Amazon Athena (Iceberg) x dbt で はじめるデータ分析！

2023/07/27

データアナリティクス事業本部 石川 覚

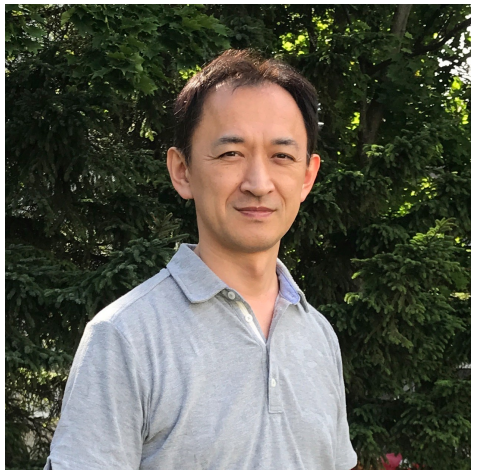
自己紹介

名前：石川 覚（いしかわ さとる）

所属：データアナリティクス事業本部
　　インテグレーション部
　　コンサルティングチーム

担当：コンサルタント、ブログ・登壇等

経歴：メーカーでSE、研究開発
　　→ITベンチャーで製品開発、受託研究
　　→クラスメソッド（2014/6～）



2023 Japan AWS Top Engineers
2023 Japan AWS All Certifications Engineers

好きなサービス：Amazon Redshift/Athena、Google BigQuery

アジェンダ

1. はじめに
2. Amazon Athena（Iceberg）
   - 従来のデータレイクの技術の課題
   - Apache Icebergの特長
   - Apache Icebergのクエリ
3. dbt（data build tools）
   - dbtとは
   - Amazon Athena(Iceberg) x dbt なのか？
   - 現状の課題と代替案
   - 検討事項
4. 最後に

はじめに

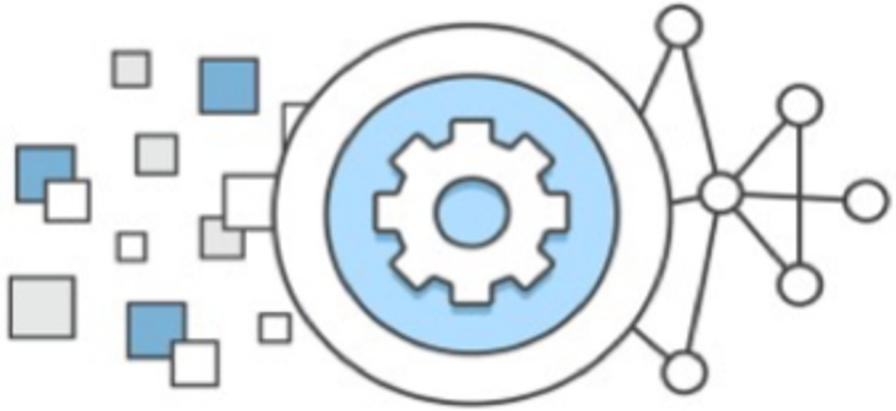
私達のチームでは、Amazon RedshiftやSnowflakeと「dbt」を用いたサーバレスなデータプラットフォームである「dbt-template」ソリューションと、コンサルティングサービスをご提供しております。

本日は、 「dbt-template」のAmazon Athena対応で得られた技術調査の結果と、テーブルフォーマット「Iceberg」と「dbt」対応について、ちょっぴりDiveDeepしたいと思います。

前半は、現在イチオシAmazon Athenaのテーブルフォーマット「Iceberg」のご紹介、後半はdbtをAmazon Athenaに対応させるアダプタ「dbt-athena」を紹介します。

Amazon Athena（Iceberg）

Amazon Athenaとは



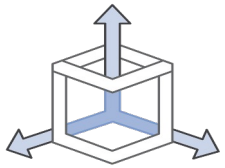
Amazon Athenaは、Amazon Web Services（AWS）が提供するインタラクティブなクエリサービスで、**標準的なSQLを使用して Amazon S3に保存されているデータをクエリ**できます。サーバーレスサービスであるため、**インフラを管理する必要がなく、データのクエリや分析に集中**することができます。

