Alibaba Cloud
大数据#算服#

プロダクト紹介

Document Version20191021

目次

1 MaxCompute とは

ビッグデータコンピューティングサービスである **MaxCompute** (旧 **ODPS**) は、**GB**、**TB**、**PB** 級のデータに対応した高速なデータウェアハウスソリューションです。

MaxCompute は、従来の各種分散コンピューティングモデルに対応しており、ビジネスコストを削減し、データセキュリティを確保しながら、大量データの計算に関する課題を解決できます。

また、**MaxCompute** は **DataWorks** とシームレスに統合されるため、**MaxCompute** のデータ同期化、タスク開発、データ ワークフロー開発、データの操作、保守、管理をワンストップで実行できます。 詳細は、「*DataWorks*」をご参照ください。

MaxCompute は、主に構造化データの一括保存と一括計算に使用されます。 大容量のデータウェアハウスソリューションだけでなく、ビッグデータ分析機能とモデル化サービスも提供します。 データ収集技術の多様化と広域化が進むにつれ、産業界でのデータの大容量化が加速しています。 データ容量が、従来のソフトウェア業界では扱うことができなかった **100 GB**、**TB**、更には **PB** 級にまで拡大しています。

大容量データの場合、単一サーバーでは処理能力が限られるため、分散コンピューティングに移行するのが一般的です。 しかし、分散コンピューティングモデルの維持管理は容易でなく、高い専門性がデータアナリストに求められます。 分散モデルを使用する場合、データアナリストはビジネスニーズを理解するだけでなく、基盤のコンピューティングモデルにも精通する必要があります。 **MaxCompute** を使用することで、大量データの分析と処理を簡便化できます。また、大容量データを分析するのに、分散コンピューティングに関する詳細な知識は必要ありません。

> 注：
> **Alibaba** グループでは、大規模インターネット企業のデータウェアハウスと **BI** 分析、**E-commerce** サイトの **Web** ログ解析とトランザクション分析、およびユーザー特性とインタレストマイニングなどの分野で、**MaxCompute** を幅広く使用しています。

MaxCompute ラーニングパス

「*MaxCompute ラーニングパス*」を使用すると、**MaxCompute** に関する概念、基本操作、および高度な操作を素早く学習できます。

プロダクトの利点

- 大規模コンピューティングとストレージ

  MaxCompute は、大規模データ (最大で **PB** 級) の保存と処理に適しています。

- 複数の計算モデルに対応

  MaxCompute は、**SQL**、**MapReduce**、**Graph**、**MPI** 反復アルゴリズム、およびその他のプログラミング モデルを基盤としたデータ処理方法に対応しています。

- 堅牢なデータセキュリティ

  MaxCompute は、すべての **Alibaba Group** ビジネスのオフライン解析を **7** 年以上に渡って安定的に支え、マルチレイヤーサンドボックス保護およびモニタリング機能を備えています。

- 優れた費用対効果

  MaxCompute は、オンプレミスのプライベートクラウドモデルに比べ、調達費を **20%** から **30%** 削減できます。

機能

- データトンネル
  - 大規模な履歴データチャネルに対応しています。

    TUNNEL は、並列性の高いデータのアップロードとダウンロードサービスを提供します。 **TUNNEL** サービスは、**TB** 級や **PB** 級の日次データのインポートとエクスポートに対応しているため、全データや履歴データを一括インポートする際に特に便利です。 また、**TUNNEL** サービスは **Java** プログラミングインターフェイスを実装しており、**MaxCompute** クライアントツールには、ローカルファイルやサービスデータの交換に対応したコマンドが用意されています。
  - リアルタイムおよび増分データチャネル

    リアルタイムのデータアップロード向けには、レイテンシを低減して使いやすい **DataHub** サービスが **MaxCompute** に搭載されています。 **DataHub** サービスは、増分データのインポートに特に適しています。 **DataHub** は、**Logstash** 、**Flume**、 **Fluentd** 、**Sqoop** などの多様なデータ転送プラグインにも対応しています。また、ログ機能も備えています。 ログは **MaxCompute** に送られ、次に **DataWorks** でログ解析とマイニングが実行されます。
- コンピューティングおよび解析タスク

  MaxCompute は、複数のコンピューティングモデルに対応しています。

  - **SQL: MaxCompute** では、データはテーブル形式で保存されます。 **MaxCompute** では、外部インターフェイス用の **SQL** クエリ関数を使用できます。 **MaxCompute** は従来

