Cloud Spanner のパフォーマンス分析に関する機能紹介と最新情報

INSIDE Games and Apps

Google Cloud カスタマーエンジニア

Mourad El Azhari
エルアズハリ ムラド

Google Cloud

アジェンダ

- Cloud Spanner の概要と特徴
- パフォーマンス調査とイントロスペクション
- 最新情報

Cloud Spanner

リレーショナル データベースの構造の利点と、
非リレーショナルのスケーラビリティを組み合わせ、
世界中に分散され、強い整合性を持った、
エンタープライズ グレードのフルマネージド データベース。

クラウド ネイティブなリレーショナル データベース
Cloud Spanner の特徴

特徴 1 - 運用構築の手間が無いRDBMS

フルマネージドなデータベースで容易に構成可能、パッチ適用なども全自動
テーブル構造に対して **SQL** でのクエリや、**ACID トランザクション**をサポート

特徴 2 - 高可用性と強整合性を両立

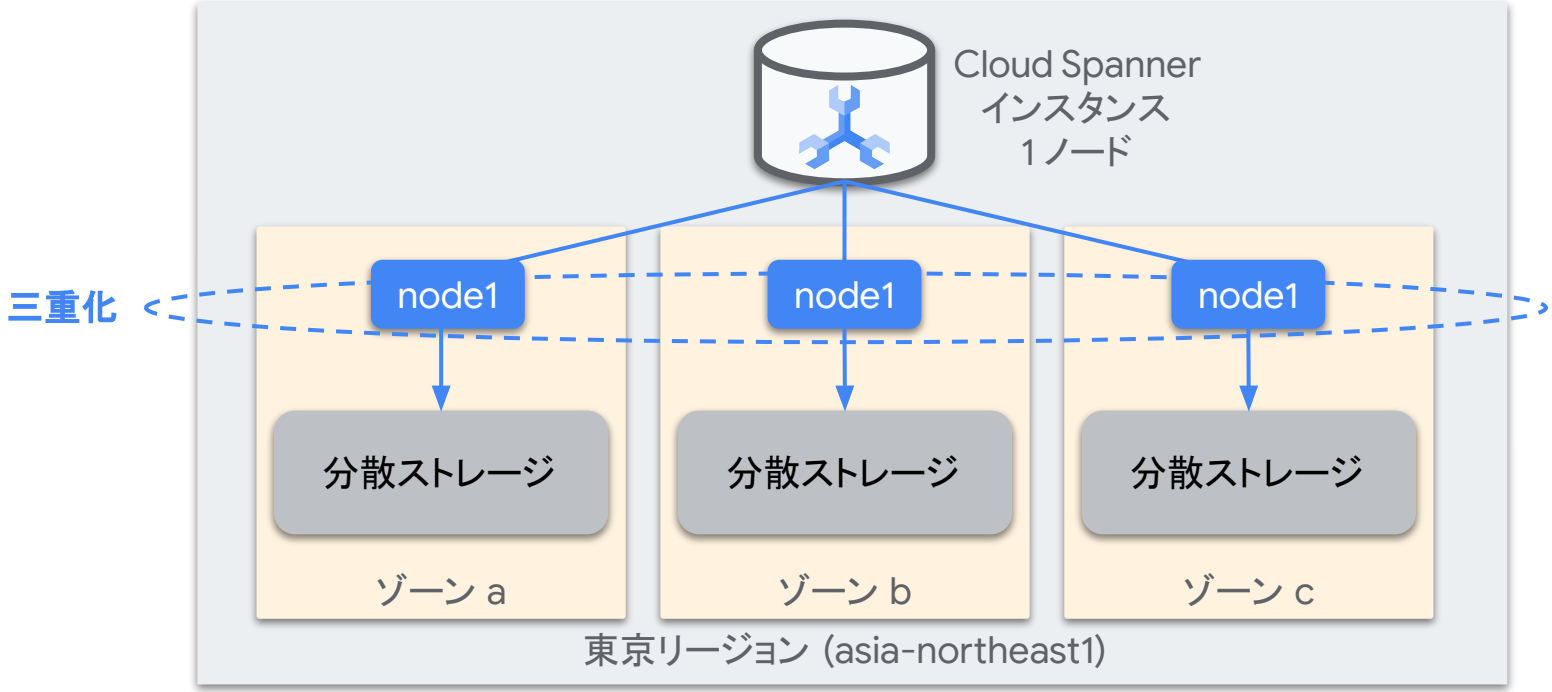
強整合性を保ちつつ複数リージョン間で **DR 構成** 可能で **99.999% の可用性**を提供、
パッチ適用やノード追加によるダウンタイムも無し

