[AWS Black Belt Online Seminar] Amazon Redshift Advanced Guide – 最新ベストプラクティスとアップデート

サービスカットシリーズ

Solutions Architect 大薗 純平
2020/7/29

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

大薗 純平 (おおぞの じゅんぺい)

@jostandard

アマゾン ウェブ サービス ジャパン
ソリューションアーキテクト

好きなサービス :
Amazon Redshift をはじめとした
Analytics サービス

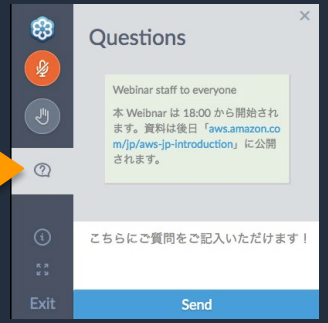
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2020年7月29日現在のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

本日のアジェンダ

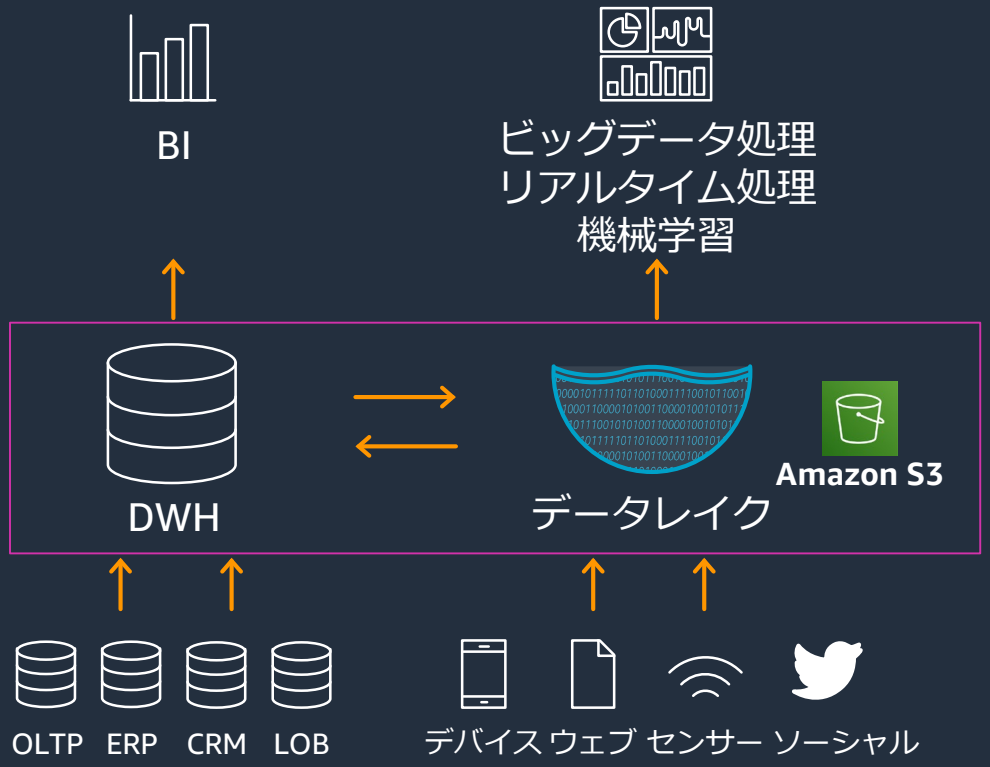
- Amazon Redshift 概要
- Amazon Redshift Spectrum を活用する
- ワークロード管理 (WLM) を活用する
- 最近のアップデート
- まとめ

Amazon Redshift 概要

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

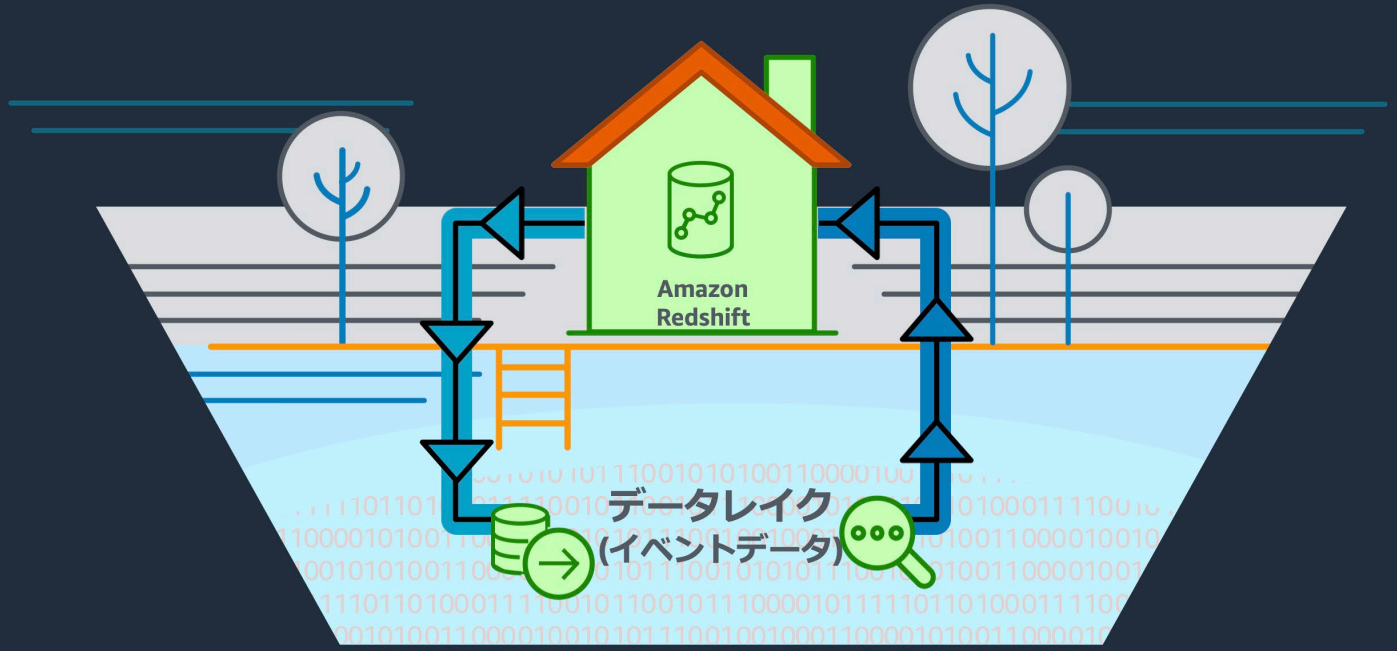
Amazon Redshift とは



Amazon Redshift

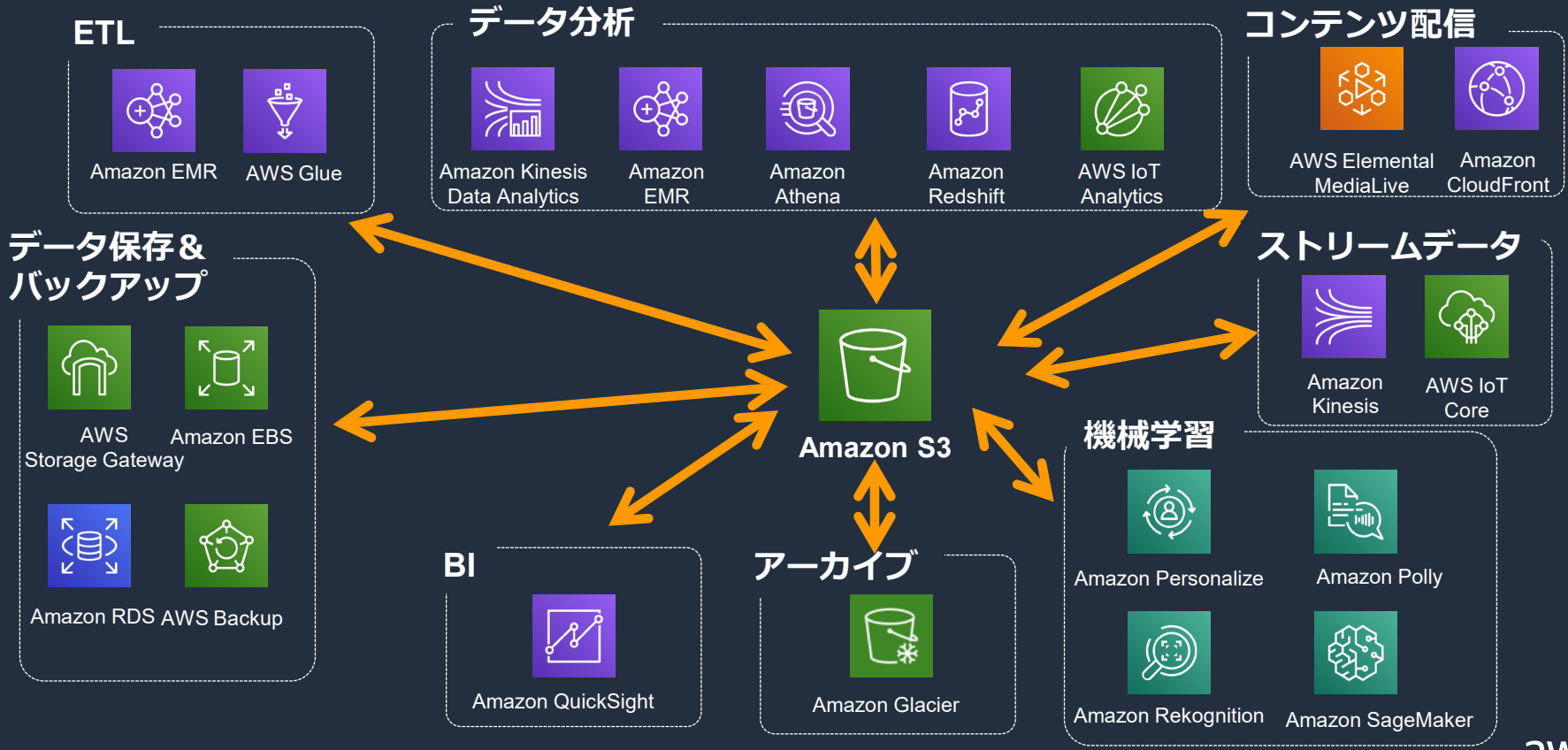
高速、スケーラブルで
費用対効果の高い
データウェアハウス
およびデータレイク
分析マネージドサービス