のデータベースソフトウェアと同じように操作できるだけでなく、**PB** 級のデータを処理することもできます。

> 注：
>
> ■ **MaxCompute SQL** は、トランザクション、インデックス、更新と削除操作に対応していません。
>
> ■ **MaxCompute SQL** 構文は、**Oracle** や **MySQL** とは異なり、他のデータベースの **SQL** 文を **MaxCompute** にシームレスに移行できないので注意してください。
>
> ■ **MaxCompute SQL** では、クエリを秒からミリ秒単位で実行できますが、結果をミリ秒単位で返すことはできません。
>
> ■ **MaxCompute SQL** の利点は、ラーニングコストを低く抑えられることです。 複雑な分散コンピューティングに関する知識は必要ありません。 データベース操作の経験があれば、**MaxCompute SQL** を短期間で習得できます。

- *UDF*: ユーザー定義関数

  MaxCompute には、コンピューティングに関するユーザーニーズを満たす数多くの組み込み関数が実装されていますが、カスタム関数を作成することもできます。

- *MapReduce*: **MapReduce** は、**MaxCompute** に実装されている **Java MapReduce** プログラミングモデルです。 **Java** プログラミングインターフェイスが採用され、開発プロセスを簡素化する設定になっていますが、**MapReduce** を使用する前に、分散コンピューティングの基本概念を理解し、関連のあるプログラミング経験を積むことを推奨します。 **MaxCompute MapReduce** では、**Java** プログラミングインターフェイスを使用できます。

- *Graph*： **MaxCompute** の **Graph** 機能は、反復グラフ計算向けの処理フレームワークです。 グラフ計算ジョブでは、グラフを使ってモデルが構築されます。 グラフは頂点と辺から構成され、 頂点と辺には値が含まれます。 グラフの編集と展開が繰り返し実行された後、最終結果が得られます。 代表的なアプリケーションには、**PageRank**、**SSSP** アルゴリズム、**K** 平均法アルゴリズムなどがあります。グラフは編集と展開が繰り返し実行され、最終的な結果が表示されます。 代表的なアプリケーションには、*PageRank*、単一始点最短距離アルゴリズム、*K* 平均法クラスタリングアルゴリズムなどがあります。

・ **SDK**

開発者向けに便利なツールキットが用意されています。 詳細は、「*MaxCompute SDK*」をご参照ください。

- 安全性

  Maxcompute には、データを保護するための強力なセキュリティサービスが搭載されています。詳細は、「セキュリティガイド *(security guide)*」をご参照ください。

次のステップ

MaxCompute プロダクトの利点、機能の特徴、その他の関連事項について学習しました。次のチュートリアルに進むことができます。 次のチュートリアルでは、 **MaxCompute** の課金方法について説明します。 詳細は、「料金 *(Product Pricing)*」をご参照ください。

2 定義

2.1 Maxcompute 用語集

ここでは、**MaxCompute** の共通概念と用語について説明します。 詳細は、ドキュメント内のリンクをご参照ください。

A

- **AccessKey**

  Access Key (短縮形は **AK**。**Access Key Id** と **Access Key Secret** で構成されます) は、**Alibaba Cloud API** にアクセスするためのキーです。 **Alibaba Cloud** 公式サイトでクラウドアカウントを登録した後に、**AccessKey** 管理ページで作成できます。**MaxCompute** や他のクラウドプロダクトにアクセスする際、**Accesskey** によってユーザー識別と署名照合が行われます。 **AccessKey Secret** は機密情報として取り扱う必要があります。

- セキュリティ

  MaxCompute のマルチテナント型データセキュリティシステムの主な機能は、プロジェクトスペース内のユーザー認証、ユーザー管理、権限管理、ならびにプロジェクトスペース間のリソース共有、およびプロジェクトスペース内のデータ保護です。 **MaxCompute** のセキュリティ操作に関する詳細は、「セーフティガイド」をご参照ください。

C

- コンソール

  MaxCompute コンソールは、**Windows** および **Linux** 環境で稼働するクライアントツールです。 コンソールを介してプロジェクトを管理したり、**DDL** や **DML** などの操作の実行コマンドを送信できます。 ツールのインストールと共通パラメーターについては、「クライアント」をご参照ください。

