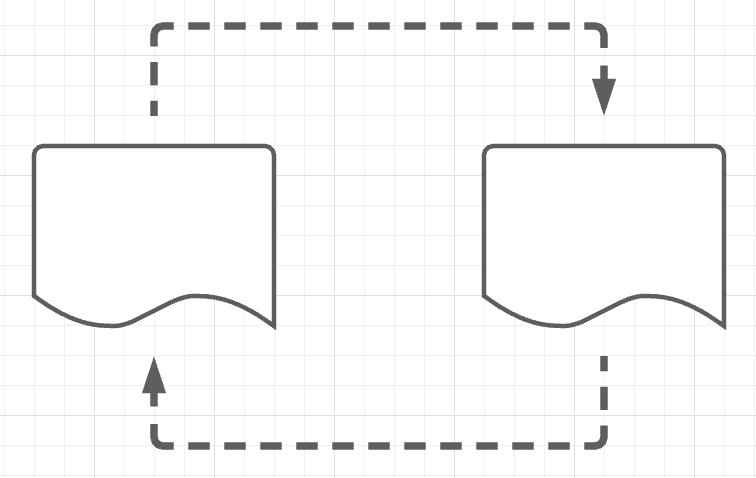
**CDC - Change Data Capture**

CDC is a software design pattern used to determine and track the data that has changed. This is a data integration approach based on which changes are captured and delivered to different consumption systems to take certain action on changed data.

There are several methodologies available to capture a change in the system. In this methodology one system has the data changed from previous point in time where second system needs to perform an action on changed data.

[](file:///Users/nnagaraju/IDEAWorkspace/deltaload/cdc-0.png)

**CDC - ways to capture**

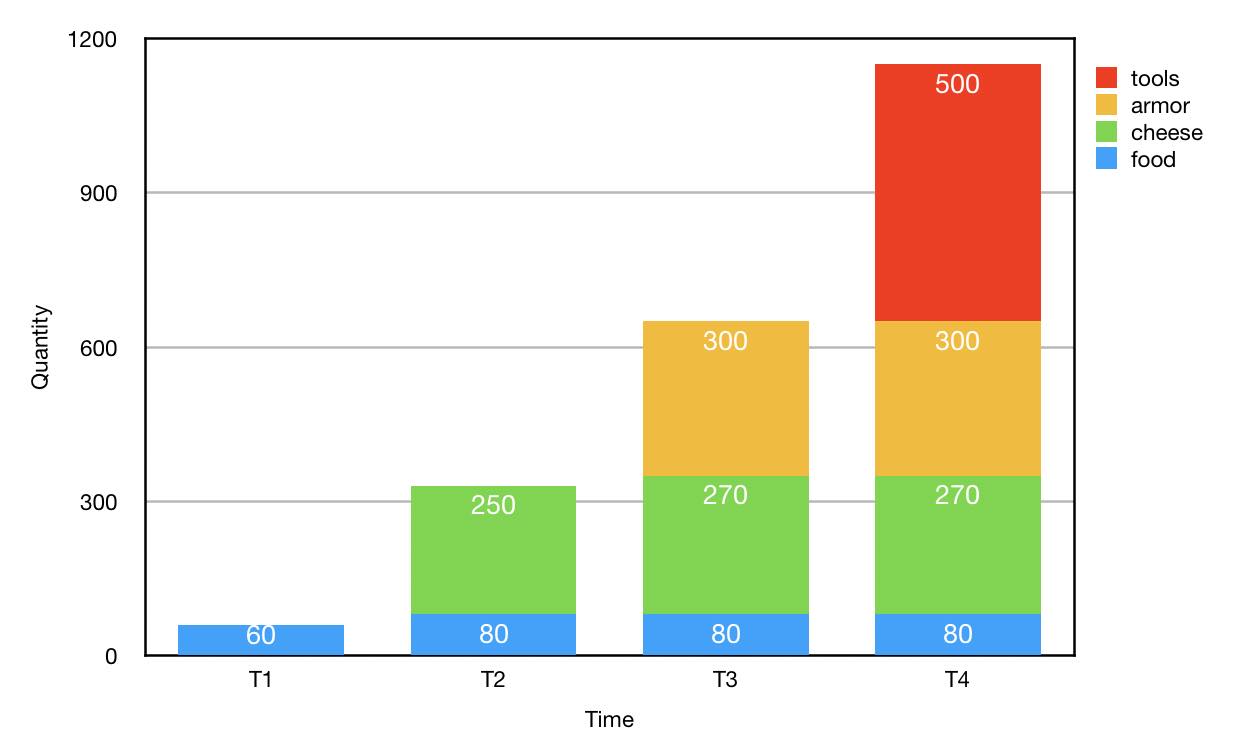
* Timestamps
  + A column such as LAST\_UPDATE, etc. are common.
* Version numbers
  + A column such as VERSION\_NUMBER represents current and previous versions for similar rows
* Status indicators
  + A column such STATUS indicates where a change is significant or not
* Time/Version/Status
  + This combines the above 3 options
* Triggers
  + An event can be fired in reaction to a change then updates another system/table/queue/topic etc
* Event programming
  + Involves a programming for the generated change event to update another system/table/queue or perform some action
* Log scanners
  + Most DBMS systems maintain transaction log of each change that happens to table data. This option provides a non-intrusive approach to capture changes across the database rather than single table.