Amazon Redshift によるレイクハウスアプローチ



データウェアハウスはデータレイクアーキテクチャの一部であり、
データを還流させることによって他のツールとの連携もよりスムーズに行える

データレイク (Amazon S3) を中心にサービスが連携



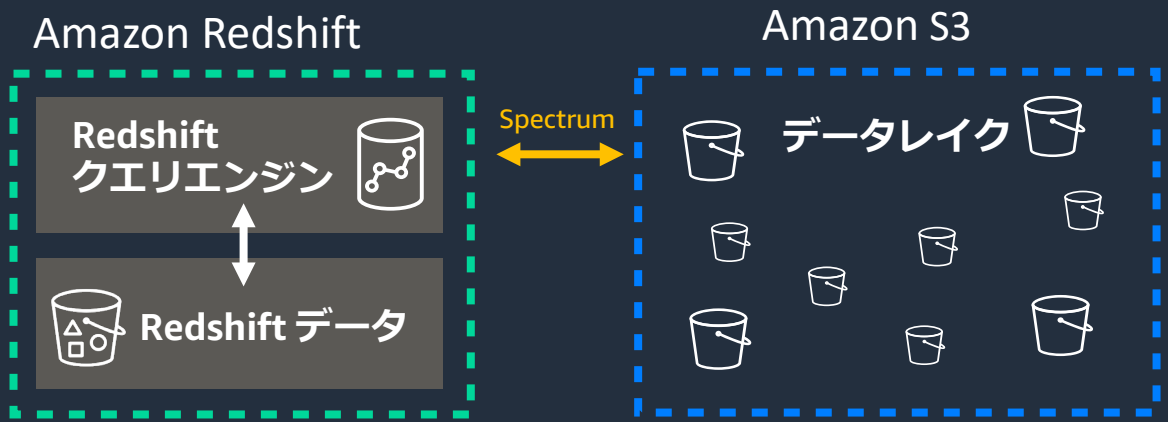
Amazon Redshift Spectrum を活用する

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

Amazon Redshift Spectrum

Redshift クエリを Amazon S3 に拡張し、データレイクアーキテクチャを強化

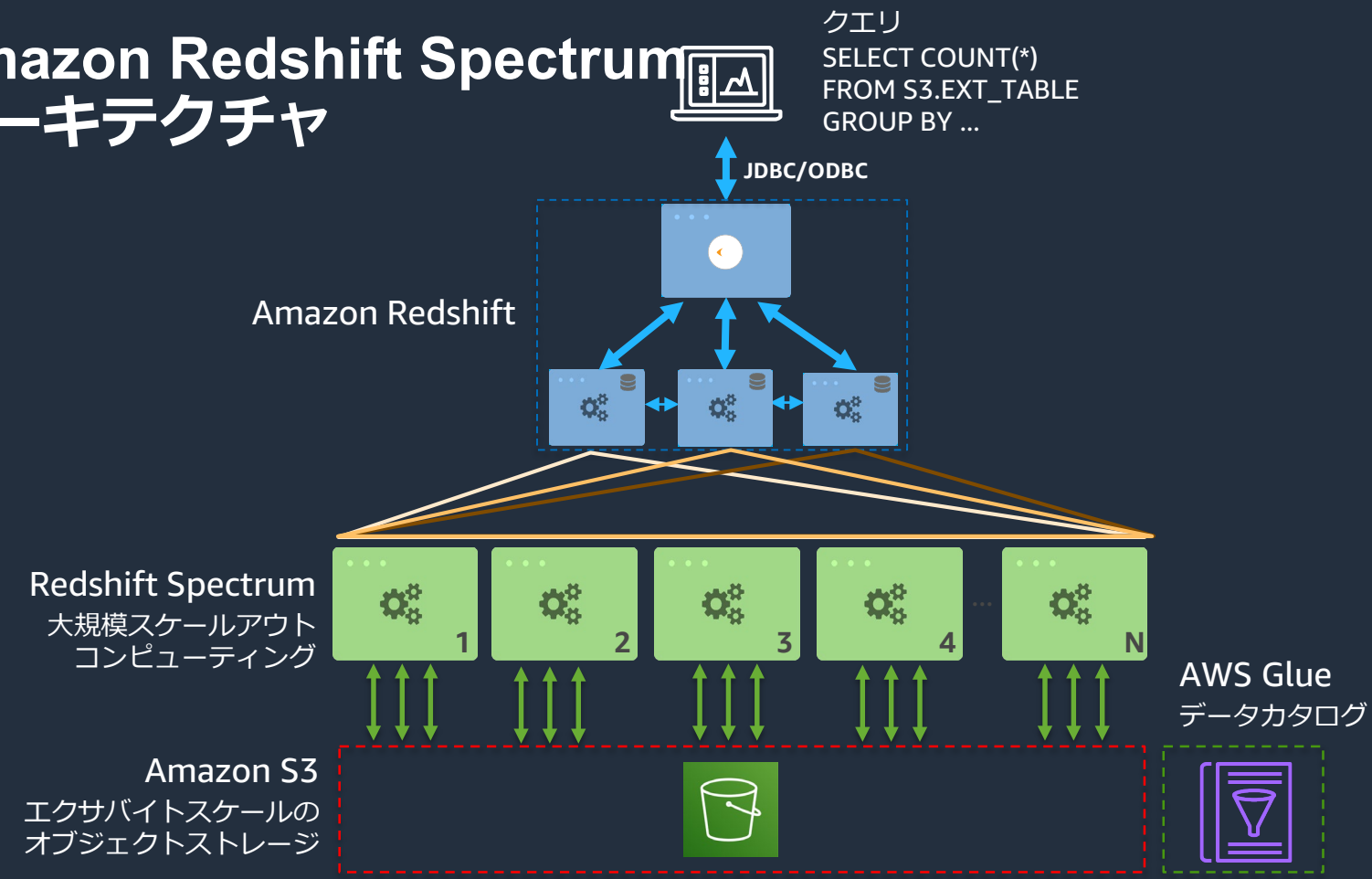


S3 上のデータに
直接クエリが可能

Redshift 上のテーブル
と S3 上のデータを結合
してクエリを実行可能

多くのオープンファイル
フォーマットをサポート

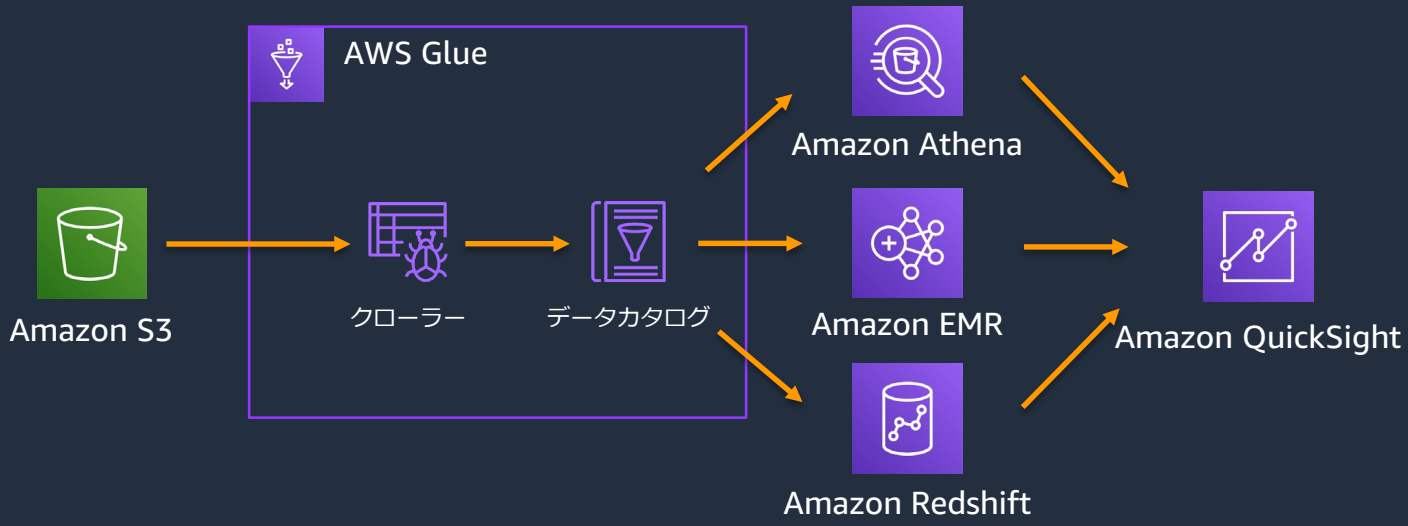
Amazon Redshift Spectrum アーキテクチャ



Spectrum の基本的な使い方は、「AWS Black Belt Online Seminar 2017 Amazon Redshift Update 最近追加された新機能と Redshift Spectrum」 をご参照
https://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-online-seminar-2017-amazon-redshift-update#13

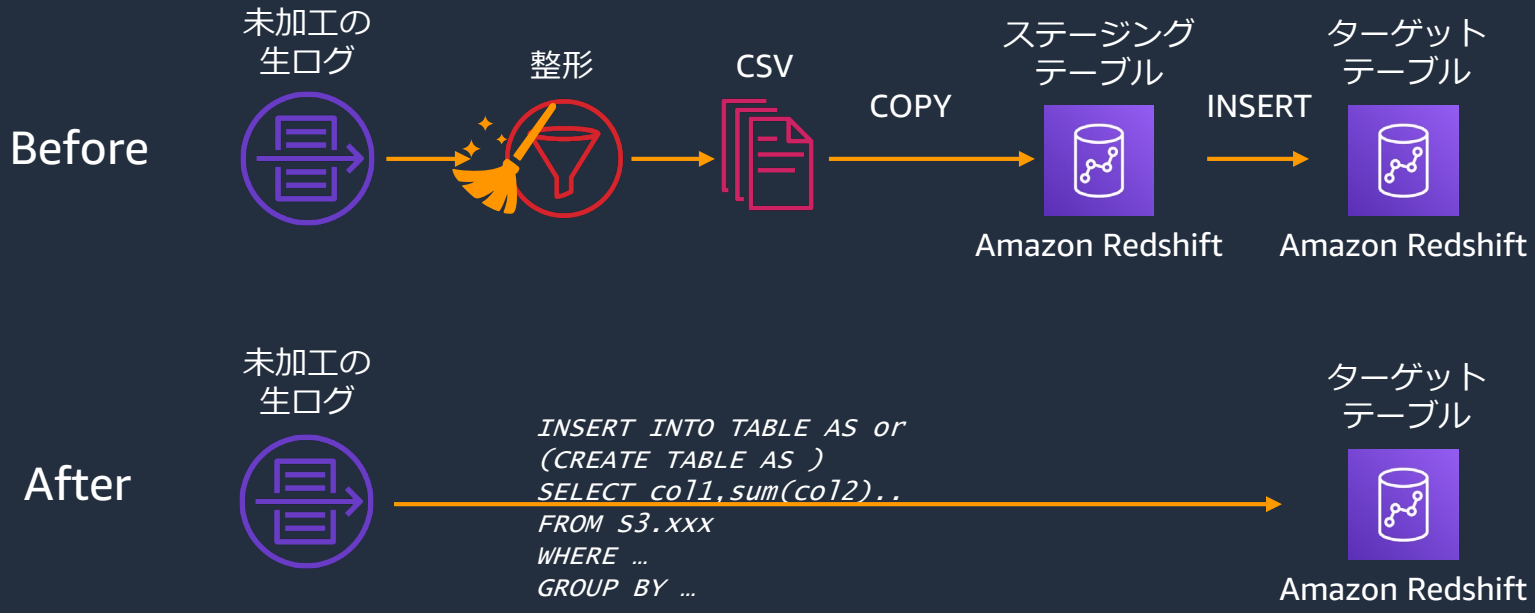
Spectrum ユースケース : データレイクアーキテクチャ

- S3 上の多様なフォーマットのデータを用途に合わせたクエリエンジンで分析
- アクセス頻度の低いデータを S3 にオフロード
- 同じデータに対して他の AWS サービスによる分析を可能に



Spectrum ユースケース : データインジェストの簡素化

- Spectrum 外部テーブル経由の SQL を活用し、 Redshift にデータロードする前のデータ加工と変換ステップをシンプルに



Spectrum ユースケース : ETL ツールとして活用

- Amazon Redshift を効率的でコスト効果の高い ETL ツールとして活用
- シンプルな SQL で CSV から Parquet へ変換
- Redshift は Pause & Resume 機能を活用

Amazon Redshift Spectrum ベストプラクティス

1. ファイル/テーブル属性
2. パーティションプルーニング
3. Min/Max プルーニング
4. Spectrum 層へのプッシュダウン
5. データレイアウト
6. 実行計画の最適化
7. Redshift クラスターサイズ

1. ファイル/テーブル属性

- ファイルフォーマット
    - 列指向フォーマットを選択(Parquet/ORC)
        - ファイル内の列のブロックの単位で圧縮し、ファイル全体で圧縮しないようにする
    - 必要な列のみの選択の徹底
        - コスト抑制とパフォーマンス向上
- 最適なファイルサイズ
    - >= 128MB(列指向), >= 64MB(行指向)
- 適切なデータ型を選択
    - string Vs varchar(<max length>)