D

- データ型

  MaxCompute テーブルのすべての列に対応したデータ型を指します。 サポート対象のデータ型については、「基本概念 > データ型」をご参照ください。

- **DDL**

  データ定義言語。テーブルの作成やビューの作成などを実行します。**MaxCompute** の **Div** 構文については、「ユーザーガイド > *DDL*」をご参照ください。

- **DML**

  データ操作言語。 データ挿入操作などを実行します。**MaxCompute DML** 構文については、「*Insert* 操作」をご参照ください。

F

- **Fuxi**

  Fuxi は、**Flying Platform** の中核でリソース管理とタスクのスケジューリングを実行するモジュールです。 また、アプリケーション開発向けの基本プログラミングフレームワークも提供します。 **MaxCompute** のボトムタスクのスケジューリングモジュールには、**Fuxi** のスケジューリングモジュールが採用されています。

I

- インスタンス

  ジョブの特定のインスタンスは、**Hadoop** ジョブの概念と同様に、実際に実行されるジョブを示します。 詳細は、「基本概念 > タスクインスタンス」をご参照ください。

M

- **MapReduce**

  MaxCompute は、データ処理用のプログラミングモデルで、通常は大規模データセットの並列操作に使用されます。 **MapReduce** に実装されているインターフェイス **(Java API)** を使用して、**MaxCompute** でデータを処理するための **MapReduce** プログラムを作成できます。 **MapReduce** プログラミングの構想は、データ処理の方法を **Map** (マッピング) と **Reduce** (プロトコル) に分けることです。

  正式に **Map** を実行する前に、分割が必要です。 スライスは入力データを同じサイズのブロックに切り分けたもので、各スライスが **1** つのマップとして機能します。 複数の **Map Worker** が一緒に動作できるように、**Worker** の入力が処理されます。各 **Map Worker** ごとにデータの読み取り、計算、処理が実行され、**Reduce** 機能によって中間結果が統合された後に、最終結果が出力されます。 詳細は、「ユーザーガイド > *MapReduce*」をご参照ください。

O

- **ODPS**

  ODPS は、**MaxCompute** の旧名です。

P

- パーティション

  パーティションは、テーブルのパーティションフィールド **(1** つまたは複数の組み合わせ**)** をベースとして、データストアを分割します。 つまり、パーティションのないテーブルの場合、データはテーブルのあるディレクトリの直下に格納されます。 パーティションのあるテーブルの場合、各パーティションがテーブル内の **1** つのディレクトリと対応し、データはパーティションのディレクトリごとに分かれて格納されます。 パーティションの詳細は、「基本概念 > パーティション」をご参照ください。
- プロジェクト

  プロジェクトは、**MaxCompute** の基本の組織単位です。 従来のデータベースやスキーマの概念と同様に、マルチユーザーの分離とアクセス制御の主要な境界です。 詳細は、「基本概念 > プロジェクト」をご参照ください。

R

- ロール

  ロールは、**MaxCompute** のセキュリティ機能で使用される概念であり、同じ権限を持つユーザーの集合体をいいます。 同時に **1** つのロールに複数のユーザーを割り当てることができます。また、**1** 人のユーザーが複数のロールに属することもできます。 すべてのロールの権限付与が完了すると、同じロールが割り当てられたすべてのユーザーは、同じ権限を持ちます。ロール管理の詳細は、「ユーザーガイド > ロール管理」をご参照ください。
- リソース

  リソースは、**MaxCompute** 特有の概念です。 **MaxCompute** のカスタム関数 **(UDF)** や **MapReduce** 関数を使用する場合は、関数を完成させるためのリソースが必要です。 詳細は、「基本概念 > リソース」をご参照ください。

S

- **SDK**

  ソフトウェア開発キット。 一般的に、ソフトウェアエンジニアが特定のソフトウェアパッケージ、ソフトウェアインスタンス、ソフトウェアフレームワーク、ハードウェアプラットフォーム、オペレーティングシステム、ドキュメントパッケージなどに対応したアプリケーションソ

フトウェアを構築するために使用する一連の開発ツールを指します。 現在、**MaxCompute** では、*#unique_25* と **Python SDK** を使用できます。

