Engineer

データレイク

≈ 元データを元の形のまま保管する場所

どのようなものか

データレイクは、データの発生元が提供する最も細かな粒度で、届いたままの元データを保管する場所です。これまでに出ていない問いにも、後からそのデータを使って答えられるようにします。実際には、分散ファイルシステムやオブジェクトストレージ上のファイルにデータを置き、クエリーエンジンがそのファイルを直接読みます。ファイル形式には、Apache Parquetなどの列指向形式がよく使われます。

この用語を生んだのは、PentahoのJames Dixonです。Pentahoが初のHadoop対応版を発表した週に書いた2010年10月14日のブログ記事で、Hadoopを使う企業から聞いた話をまとめました。十社のうち八社か九社が扱っていたのは非構造化データではなく、構造化または半構造化データで、通常は一つのアプリケーションやシステムから来るものでした。あらかじめ分かっている問いも少しありましたが、後からさらに多くの問いが生まれ、複数の利用者グループがそれぞれ答えを求めていました。また、データ量が大きく、リレーショナルデータベースには技術的に収められないか、収めようとすると費用が見合いませんでした。従来の対処法は、特に関心のある属性を選び、集計してデータマートに入れることでした。これで答えられるのは事前に選んだ問いだけであり、集計によって最も細かな情報は二度と取り出せなくなります。

Dixonは代わりに、水にたとえました。データマートはボトル入りの水です。浄化して容器に詰め、利用しやすい形に整えています。一方、データレイクは、彼の言葉では「より自然な状態にある大きな水域」です。一つの水源から水が流れ込み、問いを持つ人は誰でも、見に来たり、水に入ったり、試料をくんだりできます。

四年後、Dixonはこの用語を再び取り上げました。さまざまな定義で使われていたためです。彼は、自分の定義を改めて示しました。レイクに入るのは一つの発生元のデータであり、事前に集計しません。一つの組織に複数のレイクがあって

も構いません。複数のシステムのデータを結合したものは、彼のたとえではレイクではなく、水を配した庭園でした。また、2010年の時点で、レイクはデータベース、データマート、データウェアハウスに取って代わるものではなかったとも明言しています。彼が応答した定義のいくつかは、多数のシステムの元データを一か所に集める共有ストレージを指していました。こちらは、より広い意味での使い方です。同僚がどちらを意味しているのか、確認しておく必要があります。

概念を判断に生かす

「データレイクとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 営業、運用、経理の数値をまとめ、同じ会議に部門ごとに異なる答えを持ち寄る状況を解消します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 地理情報パイプラインを50倍に高速化 — 50x 地理データパイプラインの高速化

地理情報パイプラインを50倍に高速化 お問い合わせ →

なぜ重要なのか

どの集計にも、どの問いを重視するかという判断が含まれます。店舗ごとの一日の売上合計からは、各店舗がどれだけ売ったかが分かります。しかし、最も忙しかった時間帯、閲覧されたのに購入されなかった商品、ある顧客が注文前に三度戻ってきたかどうかは分かりません。合計だけを残すと、こうした問いには二度と答えられなくなります。答えを出せるはずの行を、合計を計算する際に捨ててしまったからです。Dixonの主張は、問いはデータより後に生まれるので、元の記録を最も細かな粒度で残し、問いが分かってから集計すべきだというものでした。

だからといって、データウェアハウスが不要になるわけではありません。Dixonは、組織のデータマートやデータウェアハウスにはレイクから取り出した集計を入れ、その都度生じる問いにはレイクを直接使うと説明しました。組織内で合意した数値を置く場所は、引き続きデータウェアハウスです。新しい問いに答えるために、ウェアハウスには残していなかった詳細が必要になったときは、レイクを使います。ウェアハウスのテーブルの定義が変わり、そのテーブルを作り直すときにも使います。

ファイル形式も、考え方と同じくらい重要です。Parquetは、行グループの各列を個別のチャンクとして格納し、ファイル末尾のメタデータに各チャンクの開始位置を記録します。読み取り側は最初にそのメタデータを確認し、必要な列チャンクだけを取得します。そのため、四十列のうち三列を使うクエリーでは、残り