Amazon Athenaの特長

Amazon Athena サーバレスのクエリエンジン

サーバレスでインフラ管理の必要なし

ODBC/JDBC 経由で BI ツールから直接クエリ



大規模データに対しても高速なクエリ

スケールアウトでエクサバイト級に対応

事前のデータロードなしにS3に直接クエリ

オープンファイルフォーマット

スキャンしたデータに対しての従量課金

Amazon Athenaのメタデータ管理とクエリエンジン

メタデータ管理: Hive Style

- 複雑なデータタイプ
- 多数のフォーマットに対応
- データパーティショニングのサポート

クエリエンジン：Trino

- ANSI-SQL 互換
- インメモリー分散クエリーエンジン
- Athena Version3は、2023年11月時点のTrinoの最新バージョン

Trino delivers for Amazon Athena: https://trino.io/blog/2022/12/01/athena.html

従来のデータレイクの技術課題

一般にS3データレイク上の**データファイルにクエリ**するには、Glueデータカタログのテーブルに、**Hiveテーブルスタイルで格納されたデータを定義**します。

しかし、テーブル内のデータの更新（UPDATE）や削除（DELETE）をするためには、S3データレイク上のデータファイルを直接操作したり、**複数のパーティションやクエリを安全に実行する事**はできませんでした。

これらの課題を解決すべく登場したのが、**Icebergや、Hudi、Delta Lake**はじめとするデータフォーマットです。

従来のデータレイクの技術課題

Iceberg、Hudi、Delta Lakeは、これらは技術課題に対して解決策を提供しています。

- スモールファイルによるパフォーマンス劣化

- データの編集・削除が苦手
- 同時アクセス時の整合性を担保できない
- データが追加・更新されると過去の状態が復元できない

従来のデータレイクの技術課題

Iceberg、Hudi、Delta Lakeは、これらは技術課題に対して解決策を提供しています。

- スモールファイルによるパフォーマンス劣化
  - **OPTIMIZE**によるデータのコンパクション
  - **VACUUM**による不要なデータの削除
- データの編集・削除が苦手
  - **UPDATE、DELETE、MERGE**のサポート
- 同時アクセス時の整合性を担保できない
  - **ACIDトランザクション**のサポート
- データが追加・更新されると過去の状態が復元できない
  - **タイムトラベル機能**による過去の時点のデータにアクセス

現時点では、Amazon AthenaとAWS Glue（SparkSQL）から更新が可能である、**dbtで利用可能なIceberg**をピックアップします。

Apache Icebergの特長

テーブルフォーマット **Iceberg** は、re:Invent2021にてサポートが発表されました。Icebergテーブルフォーマットを用いて、AthenaからSELECTはもちろん、**ACIDにINSERT、UPDATE、DELETE、タイムトラベル、オプティマイズ、VACUUM**の操作を可能です。

参考: [Amazon Athena Apache Icebergテーブルフォーマットによる ACID Transactionを試してみました！ #reinvent]

Apache Icebergは、ビッグデータワークロードのために**Netflix社によって開発**された運用実績のあるオープンソースのデータテーブル形式です。Apache ParquetやApache ORCといった既存のフォーマットにおける、テーブルスキーマのエボリューション、トランザクションの一貫性、タイムトラベル機能などの制限に対処するために設計されました。

Apache Icebergのクエリ – テーブル作成


```
CREATE TABLE iceberg_table (
  id int,
  name string,
  category string,
  sale_datetime timestamp
)
PARTITIONED BY (category, day(sale_datetime))
LOCATION 's3://<mybucket>/iceberg_table/'
TBLPROPERTIES (
  'table_type'='iceberg'
);
```

- TBLPROPERTIESのtable_typeに**iceberg**を指定するのみ
- icebergのパーティションは、**Hidden Partition**と呼び、キーに関数を指定できる
- 左記の例のようにtimestamp型のカラムにday()関数を指定すると、内部的には日付ごとにパーティションされ、パーティション用のカラムを別途作成する必要はない

Apache Icebergのクエリ – UPSERT（MERGE）


```
MERGE INTO iceberg_table AS target
USING iceberg_table_tmp AS src ON (
    target.id = src.id
)
WHEN MATCHED THEN
UPDATE
SET "id" = src."id",
    "name" = src."name",
    "category" = src."category",
    "sale_datetime" = src."sale_datetime"
WHEN NOT MATCHED THEN
INSERT (
    "id",
    "name",
    "category",
    "sale_datetime"
    )
VALUES (
    src."id",
    src."name",
    src."category",
    src."sale_datetime"
    );
```