- 権限付与

  プロジェクトスペースの管理者またはプロジェクトオーナーは、**MaxCompute** のオブジェクト (タスクやリソースなど) で特定の操作 (読み取り、書き込み、表示など) を実行する権限をユーザーに付与します。 権限付与の操作については、「ユーザー管理」をご参照ください。

- サンドボックス

  セキュリティ機能の制限事項: **MaxCompute MapReduce** および **UDF** プログラムを分散環境で実行する場合、**Java** サンドボックスにより制限されます。

T

- テーブル

  テーブルは、**MaxCompute** のデータストレージ単位です。 詳細は、「基本概念 > テーブル」をご参照ください。

- **Tunnel**

  MaxCompute のデータチャネルは、並列性の高いオフラインのデータアップロードとダウンロードサービスを提供します。 **Tunnel** サービスを使用すると、**MaxCompute** への一括アップロードやダウンロードを実行できます。 関連コマンドについては、「**Tunnel** コマンド操作」または「一括データチャネル **SDK**」をご参照ください。

U

- **UDF**

  汎用 **UDF** (ユーザー定義関数) です。**MaxCompute** に実装されている **Java** プログラミングインターフェイスを使用して、カスタム関数を開発できます。詳細は、「ユーザーガイド > *UDF*」をご参照ください。

  狭い意味で、**UDF** は入力と出力が **1** 対 **1** の関係にあるユーザー定義スカラー関数のことを指します。つまり、データが **1** 行読み取られると、出力値が **1** つ書き出されます。

- **UDAF**

  ユーザー定義集計関数。入力と出力が多対 **1** の関係にあるカスタム集計関数です。複数の入力レコードを集計して、**1** つの値が出力されます。 **SQL** では **Group By** 文と組み合わせて使用することもできます。 詳細は、「*Java UDF > UDAF*」をご参照ください。

- **UDTF**

  User Defined Table Valued Function。詳細は「*Java UDF>UDAF*」をご参照ください。

2.2 プロジェクト

プロジェクトは、**MaxCompute** の基本操作単位です。 従来のデータベースやスキーマの概念と同様に、プロジェクトによって **MaxCompute** のマルチユーザーの分離やアクセス制御の境界が設定されます。

ユーザーは同時に複数のプロジェクトの権限を持つことができます。また、関連する権限が付与されると、ユーザー自身のプロジェクトから別のプロジェクトのテーブル、リソース、関数、インスタンスといったオブジェクトにアクセスできます。

プロジェクト (下記の例では「**my project**」) に移動するには、次のように **Use Project** コマンドを使用します。

```
use my_project -- Use this command to enter the project space named
my_project.
```

上記コマンドを実行すると、「**my_project**」という名前のプロジェクトに移動して、このプロジェクト内のすべてのオブジェクトを操作できます。 **Use Project** コマンドは、**MaxCompute** クライアントに実装されています。この部分について詳しく説明する前に、本ドキュメントでは、一般的に使用されるコマンドを簡単に紹介します。詳細は、「*共通コマンド (Common Commands)* 」をご参照ください。

 注:

MaxCompute のプロジェクトは、**DataWorks** のワークスペースです。

2.3 テーブル

テーブルは、**MaxCompute** のデータストレージの単位であり、 行と列で構成される **2** 次元データ構造です。 各行はレコードを表し、各列は同じデータ型のフィールドを表します。 **1** つのレコードには、**1** つ以上の列を含めることができます。 列名とデータ型は、テーブルのスキーマで構成されます。

MaxCompute の各種コンピューティングタスクの操作オブジェクト (入力、出力) はテーブルです。 テーブルの作成、テーブルの削除、テーブルへのデータインポートを実行できます。

 注:

DataWorks のデータ管理モジュールを使用すると、**MaxCompute** テーブルに格納されるデータのライフサイクルを登録、編成、変更できます。また、ライフサイクルの管理権限を付与することもできます。 詳細は、「*データ管理の概要 (data management overview)*」をご参照ください。

MaxCompute v2.0 は、内部テーブルと外部テーブルの **2** 種類のテーブルに対応しています。**MaxCompute2.0** バージョンでは、新たに外部テーブルが使用できるようになりました。

- 内部テーブルの場合、全データが **MaxCompute** テーブルに保存されます。テーブル内の列は、**MaxCompute** で対応している*#unique_16*であれば、どれでも構いません。
- 外部テーブルの場合、データは **MaxCompute** に保存されずに、 *OSS* や *OTS* に保存されます。 **MaxCompute** には、テーブルのメタ情報のみが記録されます。 **MaxCompute** の外部テーブルを使用すると、ビデオ、オーディオ、遺伝子情報、気象情報、地理情報などの非構造化データを **OSS** や **Table Store** で処理できます。

