IT資格道場

DP-750 学習用テキスト

予備知識ゼロから合格ラインへ ―― 読んで理解する1冊目

無料サンプル(第1章まで)

DP-750 出題範囲(Skills measured)2026-03-11版 準拠
2026年11月版

SAMPLE

© IT資格道場｜無料サンプル｜転載・再配布・AI学習利用禁止

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

SAMPLE

目次

この教材の使い方

試験の基本情報

試験の設計図：4 分野と配点

構成

4 ステップ学習法

第1章 Set up and configure an Azure Databricks environment [1.1.1/1.1.2/1.1.3/1.1.4/1.1.5/1.2.1/1.2.2/1.2.3/1.2.4/1.2.5/1.2.6/1.2.7/1.2.8]

コンピューティングの種類の選択（Choose an appropriate compute type）[1.1.1]

コンピューティングのパフォーマンス設定（Configure compute performance settings）[1.1.2]

コンピューティングの機能設定（Configure compute feature settings）[1.1.3]

コンピューティングのライブラリのインストール（Install libraries for a compute resource）[1.1.4]

コンピューティングへのアクセス許可（Configure access permissions to a compute resource）[1.1.5]

Unity Catalog の名前付け規則と階層（Apply naming conventions）[1.2.1]

カタログの作成（Create a catalog）[1.2.2]

スキーマの作成（Create a schema）[1.2.3]

ボリュームの作成（Create volumes）[1.2.4]

テーブル・ビュー・マテリアライズド ビューの作成（Create tables, views, and materialized views）[1.2.5]

接続による外部カタログの実装（Implement a foreign catalog by configuring connections）[1.2.6]

マネージド テーブルと外部テーブルの DDL（Implement DDL operations on managed and external tables）[1.2.7]

SAMPLE

サンプルはここまでです

SAMPLE

3 冊の使い方は、ダウンロード欄の案内を参照してください。

SAMPLE

この教材の使い方

このテキストは、Microsoft の Associate 資格 **Microsoft Certified: Azure Databricks Data Engineer Associate(試験コード DP-750、試験名 Implementing Data Engineering Solutions Using Azure Databricks)** に合格することを目的に作られました。DP-750 は、Azure Databricks 上でデータの取り込み・変換・配備・運用を**データ エンジニアの視点で作って回す**資格で、製品の名前を知っているかではなく、「この要件なら、どのコンピューティング・どの設定・どの操作を選ぶか」を問われます。

Azure Databricks には、サーバーレスとクラシックのコンピューティング、Unity Catalog の各オブジェクト、ノートブック、Lakeflow Jobs、Lakeflow Spark Declarative Pipelines など、似た役割の機能がいくつもあり、同じ「データを入れる」「権限を絞る」「遅さを直す」でも、機能によって使える設定と手順が違います。このテキストは、その違いを軸に、公式の出題範囲(4 分野)に沿って説明していきます。

> 公式の想定読者は、SQL と Python でデータを取り込んで変換し、Azure Databricks で処理の基盤を実装して運用する人。

深い開発経験は不要で、**各機能の「何をする道具か」と「選ぶ条件・設定の勘どころ」を押さえれば合格ラインに届きます**。SQL・Python・CLI のコード例は出てきますが、**丸暗記は不要**で、「どのコードが何をするか」の対応が読めれば十分でしょう。

SAMPLE

試験の基本情報

項目	内容
試験名	Microsoft Certified: Azure Databricks Data Engineer Associate(DP-750)
出題数	50 問
試験時間	120 分
合格ライン	1,000 点満点中 **700 点**(スケールスコア)
受験料	¥20,300
前提資格	なし
受験方法	ピアソン VUE。テストセンターまたはオンライン監督試験
言語	日本語で受験可(英語ほか多言語提供)

出題形式は事前に公表されません。択一・複数選択のほか、ドラッグ アンド ドロップ、ホットエリア、ケーススタディ、「問題と解決策」が出ることがあります。受験料や実施形式は、いつまでも同じとはかぎりません。申し込み時に、公式ページの最新情報を確認してください。

試験の設計図：4 分野と配点

このテキストが準拠する Skills measured は **2026 年 3 月 11 日版**。

分野	配点	一言でいうと
① Azure Databricks 環境の設定と構成	**15〜20%**	選ぶ・作る：コンピューティングと Unity Catalog のオブジェクト

SAMPLE

分野	配点	一言でいうと
② Unity Catalog オブジェクトのセキュリティとガバナンス	**15〜20%**	守る・管理する：権限・行と列の制限・シークレット・系列・共有
③ データの準備と処理	**30〜35%**	入れる・変える：取り込み・変換・品質の検査
④ データ パイプラインとワークロードの配備と保守	**30〜35%**	回す・直す：ジョブ・配備・監視・最適化

分野③と④で、配点の 6 割以上を占めます。 分野①と②は配点が小さい分、1 問ごとの選択肢の差が細かく、設定名と権限名の正確さが点を分けます。

構成

このテキストは 4 分野を 4 つの章に分ける構成。並べ方は**公式の分野順そのまま**です。

- 第 1 章 … 分野①(環境の設定と構成)
- 第 2 章 … 分野②(Unity Catalog のセキュリティとガバナンス)
- 第 3 章 … 分野③(データの準備と処理)
- 第 4 章 … 分野④(パイプラインとワークロードの配備と保守)

見出しの末尾の [1.1.1] などの数字は、公式の出題範囲の番号です。

4 ステップ学習法

購入者限定テキストは 3 冊で 1 セット。次の順番で使うと最短です。

- **STEP 1(理解する)**: 「学習用テキスト」を通読し、Azure Databricks の地図と「選ぶ条件・設定の勘どころ」を頭に入れる

SAMPLE

- **STEP 2(解き方を掴む)**:「頻出度別ノート」で頻出度 A の論点を解いて、解き方を掴む
- **STEP 3(腕試し)**: 本サイトの問題演習で出題形式に慣れる
- **STEP 4(再確認)**:「頻出度別ノート」に戻って頻出度 A の論点を再確認する(試験直前もここ)

「暗記用ノート」は STEP には数えず、空き時間や用語の再確認に使ってください。

DP-750 は操作を問う試験なので、可能なら Azure Databricks のワークスペースでクラスターやジョブを作って、実際に触ると定着が段違いです。時間がなければ、先に「頻出度別ノート」で頻出度 A の論点を確認し、それを含む章から読み始めても構いません。

SAMPLE

第1章 Set up and configure an Azure Databricks environment [1.1.1/1.1.2/1.1.3/1.1.4/1.1.5/1.2.1/1.2.2/1.2.3/1.2.4/1.2.5/1.2.6/1.2.7/1.2.8]

この章は、ワークスペースの中に何を置き、どう設定するかを扱います。前半がコンピューティングの選び方と設定、後半が Unity Catalog のオブジェクトの作り方で、どちらも「この要件ならどの種類・どの設定・どの権限か」を組にして覚えます。

コンピューティングの種類の選択（Choose an appropriate compute type） [1.1.1]

Azure Databricks のコンピューティングは、サーバーレス コンピューティング、クラシック コンピューティング、SQL ウェアハウスの 3 系統です。サーバーレスは Databricks が管理する資源にすぐ接続する方式、クラシックは自分のサブスクリプションにクラスターを立てて設定を細かく変える方式、SQL ウェアハウスは SQL による分析と BI に特化した資源です。[1]

サーバーレスの利用条件は、ワークスペースで Unity Catalog が有効なことと、対応リージョンにあることの 2 つで、有効化の操作は要りません。条件を満たすワークスペースでは、他の資源を選ばない限り、新しいノートブックでコードを実行した時点でサーバーレスが既定で接続されます。既定の実行タイムアウトは 2.5 時間（9,000 秒）で、ノートブックごとに `spark.databricks.execution.timeout` で上書きでき、ワークスペース管理者は全ノートブックの既定も変えられます。サーバーレスはバージョンを持たない製品で、Databricks が常に最新へ自動で更新します。[2]

SAMPLE

人が対話で使う資源は汎用（all-purpose）コンピューティング、Lakeflow Jobs のタスクが使う資源はジョブ コンピューティングです。クラシックのジョブ コンピューティングは、ジョブの定義の中で設定します。サーバーレスを使えるタスクなら、サーバーレス ジョブが推奨です。[3]

タスクの種類	推奨	使えるもの
ノートブック・Python スクリプト・Python wheel	サーバーレス ジョブ	サーバーレス ジョブ・クラシック ジョブ・クラシックの汎用
JAR	サーバーレス ジョブ	サーバーレス ジョブ・クラシック ジョブ・クラシックの汎用
SQL・dbt	サーバーレス SQL ウェアハウス	サーバーレス・Pro の SQL ウェアハウス
Lakeflow パイプライン	サーバーレス パイプライン	サーバーレス・クラシックのパイプライン
Spark Submit	クラシック ジョブ	クラシック ジョブだけ

