図解でわかる

# Snowflakeのデータ階層と活用方法

## RAW・DWH・データマートからBI・AI活用まで




Start Challenge
Consulting co., ltd.



Strictly Confidential

# 企業のデータ活用で起きている問題

部署・システムごとにデータが分散していることが、**集計作業の長時間化と数字の不一致**を生み、
「分析しても行動につながらない」状態の主要因となっている。

## 現状（As-Is）｜よくある状態

データの置き場所が分散している

| | | |
|---|---|---|
| 販売実績 **基幹システム** | 商談情報 **Salesforce** | 予算 **Excel** |
| 経費 **会計システム** | Web実績 **GA4** | |

運用も個別化している

- 担当者が個別に集計している
- 部署ごとに「売上」の定義が違う
- 会議資料の数字が一致しない

## 結果として起きること（課題）

- 集計作業に時間がかかる
- データの正しさを確認できない
- 会議が数字確認だけで終わる
- 分析しても行動につながらない
- **AIに正確なデータを渡せない**

**本資料の位置づけ**　Snowflakeによるデータ基盤は、こうした分散したデータを**段階的に整理する**ために使われる。本資料ではその階層設計と活用の流れを解説する。

事実：業務データが複数システムに分散している／解釈：分散が集計工数と定義齟齬を生む／提案：階層設計による段階的整理　出典：本資料の構成案（社内ヒアリング想定）

# Snowflakeは何をするサービスなのか

Snowflakeは、企業内のデータを**集約・加工・分析・共有**するためのクラウド型データプラットフォームであり、
保存領域と計算処理を分離できる点が、複数部門での同時利用を可能にしている。

## 主な活用

- 各システムのデータ統合
- 大量データの保存
- SQLによるデータ加工
- BIダッシュボードへの接続
- 部門間・企業間のデータ共有
- 機械学習・生成AIへのデータ提供
- 定期レポートの自動化

## Snowflakeを使うメリット

- 保存領域（ストレージ）と計算処理（コンピュート）を分離できる
- 利用状況に応じて処理性能を変更できる
- 複数部門が同じデータを利用できる
- データをコピーせず共有できる
- アクセス権限を細かく設定できる

**構造上のポイント**
保存と処理が分かれているため、「データ取込み」と「ダッシュボード閲覧」を互いに影響させずに並行実行できる。（詳細はP12）




事実：Snowflakeはストレージとコンピュートを分離した構成をとる／解釈：部門横断の同時利用に適する　出典：本資料の構成案

# Snowflakeの基本的な階層

SnowflakeのデータはAccount → Database → Schema → Tableの入れ子構造で管理される。
この階層を先に理解しておくことが、後述するRAW・DWH・データマートの設計理解の前提となる。

## オブジェクトの入れ子構造

**Snowflake Account**
└ **Database**
　└ **Schema**
　　├ Table
　　├ View
　　├ Dynamic Table
　　├ Stage
　　├ Stream
　　└ Task

Table・Viewはデータの保持と参照、Stageはファイルの受け渡し、Stream・Taskは差分検知と定期処理に用いる。

| 階層 | 意味 | 具体例 |
|---|---|---|
| **Account** | Snowflakeの利用環境全体 | 会社・グループ |
| **Database** | データを大きく分類する箱 | RAW、DWH、DATAMART |
| **Schema** | 業務や用途別の分類 | 営業、財務、顧客 |
| **Table** | データを保存する表 | 売上明細、顧客一覧 |
| **Column** | データ項目 | 顧客名、売上金額 |
| **Row** | 1件ごとの記録 | 1回の取引、1件の商談 |

### 押さえるべき点

DatabaseとSchemaの切り方が、そのままデータ基盤の「分かりやすさ」と「権限管理のしやすさ」を決める。
次ページでは、この階層を身近な例に置き換えて整理する。