DUAL 表の使用：

- **Oracle** などのデータベースとは異なり、**MaxCompute** では **DUAL** 表が自動的に作成されません。
- テストテーブルとして **DUAL** 表を使い慣れている場合は、`CREATE TABLE IF NOT EXISTS DUAL (DUMMY VARCHAR(1));` コマンドを手動で実行して、テスト用のフィールドが **1** つだけある **DUAL** という名前の空のテーブルを作成できます。

  注：
  現在、**MaxCompute SQL** と新バージョンの**Mapreduce** で使用できる **Set** コマンドは、次の **2** 通りに分けられます。

  - セッションレベル: 新しいデータ型 (**Tinyint**、**Smallint**、**Int**、**Float**、**Varcha**、**TIMESTAMP BINARY**) を使用するには、テーブル文の前に **set** 文を追加する必要があります。

    ```
    set odps.sql.type.system.odps2=true;
    ```

    テーブル文と一緒に実行コマンドを送信します。
  - プロジェクトレベル: 新たにプロジェクトレベルの **Set** コマンドが使用できるようになりました。 プロジェクトオーナーは、必要に応じてプロジェクトを設定できます。

    ```
    setproject odps.sql.type.system.odps2=true;
    ```

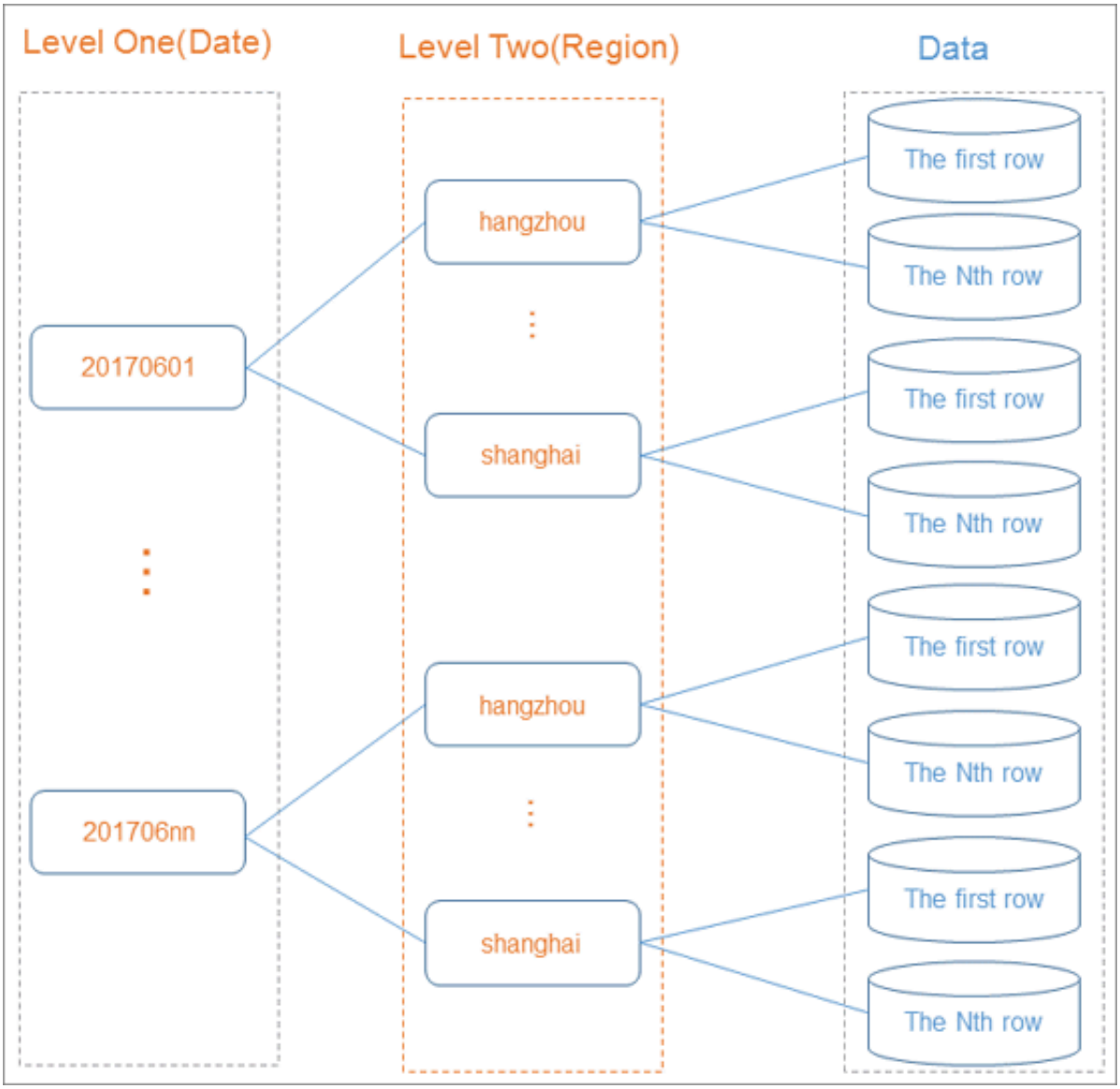
- **DUAL** 表の使い方は、**Oracle** の場合と同じです。`select getdate() from dual;` のように使用します。