サーバーレス ジョブには制限もあります。連続実行できるのは、`Trigger.AvailableNow` のような有限のトリガーを使う Structured Streaming だけで、既定のトリガーや時間間隔のトリガーは使えません。サーバーレスの費用は、ジョブとパイプラインの性能モードでも変わります。標準モードはパフォーマンス最適化モードより安く、数分の起動の遅れを許せる処理に向きます。

SQL ウェアハウスの主な種類は、サーバーレス、Pro、クラシックの 3 つです（ほかにベータの Lakehouse Real-Time があります）。[4] 違いは性能機能と起動時間です。[5]

SAMPLE

項目	サーバーレス	Pro	クラシック
Photon	対応	対応	対応
Predictive IO	対応	対応	非対応
Intelligent Workload Management	対応	非対応	非対応
起動時間	通常 2〜6 秒	約 4 分	約 4 分
計算層の置き場所	Databricks 側	自分の Azure サブスクリプション	自分の Azure サブスクリプション

サーバーレスに対応したリージョンで要件を満たすワークスペースなら、画面で作ると既定の種類はサーバーレスですが、SQL ウェアハウス API を既定のパラメーターで使うと、既定はクラシックになります。Pro を選ぶのは、サーバーレスが使えないリージョンにいる場合と、独自のネットワークでフェデレーションやオンプレミスとの接続が要る場合です。クラシックは、入門的な性能で足りる対話的な探索に使います。

たとえば、毎晩動くノートブック タスクの定時ジョブで、クラスターの管理を減らしたいなら、サーバーレス ジョブです。同じジョブに Spark Submit タスクが入っているなら、そのタスクだけはクラシック ジョブでなければ動きません。多数の利用者が同時に開く BI のダッシュボードには、起動が速くワークロード管理が効くサーバーレス SQL ウェアハウスを選びます。サーバーレスの無いリージョンで、ウェアハウスから自社のネットワーク内のデータベースへフェデレーションする場合は、Pro になります。

アクセス モードは、コンピューティングを誰が使え、どのデータに届くかを決める安全機能で、クラシック コンピューティングには必ず 1 つ付いています。[6]

SAMPLE

アクセス モード	旧称	使い方	対応言語
標準（Standard）	共有（shared）	複数の利用者が使え、利用者間でデータが分離される	Python・SQL・Scala
専用（Dedicated）	シングル ユーザー（single user）	1 人の利用者または 1 つのグループに割り当てる	Python・SQL・Scala・R

選択欄の既定は Auto で、標準に解決されますが、機械学習ランタイム、GPU インスタンス、Databricks Runtime 14.3 未満のどれかを選ぶと専用に解決されます。Databricks は、必要な機能が標準で使えない場合を除き、標準を推奨しています。旧方式の No isolation shared は非推奨です。

標準では、Databricks Runtime for ML、GPU コンピューティング、R、Spark Submit タスクは使えません。ML のライブラリは、標準ではコンピューティング スコープのライブラリとして入れます。専用では、外部カタログを問い合わせるのに接続（connection）の所有者でなければならず、接続に対する CREATE FOREIGN CATALOG 権限だけでは足りません。標準とサーバーレスなら、通常のカタログ権限で足ります。クラスター ログを Unity Catalog のボリュームへ配信する設定は、標準か、利用者に割り当てた専用で使えます。グループに割り当てた専用では使えません。

複数のアナリストが 1 つのクラスターを共用し、互いのデータを分けたいなら標準です。GPU で深層学習を動かす、または R のノートブックを動かすなら、専用です。

SAMPLE

コンピューティングのパフォーマンス設定（Configure compute performance settings）[1.1.2]

コンピューティングは、1 台のドライバー ノードと、0 台以上のワーカー ノードで構成されます。Spark のエグゼキューターはワーカー上で動き、ワーカー 1 台につき 1 つです。分散処理はすべてワーカーで行われるので、ワーカーが 0 台だと Spark のコマンドは実行できません。ドライバーでは Spark 以外のコマンドだけ動きます。[7]

CPU とメモリの大きさはノードの種類（インスタンスの種類）で決まり、メモリ重視や計算重視など、ファミリーごとに向く処理が違います。ドライバーとワーカーには別々の種類を選べますが、既定ではドライバーがワーカーと同じです。`collect()` で大量のデータをドライバーへ集めて分析する計画なら、メモリの大きいドライバーを選びます。ドライバーはアタッチされた全ノートブックの状態を持つので、使わないノートブックは外します。

単一ノードは、画面の Single node チェックボックスで作り、ドライバーがマスターとワーカーを兼ねて、ワーカー ノードはありません。エグゼキューターのスレッドは、論理コア数からドライバー用の 1 コアを引いた数だけ作られます。小さなデータや、分散しない単一ノードの機械学習ライブラリに向きます。複数ノードのコンピューティングは、ワーカー 0 台までは縮められません。ワーカー 0 台にしたい場面では、単一ノードを使います。単一ノードを複数ノードへ変換することはできません。

自動スケールは、Enable autoscaling をオンにして、Worker type の隣の Min と Max に最小と最大のワーカー数を入れます。オフなら、固定のワーカー数を 1 つ入れます。方式はワークスペースの料金プランで決まります。[8]

SAMPLE

項目	最適化された自動スケール	標準の自動スケール
使うプラン	Premium	Standard
スケール アップ	min から max まで最大 2 回のイベント	8 ノードを足し、以降は指数的に増やす
スケール ダウン	ジョブ コンピューティングは直近 40 秒、汎用は直近 150 秒の低利用で縮める。アイドルでなくてもシャッフル ファイルの状態を見て縮められる	ノードの 90% が 10 分間ビジーでなく、コンピューティング全体が 30 秒以上アイドルのとき。1 ノードから指数的に縮める

縮める判断の間隔は Spark 設定 `spark.databricks.aggressiveWindowDownS`（秒）で変えられ、最大は 600 です。値を増やすと縮み方が遅くなります。自動スケールは spark-submit ジョブでは使えません。Structured Streaming では縮み方に制限があり、ストリーミングには拡張自動スケールを持つ Lakeflow の宣言型パイプラインが勧められています。

静的なコンピューティングを 5〜10 ノードの自動スケールへ設定し直すと、現在の大きさは範囲内へ即座に直されます。初期が 6 なら 6 のまま、12 なら 10、3 なら 5 です。

Databricks の自動スケールを使うコンピューティングで、Spark の動的割り当て `spark.dynamicAllocation.enabled` を有効にしてはいけません。両者の判断がぶつかり、`NODES_LOST` のエラーなどの原因になります。

自動終了は、最後のコマンドから指定した分数が過ぎると終了する設定です。非アクティブ時間は 10〜10,000 分（約 7 日）で、0 を入れるかチェックを外すと自動終了しません。Create cluster API の `autotermination_minutes` も同じ範囲です。SQL ウェアハウスには別の設定があります。自動停止（Auto Stop）

SAMPLE

の既定は、Pro とクラシックが 45 分（最小 10 分）、サーバーレスが 10 分（画面の最小は 5 分、API なら 1 分まで）です。[9]

スポット インスタンスは費用を下げる選択で、最初のインスタンスであるドライバーは常にオンデマンドです。追い出されたスポットは、取得できなければオンデマンドで補われます。ドライバーのノードの種類にスポットのプールを選んではいけません。

インスタンス プールは、待機中のインスタンスをためておき、クラスターの起動を速くする仕組みです。サイズは 3 つのパラメーターで決めます。[10]

パラメーター	意味
Minimum Idle Instances	プールが常に待機で持つ最小数。自動終了の設定にかかわらず終了されない
Maximum Capacity	プールが確保できる最大数（任意）。自動スケール中に超える要求があると `INSTANCE_POOL_MAX_CAPACITY_FAILURE` のエラーになる
Idle instance auto termination	最小待機数を超えた分の待機インスタンスを、何分で終了するか

プールのインスタンスの種類は 1 つで、作成後は変更できません。プールにつないだクラスターは、ドライバーとワーカーに同じ種類を使います。Preloaded Databricks Runtime version を選ぶと、そのランタイムをあらかじめ待機インスタンスに載せるので、クラスターがさらに速く起動します。None にすると、起動のたびにダウンロードされて遅くなります。プールのスポットは All Spot で、ドライバーもスポットになり、追い出された分はオンデマンドで補われません。クラスターのスポットと動きが違います。

SAMPLE

プールと自動スケールを組み合わせるときは、要求するクラスターの大きさが最小待機数以下なら、起動は速くなります。要求が最小待機数より大きいと、起動の速さはプールなしと変わりません。最大サイズがプールの Maximum Capacity を超えると、自動スケールで上限を超えて求めた時に、その要求が `INSTANCE_POOL_MAX_CAPACITY_FAILURE` のエラーで失敗します。