- Icebergでは、MERGEを用いたUPSERTが可能
- もちろん、DELETE・INSERTでもUPSERTも可能

Apache Icebergのクエリ – タイムトラベル


```
SELECT * FROM iceberg_table
FOR TIMESTAMP AS OF TIMESTAMP '2023-07-24 06:13:00 UTC';
```

- タイムトラベル機能は、指定した日時のクエリ結果を取得できます。また、スナップショットのIDを用いて、バージョントラベルクエリも可能です。
- つまり、AthenaでIceberg テーブルフォーマットを用いると、DWHと同様の使い勝手でデータを保存、操作できるようになります。

(data build tools)

dbt（data build tools）

dbt（data build tool）は、**SQLによってデータ変換を構築**、オーケストレーションするためのフレームワークを提供します。データモデルの定義、データ変換と集計のためのSQLクエリの記述、データ変換と依存関係の管理が可能です。

dbt を使用すると、モジュラリティ、CI/CD、組み込みドキュメントなどの**ソフトウェアエンジニアリングのベストプラクティスに従ってデータ変換コードのデプロイが可能**です。データモデルのビジネスロジックをSQLで実装、コードベースのテストを自動化、コードのバージョン管理ができます。

dbtの特長

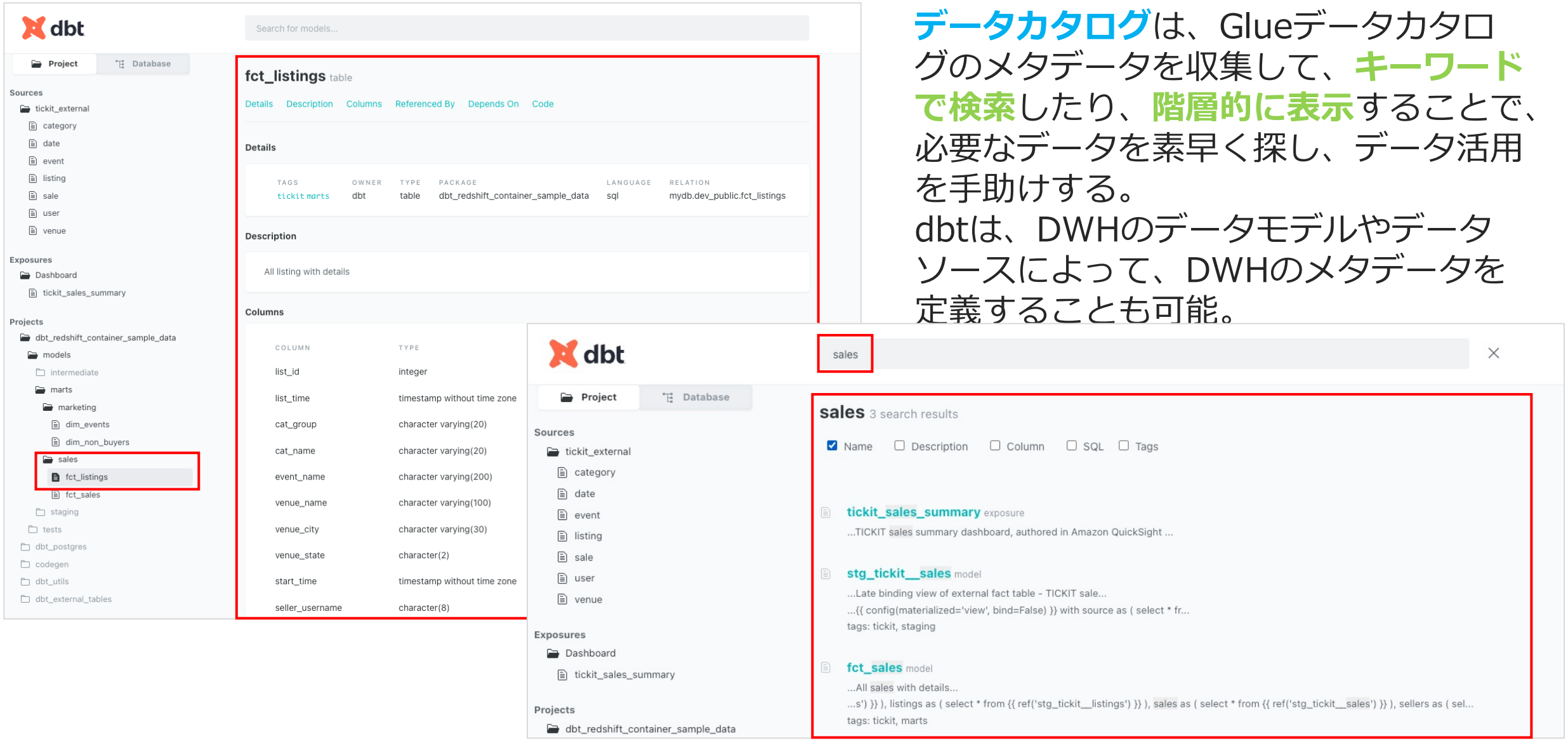
- **データモデル**を**SQLで構築**、ビジネスロジックを実装
- **ビジネスロジックの依存関係**から**DAG**を自動生成
- **DAG**に基づて**データ変換の並列で実行、データリネージ**
- **データモデル**に基づいた**データカタログ**
- **コードベース**のデータの**テストを自動化**
- **コードのバージョン管理**による**CI/CD**
- **マクロとref文**により**再利用可能なコード**

