUUM our table!
- CDF records follow the same retention policy of the table. VACUUM command deletes CDF data.
- By default, the Delta engine will prevent VACUUM operations with less than 7 days of retention. To manually run VACUUM for these files;
  - Disable Spark's retention duration check (retentionDurationCheck.enabled)
  - Run VACUUM with DRY RUN to preview files before permanently removing them
  - Run VACUUM with RETAIN 0 HOURS!

Can I perform DML on a Live Table? (i.e. GDPR)

Example GDPR use case

Using streaming live tables for ingestion and live tables after



How can you fix it?

Updates, deletes, inserts and merges on streaming live tables.

Ensure compliance for retention periods on a table.

```
DELETE FROM my_live_tables.records
WHERE date < current_time() - INTERVAL 3 years
```

Scrub PII from data in the lake.

```
UPDATE my_live_tables.records
SET email = hash(email, salt)
```

DML works on streaming live tables only

DML on live tables is undone by the next update

Live Table

```
UPDATE users
SET email = obfuscate(email)
```

```
CREATE OR REPLACE
users AS ….
```

Users

id	email
1	user@gmail.com

Streaming Live Table

```
UPDATE users
SET email = obfuscate(email)
```

```
INSERT INTO users …
```

Users

id	email
1	****@gmail.com
2	****@hotmail...

Demo: Propagating Deletes with CDF

Knowledge Check

Which of the following terms refers to irreversibly altering personal data in such a way that a data subject can no longer be identified directly or indirectly?

Select one response

A. Tokenization
B. Pseudonymization
C. Anonymization
D. Binning

Which of the following is a regulatory compliance program specifically for Europe?

Select one response

A. HIPAA
B. PCI-DSS
C. GDPR
D. CCPA

Which of the following are examples of generalization?

Select two responses

A. Hashing
B. Truncating IP addresses
C. Data suppression
D. Binning

Which of the following can be used to obscure personal information by outputting a string of randomized characters?

Select one response

A. Tokenization
B. Categorical generalization
C. Binning
D. Hashing

Change feed can be read by which of the following?

Select two responses

A. Version
B. Date modified
C. Timestamp
D. Size