**Example**

The below example items inventory shows the changes as time moves on.

**Items Inventory:**

The following data shows how items added or removed or updated at given point of time along with the history.

[](file:///Users/nnagaraju/IDEAWorkspace/deltaload/cdc-1.jpg)

**At Time T1, an inventory of food items started wtih below price values**

T1:

{"category": "food", "itemid": "MB-AG-G0", "price": 4.0, "description": "Wine"},

{"category": "food", "itemid": "FI-AG-G08", "price": 1.0, "description": "Bread"},

{"category": "food", "itemid": "BS-AG-G09", "price": 10.0, "description": "Crackers"},

**At Time T2, some of food prices updated and new cheese items added to inventory**

T2:

{"category": "food", "itemid": "FI-AG-G08", "price": 2.0, "description": "Bread"},

{"category": "cheese", "itemid": "SC-MG-G10", "price": 4.0, "description": "Roquefort"},

{"category": "cheese", "itemid": "ST-RF-M04", "price": 4.0, "description": "Manchego"},

{"category": "cheese", "itemid": "SP-FA-R08", "price": 6.0, "description": "Feta"},

{"category": "cheese", "itemid": "SC-MG-G10", "price": 20.0, "description": "Roquefort"},

{"category": "cheese", "itemid": "SC-QT-G12", "price": 35.0, "description": "Camembert"},

**At Time T3, some of cheese prices updated and new tools items added to inventory**

T3:

{"category": "cheese", "itemid": "ST-RF-M04", "price": 6.0, "description": "Manchego"},

{"category": "tools", "itemid": "WC-SH-A01", "price": 90.0, "description": "'Epic Fondue Set'"},

{"category": "tools", "itemid": "CU-PG-G06", "price": 65.0, "description": "'Cheese Board of Glory'"},

{"category": "tools", "itemid": "WC-SH-A02", "price": 9.0, "description": "'The Best Cheese Knife'"},

**At Time T4, just new armor items added to inventory**

T4:

{"category": "armor", "itemid": "DB-SG-G01", "price": 25.0, "description": "'Vegan Friendly Gloves'"},

{"category": "armor", "itemid": "DB-SG-G01", "price": 25.0, "description": "'Vegan Friendly Gloves'"},

{"category": "armor", "itemid": "DC-SG-G02", "price": 30.0, "description": "'Superbly Comfortable Boots'"}