2. パーティションプルーニング

- 頻繁にフィルターされる列でパーティショニングする
  - 不要なパーティションのスキャンをスキップできるため、クエリパフォーマンスが向上
  - コストの削減にも寄与する
  - パーティションキーは時系列が一般的
  - パーティションの粒度を細かくしすぎない
    - コストは抑制できるがクエリパフォーマンスに悪影響を及ぼす可能性がある

year/month　　year/month/day/hour
20*12 → 240　Vs　20*12*30*24 → 172,800

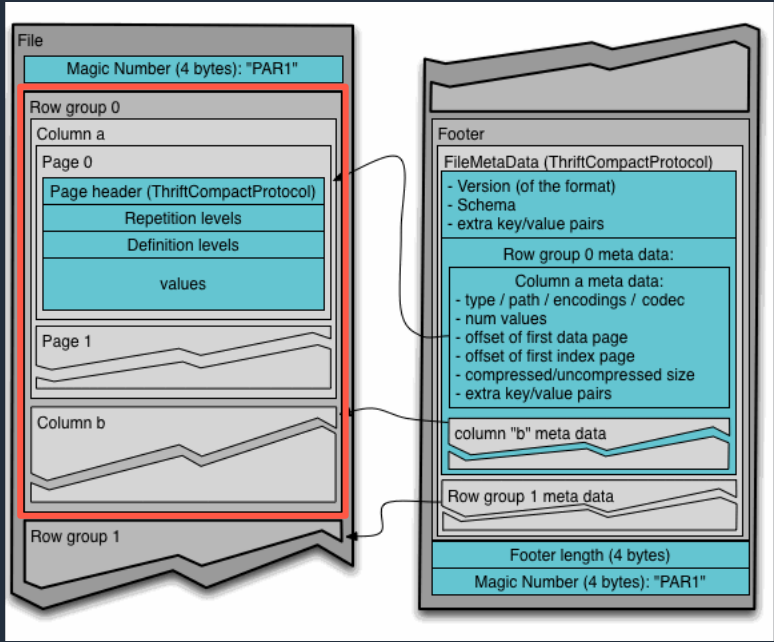
svl_s3partition_summary の total_partitions と qualified_partitions をチェック

3. Min/Max プルーニング

- Parquet, ORC など列指向フォーマットの最小/最大値の統計を活用する
- 処理不要な Row Group / Stripe はスキャンをスキップ可能
    - コストを抑え、性能を向上
- つまり Redshift ローカルテーブルのゾーンマップ(*)と同じ考えが適用できる
- 頻繁に絞り込みを行うカラムでデータをソートしておくと効果的

Parquet ファイルフォーマット

https://parquet.apache.org/documentation/latest/



* 20200318 AWS Black Belt Online Seminar Amazon Redshift をご参照

https://www.slideshare.net/AmazonWebServicesJapan/20200318-aws-black-belt-online-seminar-amazon-redshift#35

4. Spectrum 層へのプッシュダウン

- Spectrum にプッシュダウン可能な処理を意識する
- 多くの集計関数
    - COUNT, SUM, AVG, MIN, MAX, GROUP BY など
- 文字列関数
    - regex_replace, to_upper, date_trunc など
- 等価条件、LIKE、IS NULL、CASE WHEN など
- 単一カラムの DISTINCT は Spectrum にプッシュダウンされるが、
  複数カラムの DISTINCT は GROUP BY に書き換える必要あり

クエリの処理がプッシュダウンされていることを確認するために
`SVL_S3QUERY_SUMMARY` の `s3_scanned_rows / s3query_returned_rows` をチェック

5. データレイアウト

- 小さなマスターテーブル/頻繁にクエリされるデータは Redshift 内部へ、ファクトテーブル/アクセス頻度の低いデータは S3 へ
- ホットデータを Redshift 内部へ、ウォーム/コールドデータを S3 へ
- Redshift 内部テーブルと Spectrum 外部テーブル を Union する Late binding view を活用し、ホットデータとウォーム/コールドデータを 1 つのクエリで分析

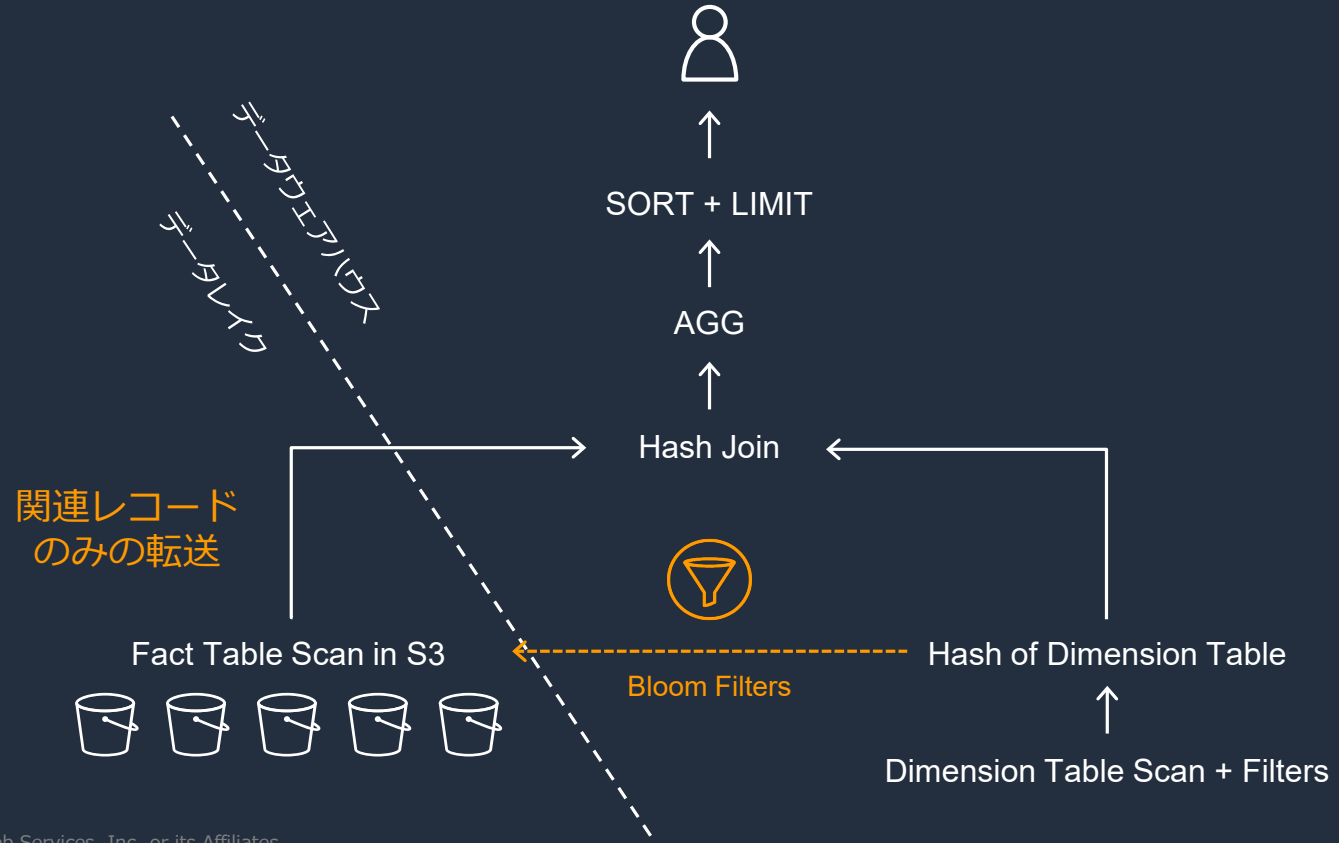
Late binding view

```
CREATE VIEW total_sales as
SELECT … FROM sales_last_month
UNION ALL
SELECT … FROM s3_external_schema.sales_historical
WITH NO SCHEMA BINDING;
```

Bloom filters で Spectrum の結合クエリの性能が向上

外部テーブルと内部テーブルの結合を効率化して最大 2 倍高速に

New



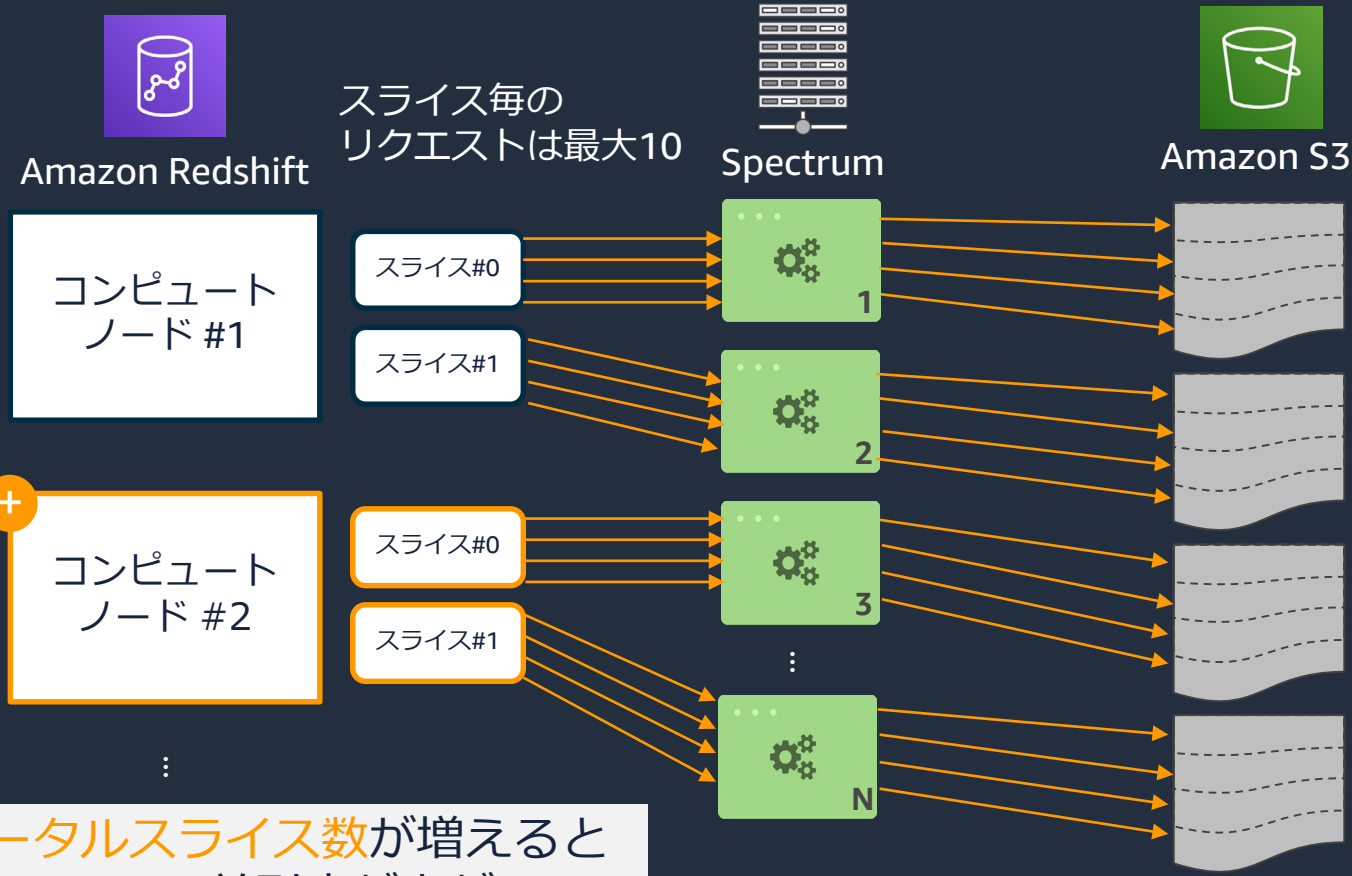
6. 実行計画の最適化

- テーブルプロパティの numRows を更新する
  - 結合順などを最適化

```
ALTER TABLE spectrum_schema.event SET TABLE PROPERTIES
('numRows'='122857504');
```

- コストとパフォーマンスを常に意識してクエリを書く
  - S3 からのスキャンを最小化
    - 必要なカラムのみの選択
    - 必要に応じてパーティショニングする
  - Redshift ノードに転送するデータを極小化
    - 効率的なアグリゲーションとフィルタ

