

Populating column-oriented databases

INTRODUCTION TO NOSQL

SQL

Jake Roach
Data Engineer

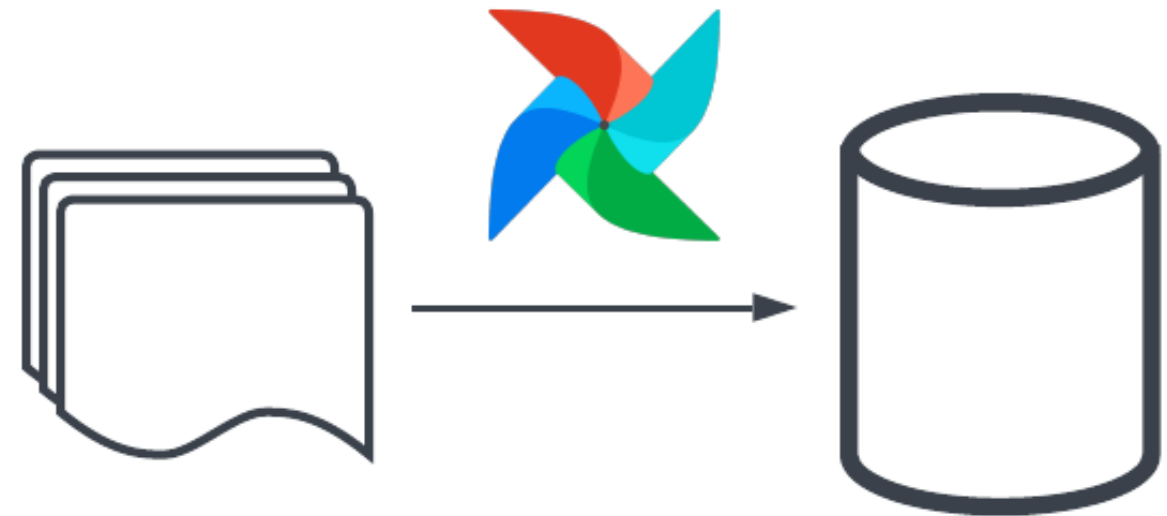
Populating row-oriented vs. column-oriented databases

Row-oriented:

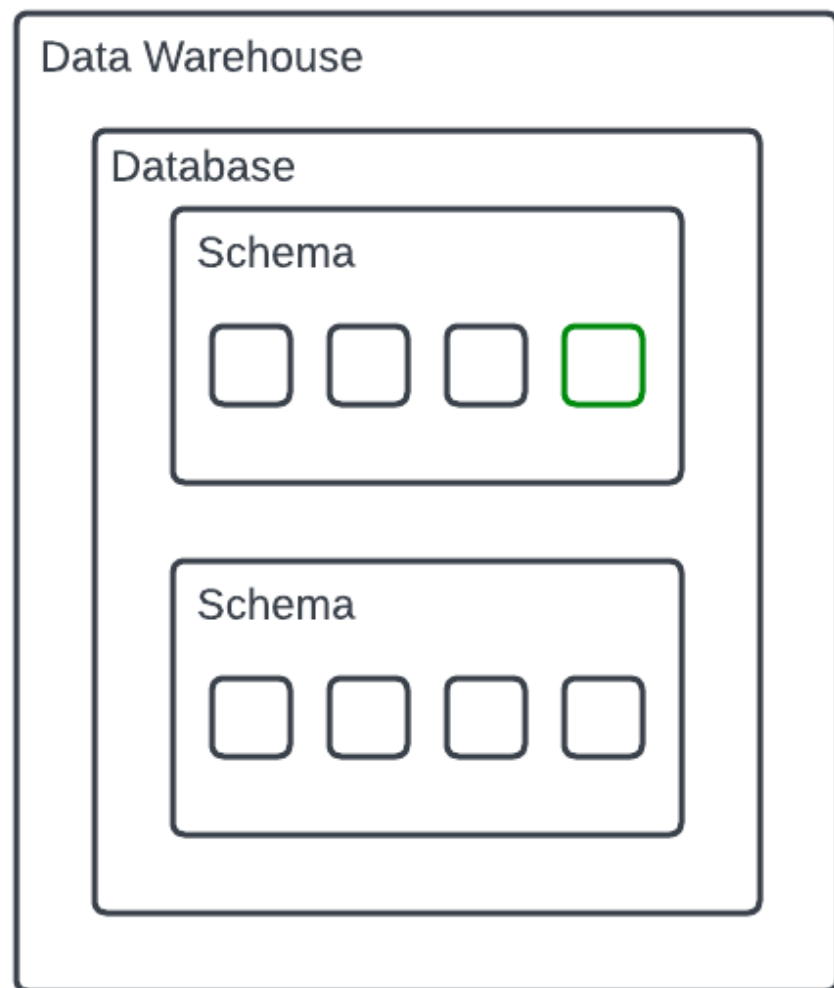
- Optimized for transactional use-cases
- Best performance when inserting, updating or deleting individual records