たとえば最小待機数 5、Maximum Capacity 10 のプールに、最小 4・最大 12 の自動スケール クラスターをつなぐとします。最小の 4 は最小待機数 5 以下なので、起動は速くなります。最大の 12 は Maximum Capacity の 10 を超えるので、自動スケールで 10 台を超えて求めると、その要求が失敗します。直すには、プールの上限を 12 以上へ上げるか、クラスターの最大を 10 以下へ下げなければなりません。

ワークロードがサーバーレスに対応するなら、プールの代わりにサーバーレスを使うよう Databricks は勧めています。[10]

コンピューティングの機能設定（Configure compute feature settings）[1.1.3]

Photon は、SQL、DataFrame の呼び出し、ETL、ステートレスなストリーミングを速くする、ネイティブの C++ で書かれたベクトル化クエリ エンジンです。Spark のオプティマイザー（Catalyst）が計画を立て、実行の層を Photon が受け持ちます。Photon が対応しない演算に出会うと、その演算の残りは自動で Spark ランタイムに切り替わります。[11]

場面	Photon の扱い
SQL ウェアハウス	既定のエンジン。サーバーレス コンピューティングとサーバーレス パイプラインでも有効

SAMPLE

場面	Photon の扱い
クラシックの汎用・ジョブ・クラシック パイプライン	Databricks Runtime 9.1 LTS 以上で既定で有効。Use Photon Acceleration チェックボックスで切り替える
Clusters API・Jobs API	明示的に `runtime_engine` を `PHOTON` にする
Pipelines API	`photon` を `true` にする

効きやすいのは、大きなデータを扱う長めのクエリです。通常 2 秒未満で終わるクエリは、実行時間の大半が計画と割り当てなので、目立った改善はありません。書き込みでは、UPDATE・DELETE・MERGE INTO・INSERT・CREATE TABLE AS SELECT が速くなります。ストリーミングはステートレスだけが対象で（ソースは Delta・Parquet・CSV・JSON・Kafka・Kinesis、書き込み先は Delta か Parquet）、ステートフルなストリーミングは対象外です。

Photon の有効化が前提になる機能は、読み書きの Predictive I/O と、MERGE・UPDATE・DELETE の動的ファイル プルーニング（Dynamic file pruning）の 2 つです。Photon のインスタンスは、Photon なしの同じインスタンスと DBU の消費率が違います。[11]

Databricks Runtime は、コンピューティングで動くコア コンポーネントの集合で、すべての版に Apache Spark が含まれます。選び方についての勧めは、次の 3 つです。[12]

用途	選ぶ版
汎用コンピューティング	最新の版（最新の最適化と、事前に入っているパッケージとの互換を得る）

SAMPLE

用途	選ぶ版
運用のワークロードを動かすジョブ コンピューティング	LTS（互換性の問題を避け、版を上げる前に十分に試せる）
データ サイエンスと機械学習	Databricks Runtime ML

版は、ベータの後に GA となり、機能開発の期間が約 6 か月続きます。その後 LTS に移り、安定性とセキュリティの修正が、GA から 3 年後のサポート終了日まで届きます。Databricks Runtime 18.x 以前の LTS でない版は、GA から 6 か月でサポートが終わります。[13]

版	Apache Spark	リリース日	サポート終了日
19	4.3.0	2026 年 7 月 23 日	LTS に移る時点で決まる
18 LTS	4.1.0	2026 年 6 月 10 日	2029 年 6 月 10 日
17.3 LTS	4.0.0	2025 年 10 月 22 日	2028 年 10 月 22 日
16.4 LTS	3.5.2	2025 年 5 月 9 日	2028 年 5 月 9 日
15.4 LTS	3.5.0	2024 年 8 月 19 日	2027 年 8 月 19 日
14.3 LTS	3.5.0	2024 年 2 月 1 日	2027 年 2 月 1 日

Databricks Runtime 19 より前は、18.0・18.1 のようなマイナーな機能版が個別にリリースされていました。

ランタイムとアクセス モードは連動します。Auto のアクセス モードは、機械学習ランタイムを選ぶと専用に解決され、標準では Databricks Runtime for ML も GPU も使えません。標準で機械学習をしたい場合は、ライブラリをコンピューティング スコープで入れます。Databricks Runtime 19 以上の標準アクセス モードには、依存関係モード（Dependency mode）というベータの設

SAMPLE

定があります。Environments はワークロード単位でベース環境を使い、Cluster libraries は従来のコンピューティング スコープのライブラリや init スクリプトを使う方式です。

本番の夜間バッチを、予告なしの互換性の変化から守りたいなら、ジョブ コンピューティングに LTS を固定します。アナリストの探索用の汎用コンピューティングは、LTS に固定する必要がありません。機械学習の学習ジョブは ML ランタイムを選び、アクセス モードは専用になります。Photon は、集計と結合が重い長時間のバッチに効きます。ステートフルなストリーミングの集計や、2 秒未満で終わる小さなクエリには、期待した効果が出ません。

コンピューティングのライブラリのインストール（Install libraries for a compute resource）[1.1.4]

ライブラリのスコープは 2 通りです。コンピューティング スコープのライブラリは、そのコンピューティングで動く全ノートブックとジョブで使えます。ノートブック スコープのライブラリは、そのノートブックと、ノートブックをジョブとして動かした実行でだけ使えます。[14]

コンピューティング スコープのライブラリは、次の場所から入れられます。

- PyPI・Maven・CRAN などのパッケージ リポジトリ
- ワークスペース ファイル
- Unity Catalog のボリューム
- クラウドのオブジェクト ストレージ
- ローカルのパス

Python、JAR、R のライブラリを扱えますが、場所とライブラリの種類とアクセス モードの組によっては使えません。Unity Catalog が有効で Databricks

SAMPLE

Runtime 13.3 LTS 以上なら、標準アクセス モードのコンピューティングには、Unity Catalog のボリュームからの導入が推奨です。使う側には、ボリュームの READ VOLUME、親カタログの USE CATALOG、親スキーマの USE SCHEMA が要ります。Maven の座標や JAR のパスは許可リストに入れる必要があり、標準アクセス モードで許可リストへ追加するには MANAGE ALLOWLIST 権限が要ります。[14]

Unity Catalog なしの Databricks Runtime 11.3 LTS 以上では、ワークスペース ファイルからの導入が推奨で、ファイルの上限は 500 MB です。DBFS ルートに置いたライブラリは、DBFS 上のファイルを誰でも書き換えられるため、Databricks Runtime 15.1 以上で非推奨かつ既定で無効です。置き場所には、ワークスペース ファイルかボリュームを選びます。Databricks Runtime 15.0 以上なら、`requirements.txt` で Python の依存を管理でき、対応する場所のどこにも置けます。Python の egg は、Databricks Runtime 13.3 LTS 以下の専用アクセス モード（と No isolation shared）だけで使え、ボリュームやワークスペース ファイルには置けません。wheel か PyPI を使います。

クラスター ライブラリは、Compute から対象を開き、Libraries タブの Install New で入れます（REST API、CLI、Terraform でも入れられます）。インストールに 2 時間を超えたライブラリは失敗として扱われます。アンインストールは、再起動待ちの状態（Uninstall pending restart）になり、クラスターを再起動するまで反映されません。クラスター ライブラリの版を替えるときは、旧版をアンインストールしてから新しい版を入れます。[14] ポリシーにライブラリのインストールを定義した場合は、最大 500 個まで追加できます。そのポリシーで作ったコンピューティングでは、手動のインストールとアンインストールはできません。ライブラリの管理は、ワークスペース管理者がポリシー側で握ります。

ノートブック スコープのライブラリは、`%pip` マジック コマンドで入れます。

SAMPLE

```
%pip install requests==2.32.3
```

そのノートブックと、それをジョブとして動かした実行にだけ効き、セッションをまたいでは残りません。ノートブックをクラスターから切り離すと消えます。複数のノートブックで同じ版を使いたいなら、クラスター ライブラリにします。Scala・SQL・R のノートブックで Python の環境を整えるときは、`%python` と `%pip` を組み合わせます。`%uv pip` は、サーバーレス ノートブックの環境バージョン 5 以上でだけ使えます。クラシックのコンピューティングや古い環境バージョンでは使えません。ノートブック スコープのインストールが増えると、ドライバーのトラフィックが増えます。

標準アクセス モードのクラスターに、社内で作った wheel を全員に配るなら、wheel をボリュームに置き、READ VOLUME などの 3 つの権限を付けて、クラスター ライブラリとして入れます。ある 1 本のノートブックだけが新しい版を試す場合は、`%pip` です。ほかのノートブックの版は変わりません。

コンピューティングへのアクセス許可（Configure access permissions to a compute resource）[1.1.5]

コンピューティングのアクセス許可は 5 段階で、NO PERMISSIONS、CAN VIEW（ベータ）、CAN ATTACH TO、CAN RESTART、CAN MANAGE です。CAN VIEW を使うには、ワークスペースで Can view compute permission のプレビューを有効にします。[15]