参考: [Amazon Redshift + dbt ユーザー必読の書「Best Practices for Leveraging Amazon Redshift and dbt」を読んでみた]

Amazon Athena(Iceberg) x dbt なのか？

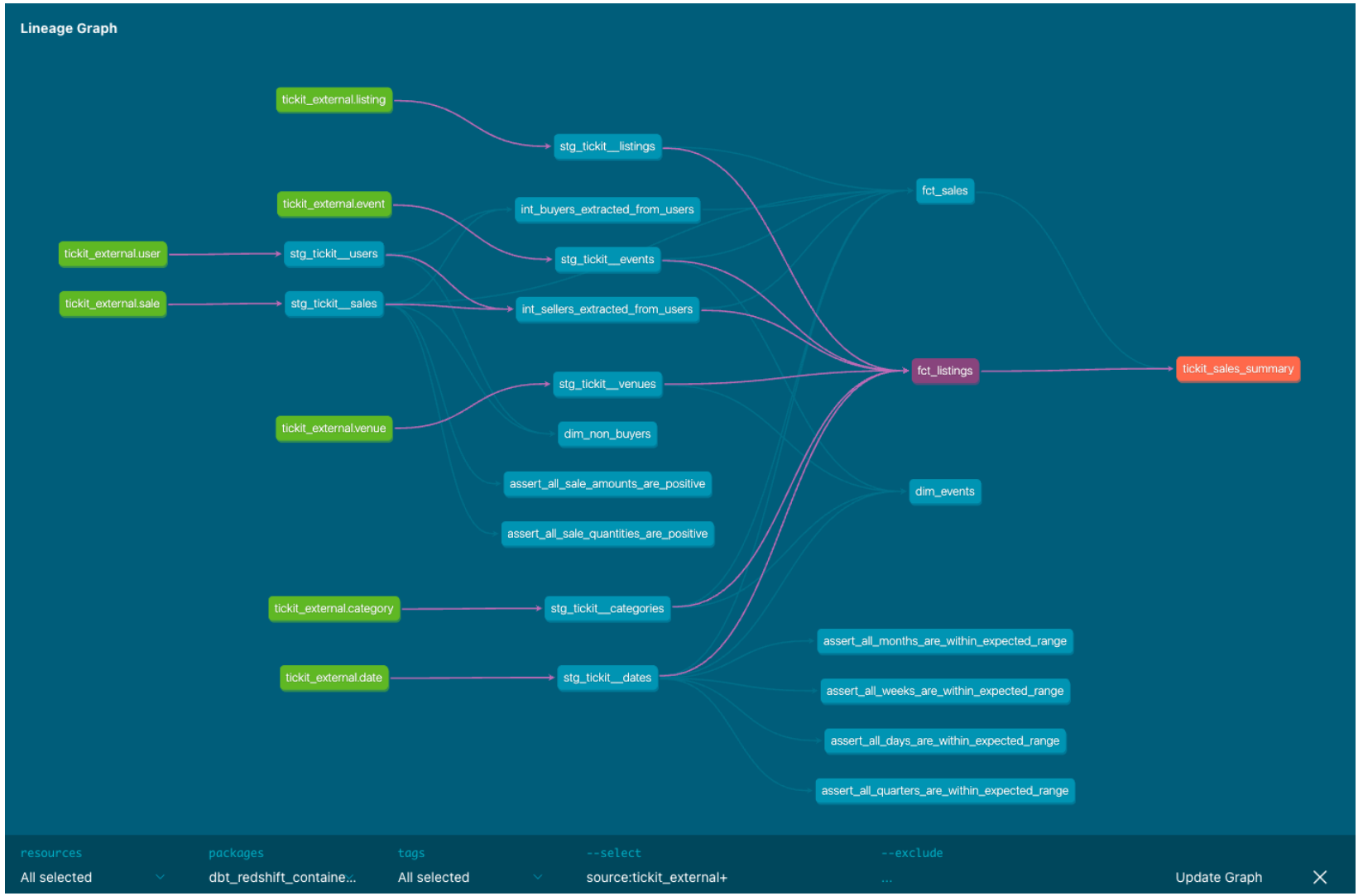
- **Amazon Athena**は、スキーマオンリードなので、**ソースデータのロードは不要**、常にライブデータをクエリできる
- **Amazon Athena**は、パーティション化されたデータも**Partition Projectionでよって直ちに読み込み可能**
- **Amazon Athena**は、**ストレージとコンピューティングが分離**されており、dbtは実行時に生成した**DAGによる並列実行と相性が良い**