の三十七列を読みません。Parquetは公開された形式で、各ファイルが自分自身のメタデータを持っています。したがって、Parquetを読めるエンジンなら、ノートパソコン上のDuckDBも含め、別途エクスポートせずに同じフォルダーへクエリーを実行できます。

実際の活用例

オンラインショップが、商品の閲覧、カートへの追加、購入を一件ずつ記録するとします。一日の合計にまとめる代わりに、すべてのイベントを残し、一日ごとに一つのフォルダーを作ってParquetファイルに書き出します。DuckDB 2.0では、記録の作成もファイルへの書き出しも、通常のSQLで行えます。

```
CREATE TABLE shop_events AS
SELECT * FROM (VALUES
  (TIMESTAMP '2026-10-26 09:14:03', 'view',     'boots-41', NULL::INTEGER),
  (TIMESTAMP '2026-10-26 09:15:40', 'cart',     'boots-41', NULL),
  (TIMESTAMP '2026-10-26 09:17:12', 'purchase', 'boots-41', 1200),
  (TIMESTAMP '2026-10-27 11:02:55', 'view',     'jacket-m', NULL),
  (TIMESTAMP '2026-10-27 11:03:30', 'view',     'boots-41', NULL),
  (TIMESTAMP '2026-10-27 11:05:01', 'cart',     'jacket-m', NULL)
) t(occurred_at, event, product, amount);

COPY (SELECT *, CAST(occurred_at AS DATE) AS day FROM shop_events)
TO 'events' (FORMAT parquet, PARTITION_BY (day));
```

`PARTITION_BY` に指定できるのは列名だけで、式は指定できません。そのため、日付は `SELECT` で計算しています。DuckDBは値ごとに一つのフォルダーを書き出します。名前はkey=value形式で、この方式をドキュメントではHiveパーティショニングと呼んでいます。

```
events/day=2026-10-26/data_0.parquet
events/day=2026-10-27/data_0.parquet
```

一日分について調べるときは、その列で絞り込みます。

```
SELECT event, count(*) AS n
FROM   read_parquet('events/*/*.parquet', hive_partitioning = true)
WHERE  day = DATE '2026-10-27'
```

```
GROUP  BY event
ORDER  BY n DESC;
-- view 2, cart 1
```

DuckDBは、パーティション列のフィルターをフォルダー名の段階まで押し下げ、一致する可能性のないファイルを読み飛ばします。このクエリーに `EXPLAIN ANALYZE` を使うと、二つのファイルのうち一つを走査したと報告されます。一年分の日別フォルダーがあれば、同じ問いで開くのは365個のうち一つです。DuckDBは既定でフォルダー名に含まれる日付、タイムスタンプ、大きな整数を検出するため、特に指定しなくても `day` 列は `DATE` として返ります。フォルダー名がkey=value形式ならHiveパーティショニングも自動的に有効になります。したがって、これらのクエリーのフラグは、意図を明示しているだけです。

次は、事前には想定していなかった問いです。最も多く閲覧された商品はどれでしょうか。

```
SELECT product, count(*) AS views
FROM   read_parquet('events/*/*.parquet', hive_partitioning = true)
WHERE  event = 'view'
GROUP  BY product
ORDER  BY views DESC, product;
-- boots-41 2, jacket-m 1
```

購入の日別合計をまとめたテーブルでは、この問いには決して答えられません。元のイベントなら、同じフォルダーにもう一つクエリーを実行するだけで答えられます。取り込むときに、すべてのイベントを残してあるからです。

当社のデータウェアハウス設計サービスが目指すのは、すべての数値が一致する一つの場所です。二つの部署が同じ会議に異なる答えを持ち込まないようにします。この例のようなレイクは、そうしたウェアハウスの上流に位置します。ウェアハウスには合意した数値を置き、次の問いに備えて元のイベントを手元に残します。

問題が起きるところ

誰も説明できないファイル。 Parquetファイルが並ぶフォルダーに、どこから来たのか、誰が管理するのか、各列が何を意味するのかという記録がなければ、使えるのは作った本人だけです。2014年に、レイクにはメタデータがないという批判が出た際、Dixonは、レイクという考え方のどこにもそうした制約はないと