2.4 パーティション

MaxCompute の処理効率を向上させるために、テーブルの作成時にパーティションを定義できます。 具体的には、テーブル内の複数フィールドをパーティションとして指定できます。 機能的観点からみると、パーティションは、ファイルシステム下のディレクトリに相当します。

MaxCompute では、各フィールドを個別のパーティションとして指定できます。 あるいは、複数のフィールドを **1** つのパーティションとして指定することもできます。これにより、複数階層のディレクトリと同様の機能を備えます。

データを使用するときにアクセスするパーティションを指定すると、対応するパーティションのみが読み取られるため、テーブル全体のスキャンが回避され、処理効率が向上し、コストの削減が実現します。



パーティションタイプ

MaxCompute 2.0 では、サポート対象のパーティションタイプが増え、現在 tinyint 型、smallint 型、Int型、bigint 型、varchar 型、および string 型のパーティションに対応しています。

 注：

MaxCompute 2.0 より前のバージョンは、STRING 型のパーティションにのみ対応しています。 bigint 型のパーティションを指定できますが、これまでの背景から、テーブルのスキーマ表現をbigint 型として指定する場合を除き、その他のケースでは、実際には string 型として扱われます。 例：

```
create table parttest (a bigint) partitioned by (pt bigint);
insert into parttest partition(pt) select 1, 2 from dual;
insert into parttest partition(pt) select 1, 10 from dual;
select * from parttest where pt >= '2';
```

上記 SQL 文を実行すると、戻ってくる結果は、1 行のみです。10 は STRING 型として扱われ、2 と比較されたので、結果は返されません。

制限事項

パーティションを使用する際、次の制限事項が適用されます。

- 1 つのテーブルの最大パーティション階層数は 6 です。
- 1 つのテーブルの最大パーティション数は 60,000 です。
- 1 つのクエリの最大クエリパーティション数は 10,000 です。

例：

```
-- create a two-level partition table with the date as the level one
partition and the region as the level two partition
create table src (key string, value bigint) partitioned by (pt string,
region string);
```

クエリの実行時、WHERE 条件フィルタで、パーティション列をフィルタ条件として使用できます。

```
select * from src where pt='20170601' and region='hangzhou';  -- This
example is the correct method of using WHERE conditional filter. When
MaxCompute generates a query plan, only data of the region 'hangzhou'
under the '20170601' partition is accessed.
select * from src where pt = 20170601;   -- This example is an
incorrect method of using the WHERE conditional filter. In this
 example, the effectiveness of the partition filter cannot be
guaranteed. pt is a STRING type. When the STRING type is compared with
```

```
 BIGINT type (20170601), MaxCompute converts both to DOUBLE type, and
loss of precision occurs.
```

パーティションに対する一部の **SQL** 操作は効率的ではなく、課金料金が高くなることがあります。「動的パーティションの使用 *(using dynamic partition)*」が例として挙げられます。

MaxCompute の一部の操作コマンドは、パーティションテーブルを処理する場合と非パーティションテーブルを処理する場合とで構文が異なります。詳細は、「*DDL 文 (DDL statement)* 」および「動的パーティションの使用 **(using dynamic partitioning)** 」をご参照ください。