dbtのデータカタログ



データカタログは、Glueデータカタログのメタデータを収集して、**キーワードで検索**したり、**階層的に表示**することで、必要なデータを素早く探し、データ活用を手助けする。
dbtは、DWHのデータモデルやデータソースによって、DWHのメタデータを定義することも可能。

dbtのデータリネージ



データリネージによって**データの関連を可視化**します。
データは、**モデルやタグなどでフィルタ（選択、除外）**したりすることで、モデルとの関連を動的に切り替え、可視化します。
データの関連を把握することで、複雑な依存関係の再発防止や、改修時の影響を把握できるようになります。
また、用途不明なデータを把握、リファクタリングにも応用することが可能です。

データ分析環境 - dbt-template

dbtは、コミュニティベースで開発が進められている**dbt-coreと dbt-athenaアダプタ**を採用。データ分析環境として、弊社のdbt-templateを用いて素早く環境を構築。

補足: dbt-templete の仕様

- AWSのサーバレスサービスのみで構成、IaC（CDK）
- CI/CD
  - Amazon CodeCommitによるコードの管理
  - Amazon CodeBuildによるイメージの作成、リポジトリの更新
  - Amazon CodePipelineによる実行環境のオーケストレーション
- データカタログやデータリネージの自動更新、公開
- コードベースのデータ検証
- スケジュール実行、リトライ、通知機能を提供
- 通信やストレージは全て暗号化

参考:  **dbt-athena** https://github.com/dbt-athena/dbt-athena

現状の課題と代替案 - Data Type

Iceberg data type	SQL data type
boolean	BOOLEAN
date	DATE
decimal(P, S)	DECIMAL(P, S)
double	DOUBLE
float	FLOAT
int	**TINYINT, SMALLINT, INT**
list	ARRAY
long	**BIGINT**
map	MAP
string	**VARCHAR, CHAR**
struct	STRUCT
timestamp	TIMESTAMP

Icebergでは、サポートしていないデータ型が存在する。データ型を置き換えることで解決。

- charやvarcharは、stringへの置き換える
- tinyintやsmallintは、intに置き換える

現状の課題と代替案 - TIMESTAMP型の精度


```
CREATE TABLE IF NOT EXISTS new_table_name
  WITH (table_type = 'ICEBERG') AS
select now() as n;
```

02:48:23 Runtime Error in model new_table_name (models/new_table_name.sql)
02:48:23 NOT_SUPPORTED: Timestamp precision (3) not supported for Iceberg. Use "timestamp(6)" instead. You may need to manually clean the data at location 's3://mybucket/s3_staging_dir/tables/5363c2d1-7009-4ef8-b8df-b10e6dca586f' before retrying. Athena will not delete data in your account.