事実：Snowflakeのオブジェクトは Account／Database／Schema の階層下に配置される　出典：本資料の構成案

# 身近な例で理解する階層

Snowflakeの構造は「会社のオフィス」に置き換えると理解しやすい。
DatabaseやSchemaは単なる保存場所ではなく、**利用者・権限・加工段階・正式な数字を管理するための構造**である。

| Snowflake | オフィスでの例 |
|---|---|
| **Account** | 会社の建物 |
| **Database** | 部署ごとのフロア |
| **Schema** | 書類を保管するキャビネット |
| **Table** | キャビネット内のファイル |
| **Column** | ファイルに記載する項目 |
| **Row** | 1件分の記録 |

入れ子のイメージ




**重要なポイント｜階層で管理する4つの観点**

1. どの部署が使用するか
2. 誰が閲覧できるか
3. どの段階まで加工されているか
4. どの数字が正式なものか

**結論**

DatabaseとSchemaは「置き場所の分類」ではなく、
利用者・権限・加工段階・数字の正式性を分かりやすく管理するための**設計上の道具**である。

解釈：階層構造をオフィスの物理構造に対応させることで初学者の理解を促進する（説明上の比喩） 出典：本資料の構成案

# データが流れる基本の4階層

データは**RAW → STAGING → DWH／CORE → DATAMART**の順に加工され、最後にBI・AIで利用される。
この階層はSnowflakeに標準で用意されているものではなく、DatabaseとSchemaを使って企業ごとに設計する。




| 階層 | 主な役割 | データの状態 |
|---|---|---|
| **RAW** | 元データの保存 | 取得したまま |
| **STAGING** | 加工前の整理 | 形式を統一 |
| **DWH／CORE** | 全社データの統合 | 共通定義 |
| **DATAMART** | 利用目的別の整理 | 分析しやすい状態 |

**設計上の注意**

階層はSnowflakeに最初から用意されているものではない。
RAW・STAGING・DWH・DATAMARTをDatabaseとして分ける、あるいはSchemaで分けるなど、**企業ごとに設計方針を決める必要がある**。

**階層を分ける目的**

① 元データに戻れる状態を保つ　② 品質を整える工程を分離する
③ 全社共通の定義を1か所に集約する　④ 利用者ごとに使いやすい形で提供する

事実：階層構成はSnowflakeの標準機能ではなくDatabase／Schemaによる設計事項／提案：4階層を基本形として自社要件に合わせて調整　出典：本資料の構成案

# RAW層：元データをそのまま保存する

RAW層に各システムのデータを原則そのまま保存しておくことで、**加工ミス時の復元・再計算・監査対応**が可能になる。
RAW層は「使うためのデータ」ではなく「正しさを担保するための基点」である。

## 保存するデータの例

- Salesforceの商談・顧客データ
- SAPなどの基幹システムデータ
- 会計システムの仕訳データ
- ExcelやCSVの予算データ
- GA4のアクセスデータ
- 広告媒体の実績
- IoT・センサーデータ
- 外部の市場・統計データ

原則として**加工・削除を行わない**。取得した形のまま蓄積する。

## RAW層を残す理由

**復元できる**
加工ミスがあっても元データへ戻れる

**確認できる**
データの取得状況を確認できる

**再計算できる**
加工ルールを変更して再計算できる

**説明できる**
監査や障害調査に利用できる

## 管理項目の例（メタデータ）

| |
|---|
| 取得元システム |
| 取得日時 |
| ファイル名 |
| 更新日時 |
| バッチ処理ID |
| データの対象期間 |

## 管理項目を持つ意味

いつ・どこから・どの範囲のデータが入ったかを記録しておくことで、数字が合わない場合に**どの工程で差異が生じたかを追跡**できる。

この記録がない場合、障害調査は各システムへの個別確認に依存することになる。

提案：RAW層は原則無加工で保持し、取得元・取得日時等のメタデータを併せて管理する　出典：本資料の構成案

# STAGING層：データを整える

STAGING層は、異なるシステム間で**形式・表記・単位を統一する「下ごしらえ」の工程**である。
ここで品質を整えておかないと、後続のDWH統合で顧客・商品の突合が成立しない。