できること	CAN VIEW	CAN ATTACH TO	CAN RESTART	CAN MANAGE
ノートブックをアタッチする		○	○	○

SAMPLE

できること	CAN VIEW	CAN ATTACH TO	CAN RESTART	CAN MANAGE
Spark UI とメトリクスを見る	○	○	○	○
終了する・開始と再起動する			○	○
ドライバー ログを見る				○（既定）
編集・ライブラリのアタッチ・サイズ変更・権限の変更				○

権限を設定するには、そのコンピューティングに CAN MANAGE が要ります。画面の Edit permissions から設定します。ワークスペース管理者は全オブジェクトに、作成者は自分で作ったオブジェクトに、自動で CAN MANAGE を持ちます。ドライバー ログは、シークレットが伏せられないため、既定では CAN MANAGE の利用者以外には見えません。CAN ATTACH TO と CAN RESTART にも見せるには、コンピューティングの Spark 設定で `spark.databricks.acl.needAdminPermissionToViewLogs` を `false` にします。この設定はコンピューティングごとに必要です。

コンピューティング ポリシーは、ワークスペース管理者が作る、利用者が作れるコンピューティングの設定を絞る規則の集まりで、Premium プランが必要です。ポリシーでは、決めた設定だけを許して一部の値を固定して隠したり、作れるクラスターの数に上限を付けたり、時間あたりの費用に効く属性を制限して費用を抑えたり、クラスター スコープのライブラリのインストールを強制したりできます。[16]

SAMPLE

Unrestricted cluster creation の資格を持たない利用者は、付与されたポリシーでしかコンピューティングを作れません。全利用者は既定で Personal Compute ポリシーを使え、単一マシンのコンピューティングを作れます。ポリシー名は大文字小文字を区別しません。ポリシーは、Family のテンプレートから作り始められます。

終了した汎用コンピューティングの設定は、30 日後に自動で完全に削除されます。管理者は最大 100 件までピン留めして残せます。

サーバーレスは事情が違い、ワークフローを動かすのにクラスター作成の権限は要りません。既定で全ワークスペース利用者が使え、管理者が制限できます。SQL ウェアハウスは、作成が管理者か Unrestricted cluster creation を持つ利用者に限られ、停止中のウェアハウスを手動で起動するには CAN MONITOR 以上が要ります。

データ エンジニアに、決まった範囲のクラスターだけ作らせ、費用の上限も付けたいなら、ポリシーを作ってそのグループに割り当て、Unrestricted cluster creation は外します。共同作業者には、ノートブックを動かしてもらうだけでよい場合、CAN ATTACH TO を与えます。停止中のクラスターを再起動できるようにするなら CAN RESTART を与え、設定を変えさせるなら CAN MANAGE が必要で、CAN RESTART では足りません。

Unity Catalog の名前付け規則と階層（Apply naming conventions）[1.2.1]

Unity Catalog のオブジェクトは階層になっています。いちばん上がメタストアで、その下にカタログ、スキーマ、そして表・ビュー・ボリューム・関数・モデルが並び、これらの名前は `catalog.schema.object` の 3 階層で指定します。ストレージ資格情報、外部の場所、接続、共有（share）は、カタログの下ではなくメタ

SAMPLE

ストア直下に置かれます。メタストアはリージョンごとに 1 つで、開発・テスト・本番や事業部門のすべてを 1 つのメタストアに載せるのが推奨です。[17]

名前には規則があります。[18]

規則	内容
長さ	255 文字まで
使えない文字	ピリオド(.)、スペース、スラッシュ(/)、ASCII 制御文字(00〜1F)、DELETE 文字(7F)
大文字小文字	Unity Catalog はオブジェクト名を小文字で保存する
特殊文字	ハイフンなどを含む名前は、SQL でバッククォート(`)で囲む
hive_metastore のスキーマ	ASCII の英数字とアンダースコアだけ。Unity Catalog ではスペース・/・. 以外は使える
予約名	INFORMATION_SCHEMA は予約名。同名のスキーマは作れない

カタログはデータ分離の基本の単位で、環境の範囲、チーム、事業部門、またはその組み合わせでカタログを作ると、権限の階層を使ってアクセスを管理しやすくなります。名前の付け方は、文字の規則を守る範囲で、要件に合わせて自分で選びます。

要件	分け方の例
開発と本番を分離する	環境ごとにカタログを作る(例: `dev_sales` 、 `prod_sales`)。本番のカタログを本番のワークスペースにだけ結び付ける

SAMPLE

要件	分け方の例
チームごとに書き込みを分ける	チームごとにスキーマを作り、そのチームにだけ USE SCHEMA と CREATE TABLE を付与する（親カタログの USE CATALOG も要る）
外部の相手と共有する	共有する表を共有（share）に入れ、受け取る側は `CREATE CATALOG ... USING SHARE` で共有カタログを作る。プロトコルは、現在の文書の表記で OpenSharing（以前の名前は Delta Sharing）

たとえば、開発用と本番用で同じ表の定義を使い、環境だけを切り替えたいなら、カタログ名に環境を入れ、スキーマ名と表名は同じにします。`dev_sales.silver.orders` と `prod_sales.silver.orders` なら、コードの中でカタログ名だけを差し替えれば済みます。本番のカタログとスキーマは、所有者を個人にしてはいけません。グループにします。USE CATALOG と USE SCHEMA は、データを見る必要のある利用者にだけ付与します。

カタログの作成（Create a catalog）[1.2.2]

カタログの作成には、メタストアに対する CREATE CATALOG 権限が要ります。[19]

```
CREATE CATALOG IF NOT EXISTS prod_sales
  MANAGED LOCATION 'abfss://sales@mystorage.dfs.core.windows.net/prod'
  COMMENT 'Production sales data';
```

CLI からは `databricks catalogs create` で作ります。ノートブックで作る場合は、Databricks Runtime 11.3 以上で Unity Catalog に対応したアクセス モ

SAMPLE

ードが要りますが、SQL ウェアハウスは常に対応しています。カタログを作ると、`default` と `information_schema` の 2 つのスキーマが自動で作られます。新しいカタログでは、全アカウント ユーザーが既定で BROWSE 権限を持ち、データを読む権限がなくてもメタデータを検索できます。[19]

種類	作り方	追加で要る権限
Standard	`CREATE CATALOG`	MANAGED LOCATION を指定するなら、その外部の場所の CREATE MANAGED STORAGE
Foreign	`CREATE FOREIGN CATALOG ... USING CONNECTION`	接続の所有者、または接続の CREATE FOREIGN CATALOG
Shared	`CREATE CATALOG ... USING SHARE`	メタストアの USE PROVIDER、または provider オブジェクトの所有

`MANAGED LOCATION` は、メタストアの既定とは別の、このカタログ専用のマネージド ストレージの場所です。指定するパスは、外部の場所（external location）として定義済みでなければなりません。外部の場所のパス自体か、そのサブパスを指定できます。メタストア レベルのストレージが無いワークスペースでは、カタログの作成時に必ず指定します。

`RETAIN DROPPED FOR` は、削除したマネージド テーブルを `UNDROP` で戻せる期間です。値は 0 時間（復元しない）または 7〜30 日で、未指定は 7 日です。パブリック プレビューの設定で、Foreign と Shared のカタログには使えません。

カタログをワークスペースに結び付ける（ワークスペース カタログ バインド）と、本番データを開発環境から切り離せます。何もしなければ、同じメタストア

SAMPLE

につないだすべてのワークスペースから、すべてのカタログに既定でアクセスできます。結び付けると、結び付けていないワークスペースからのアクセスは、SELECT などを明示的に付与されていても拒否されます。[20]

結び付けを定義して編集できるのは、メタストア管理者、カタログの所有者、またはカタログに対する MANAGE 権限を持つ利用者です。READ METADATA では閲覧しかできません。結び付けるときは、そのワークスペースを読み取り専用にもできます。読み取り専用にすると、そのワークスペースからの書き込みは止まります。

```
databricks catalogs update prod_sales --isolation-mode ISOLATED
databricks workspace-bindings update-bindings catalog prod_sales --json
'{"add": [{"workspace_id": 1234567890, "binding_type":
"BINDING_TYPE_READ_WRITE"}]}'
```

1 行目で分離モードを、既定の OPEN（全ワークスペースに開く）から ISOLATED に変えます。2 行目で、アクセスさせるワークスペースを割り当てます（`binding_type` は `BINDING_TYPE_READ_WRITE` が既定で、読み取り専用なら `BINDING_TYPE_READ_ONLY`）。[20]新しいワークスペースにできる既定のワークスペース カタログだけは例外で、最初から自分のワークスペース以外には開いていません。結び付けは、カタログのほかに外部の場所、ストレージ資格情報、サービス資格情報にも使えます。