特徴 3 - 自在に増減可能なスループット

ノード数と負荷状況に応じてテーブル内のデータは **自動シャーディング** され、
スケールアウトはもちろん、ノード数を減らしてスケールインも簡単

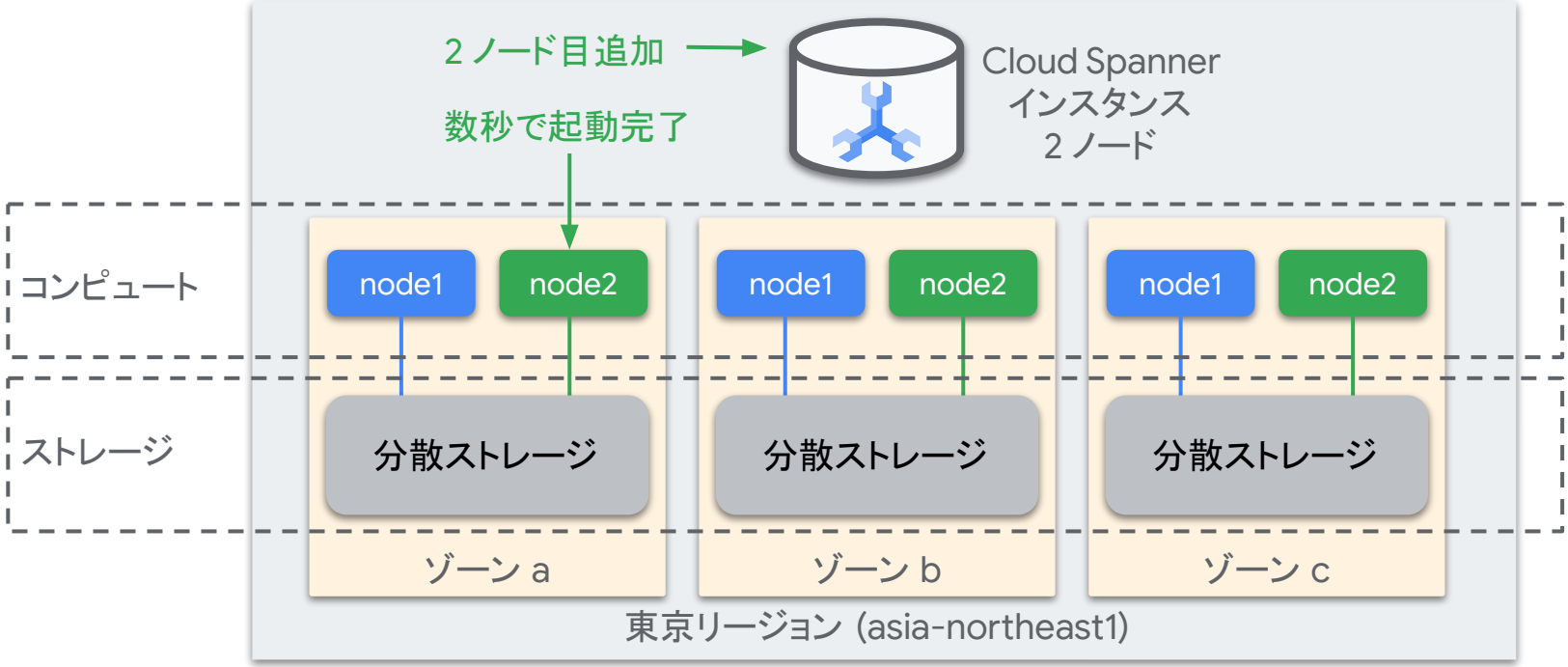
Cloud Spanner を構築すると裏側はどうなっている？

実際のデータは分散ストレージに保存されるため、
仮にノード（処理タスク）で障害が起こっても、データへ影響は無し。



ノード追加は、コンピュート ノードが追加起動するだけ

ノード追加とは、新たなコンテナが起動するだけ。データ自体は分散ストレージ経由で共有されているため、ノードの追加と削除は速やかに完了する。

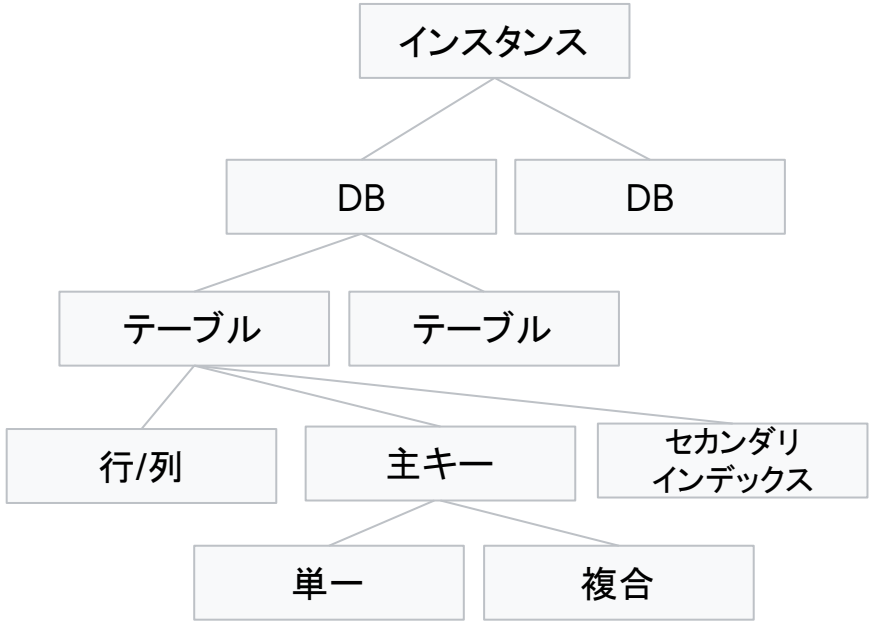


スキーマ設計の重要性

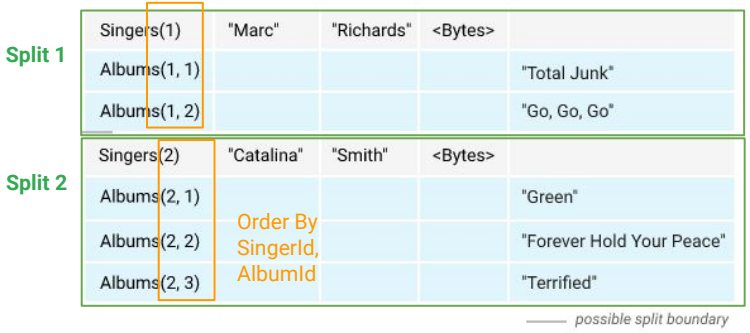
Cloud Spanner Structure

```
CREATE TABLE Singers (
  SingerId INT64 NOT NULL,
  SingerName STRING(MAX),
) PRIMARY KEY(SingerId);

CREATE TABLE Albums (
  SingerId INT64 NOT NULL,
  AlbumId INT64 NOT NULL,