## 主な加工処理

- 日付形式の統一
- 全角・半角の統一
- 電話番号や郵便番号の形式統一
- 重複レコードの除外
- 空欄・NULLの処理
- 金額の通貨・単位の統一
- テストデータの除外
- 個人情報のマスキング
- 不正な値の検出
- データ型の変換

工程上の位置




| 加工前 | 加工後 |
|---|---|
| 2026/8/21 | **2026-08-21** |
| （株）ABC | **株式会社ABC** |
| 1,000千円 | **1,000,000円** |
| 東京都新宿区 | **東京都／新宿区** |

## 整えない場合に起きること

- 同一顧客が別会社として二重計上される
- 単位違いにより売上金額が1,000倍で集計される
- テストデータが実績に混入する

**要点**

STAGING層は、分析の前にデータ品質を整える「下ごしらえ」の場所である。

事実：システム間で日付・表記・単位の形式が異なる／解釈：統一処理を独立工程として置くことで品質責任が明確になる　出典：本資料の構成案

# DWH／CORE層：全社共通データを作る

DWHは、複数システムのデータを**全社共通の定義とIDで統合する層**であり、
「全社で同じ数字を見る」ための基準となる。共通定義を決めないまま先へ進むと、部門ごとの数字の乖離が再発する。

## 統合する主なデータ

- 顧客マスタ
- 商品マスタ
- 組織・社員マスタ
- 売上・受注データ
- 予算・見込みデータ
- 在庫・仕入れデータ
- 会計・原価データ

マスタ系（顧客・商品・組織）とトランザクション系（売上・在庫・会計）を、**共通IDで接続**することが統合の中核となる。

## 顧客データ統合の例

| システム | 元の顧客名 | 共通顧客ID |
|---|---|---|
| Salesforce | 株式会社ABC | **C0001** |
| 会計システム | （株）ABC | **C0001** |
| Excel | ABC社 | **C0001** |

## 共通IDを持つ効果

表記が異なる同一顧客を1つのIDに束ねることで、
・商談 → 受注 → 請求 → 入金の一連の実績を接続できる
・顧客単位のLTVや解約率を算出できる（P10・P13）
・部門をまたいだ集計で二重計上を防止できる

## DWHで決める共通定義

売上は**受注日・出荷日・請求日**のどれで計上するか

新規顧客をどのように判定するか

利益にどの費用を含めるか

部門変更前の実績をどの組織に計上するか

キャンセルや返品をどう扱うか

DWHは、全社で同じ数字を見るための**基準**になる。

解釈：定義の不一致が数字の乖離を生む／提案：共通ID付与と計上基準の明文化をDWH構築の前提条件とする　出典：本資料の構成案

# データマート：目的別に使いやすくする

データマートは、DWHのデータを**部門や分析目的ごとに再整理したもの**である。
必要な項目に絞り集計しておくことで、BIの表示速度とKPI計算の一貫性を同時に確保できる。

| データマート | 利用者 | 主な指標 |
|---|---|---|
| **経営管理** | 役員・経営企画 | 売上、利益、予算達成率 |
| **営業** | 営業責任者 | 商談数、受注率、案件金額 |
| **マーケティング** | マーケ担当 | CV数、CPA、チャネル別成果 |
| **財務** | 財務・経理 | 原価、利益率、資金繰り |
| **人事** | 人事・部門長 | 人員数、稼働率、離職率 |
| **商品** | 商品企画 | 商品別売上、継続率、粗利 |
| **顧客分析** | 営業・CS | LTV、解約率、購入頻度 |