Column-oriented:

- Use for analytics workflows
- Perform well when loading, updating, or deleting data in-bulk



CREATE TABLE



```
CREATE TABLE books (  
    title VARCHAR(100),  
    author VARCHAR(100),  
    price FLOAT  
);
```

COPY INTO

```
COPY INTO books
FROM 'file:///data_science_books.csv'
FILE_FORMAT = (
  TYPE = 'CSV'
  FIELD_DELIMITER = ','
  SKIP_HEADER = 1
);
```

COPY INTO

FROM

- Cloud storage location
- URL
- Staged files

FILE_FORMAT

- Type of file, delimiter, other metadata information

¹ <https://docs.snowflake.com/en/sql-reference/sql/copy-into-table>

CREATE TABLE ... AS

```
CREATE TABLE premium_books AS
SELECT *
FROM books
WHERE price > 50.00;
```

```
CREATE OR REPLACE TABLE premium_books AS
SELECT *
FROM books
WHERE price > 50.00;
```

CREATE TABLE ... AS

- Provide a table name
- Creates table in the current schema

SELECT ...

- Populates table with data returned by query

OR REPLACE

- If there is an existing table, it is replaced by new table

¹ <https://docs.snowflake.com/en/sql-reference/sql/create-table>

Let's practice!

INTRODUCTION TO NOSQL

Advanced column-oriented database techniques

INTRODUCTION TO NOSQL

SQL

Jake Roach
Data Engineer

Micro-partitioning data with Snowflake

Micro-partitioning:

- Creates smaller "chunks" of rows, stored in columnar format
- Stores metadata about each partition

Allowing for:

- Query pruning to reduce the amount of data accessed
- Efficient execution of DML (data manipulation language)

¹ <https://docs.snowflake.com/en/user-guide/tables-clustering-micropartitions>

Micro-partitioning data

title	author	pages	price
R for Dummies	de Vries	432	17.99
Data for All	Thompson	230	49.99
Python Cookbook	NULL	704	51.48
The Art of Data Science	Peng	170	20.00
Emotional Data	Jame	24	4.99
Integrating Data	Inmon	134	19.95

metadata title	author	pages	price
Integrating Data	Inmon	134	19.95
Python Cookbook	NULL	704	51.48
Data for All	Thompson	230	49.99

metadata title	author	pages	price
R for Dummies	de Vries	432	17.99
Emotional Data	Jame	24	4.99
The Art of Data Science	Peng	170	20.00

Data clustering with Snowflake

Data clustering:

- Organizing or grouping similar data points together
- Automatically performed during data load

Allowing for:

- Decreasing data accessed during execution
- Improved query performance

¹ <https://docs.snowflake.com/en/user-guide/tables-clustering-micropartitions>

Data clustering

metadata	title	author	pages	price
	Integrating Data	Inmon	134	19.95
	Python Cookbook	NULL	704	51.48
	Data for All	Thompson	230	49.99

metadata	title	author	pages	price
	Integrating Data	Inmon	134	19.95
	Data for All	Thompson	230	49.99
	Python Cookbook	NULL	704	51.48

metadata	title	author	pages	price
	R for Dummies	de Vries	432	17.99
	Emotional Data	Jame	24	4.99
	The Art of Data Science	Peng	170	20.00

metadata	title	author	pages	price
	Emotional Data	Jame	24	4.99
	R for Dummies	de Vries	432	17.99
	The Art of Data Science	Peng	170	20.00

Query pruning

SELECT

title,
author,
price

FROM books

WHERE

price > 25.00;

Micro-partitioning and data clustering allow for:

- Reducing data scanned
- Fast time-to-insights

metadata	title	author	pages	price
	Integrating Data	Inmon	134	19.95
	Data for All	Thompson	230	49.99
	Python Cookbook	NULL	704	51.48

metadata	title	author	pages	price
	Emotional Data	Jame	24	4.99
	R for Dummies	de Vries	432	17.99
	The Art of Data Science	Peng	170	20.00

Let's practice!

INTRODUCTION TO NOSQL

Analytics workflows for column-oriented databases

INTRODUCTION TO NOSQL

SQL

Jake Roach
Data Engineer

Common table expressions with Snowflake

Common table expressions (CTEs):

- Named sub-queries/temporary tables, defined using the `WITH` keyword
- Creates a object that can be later queried
- Reduce the amount of data that is being queried and/or `JOIN` 'ed
- More modular, easier to troubleshoot

```
WITH <cte-name> AS (  
    SELECT  
        . . . .  
    FROM <table-name>  
    [JOIN | WHERE | ...]  
)  
  
SELECT  
    . . .  
FROM <cte-name>;
```