たとえば、本番のカタログを、本番のワークスペースからだけ書き込めるようにし、開発のワークスペースからは読むだけにしたい場合を考えます。カタログを ISOLATED にして、本番ワークスペースを読み取り/書き込み、開発ワークスペースを読み取り専用で結び付けます。開発のワークスペースを結び付けずにおくと、読むこともできません。

SAMPLE

スキーマの作成（Create a schema）[1.2.3]

スキーマの作成には、親カタログの USE CATALOG と CREATE SCHEMA が要ります。MANAGED LOCATION を指定するときだけ、そのパスを支える外部の場所の CREATE MANAGED STORAGE 権限も加わります。[21]

```
CREATE SCHEMA IF NOT EXISTS prod_sales.silver
  COMMENT 'Cleansed sales tables'
  MANAGED LOCATION
'abfss://sales@mystorage.dfs.core.windows.net/prod/silver'
  WITH DBPROPERTIES (owner_team = 'data-eng');
```

SCHEMA と DATABASE は同じ意味ですが、SCHEMA が推奨です。Unity Catalog では `LOCATION` 句は使えません。保存先は `MANAGED LOCATION` で指定し、`LOCATION` は Hive メタストア用です。指定できる句は、COMMENT、DEFAULT COLLATION、MANAGED LOCATION、RETAIN DROPPED FOR、WITH DBPROPERTIES です。DEFAULT COLLATION は Databricks Runtime 17.1 以上で、未指定ならカタログから引き継ぎます。各カタログには読み取り専用の INFORMATION_SCHEMA が自動で入るので、同じ名前のスキーマは作れません。Unity Catalog が有効なワークスペースで Hive メタストアにスキーマを作るときは、カタログ名に `hive_metastore` を使います。

管理された保存場所は、メタストア、カタログ、スキーマのどのレベルでも設定でき、データはいちばん下に定義された場所に置かれます。マネージド テーブルとボリュームのデータは、スキーマに場所があればスキーマの場所に、なければカタログの場所に、それも無ければメタストアの場所に置かれます。Databricks は、データ分離の主な単位として、カタログ レベルの保存場所を

SAMPLE

優先することを勧めています。メタストア レベルのストレージは、初期の Unity Catalog では必須でしたが、今は必須ではありません。

復元期間も同じ考え方で、カタログとスキーマの両方に `RETAIN DROPPED FOR` を設定したら、スキーマ側が優先されます。この設定は、設定した後に削除したテーブルにだけ効きます。

`DROP SCHEMA ... CASCADE` の動きは、Unity Catalog のスキーマなら、配下のマネージド テーブルのファイルが再帰的に削除され、外部テーブルのファイルは削除されません。Hive メタストアのスキーマを CASCADE で消すと、種類に関係なくそのパス以下のファイルが全部消えます。

たとえば財務チームのデータだけ別のストレージ アカウントに置く必要があり、カタログは全社で 1 つの場合は、財務スキーマに `MANAGED LOCATION` を付けます。財務スキーマのマネージド テーブルはスキーマの場所に入り、ほかのスキーマはカタログの場所のままです。CREATE SCHEMA を実行するユーザーに、財務用の外部の場所の CREATE MANAGED STORAGE がなければ、スキーマは作れません。

ボリュームの作成（Create volumes）[1.2.4]

ボリュームは、表にしないファイルを扱う Unity Catalog のオブジェクトで、`catalog.schema.volume` の 3 階層目に置かれます。表が表形式のデータを管理するのに対し、ボリュームは形式を問わない非表形式のデータを管理します。取り込み前の着地点、Auto Loader や COPY INTO のステージング、JAR や wheel の置き場に使い、マネージドと外部の 2 種類があります。[22]

項目	マネージド ボリューム	外部ボリューム
保存場所	スキーマの管理された保存場所の中	既存のクラウド ストレージの場所に登録する

SAMPLE

項目	マネージド ボリューム	外部ボリューム
削除したとき	Unity Catalog が配置と削除を管理する。削除後 7 日保持される	ボリュームを DROP しても、データはクラウドに残る
作成の構文	`CREATE VOLUME`	`CREATE EXTERNAL VOLUME ... LOCATION`
向く場面	Databricks だけが使う場合のいちばん簡単な選択	Databricks と外部システムの両方が同じファイルを使う場合

```
CREATE VOLUME prod_sales.raw.landing;


CREATE EXTERNAL VOLUME prod_sales.raw.partner_files
  LOCATION 'abfss://partner@mystorage.dfs.core.windows.net/files'
  COMMENT 'Files delivered by partner';
```

EXTERNAL を付けなければマネージドです。作成に要る権限は、親カタログの USE CATALOG、親スキーマの USE SCHEMA、スキーマの CREATE VOLUME です。外部ボリュームでは、LOCATION を支える外部の場所の CREATE EXTERNAL VOLUME も要ります。外部ボリュームの場所は、外部の場所の中のディレクトリで、ほかのボリュームや表の場所と重なってはいけません。

ファイルは次の形のパスで参照し、Spark では先頭に `dbfs:` を付けても通ります。

```
/Volumes/<catalog>/<schema>/<volume>/<path>/<file-name>
```

SAMPLE

```
SELECT * FROM csv.`/Volumes/prod_sales/raw/landing/orders.csv`;
```

`/Volumes/<catalog>/<schema>/<volume>` の部分は Unity Catalog が自動で管理する読み取り専用のディレクトリで、ファイル システムのコマンドでは作れず、消せません。ボリュームを使えるのは、SQL ウェアハウス、または Databricks Runtime 13.3 LTS 以上のクラスターです。ボリューム内のファイルを Unity Catalog の表として登録することはできません。ボリュームはパスによるアクセス専用です。

外部の場所とストレージ資格情報は、外部ボリュームと外部テーブルの前提です。ADLS Gen2 に接続するには、Unity Catalog のオブジェクトが 2 つ要ります。マネージド ID を指すストレージ資格情報（storage credential）と、パスとその資格情報を結ぶ外部の場所（external location）です。[23]

1. Azure で「Azure Databricks 用のアクセス コネクタ」を作る（リソース グループの共同作成者か所有者が要る）。
2. そのマネージド ID に、ストレージ アカウントの Storage Blob Data Contributor ロールを付ける（所有者か User Access Administrator が要る）。
3. ストレージ資格情報を作る。メタストアの CREATE STORAGE CREDENTIAL 権限が要る。
4. 外部の場所を作る。メタストアとストレージ資格情報の両方の CREATE EXTERNAL LOCATION 権限が要る。

ストレージ アカウントは、階層型名前空間が有効でなければなりません。不変ポリシー（WORM）を有効にしたコンテナーは、Unity Catalog が DELETE 権限を使うため、外部の場所に使えません。パブリック ネットワーク アクセスを無効にしているなら、Allow Azure trusted services を許可します。外部の場

SAMPLE

所の WRITE FILES は、READ FILES と一緒に付与しなければ PERMISSION_DENIED になります。

たとえばパートナー企業が書き込む既存のコンテナーを、Databricks からも読みたく、パートナーの側の書き込みも続くなら、外部ボリュームです。外部システムがストレージへ直接読み書きする分は、Unity Catalog の権限では守られません。ストレージ側のアクセス制御も合わせます。Databricks だけが使うステージングなら、マネージド ボリュームで足ります。

テーブル・ビュー・マテリアライズド ビューの作成（Create tables, views, and materialized views） [1.2.5]

テーブルは、Unity Catalog のマネージド テーブルが既定で推奨です。Delta Lake と Apache Iceberg に対応し、保存、読み書き、最適化は Unity Catalog が受け持つので、利用者が配置や整理を管理する必要はありません。予測最適化（Predictive Optimization）は既定で有効です。マネージド テーブルを削除すると、復元期間（既定 7 日）が過ぎてからデータ ファイルが削除されます。[24]

SAMPLE

```
CREATE TABLE prod_sales.silver.orders (
  order_id BIGINT,
  customer_id BIGINT,
  amount DECIMAL(10,2),
  order_date DATE
);

CREATE TABLE prod_sales.silver.orders_2026 AS
SELECT * FROM prod_sales.silver.orders WHERE order_date >= '2026-01-
01';
```

ビューは、1 つ以上の表またはビューへのクエリ結果を表す、読み取り専用のオブジェクトです。作成してもデータの処理や書き込みは起きず、クエリの文だけがスキーマに登録されます。作成には、メタストア管理者か、USE CATALOG・USE SCHEMA と対象スキーマの CREATE TABLE が要り、作成した人が所有者になります。一時ビュー（TEMPORARY VIEW）はスキーマにもカタログにも登録されません。ノートブックやジョブでは、そのノートブックやスクリプトの中だけで有効で、ノートブックを切り離すと消えます。Databricks SQL では、クエリ単位で有効です。

```
CREATE OR REPLACE VIEW prod_sales.gold.big_orders AS
SELECT order_id, amount FROM prod_sales.silver.orders WHERE amount >
10000;
```