### データマートの特徴

- 必要な項目だけに絞る
- 分析しやすい形に集計する
- BIツールの表示速度を改善する
- 利用者に不要な情報を見せない
- KPIの計算ロジックを統一する

### DWHとの役割分担

DWH＝**定義を1つに揃える層**
データマート＝**用途別に取り出しやすくする層**
定義を各マートで独自に持たせないことが原則（P17 失敗2）。

提案：マートは利用者・利用目的を明示して作成し、KPI計算ロジックはDWH側に集約する　出典：本資料の構成案

# Table・View・Dynamic Tableの使い分け

すべてをTableとして作るのではなく、**更新頻度とデータ量に応じて保存方式を選択する。**
履歴保持はTable、見せ方の変更はView、更新処理の簡素化はDynamic Tableが適する。

| 種類 | 特徴 | 適した用途 |
|---|---|---|
| **Table** | データを実際に保存 | 大量データ、履歴保存 |
| **View** | SQLの結果を仮想的に表示 | 軽い加工、参照先の統一 |
| **Materialized View** | 計算結果を保持 | 特定条件の高速参照 |
| **Dynamic Table** | 指定した鮮度で自動更新 | データマートの段階的な更新 |

## 使い分けの考え方

| | |
|---|---|
| 履歴を確実に残したい | **Table** |
| 元データを直接変更せず見せ方を変えたい | **View** |
| 更新処理を簡単に管理したい | **Dynamic Table** |
| 同じ重い検索を繰り返す | **Materialized View を検討** |

## 階層ごとの適用イメージ

**RAW層**
Table（取得データを蓄積・履歴保持）

**STAGING層**
View または Dynamic Table（形式統一の加工）

**DWH／CORE層**
Table（共通定義データを確定保持）

**DATAMART層**
Dynamic Table／View（目的別集計を自動更新）

**要点**

保存方式の選択は、コストと運用負荷に直結する設計判断である。

提案：階層ごとの適用イメージは一般的な構成例であり、データ量・更新要件に応じて見直す（解釈） 出典：本資料の構成案

# SnowflakeのWarehouseを理解する

Snowflakeの「Virtual Warehouse」は、データを蓄積する仕組みではなく**SQLを実行する計算リソース**を指す。
用途別にWarehouseを分けることで、データ取込みとダッシュボード閲覧が同時に発生しても影響を抑えられる。

| 用語 | 役割 |
|---|---|
| **Data Warehouse** | 分析データを統合・蓄積する仕組み |
| **Virtual Warehouse** | SQLを実行する計算リソース |

## Virtual Warehouseの活用例

| | |
|---|---|
| 取込み用 | 各システムからのデータ連携・ロード処理 |
| 加工用 | STAGING・DWH・マートの加工処理 |
| BI用 | ダッシュボード閲覧時のクエリ実行 |
| AI分析用 | データサイエンス・AI分析の重い処理 |

処理を分けることで、夜間バッチの遅延が翌朝のダッシュボード閲覧を妨げる、といった相互影響を抑制できる。

## 同じデータを、別々の計算リソースから利用する




事実：Virtual Warehouseは計算リソースであり、ストレージとは独立して構成できる／提案：用途別にWarehouseを分離し相互影響を抑制する　出典：本資料の構成案

# BIダッシュボードでの活用

SnowflakeはPower BI・Sigma・Tableauなど主要BIツールと接続できる。
重要なのは「表示するだけ」で終わらせず、**数字の変化から次のアクションにつなげる設計**である。

## 接続できるBIツール・分析環境

| Power BI | Sigma | Tableau | Looker | Excel | Amazon QuickSight |
|---|---|---|---|---|---|

## 経営ダッシュボードの例（利用者：役員・経営企画）

- 全社売上
- 予算達成率
- 部門別利益
- 月次売上推移
- 商品別粗利
- 受注見込み
- キャッシュフロー

## 営業ダッシュボードの例（利用者：営業責任者・担当者）

- 担当者別商談数
- 商談ステージ別金額
- 受注率
- 平均受注単価
- 失注理由
- 次回対応未設定案件
- 受注予測

提案：ダッシュボードは指標の羅列ではなく、次の行動が判断できる構成とする（P14参照） 出典：本資料の構成案

# データからアクションにつなげる方法

KPIごとに**「検知する状態」と「推奨アクション」を事前に定義**しておくことで、
ダッシュボードの閲覧が担当者の具体的な行動に接続される。

| KPI | 検知する状態 | 推奨アクション |
|---|---|---|
| **商談数** | 前月比20%減（仮定値） | リード獲得施策を確認 |
| **受注率** | 目標を5ポイント下回る（仮定値） | 失注理由を分析 |
| **粗利率** | 商品別に低下 | 値引き・原価を確認 |
| **解約率** | 基準値を超過 | 対象顧客へフォロー |
| **在庫数** | 適正在庫を超過 | 発注停止・販売促進 |
| **入金遅延** | 支払期限を超過 | 営業・経理へ通知 |