7. Redshift クラスターサイズ



Split 可能なファイル
および圧縮形式の場合、
内部的に分割して
読み込む可能性がある

トータルスライス数が増えると
Spectrum の並列度が上がる

7. Redshift クラスターサイズ

- パフォーマンスを最適化するために正しく Redshift クラスターを構成する
  - Redshift の各スライスは Spectrum に対して並列にリクエストを投入
  - クラスターのトータルスライス数の合計がパフォーマンスに影響

```
SELECT query, files, splits, avg_request_parallelism
FROM SVL_S3QUERY_SUMMARY WHERE query= xxxxx;

query | files | splits | avg_request_parallelism
------+-------+--------+------------------------
xxxxx | 112   | 328    | 5
```

  - avg_request_parallelism * クラスターのトータルスライスの値よりも Splits 数の合計のほうが少ない場合、Redshift のノードを増やしても Spectrum のパフォーマンスは向上しない可能性が高い

Redshift クラスターのトータルスライス数の確認

- STV_SLICES テーブルより確認可能

例

```
select node, count(slice) from stv_slices group by node order by node;
 node | count
------+-------
    0 | 16
    1 | 16
    2 | 16
    3 | 16
    4 | 16
    5 | 16
    6 | 16
    7 | 16
(8 rows)
```

8 ノードで構成されており、
ノード毎に 16 スライスあることが確認できる

データレイクエクスポート

Amazon Redshift

Redshift テーブル上の過去データをParquet ファイルに変換

※テーブル定義は別途作成する(手動, Glue クローラー等)

Sales (Table)

```
UNLOAD
('select * from Sales where
sales_date = YYYYMMDD')
TO 's3://mybucket/unload/Sales/'
FORMAT as PARQUET
PARTITION BY (sales_date);
```

Amazon S3

Parquet

CTAS で簡単に Spectrum 外部テーブルを作成

New

Amazon Redshift

CTAS(Create Table As Select) で Redshift テーブルデータから Spectrum 用の外部テーブルを容易に作成可能

Sales (Table)

```
CREATE EXTERNAL TABLE
spectrum_schema.Sales_history
PARTITION BY (sales_date)
STORED as PARQUET
LOCATION
's3://mybucket/unload/Sales/'
AS select * from Sales
;
```

パーティション指定

ファイル形式指定

S3 Path 指定

Amazon S3

Sales_history (External Table)

CTAS で簡単に Spectrum 外部テーブルを作成


```
CREATE EXTERNAL TABLE spectrum.supplier_new stored as parquet
location 's3://mybucket/unload/parquet/supplier_ctas/'
AS SELECT * FROM supplier WHERE s_suppkey BETWEEN 1 and 5000000;
INFO:  5000000 record(s) exported successfully.
CREATE EXTERNAL TABLE
```

テーブルのプロパティ | EXTERNAL | TRUE | numRows | 5000000 | transient_lastDdlTime | 1595972202

Glue Data Catalog 上に
メタデータ(件数)も自動登録

スキーマ

	列名	データ型	パーティションキー	コメント
1	s_suppkey	int		
2	s_name	char(25)		
3	s_address	varchar(40)		
4	s_nationkey	int		
5	s_phone	char(15)		
6	s_acctbal	decimal(15,2)		
7	s_comment	varchar(101)		

Spectrum 外部テーブルへの書込(データ追加)も可能

New

Amazon Redshift

通常の INSERT INTO 構文で Spectrum 用の外部テーブルに対して Redshift テーブルデータを追加可能

Sales (Table)

```
INSERT INTO
spectrum_schema.Sales_history
select * from Sales
where sales_date = YYYYMMDD
;
```

Amazon S3

Sales_history (External Table)

Spectrum 外部テーブルへの書込(データ追加)も可能

New

```
INSERT INTO spectrum.supplier_new
SELECT * FROM supplier WHERE s_suppkey BETWEEN 5000001 AND 10000000;
INFO:  5000000 record(s) exported successfully.
INSERT
```

テーブルのプロパティ | EXTERNAL | TRUE | numRows | 10000000 | transient_lastDdlTime | 1595972619

Glue Data Catalog 上の
メタデータ(件数)も自動更新

スキーマ

	列名	データ型	パーティションキー	コメント
1	s_suppkey	int		
2	s_name	char(25)		
3	s_address	varchar(40)		
4	s_nationkey	int		
5	s_phone	char(15)		
6	s_acctbal	decimal(15,2)		
7	s_comment	varchar(101)		

マテリアライズドビューが外部テーブルを参照可能に

New

Amazon Redshift

Redshift Spectrum 外部テーブルのデータを
マテリアライズドビュー化

Sales_summary
(Materialized View)

```
CREATE MATERIALIZED VIEW
sales_summary
AS select sales_date, sum(sales)
FROM
 spectrum_schema.sales_history
GROUP BY sales_date;
```

Sales_history
(External Table)

Amazon S3

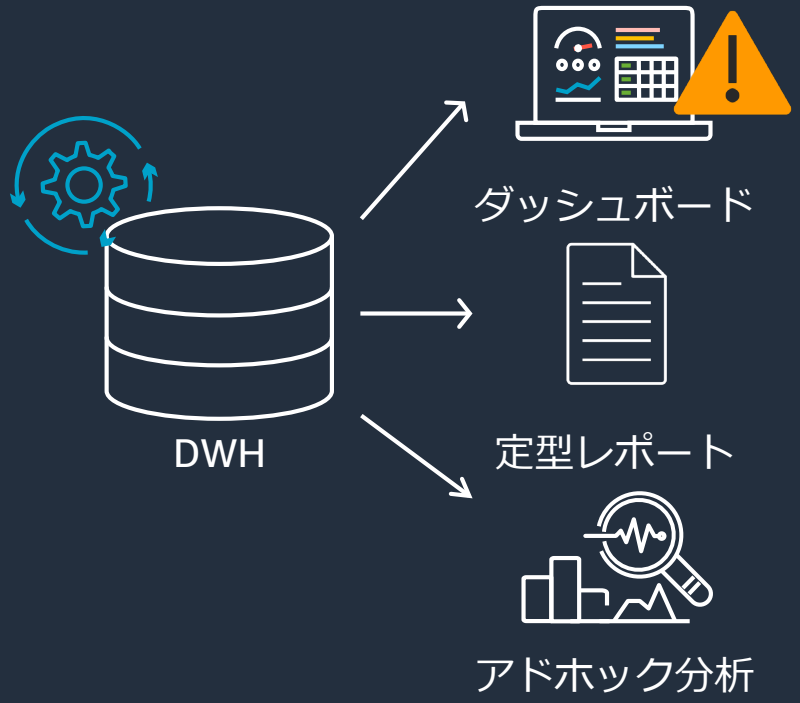
ワークロード管理 (WLM) を活用する

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

DWH のワークロード管理

- DWH に対するクエリには様々なものがあり、それぞれ処理される時間帯や特性、期待される性能 (SLA) は異なる
- すべての処理を平等に扱うべきか？
- 処理に応じて適切なリソースを割り当てたり、優先すべき処理が他の処理によって妨げられないよう **ワークロードを管理できる仕組みは重要**



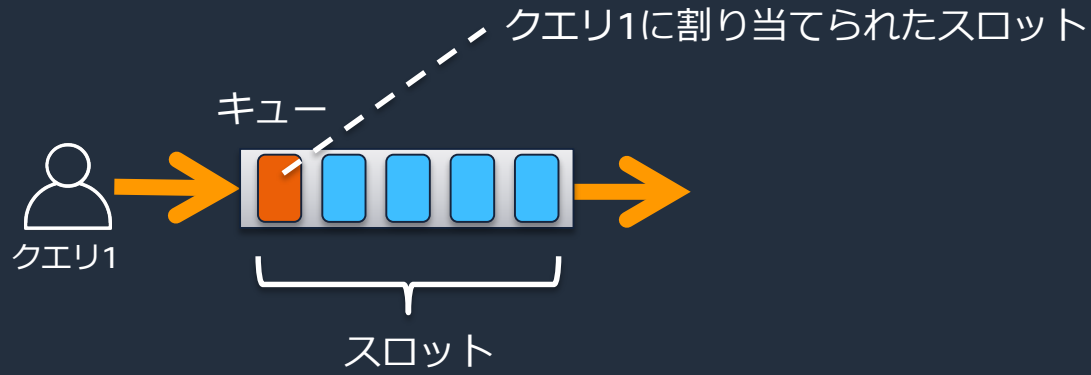
WLM (ワークロード管理) とは

- Redshift がクエリに対してどのようにリソース(メモリ/CPU/Disk IO)を割り当てるかを決定づけるしくみ
- クエリが実行されると、必ず待ち行列 (キュー) にアサインされる
- キューはデフォルトで用意されているが、別途作成して増やすことも可能で、クエリの種類(ユーザーグループ、クエリグループ)に応じてアサイン先を振り分けることが可能

WLM の基本

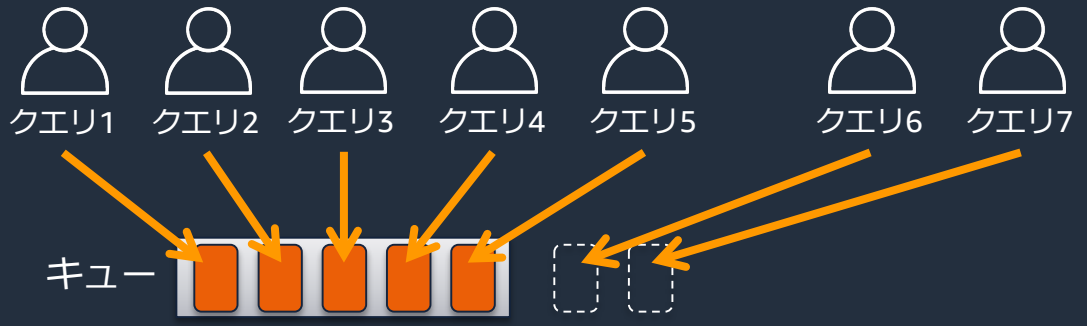
- クエリには、キューで利用可能なリソース (スロット) が割り当てられる
- 通常 1 スロットが割り当てられるが、クエリ実行前にセッション内で
  以下のコマンドを実行することで、複数スロットを割り当てることも可能

```
set WLM_QUERY_SLOT_COUNT to N; // N <= キューの総スロット数
```



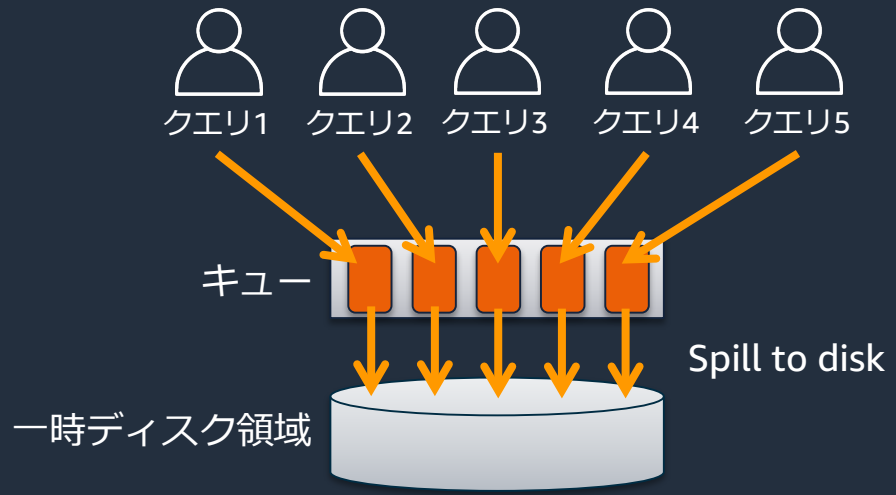
WLM の基本

- キューで利用できるスロットがクエリで専有された場合、
後続のクエリは、実行中のクエリが完了もしくはタイムアウトして
スロットが空くまで実行待ち状態になる(FIFO キュー待ち)