マテリアライズド ビューは、クエリ結果を物理的に保存する Unity Catalog のマネージド テーブルで、結果を先に計算しておくので、問い合わせが速く、変換の層を重ねる処理や BI ダッシュボードに向きます。作成できる場所は、

SAMPLE

CREATE MATERIALIZED VIEW のページでは Pro またはサーバーレスの SQL ウェアハウスとパイプラインの中、Materialized views のページではそれに加えてサーバーレスの汎用コンピューティングで動くノートブックと、文書によって書き方が違います。クラシックの SQL ウェアハウスでは作れません。[2526] 作成と更新は、ビューごとに自動で作られるサーバーレス パイプラインが処理します。[27]

```
CREATE MATERIALIZED VIEW prod_sales.gold.daily_sales
  SCHEDULE EVERY 1 DAY
AS
SELECT order_date, SUM(amount) AS total
FROM prod_sales.silver.orders
GROUP BY order_date;
```

作成者がマテリアライズド ビューの所有者で、ソースの表に対する SELECT、USE CATALOG・USE SCHEMA、スキーマの CREATE MATERIALIZED VIEW が要ります。更新できるのは所有者か REFRESH 権限の保持者で、問い合わせには SELECT が要ります。

更新の方法は、手動、時間による定期実行、上流の更新による自動実行の 3 つです。

方法	書き方	注意
手動	`REFRESH MATERIALIZED VIEW <name>;`	既定は同期。`ASYNC` を付けると非同期

SAMPLE

方法	書き方	注意
時間で定期	`SCHEDULE [REFRESH] EVERY n { HOUR | DAY | WEEK }` または `SCHEDULE CRON '<quartz 式>' AT TIME ZONE '...'`	HOUR は 1〜72、DAY は 1〜31、WEEK は 1〜8。CRON は秒・分・時・日・月・曜日の 6 フィールド
上流の更新で自動	`TRIGGER ON UPDATE [AT MOST EVERY <interval>]`	間隔は最短 1 分（既定も 1 分）。上流は 1 つのビューあたり 10 まで。本番向けに推奨

TRIGGER ON UPDATE の上流にできるのは、外部テーブルかマネージドの Delta テーブル（マテリアライズド ビューとストリーミング テーブルを含む）と、依存先が対応する種類に限られるマネージド ビューです。OpenSharing の共有テーブルは上流にできません。

更新は、増分更新（変更分だけ計算）か完全更新（全体を再計算）です。増分更新を効かせるには、ソースを行トラッキングを有効にした Delta テーブルにし、`EXPLAIN CREATE MATERIALIZED VIEW` で、そのクエリが増分更新できるかを確かめます。既定では、Databricks がコスト モデルで 2 つのどちらかを選び、`REFRESH POLICY` を指定すると、増分を優先するか、完全を優先するかを上書きできます。

項目	ビュー	マテリアライズド ビュー
結果の保存	しない（クエリの文だけ登録）	する（マネージド テーブル）
最新性	問い合わせのたびに計算するので、常に元のデータの最新	更新するまで古い内容が見える

SAMPLE

項目	ビュー	マテリアライズド ビュー
向く場面	常に最新を見せたい。計算が軽い	更新の遅れを許せる。重い集計を何度も読む

たとえば昼に 300 人が開くダッシュボードが、重い集計の表を読んでいて遅いなら、集計をマテリアライズド ビューにして、上流の表が更新されたら自動で更新される `TRIGGER ON UPDATE` を付けます。クラシックの SQL ウェアハウスしか使えない環境では、そのビューは作れません。Pro かサーバーレスを用意するか、パイプラインの中で定義します。

接続による外部カタログの実装（Implement a foreign catalog by configuring connections）[1.2.6]

Lakehouse Federation は、データを移さずに外部のデータ システムへ問い合わせる仕組みです。Unity Catalog の接続（connection）が外部のデータベースへの経路と資格情報を持ち、外部カタログ（foreign catalog）が外部のデータベースを鏡のように映します。外部カタログは読み取り専用で、権限は通常の Unity Catalog の権限で付与します。[28]

設定の手順

1. 資格情報と接続先（JDBC の URL）を持つ接続を作る。
2. その接続から外部カタログを作る。
3. 利用者に外部カタログの表への権限を付与する。
4. クエリを実行する。クエリは外部のデータベースへ押し下げられる。

SAMPLE

```
CREATE CONNECTION postgresql_connection TYPE POSTGRESQL
OPTIONS (
  host 'pg.example.com',
  port '5432',
  user secret('my_scope', 'pg_user'),
  password secret('my_scope', 'pg_password')
);

CREATE FOREIGN CATALOG IF NOT EXISTS pg_cat
USING CONNECTION postgresql_connection
OPTIONS (database 'sales');
```

`CREATE CONNECTION` は、標準に合わせて `CREATE SERVER` とも書けます。接続の種類（TYPE）は、DATABRICKS、HTTP、MYSQL、POSTGRESQL、REDSHIFT、SNOWFLAKE、SQLDW（Azure Synapse）、SQLSERVER です。

接続の種類	必須のオプション
MYSQL・POSTGRESQL・REDSHIFT・SQLDW・SQLSERVER	host・port・user・password
SNOWFLAKE	host・port・sfWarehouse・user・password
DATABRICKS	host・httpPath・personalAccessToken
HTTP	host・bearer_token

パスワードやトークンのような資格情報は、リテラルを直接書いてはいけません。`secret()` 関数でシークレットから渡します。`CREATE FOREIGN CATALOG` に必要なのは、接続の所有者であること、または接続に対する CREATE FOREIGN

SAMPLE

CATALOG 権限です。コンピューティングは、Pro またはサーバーレスの SQL ウェアハウスか、対応した版の Databricks Runtime を使います。版の下限は、文書どうしで揃っていません。Create catalogs のページでは 13.1 以上、CREATE CONNECTION のリファレンスでは 13.3 LTS 以上です。[2930] 外部データへの経路には 2 種類あります。

項目	クエリ フェデレーション	カタログ フェデレーション
実行	JDBC で外部のデータベースへ押し下げる。Azure Databricks と外部の計算の両方で動く	オブジェクト ストレージのデータを、Azure Databricks のコンピューティングだけで直接読む
書き込み	不可（読み取り専用）	不可（読み取り専用）
向く場面	アドホックなレポート、BI、運用データベースへの概念実証	Unity Catalog への段階的な移行、外部カタログとの長期的な併用
追加で要るもの	接続と外部カタログ	接続に加え、表のパスのためのストレージ資格情報と外部の場所
対応ソース	MySQL・PostgreSQL・Teradata・Oracle・Amazon Redshift・Salesforce Data 360・Snowflake・SQL Server・Azure Synapse・Google BigQuery・Databricks	Hive メタストア（従来の Databricks のものと外部のもの）・Salesforce Data 360・Snowflake・Palantir Foundry・Workday Data Connect・OneLake

書き込みは、どちらのフェデレーションでもできません。書き込みが必要なときや、フェデレーションが対応しないソースには、Spark のデータ ソース API を使います。ソースが Lakeflow Connect にも対応し、大きなデータ量や低い遅延を重視するなら、Lakeflow Connect のほうが勧められています。

SAMPLE

クエリでは Apache Spark のデータ型を使います。専用アクセス モードのコンピューティングで外部カタログを問い合わせるには、接続の所有者でなければなりません。標準アクセス モードとサーバーレスは、通常の権限で足ります。

たとえば BI チームが、運用の PostgreSQL に直接つなぎ、Unity Catalog の表の権限で守りたい場合は、接続と外部カタログを作り、チームの SELECT を付与します。外部のデータベースへ書き戻す要件が加わった場合、外部カタログは読み取り専用なので、書き込みはフェデレーションの外で行うことになり、外部カタログでは行えません。

マネージド テーブルと外部テーブルの DDL（Implement DDL operations on managed and external tables）[1.2.7]

マネージド テーブルは、データのライフサイクルまで Unity Catalog が管理します。外部テーブルは、メタデータだけを Unity Catalog が管理し、データの場所と配置、ライフサイクルは自分たちが管理します。作成するときは、場所を指定します。場所は、Unity Catalog に登録した外部の場所の中でなければなりません。[31]

```
CREATE TABLE prod_sales.bronze.raw_orders (
  order_id BIGINT,
  amount DECIMAL(10,2)
)
LOCATION 'abfss://sales@mystorage.dfs.core.windows.net/raw/orders';
```

`EXTERNAL` キーワードも使え、使うときは `LOCATION` が必須です。外部テーブルを作るには、外部の場所の CREATE EXTERNAL TABLE、親カタログの USE CATALOG、親スキーマの USE SCHEMA と CREATE TABLE が要ります。使え