**From the above data, possible questions can be raised**

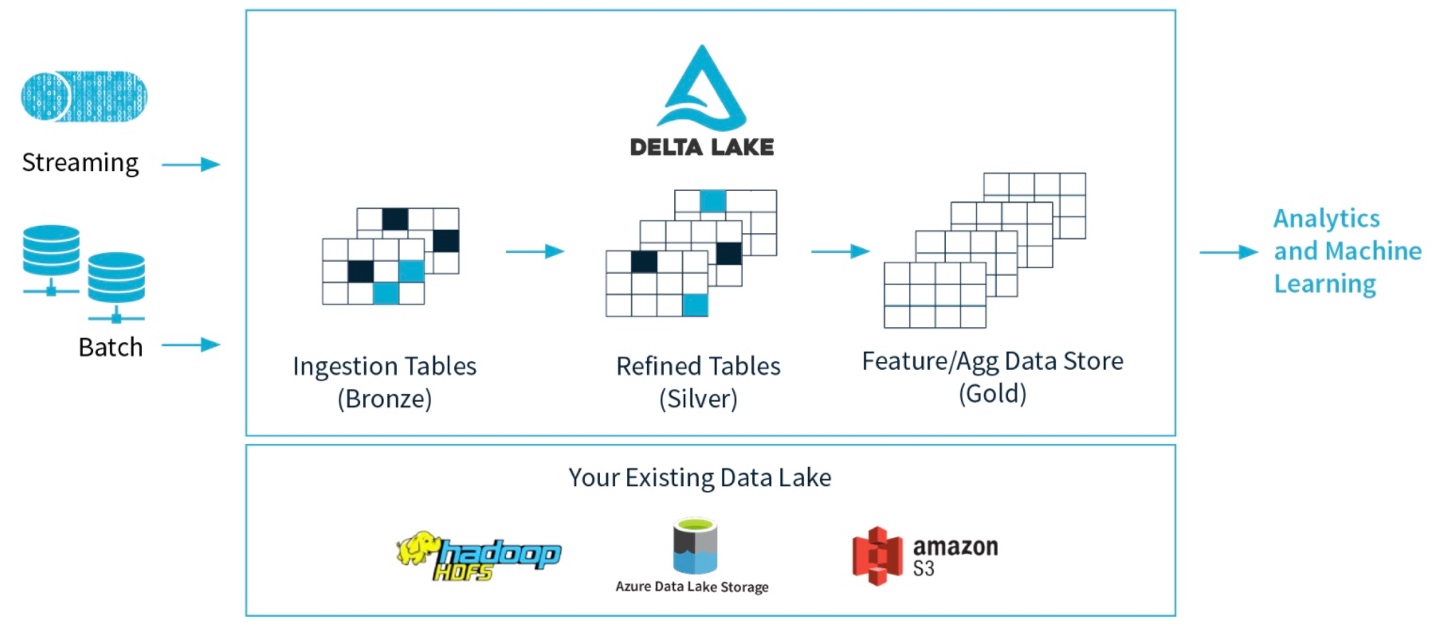
* Is there a system that
  + can show latest inventory
  + can show inventory at a given point-in-time
  + allows corrections to data
  + allows ACID transactions
  + unifies batch and streaming
  + allow stream from prior changes
  + allow check constraints on values
  + capture change data

**Let's see which system handles above scenarios..........**

**Delta Lake and Delta Engine**

**Overview**

Delta Lake is an open source storage layer that provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing. Delta Lake runs on top of existing data lake and is fully compatible with Apache Spark APIs. Automatically handles schema variations to prevent insertion of bad records during ingestion. Data versioning enables rollbacks, full historical audit trails, and reproducible machine learning experiments. Supports merge, update and delete operations. Delta Engine optimizations make Delta Lake operations highly performant, supporting a variety of workloads ranging from large-scale ETL processing to ad-hoc, interactive queries.

[](file:///Users/nnagaraju/IDEAWorkspace/deltaload/delta-0.jpg)

**Delta table**

* Load or modify the data as described in below steps, these steps makes CDC data lands into delta lake table.
  + At T1, create inventory delta lake table with food items
  + At T2, the cheese items are updated/appended into inventory delta lake table
  + At T3, the tools items are updated/appended into inventory delta lake table
  + At T4, the armor items are appended into inventory delta lake table

val path = new java.io.File("./inventory/").getCanonicalPath

// read T1 json file

df = spark.read.json(path)

// create inventory delta lake table

df.write.format("delta").mode(SaveMode.Overwrite).save(path)

val deltaTable = DeltaTable.forPath(spark, path)

// T2 update T1 and append T2 json file

deltaTable.update(condition = expr("category == 'food' AND description == 'Bread' AND itemid == 'FI-AG-G08' AND price == 1.0"), Map("price" -> lit(2.0)))

df.write.format("delta").mode(SaveMode.Append).save(path)

// T3 update T2 and append T3 json file

deltaTable.update(condition = expr("category == 'cheese' AND description == 'Manchego' AND itemid == 'ST-RF-M04' AND price == 4.0"), Map("price" -> lit(6.0)))

df.write.format("delta").mode(SaveMode.Append).save(path)

// T4 append T4 json file

df.write.format("delta").mode(SaveMode.Append).save(path)

* Data after each time delta lake table is updated

// T1: food items

+--------+-----------+---------+-----+

|category|description| itemid|price|

+--------+-----------+---------+-----+

| food| Wine| MB-AG-G0| 4.0|

| food| Bread|FI-AG-G08| 1.0|

| food| Crackers|BS-AG-G09| 10.0|

+--------+-----------+---------+-----+

// T2: updated food item and appended cheese items

+--------+-----------+---------+-----+

|category|description| itemid|price|

+--------+-----------+---------+-----+

| cheese| Roquefort|SC-MG-G10| 4.0|

| cheese| Manchego|ST-RF-M04| 4.0|

| cheese| Feta|SP-FA-R08| 6.0|

| cheese| Camembert|SC-QT-G12| 35.0|

| food| Wine| MB-AG-G0| 4.0|

| food| Bread|FI-AG-G08| 2.0|

| food| Crackers|BS-AG-G09| 10.0|

+--------+-----------+---------+-----+

// T3: update cheese item and appended tools items

+--------+----------------------------+---------+-----+

|category|description |itemid |price|

+--------+----------------------------+---------+-----+

|tools |'Epic Fondue Set' |WC-SH-A01|90.0 |

|tools |'Cheese Board of Glory' |CU-PG-G06|65.0 |

|tools |'The Best Cheese Knife' |WC-SH-A02|9.0 |

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|6.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+----------------------------+---------+-----+