### 閾値設定の考え方

表中の数値は例示であり、実際の閾値は自社の実績分布と業務判断に基づいて設定する。
重要なのは水準の厳密さよりも、**「誰が・何を見て・何をするか」が事前に決まっている**ことである。

### 実際の流れ




仮説：前月比20%減・目標比5ポイント低下は例示の閾値（仮定値）／提案：KPIごとに検知条件と担当アクションを事前定義する　出典：本資料の構成案

# AI・生成AIでの活用

Snowflakeに整理されたデータは、生成AI活用の情報源にもなる。
AI導入前に**DWHとデータマートを整備しKPI定義を統一する**ことで、回答精度を高められる。

## AI活用例

- 売上が減少した原因の要約
- 部門別実績の自動コメント作成
- 商談データから受注確度を推定
- 顧客ごとの次回提案内容を生成
- 解約リスクが高い顧客の抽出
- 会議資料の説明文章を作成
- 自然言語によるデータ検索
- 社内データを使用したRAGの構築

## 出力例（生成AIによる要約）

> 今月の売上は前月比**8%減少**しています。主な要因は関東地区の新規商談数の減少と、商品Aの受注率低下です。未対応リードへの再アプローチを推奨します。
>
> ※出力例（仮定値）

データマートのKPIを参照させることで、
・数値の変化
・主要因の特定
・次のアクション
までを含む要約を自動生成できる。

## AI活用前に必要なこと

- KPIの定義を統一する
- データ品質を確認する
- AIが参照できる範囲を制限する
- 個人情報や機密情報を管理する
- AIの回答を人が確認する

整備されていないデータをAIに参照させると、**誤った前提の回答**が生成される。

仮説：出力例の「前月比8%減」は説明用の仮定値／提案：AI活用の前提としてKPI定義統一・品質確認・参照範囲制限・人による確認を実施　出典：本資料の構成案

# 権限・セキュリティの考え方

Snowflakeでは役割（ロール）に応じてアクセス権限を設定する。
設計上の要点は、**「見られる人」と「変更できる人」を分けること**である。

| 利用者 | 閲覧できるデータ |
|---|---|
| **役員** | 全社の集計データ |
| **営業部長** | 営業部全体の実績 |
| **営業担当者** | 自分の担当顧客 |
| **人事担当者** | 人事・従業員データ |
| **BI担当者** | 分析用データマート |
| **開発担当者** | 検証環境のデータ |

## 権限の2軸で整理する

**閲覧できる範囲**

組織階層・担当顧客・データマート単位で制御する。集計値のみを見せるか、明細まで見せるかを分ける。

**変更できる範囲**

加工処理の定義変更・テーブル作成・権限付与は、限定した担当者のみに付与する。

## セキュリティ対策

- ロール単位の権限管理
- 個人情報のマスキング
- 行単位のアクセス制御
- 開発・検証・本番環境の分離
- 操作・アクセス履歴の保存
- 不要なデータの非表示
- **外部共有先の制限**

※実装可否・機能名は利用エディションにより異なる

**原則**

「見られる人」と「変更できる人」を分けることが重要である。

提案：ロール設計は組織階層と業務範囲に基づき定義し、変更権限は限定付与とする／注記：機能の利用可否は契約エディションに依存　出典：本資料の構成案

# よくある失敗と改善方法

失敗の多くは技術面ではなく、**目的・定義・運用体制を先に決めていないこと**に起因する。
「データを集めること」よりも「KPIと利用者を先に決めること」が、実行上の優先事項となる。