答えました。後には、異なる発生元のデータを結合したり、その来歴をたどったりするなら、メタデータが必要になるとも付け加えています。データセットごとに、発生元、管理者、列、読み込み時刻、保持期間をカタログに一件ずつ記録し、その記入を読み込みジョブに組み込んでください。

何千もの小さなファイル。 Parquetファイルはそれぞれ固有のメタデータブロックを持ち、読み取り側はデータを読む前にそれを読む必要があります。一分ごとに小さなファイルを一つ書くストリームでは、30日間で40,000個を超えます。全件走査のたびに、その一つ一つを開いて解析することになります。書き込みをまとめて、数を減らし、より大きなファイルにしてください。また、古いパーティションは定期的に統合します。Apache Icebergのテーブル形式では、こうした統合を `replace` スナップショットとして記録します。複数のファイルを、同じ行を保持する一つのファイルに置き換える操作です。

読み込みの途中が見えてしまう。 十個のファイルを書くジョブが六個を書いたところで止まると、フォルダーには六個だけが残り、そのフォルダーを一覧取得する処理はそれらを読みます。単なるディレクトリーにはコミットがありません。これを解決するのがテーブル形式です。Icebergは、ディレクトリー全体ではなく、個々のデータファイルをメタデータで追跡し、あるメタデータファイルを次のものへアトミックに切り替えて変更をコミットします。読み取り側は常にコミット済みのスナップショットを使います。書き込みが部分的に見えることはなく、読み取り側がロックを取ることもありません。

列名を変えたために履歴が分断される。 古いファイルには古い列名が残り、すべてのクエリーが新旧両方の名前を知っている必要が生じます。Icebergは、名前ではなくフィールドIDでデータファイル内の列を選びます。そのため、すでに書いたファイルを書き直さずに、ネストした構造内の列も含めて、名前の変更、追加、削除、移動ができます。

すべてのシステムを一つのレイクに入れる。 組織内のすべてのデータベースを一か所に注ぎ込み、全体を横断して結合するものは、Dixonが水を配した庭園と呼んだものです。彼は、企業内で複数の種類のストレージを使い、それぞれが扱うデータとクエリーに合わせて選ぶことを勧めました。助言は、用途に最適な道具を使うことでした。トランザクション処理にはトランザクション向けデータベースを、分析には分析向けデータベースを使います。答えるべき問いが実際にある一つの発生元からレイクを始め、そこからウェアハウスにデータを送り、統合され、合意された数値を作ってください。

個人データを永久に残す。 すべてを残すためのストレージは、削除要求や保持期限と衝突します。Icebergバージョン2では、データファイルを書き直さずに削除ファイルを書いて、個々の行を削除できます。つまり、元のバイト列はストレージに残ります。データファイルを書き直し、元のファイルをまだ参照している古いスナップショットが期限切れになってから、ファイルを物理的に削除して元のバイト列を取り除けるようになります。可能な範囲で元のイベントに直接識別子を入れず、保持期限に達したらフォルダー全体を削除できるようにパーティションを分け、削除後のファイル統合とスナップショットの期限切れ処理を定期的に実行してください。

参考資料

- Pentaho、Hadoop、データレイク — James Dixonがこの用語とボトル入りの水との比較を紹介した、2010年10月の投稿です。
- データレイク再考 — 一つのレイクに一つの発生元という定義、水を配した庭園のたとえ、批判への応答を述べたDixonの2014年の続編です。
- ファイル形式 — Parquetにおける行グループ、列チャンク、データの後に書かれるファイルメタデータの配置を説明しています。
- Icebergテーブル仕様 — スナップショット、アトミックなコミット、フィールドIDによるスキーマの変更、削除ファイル、スナップショットの期限切れを定めたApache Icebergの仕様です。
- Hiveパーティショニング — key=value形式のフォルダー、フィルターの押し下げ、パーティション型の検出に関するDuckDBのドキュメントです。

用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「データレイクとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/data-lake/