WLM の基本

- クエリ処理がスロットに割り当てられたメモリに収まらない場合、
  他のスロットは利用せず、一時ディスク領域を利用して処理を実行
- より低速な Disk IO ベースの処理になるため、性能が劣化してしまいがち



一時ディスク領域を使用したクエリの特定

- 分析クエリやバッチ実行中にディスク使用率が増加し、処理終了後には元に戻るといったケースは、多くの場合クエリが複雑でメモリ溢れし、処理が一時ディスクに落ちている可能性が高い

- システムビュー **SVL_QUERY_SUMMARY(*)** の **is_diskbased** 列に 't' が含まれているクエリは WLM のメモリを使い切って一時ディスク領域を使っているクエリ

- より詳細な情報を得るにはシステムテーブル **STL_QUERY_METRICS(*)** の **max_blocks_to_disk** を確認

* 該当システムテーブル/ビューの詳細は以下をご参照
https://docs.aws.amazon.com/redshift/latest/dg/r_SVL_QUERY_SUMMARY.html
https://docs.aws.amazon.com/redshift/latest/dg/r_SVL_QUERY_METRICS.html

手動 WLM と自動 WLM

WLM には 手動 WLM と 自動 WLM があり、クラスター単位でどちらを使用するかを決定する(それぞれの機能は混在して使うことができない)

手動 WLM

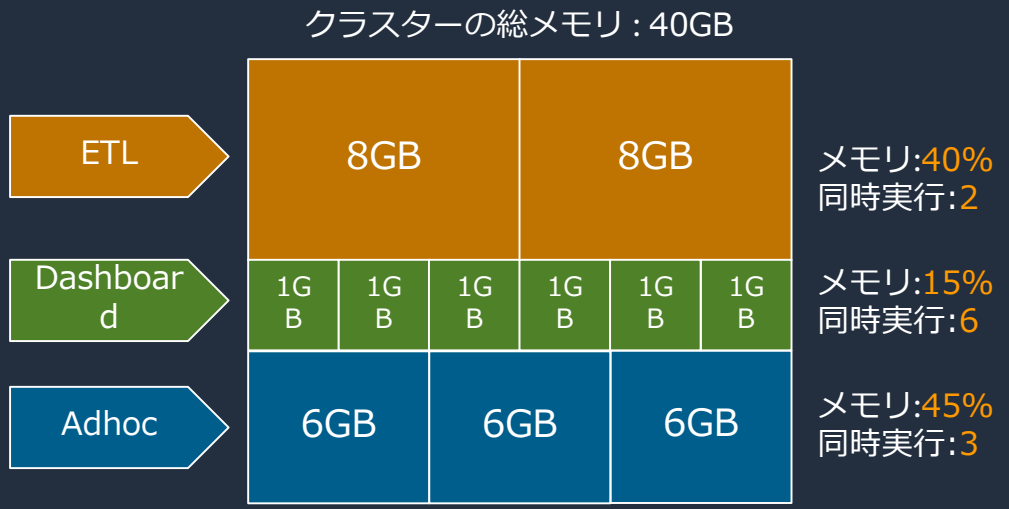
- キューの**メモリ配分**や**同時実行数**を静的に設定する
- 各ワークロードの使用メモリ領域を明確に分離できる
- 処理が一時ディスクに落ちないよう、適正な値を調整をしていく必要あり
- すべての処理で CPU, Disk IO リソースは平等に扱われる

自動 WLM (デフォルト)

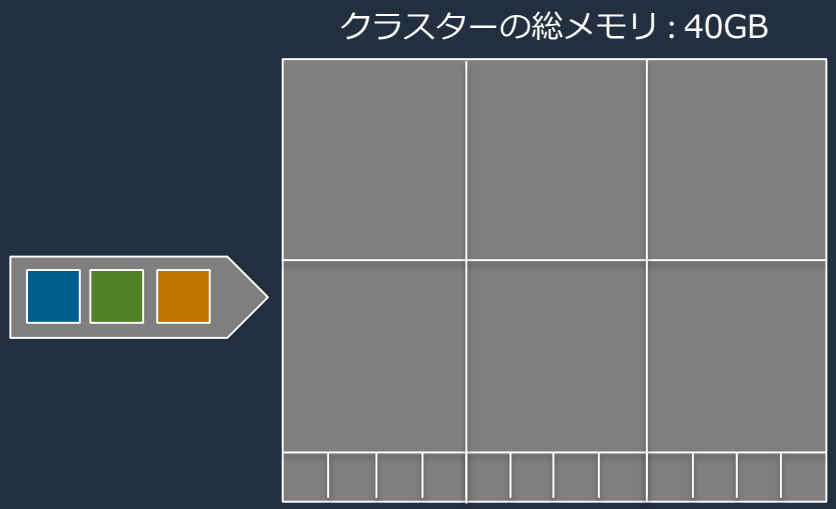
- クラスター全体で柔軟にリソースを使用し、メモリ配分や同時実行数は Redshift が自動で管理
- キューを**優先度**ごとに分け、与える CPU, Disk IO リソースに差をつけることが可能

手動 WLM と自動 WLM

手動 WLM



自動 WLM (デフォルト)



手動 WLM と自動 WLM - どちらを選択すべきなのか？

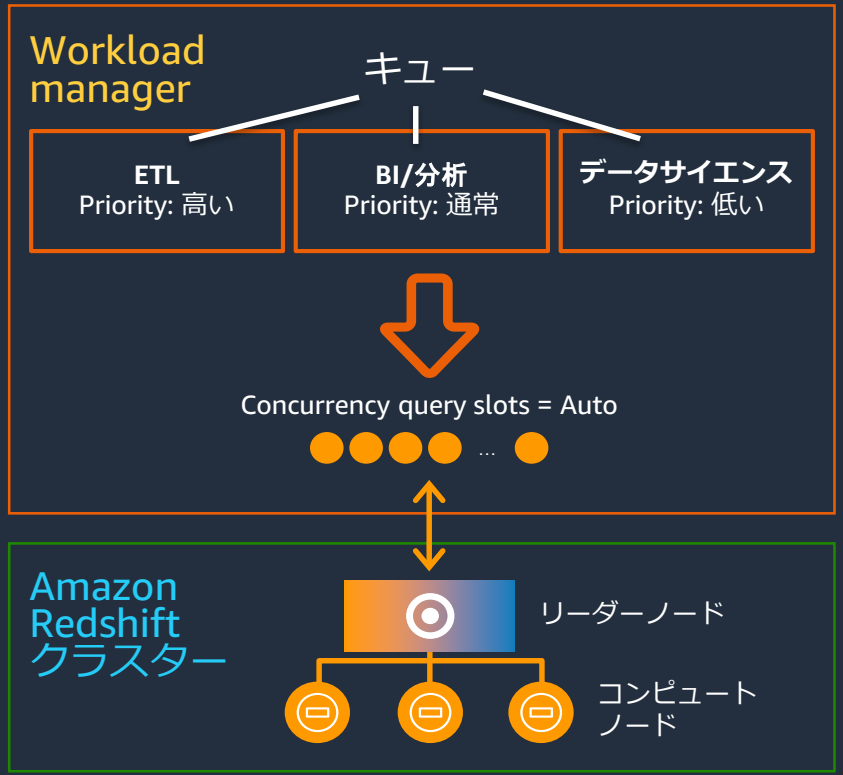
手動 WLM を選択したほうがいい場合

- 既に手動 WLM を使う形で検証済みで、問題が発生していない場合
- ある程度ワークロードが予測できており、各クエリに必要なメモリ容量、同時実行数が定義できる場合

自動 WLM を選択したほうがいい場合

- 手動 WLM によってリソースを適切に使い切れていない場合
- ワークロードの予測がつかず、手動で設計していくのが困難な場合
- 自動 WLM を使う場合、**クエリ優先度の機能を活用すべし**

自動 WLM とクエリ優先度



ビジネスの優先度に応じてキューを分け、処理の優先度を設定

優先度の高い処理には、自動的に多くの CPU, IOリソースを割り当て

優先度はコマンドまたは QMR によって動的に変更が可能(後述)

簡単な設定で、
高い SLA が求められる処理を
優先的に処理することが可能

自動 WLM とクエリ優先度

- ワークロード毎にキューを分け、相対的な優先度を設定することが可能
  - 最高
  - 高い
  - 通常 (デフォルト)
  - 低い
  - 最低

- 優先度は以下のコマンドまたはクエリモニタリングルール (QMR) によって動的に変更が可能
  - CHANGE_QUERY_PRIORITY
  - CHANGE_SESSION_PRIORITY
  - CHANGE_USER_PRIORITY

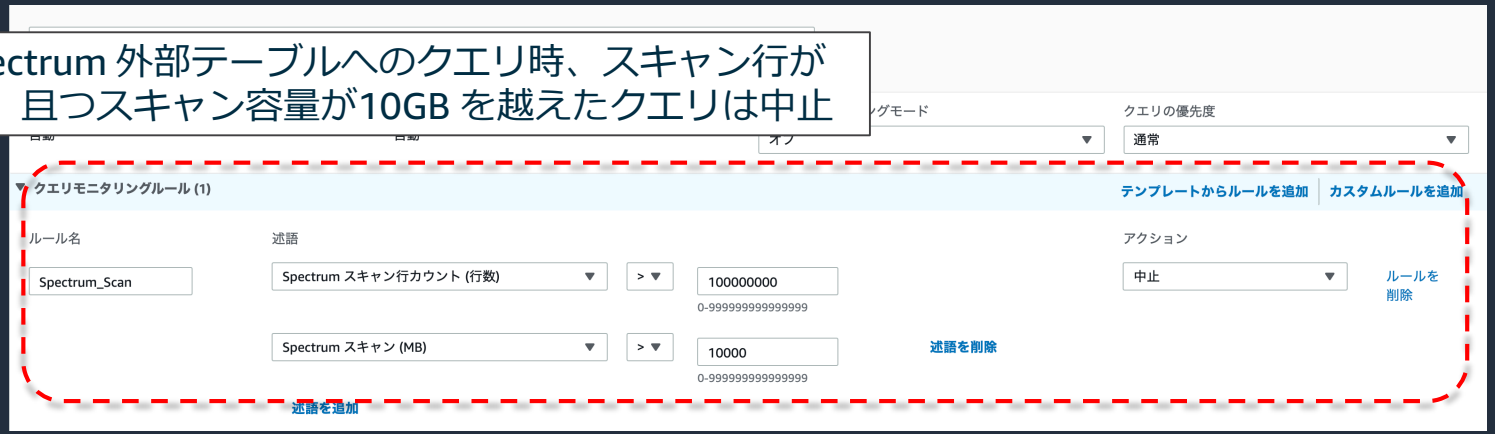
※コマンドを利用すると、上記 5 つの優先度に加えてスーパーユーザーのみが利用できる、“最高”よりも高優先度な "Critical” も設定可能

* 該当コマンドの詳細は以下をご参照
https://docs.aws.amazon.com/redshift/latest/dg/r_CHANGE_QUERY_PRIORITY.html
https://docs.aws.amazon.com/redshift/latest/dg/r_CHANGE_SESSION_PRIORITY.html
https://docs.aws.amazon.com/redshift/latest/dg/r_CHANGE_USER_PRIORITY.html

クエリモニタリングルール (QMR)

