

Snowflake と dbt

変換は green、請求は red —
Snowflake 運用シリーズ 2/3

よくある落差

- CI の `dbt build` は 毎日 **green**
- ダッシュボードのテストも通っている
- なのに クレジット請求だけ が前月比で跳ねる

変換レイヤーは整ったのに、運用が楽にならない — 今日のテーマ。

前話とのつながり

シリーズ1本目: [Snowflake と ClickOps の限界](#)

- コンソール運用の 限界 を整理した
- 本日: dbt を載せた その先 に残るギャップ

今日のねらい

dbt が 解く領域 と、ClickOps 限界（前話）が 残る領域 を切り分ける。

2. dbt が担うもの

コードとしてのデータ

モデル

- SQL を リポジトリ で管理 (staging → intermediate → marts)
- `ref()` / `source()` で 依存関係 を明示
- レビュー・差分で 意図 が残る

テスト

- `unique` / `not_null` / `relationships` など 契約
- CI で マージ前 に壊れを検知
- データの正しさ をコード側で担保

ドキュメントと lineage

- `schema.yml` で カラム説明・所有者
- `dbt docs generate` で グラフ可視化
- 「この mart は何から来たか」を コードから辿れる

3. dbt が担わないもの

プラットフォームは別レイヤー

ウェアハウス・リソース

- **WH** サイズ・自動停止・マルチクラスタ
- ロール / **GRANT** (dbt 実行ロール以外の human 操作)
- ステージ・外部ボリューム・ネットワークルール

dbt は 変換の正しさ — インフラの正しさ ではない。

パイプライン外

- Worksheets の **ad hoc** クエリ
- BI ツールからの 直接参照
- 緊急対応の 手動 **DDL**

コストの時間軸

- dbt run 1 回の成功 ≠ 月次コストの最適
- 止め忘れ WH・サイズ戻し忘れは 請求に直結
- テスト green でも 利用率・クレジット は別メーター

4. 接続点

dbt と Snowflake 運用をどう噛ませるか

環境分離

- **dev / prod** の DB ・ スキーマ ・ WH を dbt **target** で切替
- `profiles.yml` の output (ダミ一例) :

```
me_prod:

  type: snowflake

  account: "{{ env_var('SNOWFLAKE_ACCOUNT') }}"

  role: TRANSFORMER_PROD

  database: ANALYTICS

  warehouse: WH_DBT_PROD

  schema: "{{ env_var('DBT_USER') }}"
```

CI と本番スケジュール

役割	典型パターン
CI	PR で <code>dbt build</code> (テスト込み)
本番	スケジューラ / Orchestrator で deploy
境界	main マージ ≠ 即本番 (承認・タグ)

権限モデル

- **dbt** 実行ロール: 変換に必要な最小 GRANT (CREATE TABLE in schema 等)
- **human** ロール: Worksheets・Admin は 別ロール (ACCOUNTADMIN 目常禁止)
- サービスアカウント: CI 用キーは ローテーション と監査対象

5. 実務パターン

うまくいくチームの「dbt の外側」の約束事

約束事の例

- 本番スキーマ変更は **dbt** マージ経由のみ（手 DDL は例外手順）
- WH は **dbt** 用と **ad hoc** 用 を分ける（サイズポリシー別）
- コストレビューは **dbt** の **green** とは独立 の定例（週次 / 月次）

アーキの一枚絵（概念）

Sources → [dbt models / tests] → marts

↓

その外側: WH, roles, stages, ad hoc, alerts

内側 を dbt、外側 をプラットフォームチーム（または IaC）で持つ。

3 本目への布石

- 2 本目まで: 何を **dbt** に任せ、何を残すか
- 3 本目: 観測・把握 (QUERY_HISTORY、コストアラート) で外側を閉じる

持ち帰り

- dbt は データの正しさ (モデル・テスト・lineage)
- Snowflake 運用は プラットフォームの正しさ (WH・ロール・コスト) — 別軸
- green の CI だけでは 請求は守れない — 外側の約束事が要る

ありがとうございました

質問・失敗談（匿名化）歓迎・@ta93abe_