SAMPLE

るファイル形式は、DELTA、CSV、JSON、AVRO、PARQUET、ORC、TEXT で、これ以外の形式は外部テーブルに使えません。Databricks は多くの場合マネージド テーブルを勧めています。外部テーブルを選ぶのは、JSON や Avro のようにマネージド テーブルと互換性のない既存データを登録したいときと、Databricks 以外のクライアントから直接データへ届く必要があるときです。外部のクライアントがデータ ファイルへ直接届く場合、Unity Catalog の権限は強制されません。

`ALTER TABLE` の主な操作と、必要な権限は次のとおりです。

操作	例	必要な権限
列の追加	`ALTER TABLE t ADD COLUMN note STRING;`	MODIFY
列の名前の変更	`ALTER TABLE t RENAME COLUMN name TO first_name;`	所有者[32]
列の削除	`ALTER TABLE t DROP COLUMN note;`	MODIFY
プロパティの設定と解除	`ALTER TABLE t SET TBLPROPERTIES ('k' = 'v');`	MODIFY
名前の変更	`ALTER TABLE t RENAME TO t2;`	所有者
所有者の変更	`ALTER TABLE t SET OWNER TO grp;`	MANAGE か所有者

`RENAME TO` は、Unity Catalog では同じカタログの中でしか移せません。既存の Delta テーブルへ列を追加するときは、DEFAULT 値を同時には定義できま

SAMPLE

せん。追加した列は、既存の行では NULL になり、既定値は付きません。既定値を付けたいなら、追加の後に `ALTER COLUMN` で定義します。`CREATE OR REPLACE TABLE` は Delta と Iceberg だけで使え、履歴、権限、行フィルター、列マスクを引き継ぐので、DROP してから作り直すより推奨です。DROP TABLE は、表の MANAGE 権限、所有者、またはスキーマ・カタログ・メタストアの所有者が実行できます。

項目	マネージド テーブル	外部テーブル
DROP したとき	メタデータが消え、復元期間が過ぎるとデータ ファイルも消える	メタデータだけが消え、データ ファイルは残る
UNDROP	復元期間内なら戻せる	戻せる（外部の場所の CREATE EXTERNAL TABLE 権限も要る）

UNDROP は、既定で削除後 7 日以内なら戻せ、期間はカタログかスキーマの設定で 0（無効）または 7〜30 日に変えられ、スキーマ側の設定が優先です。作る時は `CREATE SCHEMA … RETAIN DROPPED FOR`、後から変える時は `ALTER SCHEMA … RETAIN DROPPED TO 7 DAYS` のように書きます（カタログも同じ）。[33]復元期間が過ぎた表は、UNDROP では戻せません。データ ファイルは 48 時間以内にクラウド ストレージから削除されます。

```
DROP TABLE prod_sales.silver.orders;
SHOW TABLES DROPPED IN prod_sales.silver;
UNDROP TABLE prod_sales.silver.orders;
UNDROP TABLE WITH ID '6ca7be55-8f58-47a7-85ee-7a59082fd17a';
```

SAMPLE

UNDROP の構文は `UNDROP { MATERIALIZED VIEW | TABLE } { name | WITH ID id }` です。同じ名前の削除済みの表が複数あるときは、`SHOW TABLES DROPPED` で ID を調べ、`WITH ID` で戻します。同じ名前の現役の表があるときは、先にそちらを `RENAME TO` で改名します。表のメタデータ（権限・列の定義・プロパティ）は戻りますが、主キーと外部キーの制約は戻りません。制約は `ALTER TABLE ADD CONSTRAINT` で作り直します。マテリアライズド ビューとストリーミング テーブルは、裏のパイプラインが残っている場合だけ戻せ、Databricks SQL で作ったものは戻せません。一時テーブルも戻せません。

たとえば誤って外部テーブルを DROP し、7 日以内なら、`UNDROP TABLE` で戻します。データ ファイルは DROP の後も外部ストレージに残っていたので、中身は変わりません。誤ってマネージド テーブルを DROP し、8 日目に気づいた場合は、復元期間が 7 日の既定なら戻せません。カタログかスキーマで期間を延ばしてあれば戻せますが、延ばした設定は、設定した後に削除した表にだけ効きます。

AI/BI Genie のデータ検出向け手順（Configure AI/BI Genie instructions）[1.2.8]

AI/BI Genie の「スペース」は、現在の画面と文書では Genie Agents（旧称 Genie Spaces）と呼ばれます。Unity Catalog に登録したデータを自然言語で問い合わせる、ドメイン向けのチャットです。アナリストが、データセット、例の SQL、SQL 式、テキストの指示を整え、業務の利用者がチャットで質問します。[34]

項目	内容
データ	Unity Catalog に登録済みの表・ビュー・メトリック ビュー。1 つの Genie Agent に最大 50

SAMPLE

項目	内容
コンピューティング	Pro またはサーバーレスの SQL ウェアハウス（サーバーレスを推奨）。設定者に少なくとも CAN USE が要る
作成と編集の権限	Databricks SQL のエンタイトルメント、データへの SELECT、Genie Agent の ACL で CAN EDIT 以上
作成者の既定の権限	作成した Genie Agent の CAN MANAGE
共有の段階	CAN MANAGE・CAN EDIT・CAN RUN・CAN VIEW。質問する利用者には少なくとも CAN VIEW か CAN RUN が要る

データへのアクセスは、ウェアハウスの資格情報が Agent に組み込まれていても、常に各利用者自身の Unity Catalog の権限で判定されます。Volumes のファイル分析はパブリック プレビュー、MCP コネクタの接続はベータの機能です。Genie の答えを正しくする材料は、4 つです。

- **例の SQL クエリ**: 典型的な質問を題にして、その答えとなる SQL を登録する。静的でも、パラメーター付きでもよい
- **SQL 関数**: 静的やパラメーター付きの SQL で書けない質問のために、Unity Catalog に登録した関数（スカラーか表値）を渡す
- **テキストの指示（一般の指示）**: 用語や業務ルールを文章で書く
- **ナレッジ ストア**: この Agent だけで使う表と列の説明、列の同義語、結合（JOIN）の関係、SQL 式（メジャー、フィルター、ディメンション）

指示の上限は 1 つの Agent に 100 件で、例の SQL クエリ 1 本、SQL 関数 1 つ、一般の指示のテキスト全体 1 つが、それぞれ 1 件と数えられます。ナレッジストアのスニペットは、表の説明・結合の関係・SQL 式を合わせて 200 件まで

SAMPLE

です。テキストの指示、例の SQL、関数、列の説明、プロンプト マッチングの設定は、この 200 件に入りません。

信頼できる資産（trusted assets）は、パラメーター付きの例の SQL クエリと SQL 関数です。チャット モードで、パラメーター付きクエリの本文をそのまま使って答えると、Genie は「検証済みの回答」を返します。追加と削除ができるのは CAN EDIT 以上の利用者で、SQL 関数を使う利用者には、その関数の EXECUTE 権限が要ります。例のクエリのパラメーターの型は、String、Date、Date and Time、Decimal、Integer で、既定は String です。値が型に合わないと、Genie は値を誤った型として扱い、答えの正確さが保証されません。

指示を書くときは、質問を、利用者が自然に尋ねる言い方で書き、その答えの SQL と組にします。組織やデータに固有の論理（たとえば、どの商談を「パイプライン」と数えるか）を例にします。ビジネスの論理は、文章の指示より SQL 式（メジャー、フィルター、ディメンション）で定義するほうが、書いたとおりに適用されるので、一貫した結果になります。Genie は非決定的に動くため、指示同士が矛盾したり曖昧だったりすると、望ましくない答えになりやすくなります。設定の作業では、矛盾を見つけて解消することが欠かせません。

指示を置く場所にも使い分けがあります。テキストの指示は、全体に当てはまる前提（会計年度の始まり、小数の丸め方、応答の言語）に限ります。一部の質問にだけ関係する内容は、例のクエリや関数にするか、該当する表の説明に書きます。表と列の説明、列の同義語は Agent ごとのメタデータで、基になる Unity Catalog の表のメタデータは変わりません。不要な列や重複した列を隠すと、Genie が迷いにくくなります。Genie Code は Agent の作成時に自動で立ち上がり、データを読んで表の説明や例のクエリを提案するので、残したい提案だけ受け入れます。

SAMPLE

たとえば「受注額」という列が複数の表にあって、Genie が違う列を使ってしまう場合は、使う表と列を限定し、列に説明を付け、受注額を SQL 式のメジャーとして登録します。よく出る質問が複雑な結合を要するなら、その SQL をパラメーター付きの例のクエリにして、信頼できる資産にすると、利用者に検証済みの回答が返ります。

SAMPLE

サンプルはここまでです

続き(第2章〜巻末)は、DP-750 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

Footnotes

1. Compute（serverless compute・classic compute・SQL warehouses の 3 系統） https://learn.microsoft.com/en-us/azure/databricks/compute/ ↩

2. Serverless compute（利用要件・versionless） https://learn.microsoft.com/en-us/azure/databricks/compute/serverless/ ／ Serverless notebooks（既定の接続・タイムアウト）https://learn.microsoft.com/en-us/azure/databricks/compute/serverless/notebooks ↩