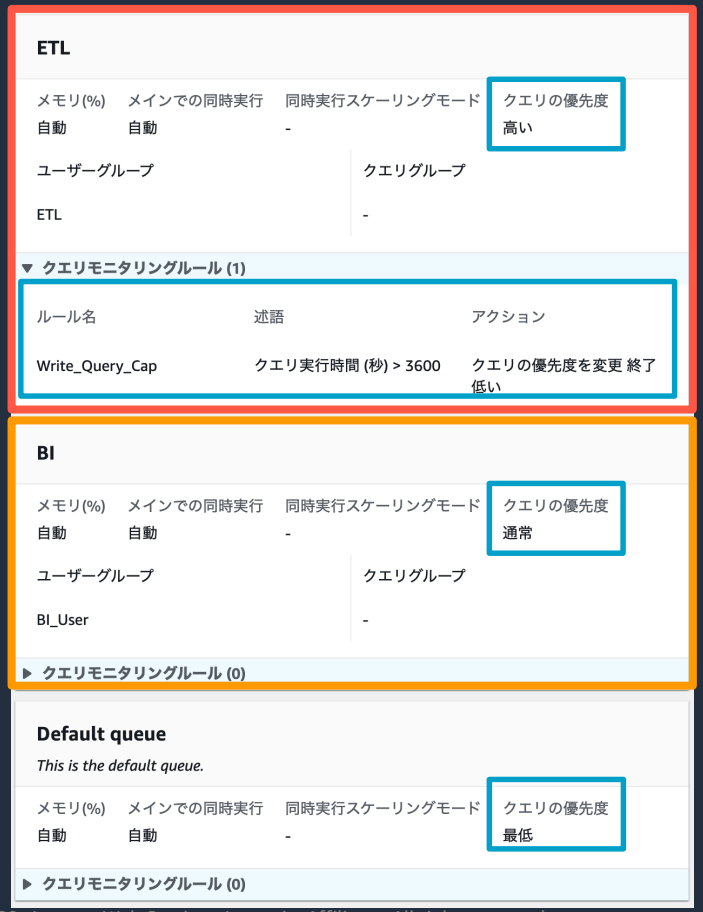
- キューには、クエリに関するルールを 25 まで作成し、そのルールの境界を越えたクエリへのアクションを定義することが可能
  - ルール : CPU 利用率、CPU 時間、ジョイン対象行数、Spectrumスキャンなど
  - アクション : LOG(ログ), ABORT(中止), Change Priority(クエリの優先度を変更) *1, HOP (ホップ) *2

例 : Spectrum 外部テーブルへのクエリ時、スキャン行が 1 億件、且つスキャン容量が10GB を越えたクエリは中止



*1 – 自動 WLM の場合のみ選択できるアクション
*2 – 手動 WLM の場合のみ選択できるアクション

自動 WLM とクエリ優先度の設定例



ETL 用キュー

- 遅延が許されず、常に一定の SLA が要求される ETL 処理の優先度は「高い」に
- ただし、処理に 1 時間以上かかった場合は優先度を「低い」に落とす (QMR で設定)

