

# WIP PR

<https://github.com/apache/iceberg-python/pull/353>

## Work Items

- Append Identity Partitions (1)
  - Detect partitions and split writes
- Append Transform Partitions (4)
  - [Requires managing an auxiliary transform partition column](#)
- Static Overwrite as Delete + Append (3)
- Dynamic Overwrite (2)

## Timelines:

- 0.7.0 release
- After

## Open Questions

- API
  - Append
  - Static Overwrite (as Delete + Append)
  - Dynamic Overwrite (from table property?)
- Implications of append + delete vs separate MergingSnapshotProducer
  - Where is the filter column expected?
    - Only In the BooleanExpression (Spark Iceberg)
    - Both?
      - If both, what's the expectation when the two don't match? **Requires looking through the entire table to find mismatches**
  - In Spark Iceberg, static overwrites only support Identity Partitions.
    - Does this still hold true?

- In Spark Iceberg, static overwrites only supports Partition Filter
  - Does this still hold true?
- ~~How is Transform Partition Parsing from filter expression supported / or not supported?~~

## Proposed APIs

### Static Overwrite as Delete + Append

- Filter = can be anything (Partition / non-partition, Identity partition)
- Supported Cases

Python

```
# Partition columns 'level' and 'dt' as Identity Partition
```

```
# case 1: arrow table partition value matches filter partition value
```

```
# expected behavior -> deletes partition level = 'INFO', dt = '2024-03-26',  
adds the same
```

```
overwrite_filter = "level = 'INFO' AND dt = '2024-03-26'"
```

```
df = pa.Table.from_pylist(  
    [  
        {"level": "INFO", "msg": "hi", "dt": date(2024, 3, 26)},  
        {"level": "INFO", "msg": "bye", "dt": date(2024, 3, 26)},  
    ],  
)
```

```
tbl.overwrite(df, overwrite_filter)
```

```
# case 2: arrow table partition value matches filter partition value, but only  
a subset of partition fields are present in the filter
```

```
# expected behavior -> deletes all values in partition level = 'INFO' but only  
two files are added for each value of 'dt'. A full scan is required to  
determine the partition values of 'dt'
```

```
overwrite_filter = "level = 'INFO'"
```

```

df = pa.Table.from_pylist(
    [
        {"level": "INFO", "msg": "hi", "dt": date(2024, 3, 26)},
        {"level": "INFO", "msg": "bye", "dt": date(2024, 3, 27)},
    ],
)

tbl.overwrite(df, overwrite_filter)

# case 3: arrow table partition value matches, but regular column value does
not

overwrite_filter = "level = 'INFO' AND dt = '2024-03-26' AND msg = 'hi'"

# expected behavior -> deletes partition level = 'INFO', dt = '2024-03-26' and
msg = 'hi'

df = pa.Table.from_pylist(
    [
        {"level": "INFO", "msg": "hi", "dt": date(2024, 3, 26)},
        {"level": "INFO", "msg": "bye", "dt": date(2024, 3, 26)},
    ],
)

tbl.overwrite(df, overwrite_filter)

# Should this result in an exception? -> maybe validate that all filter columns
are partition columns

# case 4: partition value does not match with provided filter partition value

overwrite_filter = "level = 'INFO' AND dt = '2024-03-26'"

# expected behavior -> deletes partition level = 'INFO' and dt = '2024-03-26'

df = pa.Table.from_pylist(
    [
        {"level": "INFO", "msg": "hi", "dt": date(2024, 3, 26)},
        {"level": "ERROR", "msg": "bye", "dt": date(2024, 3, 26)},
    ],
)

tbl.overwrite(df, overwrite_filter)

```

```

# Should this result in an exception? -> validate and throw

# case 5: other predicates (greater,less,set) overwrite_filter expression

overwrite_filter = "level = 'INFO' AND dt >= '2024-03-26'"

# expected behavior -> deletes partition level = 'INFO' and dt >= '2024-03-26',
# and adds the files?

df = pa.Table.from_pylist(
    [
        {"level": "INFO", "msg": "hi", "dt": date(2024, 3, 26)},
        {"level": "INFO", "msg": "bye", "dt": date(2024, 3, 27)},
    ],
)

tbl.overwrite(df, overwrite_filter)

```

## Dynamic Overwrite as detect partition + delete + append

- Supported Cases

```

Python
# Partition column = 'level' as Identity Partition and Partition column =
# 'msg_bucket_2' as Bucket Partition

# case 1

df = pa.Table.from_pylist(
    [
        {"level": "INFO", "msg": "hi"},
        {"level": "INFO", "msg": "bye"},
    ],
)

```

```
tbl.dynamic_overwrite(df) -> winner?
```

OR

```
tbl.overwrite(df) # with table property write.overwrite.mode = dynamic?
```