3. Configure compute for jobs（Recommended compute for each task） https://learn.microsoft.com/en-us/azure/databricks/jobs/compute ／ Run your Lakeflow Jobs with serverless compute（対応するタスク） https://learn.microsoft.com/en-us/azure/databricks/jobs/run-serverless-jobs ／ Serverless compute（How do I estimate costs・性能モード） https://learn.microsoft.com/en-us/azure/databricks/compute/serverless/ ↩

4. https://learn.microsoft.com/en-us/azure/databricks/compute/sql-warehouse/warehouse-types ↩

SAMPLE

5. SQL warehouse types（Performance capabilities by type・各種類の節） https://learn.microsoft.com/en-us/azure/databricks/compute/sql-warehouse/warehouse-types ↩

6. Compute configuration reference（Access modes） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ／ Standard compute requirements and limitations https://learn.microsoft.com/en-us/azure/databricks/compute/standard-limitations ／ Foreign catalogs（Query data in a foreign catalog） https://learn.microsoft.com/en-us/azure/databricks/query-federation/foreign-catalogs ↩

7. Compute configuration reference（Worker node type・Single-node compute・Driver type） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ↩

8. Compute configuration reference（Enable autoscaling・How autoscaling behaves・Autoscaling example） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ↩

9. Manage compute（Configure automatic termination） https://learn.microsoft.com/en-us/azure/databricks/compute/clusters-manage ／ Compute configuration reference（Spot instances） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ／ Create a SQL

SAMPLE

warehouse（Auto Stop） https://learn.microsoft.com/en-us/azure/databricks/compute/sql-warehouse/create ↩

10. Pool configuration reference（Pool size・Instance types・Preloaded Databricks Runtime version・Spot instances） https://learn.microsoft.com/en-us/azure/databricks/compute/pools ／ Compute configuration reference（Autoscaling with pools） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ↩ ↩2

11. What is Photon（How Photon works・Photon enablement・Features that require Photon enablement） https://learn.microsoft.com/en-us/azure/databricks/compute/photon ↩ ↩2

12. Compute configuration reference（Databricks Runtime versions・Dependency mode） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ／ Databricks Runtime support lifecycles https://learn.microsoft.com/en-us/azure/databricks/release-notes/runtime/databricks-runtime-ver ／ Databricks Runtime release notes versions and compatibility https://learn.microsoft.com/en-us/azure/databricks/release-notes/runtime/ ／ Standard compute requirements and limitations https://learn.microsoft.com/en-us/azure/databricks/compute/standard-limitations ↩

13. https://learn.microsoft.com/en-us/azure/databricks/release-notes/runtime/databricks-runtime-ver ↩

14. Libraries（Compute-scoped libraries・Recommendations for uploading libraries） https://learn.microsoft.com/en-

SAMPLE

us/azure/databricks/libraries/ ／ Notebook-scoped Python libraries https://learn.microsoft.com/en-us/azure/databricks/libraries/notebooks-python-libraries ／ Compute-scoped libraries https://learn.microsoft.com/en-us/azure/databricks/libraries/cluster-libraries ／ Create and manage compute policies（ライブラリ数の上限） https://learn.microsoft.com/en-us/azure/databricks/admin/clusters/policies ↩ ↩2 ↩3

15. Access control lists（Compute ACLs） https://learn.microsoft.com/en-us/azure/databricks/security/auth/access-control/ ／ Manage compute（Compute permissions・Configure compute permissions） https://learn.microsoft.com/en-us/azure/databricks/compute/clusters-manage ↩

16. Create and manage compute policies https://learn.microsoft.com/en-us/azure/databricks/admin/clusters/policies ／ Compute configuration reference（Compute policy） https://learn.microsoft.com/en-us/azure/databricks/compute/configure ／ Run your Lakeflow Jobs with serverless compute https://learn.microsoft.com/en-us/azure/databricks/jobs/run-serverless-jobs ／ SQL warehouses（Start a SQL warehouse） https://learn.microsoft.com/en-us/azure/databricks/compute/sql-warehouse/ ↩

17. What is Unity Catalog https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/ ／ Unity

SAMPLE

Catalog best practices（Cross-region and cross-platform sharing・Catalogs and schemas ・ Managed storage ） https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/best-practices ↩

18. Names （ Unity Catalog の名前の規則 ） https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-names ／ Create schemas （ INFORMATION_SCHEMA ） https://learn.microsoft.com/en-us/azure/databricks/schemas/create-schema ↩

19. Create catalogs （ Requirements ・ Create a catalog ） https://learn.microsoft.com/en-us/azure/databricks/catalogs/create-catalog ／ CREATE CATALOG （ Syntax ・ MANAGED LOCATION ・ RETAIN DROPPED FOR ） https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-catalog ↩ ↩2

20. Workspace-catalog binding （ How workspace-catalog binding works ・ Bind a catalog to one or more workspaces ・ CLI ） https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/access-control/workspace-catalog-binding ↩ ↩2

21. Create schemas（ Requirements ） https://learn.microsoft.com/en-us/azure/databricks/schemas/create-schema ／ CREATE SCHEMA （ Syntax ・ MANAGED LOCATION ・ CASCADE ） https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-schema ／ Unity Catalog best

SAMPLE

practices（Managed storage）https://learn.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/best-practices ↩

22. What are Unity Catalog volumes（Managed versus external volumes・Path）https://learn.microsoft.com/en-us/azure/databricks/volumes/ ／ CREATE VOLUME（Permissions・Syntax）https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-volume ↩

23. Connect to Azure Data Lake Storage（storage credential と external location の作成・Requirements）https://learn.microsoft.com/en-us/azure/databricks/connect/unity-catalog/cloud-storage/external-locations-adls ／ Manage external locations（WRITE FILES と READ FILES）https://learn.microsoft.com/en-us/azure/databricks/connect/unity-catalog/cloud-storage/manage-external-locations ↩

24. Unity Catalog managed tables https://learn.microsoft.com/en-us/azure/databricks/tables/managed ／ What is a view（Views in Unity Catalog・Temporary views）https://learn.microsoft.com/en-us/azure/databricks/views/ ／ CREATE VIEW（Syntax・権限）https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-view ↩

25. https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-materialized-view ↩

SAMPLE

26. https://learn.microsoft.com/en-us/azure/databricks/views/materialized ↩

27. Materialized views（Refresh・Incremental refresh）https://learn.microsoft.com/en-us/azure/databricks/sql/mv-st/materialized ／ CREATE MATERIALIZED VIEW（Syntax・SCHEDULE・TRIGGER ON UPDATE・Required permissions）https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-materialized-view ↩

28. What is Lakehouse Federation（query federation と catalog federation）https://learn.microsoft.com/en-us/azure/databricks/query-federation/ ／ Foreign catalogs https://learn.microsoft.com/en-us/azure/databricks/query-federation/foreign-catalogs ／ Connections https://learn.microsoft.com/en-us/azure/databricks/query-federation/connections ／ CREATE CONNECTION（connection_type・Options by connection type）https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-connection ／ CREATE CATALOG（CREATE FOREIGN CATALOG）https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-catalog ↩

29. https://learn.microsoft.com/en-us/azure/databricks/catalogs/create-catalog ↩

30. https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-connection ↩

SAMPLE

31. Unity Catalog external tables https://learn.microsoft.com/en-us/azure/databricks/tables/external ／ ALTER TABLE（Required permissions・RENAME TO・ADD COLUMN） https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-alter-table ／ DROP TABLE https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-drop-table ／ UNDROP https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-undrop-table ／ CREATE TABLE（REPLACE） https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-create-table-using ↩

32. https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-ddl-alter-table ↩

33. https://learn.microsoft.com/en-us/azure/databricks/tables/managed ↩

34. Genie Agents https://learn.microsoft.com/en-us/azure/databricks/genie-agents/ ／ Set up and manage a Genie Agent（Technical requirements and limits・Required permissions） https://learn.microsoft.com/en-us/azure/databricks/genie-agents/set-up ／ Tune Genie Agent quality（Instructions・Trusted assets・knowledge store） https://learn.microsoft.com/en-us/azure/databricks/genie-agents/tune-quality ↩

SAMPLE

DP-750 学習用テキスト(無料サンプル)

版

2026年11月版(DP-750 出題範囲(Skills measured)2026-03-11版 準拠)

発行

IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ

info@shikaku-doujou.com

Microsoft、Azure、Azure Databricks および Microsoft の製品名・サービス名は、Microsoft Corporation の米国およびその他の国における商標または登録商標です。Databricks、Apache Spark、Delta Lake、およびそれぞれのロゴと、記載されているその他の会社名・製品名・サービス名は、各権利者の商標または登録商標です。

本教材は公式シラバスの内容を基に作成したもので、Microsoft Corporation との提携・承認関係はありません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。

SAMPLE