2.5 ライフサイクル

ここでは、 **MaxCompute** テーブルのライフサイクルの概念を詳しく紹介します。

MaxCompute テーブルやパーティションのライフサイクルは、最後の更新時間から計測されます。 指定した時間が経過しても変更されないテーブルやパーティションは、 **MaxCompute** により自動的にリサイクルされます。 「指定した時間」によりライフサイクルが設定されます。

- ライフサイクル単位**:** 日数で、正の整数のみ指定できます。
- 非パーティションテーブルにライフサイクルを設定すると、ライフサイクルは、テーブルデータが最後に変更された時刻 **(LastDataModifiedTime)** からカウントされます。 指定した時間が過ぎてもテーブルデータが変更されない場合、手動による操作がなくてもテーブルは自動的にリサイクルされます (テーブルの削除操作と同様)。
- パーティションテーブルにライフサイクルを設定する場合は、ユーザーが各パーティションの **LastDataModifiedTime** に基づき、パーティションテーブルをリサイクルするかどうかを決定します。 非パーティションテーブルとは異なり、すべてのパーティションがリサイクルされても、パーティション テーブルは削除されません。

   注**:**

  ライフサイクルスキャンは、スケジュールされた時間に毎日開始され、パーティション全体がスキャンされます。ライフサイクルが過ぎても変更されないパーティションは、自動的にリサイクルされます。

  パーティションテーブルのライフサイクルを設定すると、**LastDataModifiedTime** に基づいて、パーティションをリサイクルするかが自動的に判断されます。 非パーティションテーブルとは異なり、すべてのパーティションがリサイクルされても、パーティションテーブルは削除されません。
- テーブルのライフサイクルは設定できますが、パーティションのライフサイクルは設定できません。 テーブルのライフサイクルは、テーブルの作成時に指定できます。

- ライフサイクルを指定していない場合、パーティションやテーブルが自動的にリサイクルされることはありません。

テーブルの作成時にライフサイクルを指定または変更する方法、およびテーブルの **LastDataModifiedTime** を変更する方法については、「*#unique_17*」をご参照ください。

2.6 リソース

ここでは、**MaxCompute** リソースの概念を紹介します。**MaxCompute** の操作には、リソースの依存関係を設定できます。

リソースの概念

リソースは、**MaxCompute** 特有の概念です。 ユーザー定義関数 (詳細は、「*UDF*」をご参照ください) または *MapReduce* を使用してタスクを実行するには、リソースを使用する必要があります。

- **SQL UDF: UDF** を記述後、**Jar** パッケージとしてコンパイルし、コンパイルしたパッケージをリソースとして **MaxCompute** にアップロードする必要があります。 これにより、**UDF** の実行時に 、対応する **JAR** パッケージが自動的にダウンロードされ、記述したコードが取得されます。 **JAR** パッケージは、**MaxCompute** リソースの一種です。
- **MapReduce: MapReduce** プログラムを記述後、**Jar** パッケージとしてコンパイルし、コンパイルしたパッケージをリソースとして **MaxCompute** にアップロードする必要があります。 これにより、**MapReduce** ジョブの実行時に、対応する **JAR** パッケージが自動的にダウンロードされ、記述したコードが取得されます。 テキストファイルと **MaxCompute** テーブルは、異なるタイプのリソースとして **MaxCompute** にアップロードできます。 これで、**UDF** や **MapReduce** の実行時に、アップロードしたリソースを読み込んだり、使用したりできるようになります。

リソースタイプ

MaxCompute は、リソースを読み込んだり使用したりするためのインターフェイスを備えています。 詳細は、「*リソースの使用例 (Use Resourse Example)* 」および「*UDTF の使用方法 (UDTF Usage)* 」をご参照ください。

 注 :

ユーザー定義関数 (*UDF*) および *MapReduce* のリソース読み込み機能に関する制限についての詳細は、 「*アプリケーションの制限事項 (Application Restriction)*」をご参照ください。

MaxCompute が対応可能な 単一リソースの最大容量は、**500 MB** です。 **MaxCompute** のリソースタイプは、次の通りです。

- ファイル型
- テーブル型**: MaxCompute** 内のテーブル

> 注 **:**
> 現在、**MapReduce** で参照されるテーブル内では、**BIGINT** 型、**DOUBLE** 型、**STRING** 型、**DATETIME** 型、**BOOLEAN** 型のフィールドのみ使用できます。