CTASでTIMESTAMP型がエラーになる。

- 不具合ではなく、現行の仕様（今後に期待）
- dbtは、materialized=‘table’または、materialized=‘incremental‘の場合、CTASが実行され、timestamp型が含まれると左記のエラーとなるため、string型などに変更が必要

現状の課題と代替案 - partitioned byで truncate()エラー 25

dbtからIcebergのhidden partitionで**truncate()関数を使用するとエラー**になる。

- Athenaからは問題なく使えるため、恐らくdbt-athenaアダプタとの不具合の可能性がある
- dbtではなく、事前にテーブルを作成することで問題を回避できる

考慮事項 - StartQueryExecution API コールのクォータ

dbtは、DAGに基づき**threadパラメタに応じてクエリを並列実行**できるが、StartQueryExecution API コールの**クォータを超えないように設定**する必要がある

- StartQueryExecution API コールは、20/sec
- dbt-athenaアダプタのサイトでは、推奨値が、8/sec

API 名	1 秒あたりのデフォルトコール数	バーストキャパシティ
StartQueryExecution, StopQueryExecution	20	最大 80
GetQueryExecution, GetQueryResults	100	最大 200

例えば、StartQueryExecution の場合は、1 秒あたり最大 20 回呼び出すことができます。さらに、API が 4 秒間呼び出しを行わなかった場合、アカウントのバーストキャパシティーが最大 80 回まで累積されます。この場合、アプリケーションはバーストモードでこの API を最大 80 回呼び出すことができます。

引用: Amazon Athena ユーザーガイド - アカウントあたりの API コールのクォータ

考慮事項 - 100を超えるパーティションの作成には注意

Amazon Athenaには、100を超えるパーティションの作成はエラーとなる。

- 従来からのAmazon Athenaの仕様
- INSERTやCTASにおいて発生
- VACUUMも同様にエラーになるが、100パーティションまでは頑張って実行してくれるので、エラーにめげず繰り返し実行するという方法もある
- 100パーティションの制限を回避するには、一度のテーブル更新で100パーティションを超えないように工夫することで回避できなくもない
- 初期データの移行時やVACUUMなど、この制限を考えたくない場合は、Glue ETL Jobで実行する方法もある

参考: Amazon Athena ユーザーガイド - CTAS および INSERT INTO を使用して 100 パーティションの制限を回避する

まとめ

まとめ

- 現在（2023/07/27時点）、Aamzon AthenaとAWS Glueの両方が更新系クエリをサポートしているのは、Apache Iceberg のみ
- 2023/07/25、Amazon RedshiftもIcebergテーブルのクエリをサポート（Preview）のアナウンス
- Aamzon Athena（Iceberg）は、 コスト効率の良さと、DWHのような柔軟さを兼ね備えている
- ソースデータは、Partition Projectionとの組み合わせでロード不要、dbtとの相性が良い
- Apache Icebergは、 AWS GlueのSparkSQLにおいても普通のSQLのようにクエリを書ける点で使い勝手が良い
- TIMESTAMP型の精度の問題でCTASでエラーになる仕様が見直されるとIcebergとdbtの強みが活かせるので、今後に期待したい

I hope enjoy
your Amazon Athena (Iceberg) x dbt Life ♪

プロトタイプソリューションアーキテクトの募集 New!

サーバレスなデータ分析基盤のプロトタイプ開発、導入支援 👉

カジュアル面談も受付中です！ お気軽にどうぞ！

参考: データアナリティクス事業本部 - プロトタイプソリューションアーキテクトの募集要項

classmethod

プロトタイプソリューションアーキテクトの募集 New!

サーバレスなデータ分析基盤のプロトタイプ開発、導入支援 👉

カジュアル面談も受付中です！ お気軽にどうぞ！

参考: データアナリティクス事業本部 - プロトタイプソリューションアーキテクトの募集要項