¹ <https://docs.snowflake.com/en/user-guide/queries-cte>

Writing common table expressions

```
WITH premium_books AS (  
  SELECT  
    title,  
    author,  
    avg_reviews  
  FROM books  
  WHERE price > 25.00  
)  
SELECT  
  author,  
  MIN(avg_reviews) AS min_avg_reviews,  
  MAX(avg_reviews) AS max_avg_reviews  
FROM premium_books  
GROUP BY author;
```

- Creating a `premium_books` temporary object
- Using `premium_books` downstream

Can creating multiple temporary objects:

```
WITH  
  <first-name> AS (...),  
  <second-name> AS (...),  
  ...  
  ...  
;
```


Views with Snowflake

Views:

- Allow query results to be accessed like a table
- Non-materialized and materialized

```
CREATE VIEW <view-name> AS
SELECT
    ...
FROM <table-name>
[WHERE | JOIN | ...];
```

Creating views with Snowflake

```
CREATE VIEW premium_books AS
  SELECT
    title,
    author,
    avg_reviews
  FROM books
  WHERE price >= 25.00;
```

```
SELECT * FROM premium_books;
```

- Query executes when `premium_books` is called
- "Named definition" of a query

```
CREATE MATERIALIZED VIEW premium_books AS
  SELECT
    title,
    author,
    avg_reviews
  FROM books
  WHERE price >= 25.00;
```

```
SELECT * FROM premium_books;
```

- Results are stored upon execution
- Better query performance, requires refreshing

Let's practice!

INTRODUCTION TO NOSQL

Working with semi-structured data in Snowflake

INTRODUCTION TO NOSQL

SQL

Jake Roach
Data Engineer

Semi-structured data in Snowflake

```
{  
  "ISBN_13": "978-1685549596",  
  "publisher": "Notion Press Media",  
  "size": {  
    "dimensions": "8.5 x 1.01 x 11 inches",  
    "weight": "2.53 pounds"  
  }  
}
```

- Allows data to be stored in "raw" format
- **VARIANT** type
- Store each object in a single column

Semi-structured data types in Snowflake

Snowflake also supports the **OBJECT** and **ARRAY** types

- **OBJECT** is similar to dictionaries in Python
- **ARRAY** is similar to lists in Python

VARIANT type

- Stores semi-structured data in a single column

library
<pre>{ "ISBN_13": "978-1685549596", "publisher": "Notion Press Media", "size": { "dimensions": "8.5 x 1.01 x 11 inches", "weight": "2.53 pounds" } } { "ISBN_13": "978-0596153939", "publisher": "O'Reilly Media", "size": { "dimensions": "8 x 0.98 x 9.25 inches", "weight": "1.96 pounds" } }</pre>

Querying semi-structured data with bracket notation

library
<pre>{ "ISBN_13": "978-1685549596", "publisher": "Notion Press Media", "size": { "dimensions": "8.5 x 1.01 x 11 inches", "weight": "2.53 pounds" } }</pre>
<pre>{ "ISBN_13": "978-0596153939", "publisher": "O'Reilly Media", "size": { "dimensions": "8 x 0.98 x 9.25 inches", "weight": "1.96 pounds" } }</pre>

Query:

```
SELECT
  library['ISBN_13']
FROM books;
```

Result:

library['ISBN_13']
"978-1685549596"
"978-0596153939"

Querying semi-structured data with dot notation

library
<pre>{ "ISBN_13": "978-1685549596", "publisher": "Notion Press Media", "size": { "dimensions": "8.5 x 1.01 x 11 inches", "weight": "2.53 pounds" } }</pre>
<pre>{ "ISBN_13": "978-0596153939", "publisher": "O'Reilly Media", "size": { "dimensions": "8 x 0.98 x 9.25 inches", "weight": "1.96 pounds" } }</pre>

Query:

```
SELECT
  library:ISBN_13,
  library:publisher
FROM books;
```

Result:

library:ISBN_13	library:publisher
"978-1685549596"	"Notion Press Media"
"978-0596153939"	"O'Reilly Media"

Querying nested semi-structured data

library

```
{
  "ISBN_13": "978-1685549596",
  "publisher": "Notion Press Media",
  "size": {
    "dimensions": "8.5 x 1.01 x 11 inches",
    "weight": "2.53 pounds"
  }
}
{
  "ISBN_13": "978-0596153939",
  "publisher": "O'Reilly Media",
  "size": {
    "dimensions": "8 x 0.98 x 9.25 inches",
    "weight": "1.96 pounds"
  }
}
```

SELECT

```
library:ISBN_13,
library:size.dimensions,
library:size.weight,
```

FROM books;

SELECT

```
library["ISBN_13"],
library["size"]["dimensions"],
library["size"]["weight"],
```

FROM books;

library:ISBN_13	library:size:dimensions	library:size:dimensions
"978-1685549596"	"8.5 x 1.01 x 11 inches"	"2.53 pounds"
"978-0596153939"	"8.5 x 1.01 x 11 inches"	"1.96 pounds"

Let's practice!

INTRODUCTION TO NOSQL