// T4: appended armor items

+--------+----------------------------+---------+-----+

|category|description |itemid |price|

+--------+----------------------------+---------+-----+

|armor |'Vegan Friendly Gloves' |DB-SG-G01|25.0 |

|armor |'Superbly Comfortable Boots'|DC-SG-G02|30.0 |

|tools |'Epic Fondue Set' |WC-SH-A01|90.0 |

|tools |'Cheese Board of Glory' |CU-PG-G06|65.0 |

|tools |'The Best Cheese Knife' |WC-SH-A02|9.0 |

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|6.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+----------------------------+---------+-----+

**Time Travel**

* The below statements to retrieve latest as well as point-in-time snapshots

val path = new java.io.File("./inventory/").getCanonicalPath

spark.sql("CREATE TABLE inventory (category string,description string, itemid string, price double) USING DELTA LOCATION './inventory/'")

//latest

spark.read.format("delta").load(path).show

// At T4 or latest

spark.read.format("delta").option("versionAsOf", 5).load(path).show

// At T3

spark.read.format("delta").option("versionAsOf", 4).load(path).show

// At T2

spark.read.format("delta").option("versionAsOf", 2).load(path).show

// At T1

spark.read.format("delta").option("versionAsOf", 0).load(path).show

**Time travel snapshot data**

//Latest Data

+--------+----------------------------+---------+-----+

|category|description |itemid |price|

+--------+----------------------------+---------+-----+

|armor |'Vegan Friendly Gloves' |DB-SG-G01|25.0 |

|armor |'Superbly Comfortable Boots'|DC-SG-G02|30.0 |

|tools |'Epic Fondue Set' |WC-SH-A01|90.0 |

|tools |'Cheese Board of Glory' |CU-PG-G06|65.0 |

|tools |'The Best Cheese Knife' |WC-SH-A02|9.0 |

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|6.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+----------------------------+---------+-----+

// T4 point-in-time snapshot

+--------+----------------------------+---------+-----+

|category|description |itemid |price|

+--------+----------------------------+---------+-----+

|armor |'Vegan Friendly Gloves' |DB-SG-G01|25.0 |

|armor |'Superbly Comfortable Boots'|DC-SG-G02|30.0 |

|tools |'Epic Fondue Set' |WC-SH-A01|90.0 |

|tools |'Cheese Board of Glory' |CU-PG-G06|65.0 |

|tools |'The Best Cheese Knife' |WC-SH-A02|9.0 |

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|6.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+----------------------------+---------+-----+

// T3 point-in-time snapshot

+--------+-----------------------+---------+-----+

|category|description |itemid |price|

+--------+-----------------------+---------+-----+

|tools |'Epic Fondue Set' |WC-SH-A01|90.0 |

|tools |'Cheese Board of Glory'|CU-PG-G06|65.0 |

|tools |'The Best Cheese Knife'|WC-SH-A02|9.0 |

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|6.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+-----------------------+---------+-----+

// T2 point-in-time snapshot

+--------+-----------+---------+-----+

|category|description|itemid |price|

+--------+-----------+---------+-----+

|cheese |Roquefort |SC-MG-G10|4.0 |

|cheese |Manchego |ST-RF-M04|4.0 |

|cheese |Feta |SP-FA-R08|6.0 |

|cheese |Camembert |SC-QT-G12|35.0 |

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|2.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+-----------+---------+-----+

// T1 point-in-time snapshot

+--------+-----------+---------+-----+

|category|description|itemid |price|

+--------+-----------+---------+-----+

|food |Wine |MB-AG-G0 |4.0 |

|food |Bread |FI-AG-G08|1.0 |

|food |Crackers |BS-AG-G09|10.0 |

+--------+-----------+---------+-----+

**Supported Fetaures**

* Table batch reads and writes
* Table streaming reads and writes
* Table deletes, updates, and merges
* Concurrency control
* Table versioning
* Table optimize
* Change data feed
* Constraints