BI 用キュー

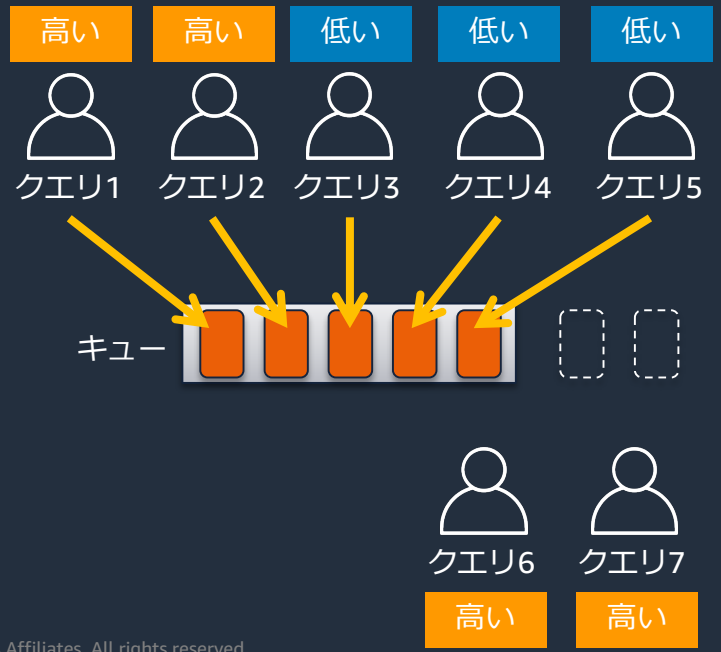
- BI ツールによる分析処理の優先度は「通常」に

その他

- 上記以外の処理の優先度は「最低」に

クエリ優先度の動き

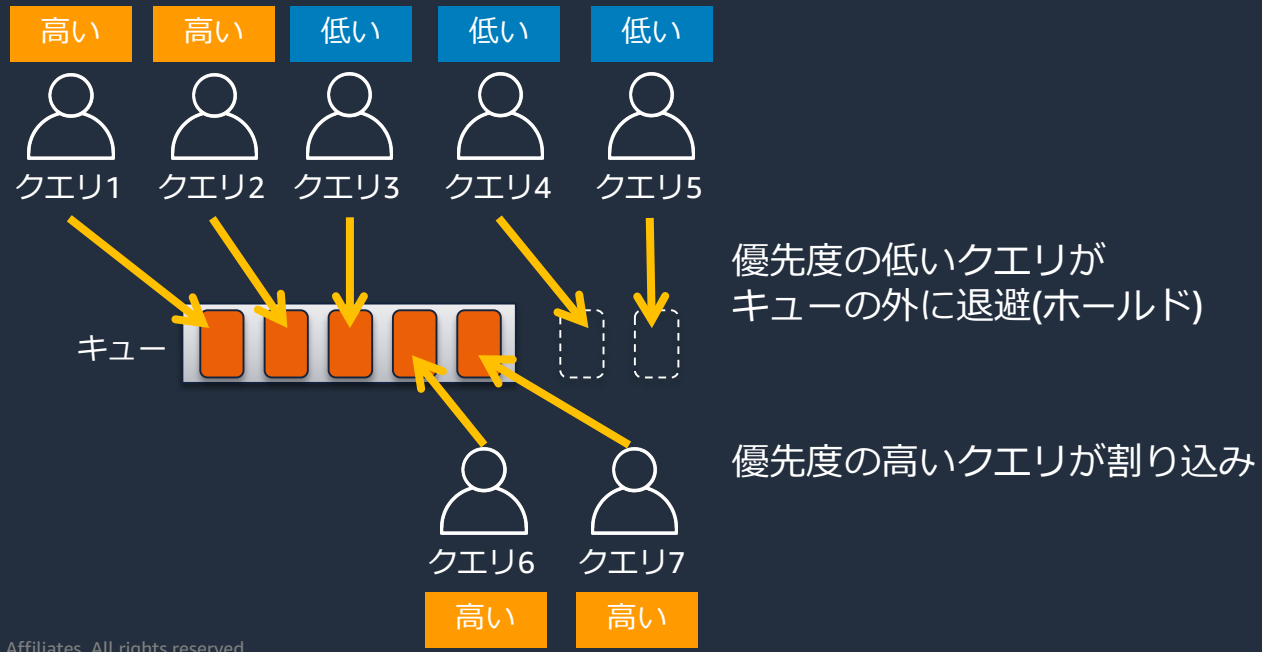
キューで実行されているクエリよりも高い優先度のクエリがきた場合は？



※手動 WLM の場合は FIFO になる

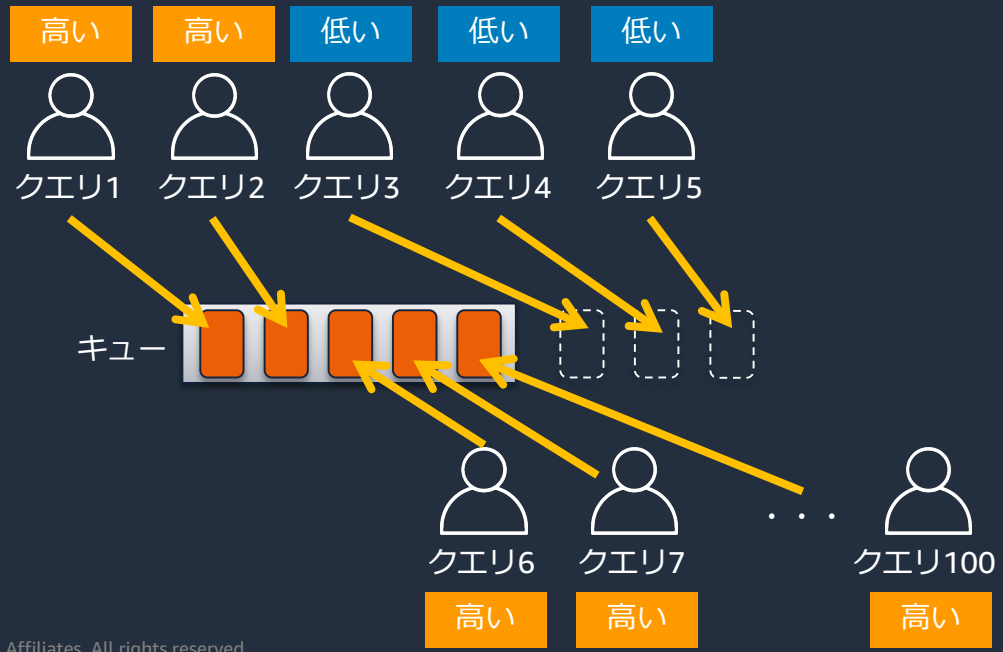
クエリ優先度の動き

キューで実行されているクエリよりも高い優先度のクエリがきた場合は？
＝＞優先度の高いクエリが優先して処理される



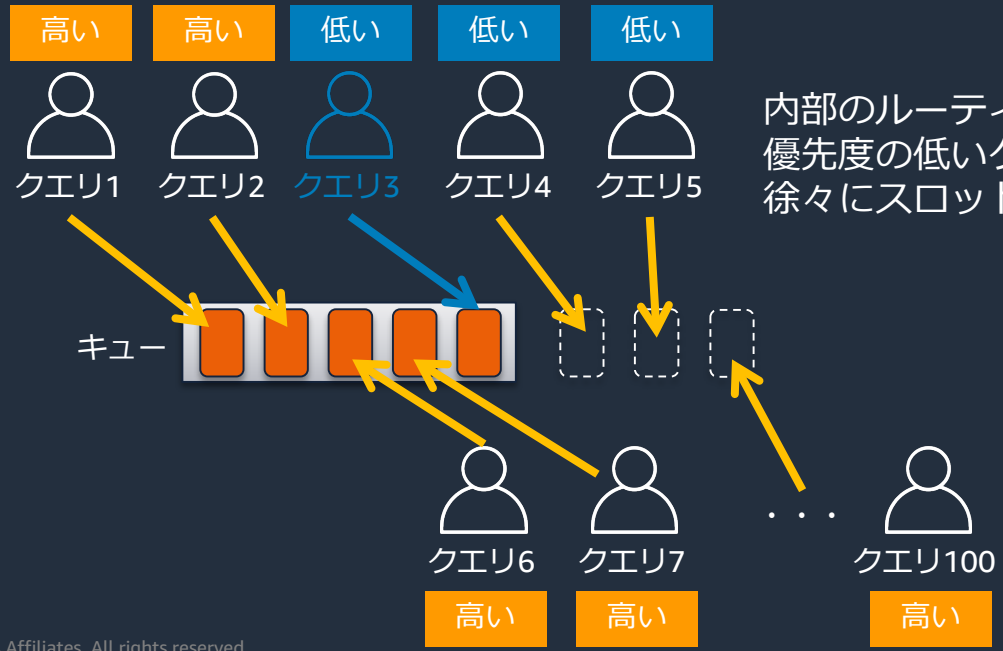
クエリ優先度の動き

優先度の高いクエリが間断なくくる場合、優先度の低いクエリはどうなるか？



クエリ優先度の動き

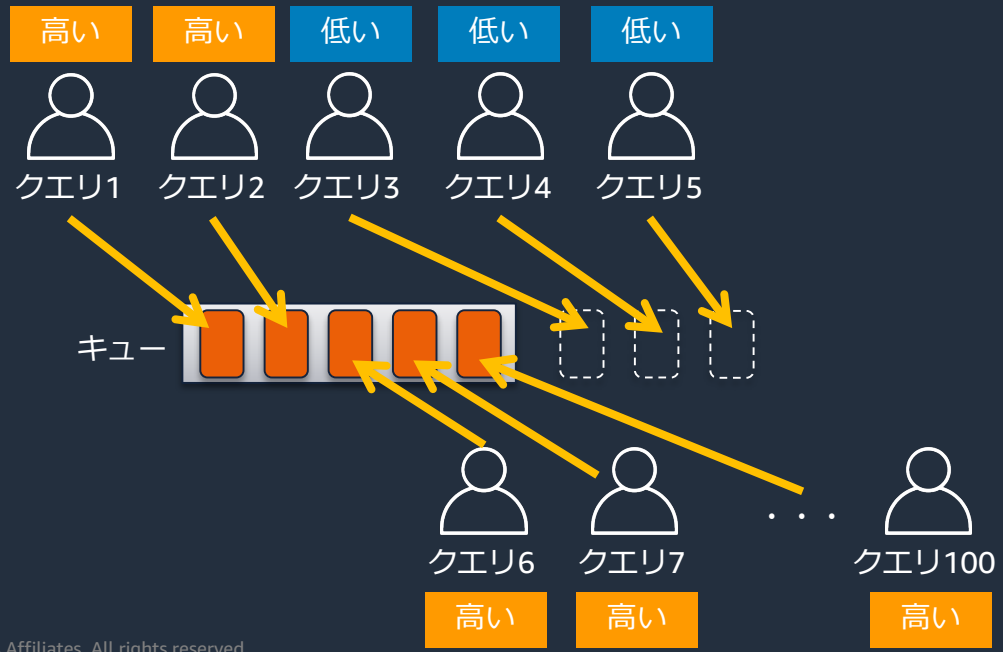
優先度の高いクエリが間断なくくる場合、優先度の低いクエリはどうなるか？
＝＞優先度の低いクエリが永遠にキュー待ちになることがないよう動作する



内部のルーティングアルゴリズムに則り、
優先度の低いクエリも時間をかけて
徐々にスロットにアサインされていく

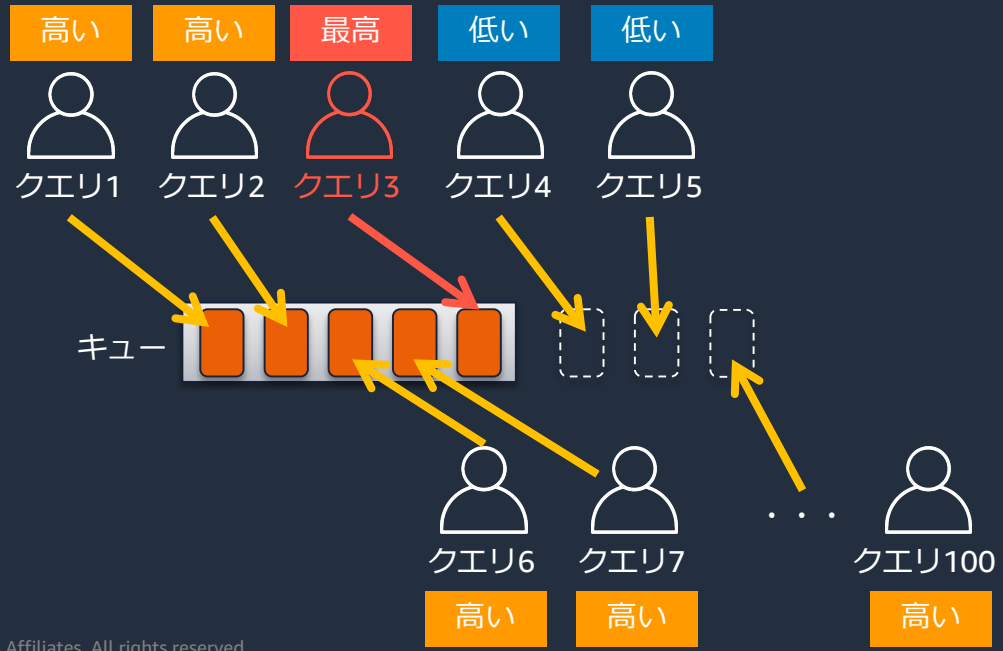
クエリ優先度の動き

優先度の低いクエリに、より速くスロットをアサインするためには？



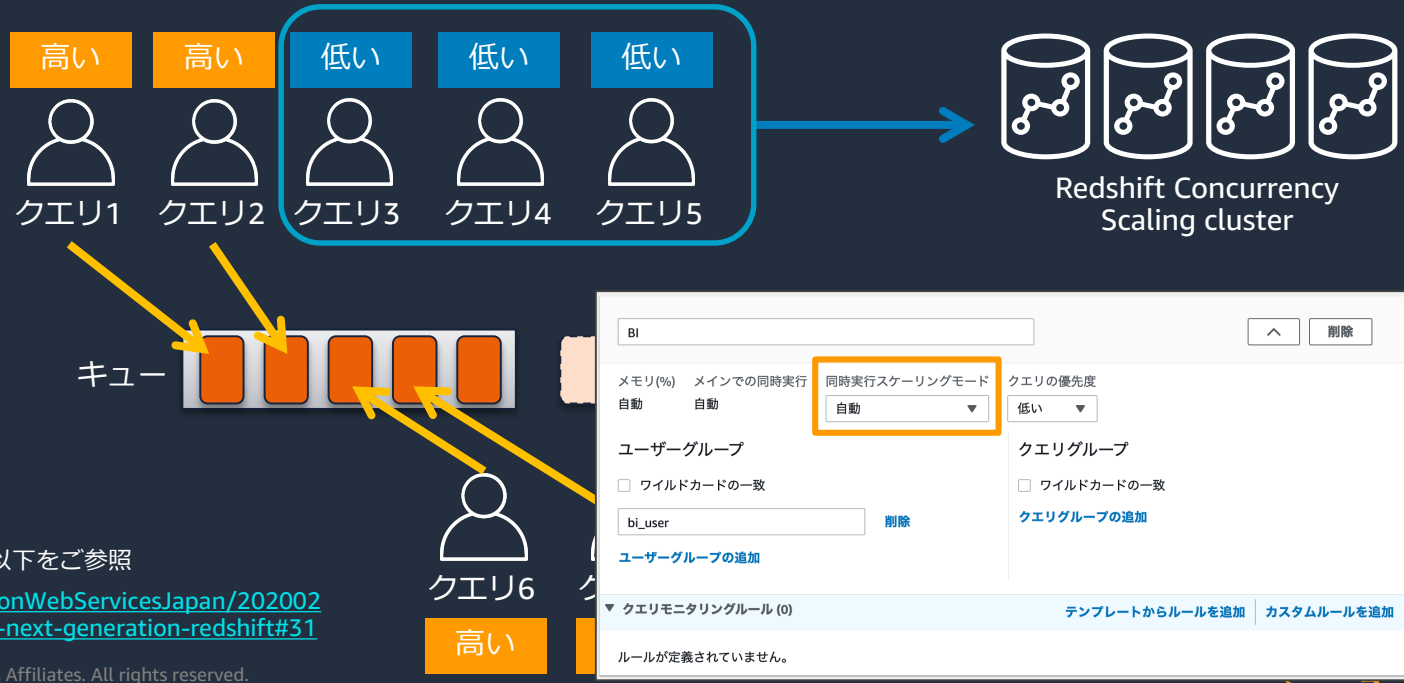
クエリ優先度の動き

優先度の低いクエリに、より速くスロットをアサインするためには？
=＞コマンドや QMR で優先度をより高いものに変更することで対応



クエリ優先度の動き

優先度の低いクエリに、より速くスロットをアサインするためには？
＝＞キューの「同時実行スケーリングモード」を自動に設定し、Concurrency Scaling を利用



* Concurrency Scaling の詳細は以下をご参照

https://www.slideshare.net/AmazonWebServicesJapan/20200218-aws-black-belt-online-seminar-next-generation-redshift#31

最近のアップデート

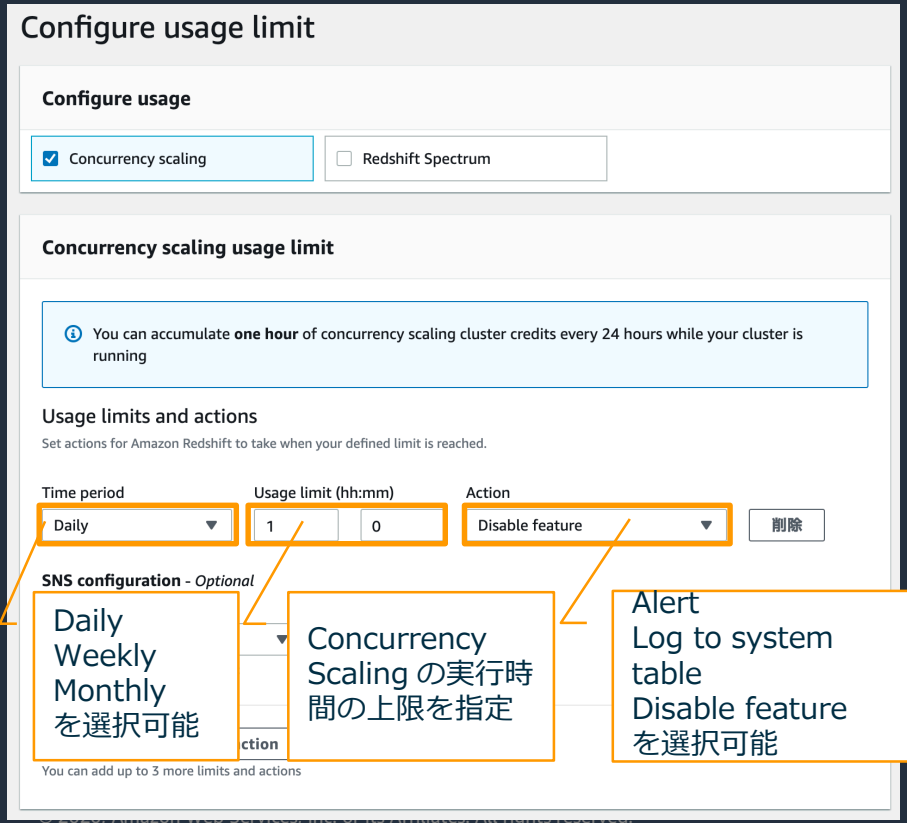
AWS 公式 Webinar
https://amzn.to/JPWebinar