### 失敗1｜すべてのデータを先に集める

**問題**
利用目的がなく、使われないデータが増える

**改善**
先にKPIと利用者を決める

### 失敗2｜部署ごとに別のデータマートを作る

**問題**
売上や顧客の定義が再び分かれる

**改善**
DWHで共通定義を作ってから展開する

### 失敗3｜BIツールで複雑な加工を行う

**問題**
ダッシュボードごとに計算結果が変わる

**改善**
共通計算はSnowflake側へ集約する

### 失敗4｜更新頻度をすべてリアルタイムにする

**問題**
コストと運用負荷が増える

**改善**
日次・時間単位・リアルタイムを使い分ける

### 失敗5｜作成者しか構造を理解できない

**問題**
担当者変更後に運用できない

**改善**
データ定義・処理・責任者を文書化する

**共通する原因**

いずれの失敗も、**「利用目的・共通定義・運用責任」を後回しにしたこと**に起因する。
技術的な構築の前に、この3点を先に確定させることが必要である。

解釈：5つの失敗はいずれも設計・運用ルールの未整備に起因する／提案：KPI定義・共通定義・文書化を構築前に確定　出典：本資料の構成案

# 導入ステップとチェックリスト

重要なのはデータを保存することではなく、**RAWで残し・STAGINGで整え・DWHで統合し・マートで使いやすくし・BI／AIから業務アクションにつなげる**一連の流れを設計することである。

## 導入の基本ステップ

1. 利用目的と経営課題を決める
2. KPIとアクションを定義する
3. 必要なデータを洗い出す
4. データの取得元を確認する
5. RAW・STAGING・DWH・DATAMARTを設計する
6. データ連携と加工処理を構築する
7. BIダッシュボードを作成する
8. 権限・セキュリティを設定する
9. 利用部門でテストする
10. 運用・改善を継続する

## 最終チェックリスト

- ☐ 利用者と利用目的が明確か
- ☐ KPIの計算方法が決まっているか
- ☐ データの取得元が整理されているか
- ☐ RAWデータを保持しているか
- ☐ 顧客・商品・組織の共通IDがあるか
- ☐ DWHとデータマートの役割が分かれているか
- ☐ 更新頻度が業務要件に合っているか
- ☐ BIからアクションにつながる設計か
- ☐ 権限と個人情報対策ができているか
- ☐ データの管理責任者が決まっているか
- ☐ 利用量とコストを確認できるか
- ☐ データ定義書が整備されているか

**本資料のまとめ**

RAWで残す、STAGINGで整える、DWHで統合する、データマートで使いやすくする、BI・AIから業務アクションにつなげる。
この一連の流れを設計することで、データは「保管されているだけ」の状態から「経営や現場で使われる状態」へ変わる。

提案：導入は目的定義（1–2）→データ整理（3–4）→階層設計・構築（5–6）→活用・統制（7–8）→定着（9–10）の順で進める　出典：本資料の構成案

Contact

# お問い合わせ

Snowflakeによるデータ基盤の構想策定から、階層設計・BI／AI活用の実装・運用定着まで一貫してご支援します。
本資料の内容に関するご質問、個別のご相談は下記までお問い合わせください。

Start Challenge
Consulting co., ltd.
挑む力が、世界を照らす

## お問い合わせ先


| | |
|---|---|
| 会社名 | **Start Challenge Consulting** |
| 担当 | 平出 |
| E-mail | **info@startchg.com** |


**ご支援メニュー**

データ活用構想の策定（KPI・利用者定義）／データ階層設計（RAW・STAGING・DWH・DATAMART）
BIダッシュボード構築（Power BI・Sigma・Tableau）／生成AI活用・RAG構築／権限・セキュリティ設計／運用定着支援

本資料は情報提供を目的として作成したものであり、記載内容の完全性・正確性を保証するものではありません。 発行：Start Challenge Consulting co., ltd.