- **Java JAR** パッケージにコンパイルされた **Jar** 型
- アーカイブ型。圧縮型で、リソース名の接尾辞によって決まります。 サポート対象の圧縮形式は、**.zip**、**tgz**、**tar.gz**、**tar**、**jar** です。

リソースに関する詳細は、「リソースの追加 *(Add Resource)* 」、「リソースの削除 *(Drop Resource)*」、「リソース一覧の表示 *(List Resources)*」、および「リソース情報の表示 *(Describe Resource)*」をご参照ください。

2.7 関数

ここでは、**MaxCompute** で計算機能を実行するために使用する関数の概要を紹介します。

MaxCompute には、**SQL** コンピューティング機能が搭載されています。 **MaxCompute SQL** では、組み込み関数 *(system's built-in functions)* を使用して、一般的なコンピューティングタスクと集計タスクを実行できます。 組み込み関数が要件を満たさない場合は、**MaxCompute** に実装されている **Java** プログラミングインターフェイスを使用して、 ユーザー定義関数 **(UDF)** を開発できます。

UDF は、スカラー値関数、ユーザー定義集計関数 **(UDAF)**、およびユーザー定義テーブル関数 **(UDTF)** に分類できます。

UDF コードを記述後に **JAR** パッケージにコンパイルし、コンパイルした **JAR** パッケージを、**MaxCompute** にアップロードする必要があります。 これで、**MaxCompute** に **UDF** を登録できます。

> 注 **:**
> **UDF** の使用方法は、 組み込み関数の場合と同じです。**SQL** で **UDF** 名と入力関連パラメーターを指定します。

詳細は、 「関数の紹介 *(Function introduction)*」をご参照ください。

2.8 タスク

タスクは、**MaxCompute** の基本的なコンピューティング単位です。
SQL、**DML**、**MapReduce** 関数などのコンピューティングタスクは、タスクごとに実行されます。

SQL DML 文、*MapReduce* などのように、ユーザーが送信するタイプのタスクのほとんどでは、最初に **MaxCompute** によってタスクが分析され、タスク実行計画が生成されます。 タスク実行計画は、相互依存関係のある複数の実行ステージで構成されます。 実行計画は、依存関係のある複数のステージで構成されます。

現在、実行計画は有向非循環グラフとしてローカルで表示できます。グラフの **Vertix** はステージを示し、**Edge** はステージの依存関係を示します。 **MaxCompute** では、グラフ (実行計画) の依存関係に基づいて、各ステージが実行されます。 **1** つのステージには、ワーカーと呼ばれる複数のスレッドが含まれ、 ワーカーによってステージ内のコンピューティングが実行されます。同一ステージ内にある複数のワーカーのロジックは全く同じですが、処理するデータが異なります。 計算タスクは、**MaxCompute** インスタンスで直接実行されます。たとえば、 インスタンスのステータス表示やインスタンスの停止などがあります。

SQL の **DDL** 文などの 計算タスク以外の **MaxCompute** タスクの場合は、**MaxCompute** のメタ情報の読み取りと変更のみを実行できます。 つまり、タスクから実行計画の分析と生成は実行できません。

> 注:
> すべてのリクエストが、**MaxCompute** タスクに変換されるわけではありません。たとえば、プロジェクト、リソース、*UDF*、インスタンスの操作は、**MaxCompute** タスクなしで完了できます。

2.9 インスタンス

本章では、**MaxCompute** のタスク インスタンスとステータスを紹介します。

MaxCompute では、ほとんどのタスクは、**MaxCompute** インスタンスで開始されます。
MaxCompute のインスタンスは、実行中と終了の **2** つのフェーズのいずれかになります。

実行中フェーズのステータスは **"Running"** です。終了フェーズのステータスは、**"Success"** (成功)、**"Failed"** (失敗)、**"Canceld"** (キャンセル済み) のいずれかです。 **MaxCompute** によっ

て割り当てられたインスタンス ID を使用して、ステータスを照会または変更することができます。 例は、以下のとおりです。

```
status <instance_id>; --View the status of a certain instance.
            kill <instance_id>; --Stop an instance and set its status
as ‘Canceled’.
            wait <instance_id>; --View the running logs of a certain
instance.
```