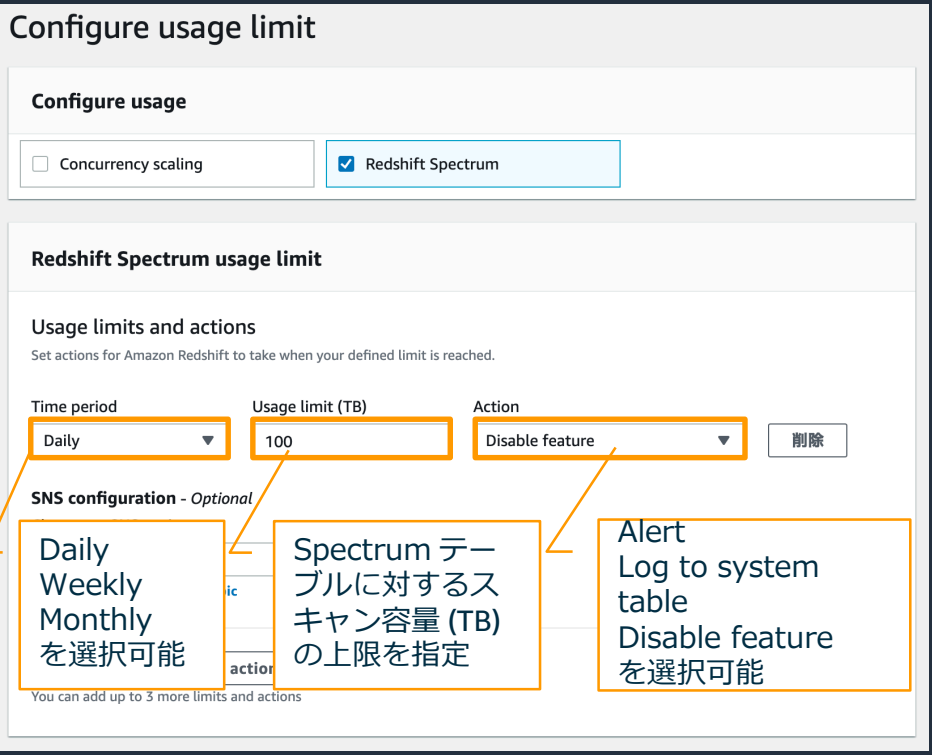
過去資料
https://amzn.to/JPArchive

Concurrency Scaling and Spectrum Usage Limit

New

日/週/月毎の単位で使用上限を設定し、コストをコントロールすることが可能に





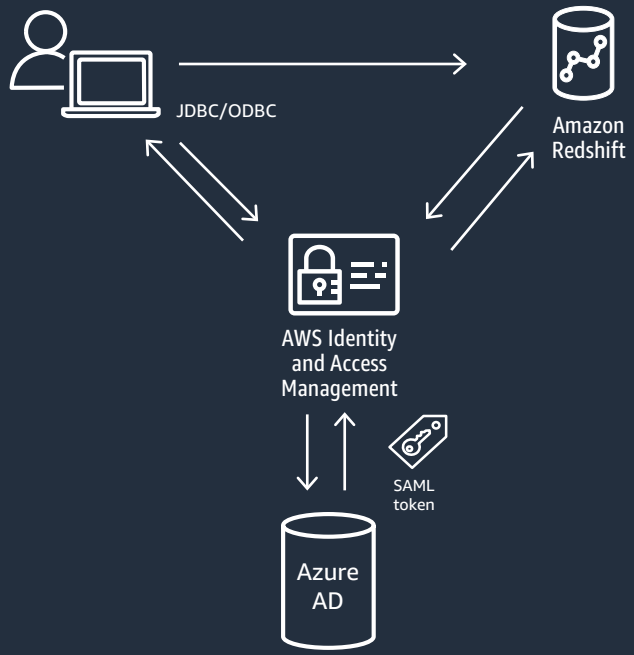
Amazon Redshift の料金
https://aws.amazon.com/jp/redshift/pricing/

Azure Active Directory SSO 連携と多要素認証 (MFA)

New

SAML 準拠のシングルサインオン(SSO)

- Redshift JDBC/ODBC ドライバー(*) は業界標準の SAML ワークフローとブラウザベースの多要素認証 (MFA) をサポート
- ADFS, Okta, PingFederate といったオンプレミスの SSO プロバイダに加えて Azure AD との連携も可能に



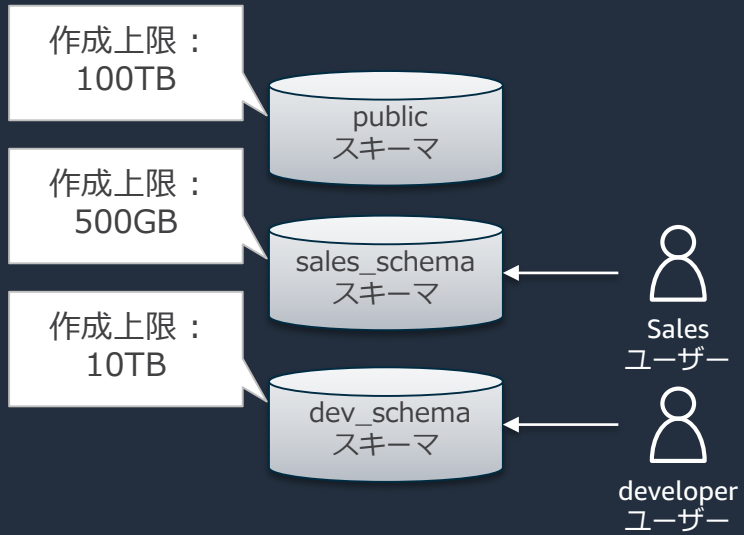
* JDBC ドライバーバージョン 1.2.37.1065 以降
ODBC ドライバーバージョン 1.4.11.1000 以降

スキーマのストレージ制御 (Schema Quota)

スキーマが使用するディスク容量を制限することが可能

New

```
ALTER SCHEMA public QUOTA 100 TB;
CREATE SCHEMA sales_schema AUTHORIZATION sales QUOTA 500 GB;
CREATE SCHEMA dev_schema AUTHORIZATION developer QUOTA 10 TB;
```



各スキーマの使用状況は以下のシステムテーブルより
モニタリング可能

- SVV_SCHEMA_QUOTA_STATE
- STL_SCHEMA_QUOTA_VIOLATIONS

schema	quota	disk_usage	disk_usage_pct
public	104857600	10022023	9.56
sales_schema	512000	0	0
dev_schema	10485760	0	0

まとめ

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

まとめ

- Amazon Redshift には多くの機能があり、それらをうまく活用することにより様々なユースケースに対応することが可能

- Amazon Redshift Spectrum はアップデートの多い機能のひとつ
積極的に活用することによって S3 データレイクとの連携を密にし、
更にコスト対効果が高くスケーラブルな分析環境を構築することが可能

- DWH に対するクエリには様々なものがあり、それぞれ処理される時間帯や特性、期待される性能 (SLA) は異なる
ワークロード管理 (WLM) の機能を活用しながらワークロード全体の最適化を図っていくことは重要

Q&A

お答えできなかったご質問については
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

参考資料

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

WLM クエリ優先度を設定してみよう

前提

- 2 種類の異なるワークロードがあると仮定し、
  各々の実行ユーザーが所属するユーザーグループ名は以下の通り
    - ETL (バッチ) : etl_user
    - BI 分析 : bi_user

- 上記のユーザーグループに所属したユーザーからクエリが投げられた際に
  利用するキューを、クエリ優先度付きで作成する

- BI 分析よりも、ETL (バッチ) の処理を優先させたい

WLM クエリ優先度を設定してみよう

Redshift コンソール
設定 > ワークロード管理 から
パラメータグループを新規作成
※デフォルトパラメータグループは編集できない

WLM クエリ優先度を設定してみよう

WLM モードが「自動 WLM」であることの確認 (違っていれば変更)

WLM クエリ優先度を設定してみよう

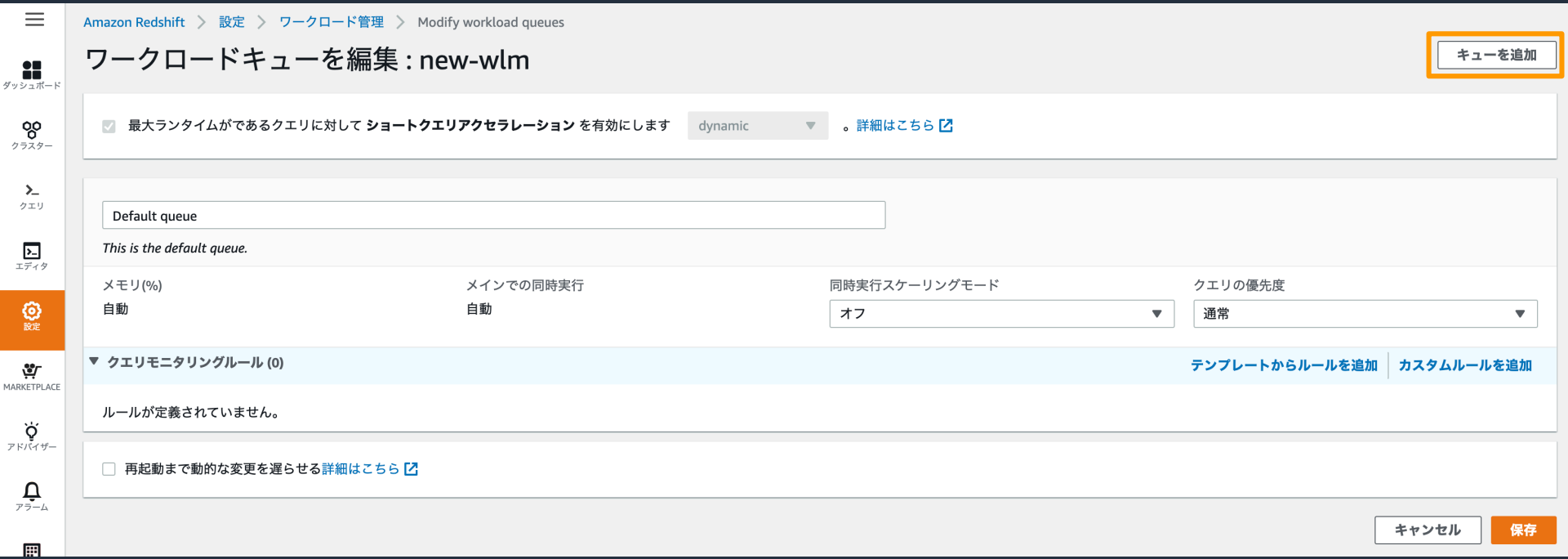
WLM モードが「自動 WLM」であることの確認 (違っていれば変更)

WLM クエリ優先度を設定してみよう

ワークロードキューを編集する

WLM クエリ優先度を設定してみよう

キューを 2 つ追加する (「キューを追加」を 2 回押す)

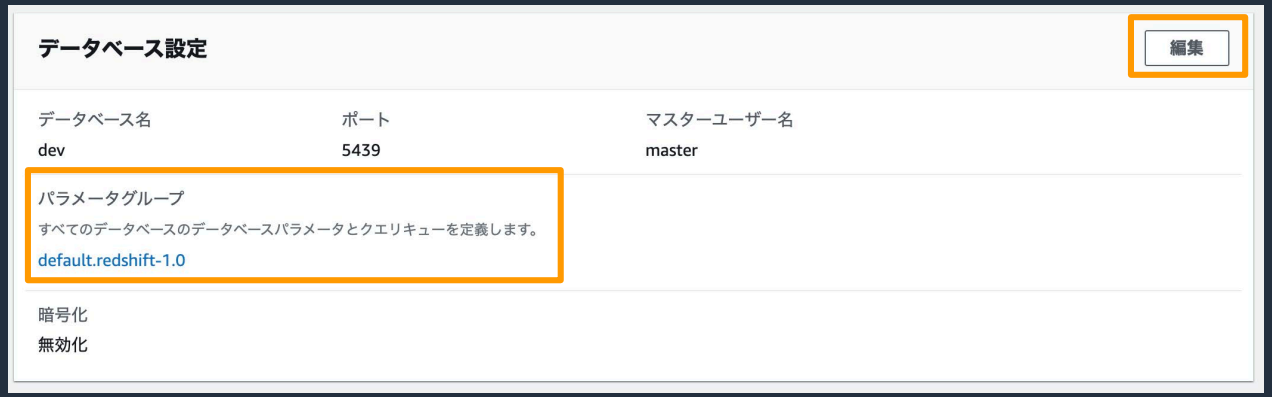


WLM クエリ優先度を設定してみよう

情報を入力して保存

WLM クエリ優先度を設定してみよう

クラスターメニューのプロパティタブよりデータベース設定を編集



WLM クエリ優先度を設定してみよう

作成したパラメータグループを選択してクラスターを変更

WLM クエリ優先度を設定してみよう

再起動を保留中の状態になったら、クラスターの再起動を実施

Amazon Redshift パラメータグループ
https://docs.aws.amazon.com/ja_jp/redshift/latest/mgmt/working-with-parameter-groups.html

クラスターに既に関連付けられているパラメータグループのパラメータ値を変更したり、別のパラメータグループをクラスターに関連付けたりすることもできます。このような場合、更新したパラメータ値を有効にするためにクラスターの再起動が必要になることがあります。