Cloning
========

you can create a copy on an existing delta table.

clones can be of 2 types

- shallow clone (only metadata is copied)

source table

cloned table

- deep clone (copies both metadata and data)

what if I delete all the records from the source table orders_managed

and then do a VACUUM RETAIN 0 HOURS
==========

Deep Clone
==========

- any changes made to either deep clone or shallow clone, affects only the clone themself and not the source table.

- cloning a table is not the same as CTAS create table as select *.

```
create orders_copy as
select * from orders_managed;
```

- you can use deep clone to perserve the state of a table at a certain point in time for archival purpose.

```
create or replace table archive_table CLONE my_prod_table;
```

- to test a workflow on a prod table without corrupting the table, you can easily create a shallow clone. this allows you to run arbitary worflows on the cloned

table that contains all the prod data but does not affect production workloads.

- shallow clones on external tables must be external tables. shallow clone on managed table must be managed table.

- for managed tables, dropping the source table breaks the target table for shallow clones.

- shallow clones on external table are not impacted by dropping the source.

create table target_table shallow clone source_table version as of 3;

======

Change Data Feed (CDF)
==========

Delta logs - file level changes / dataset level changes

CDF is used to track row level changes between different versions.

v0 - v2

inserts
deletes
pre-image
post-image

- process only changed data (incremental)

```
CREATE TABLE demo (
  id INT,
  name STRING,
```

```
  amount DOUBLE
)
USING DELTA
TBLPROPERTIES (delta.enableChangeDataFeed = true);
```

```
SET spark.databricks.delta.properties.defaults.enableChangeDataFeed = true;
```

availableNow = True

process all the data that is currently available.

runs as a finite job (like a batch), then stops automatically

if you re run it later with the same checkpoint, it will pick up only new changes.

best for

serverless (since continuous triggers are not supported)

periodic job (eg hourly / daily) - schedule notebook as a job, no need to keep the cluster alive.

Delta is a very famous lakehouse file format.

iceberg

Parquet

the only difference between Delta and icerberg is about the metadata

Delta table

iceberg client has to read

making a copy of this delta table into iceberg format

it will be rewrite of data and metadata

100 GB

parquet - 1 copy
metadata - delta
metadata - iceberg

Read Delta tables with iceberg clients

This functionality was previously called Delta Lake Universal Format (UniForm)

Multiple formats, single copy of data

=====

Small file problem

you have 10000 files each file is of roughly 1 mb

select avg(sales) from orders_managed

100 files of 100 mb each

you have a table

and every one hour there is some incremental update which is coming

optimize

manual optimize

16 mb to 1 GB

SET spark.databricks.delta.optimize.maxFileSize = <bytes>;

bin packing algo

100 mb
300 mb
300 mb
100 mb
600 mb
300 mb

600 mb
300 mb
300 mb
300 mb
100 mb
100 mb

bin 1 - 600 mb, 300 mb, 100 mb
bin 2 - 300 mb, 300 mb, 100 mb

Auto Optimize

what is the triggering point for auto optimize to run.

spark.databricks.delta.autoCompact.minNumFiles

(minimum files which are present to trigger)

spark.databricks.delta.optimize.maxFileSize = 1 GB

spark.databricks.delta.autoCompact.minNumFiles (triggering point)

========

Data Skipping
==========

1000 files

800 files

20 files

Data Layout of your table

Partitioning / zorder

Partitioning

sales table

select __ from sales where country = India

50 such folders

country=India
country=US
country=Australia
.
.

hive style partioning

challenges with partitioning

- you can apply on columns with low cardinality
- no obvious benefits for data skipping
- if usage pattern changes partitioned needs to be changed
- to change partitioning, table must be rewritten.
- can have small file problem
- high chances of data skewness

Z ordering
==========

trip distance

file 1 - 0, 500
file 2 - 2, 400
file 3 - 10, 450
file 4 - 5, 350
file 5 - 20, 100

trip_distance > 50 and trip_distance < 100

zorder

file 1 - 0, 30

file 2 - 31, 80

file 3 - 81, 200

file 4 - 201, 400

file 5 - 401, 500

z ordering is a technique to colocate related information in the same set of files.

this co-locality is automatically used by delta lake on databricks for data skipping.

If you expect a column to be commonly used in predicates and that has high cardinality then use zorder by

partitioning (low cardinality columns)

zordering (on high cardinality columns)

Liquid Clustering
==========

Partitioning

zorder

we have struggle to determine the partitioning column and zorder column

if the reading pattern changes then we have to rewrite all the data

Liquid Clustering

2020 - small

2021 - small

2022 - large

2023 - medium

2024 - small

2025 - small

medium size is the ideal size

2020, 2021

2022-1, 2022-2

2023

2024, 2025

01-01 02-01. 03-01

India small small. medium

US small. very small. very small

Australia. small. very small. very small

Partitioning and zorder

'
'

partitioning or zorder

liquid clustering

- incremental clustering
- flexibility to change the columns (Auto)
- avoid rewriting all the data

- avoid write conflicts

partitioning

5 partitions

IN - Merge

IN - Merge

The following are the scenarios when you can benefit from liquid clustering:

- Tables ofter filtered by high cardinality columns..
- tables with significant skew in data distribution
- tables with access patterns that change over time
- tables with concurrent write requirements.
- tables that require a lot of maintainance and tuning efforts

you can enable liquid clustering on a brand new table

or on a existing table

after liquid clustering is enables, run optimize jobs as usual to incrementally cluster new data.

if you have predictive optimization enabled. optimize runs automatically.

ideal data layout

- files between 16 mb to 1 gb
- non overlapping data ranges
- cost of maintaining the data layout must be cheap

Predictive optimization
==========

it removes the need to manually manage the maintainance operations for UC managed tables on azure databricks.

1. optimize - triggers incremental clustering for liquid enable tables.
improves query performance by optimizing file sizes.

2. VACUUM - reduces storage costs by deleting data files no longer referenced by the table.

3. ANALYSE - triggers incremental update of stats to improve query performance

serverless compute for these background jobs. your account is billed for compute associated with these workloads using a serverless jobs SKU.