  AlbumName STRING(MAX),
) PRIMARY KEY(SingerId, AlbumId);
```



テーブル インターリーブ



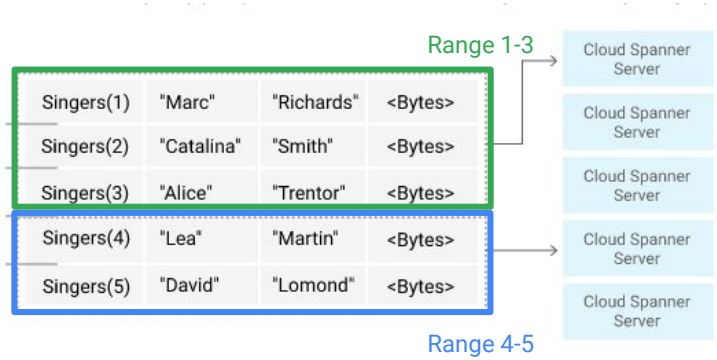
特徴

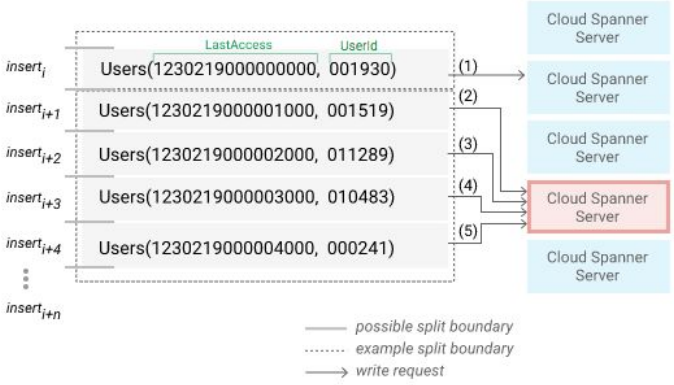
- 親子テーブルを同じスプリットにコロケーションさせる
- 子レコードを作成する前に、親行が存在している必要
- 最初に親テーブルの行で順序付けられ、次に主キーに基づいて子テーブルで順序付けられます
- インターリーブされたテーブルの主キーを介して結合します（パフォーマンス向上）

利用しない場面

- 単一行とその子孫テーブルのデータが増え続けるような場合
- 親とは無関係に子テーブルにアクセスするようなクエリパターンの場合

ホットスポットを回避 - 主キーの設計





問題

- データはキー範囲を通じてサーバー間で分割される
- 単調に増やすと、範囲の最後に新しいレコードが追加され、常に同じサーバーにヒットする
- スプリット(分割) に負荷が高くなると、新しい分割が作成される負荷分散

対応

- 十分に分散する値を主キーに使用する
  - 例）UUID（バージョン 4）
  - どうしても単調増加/ 減少する値を使用したい

例）値をハッシュし、主キーとして使用する。またはハッシュした値の列と元のキー列をあわせて複合主キーとして使用する

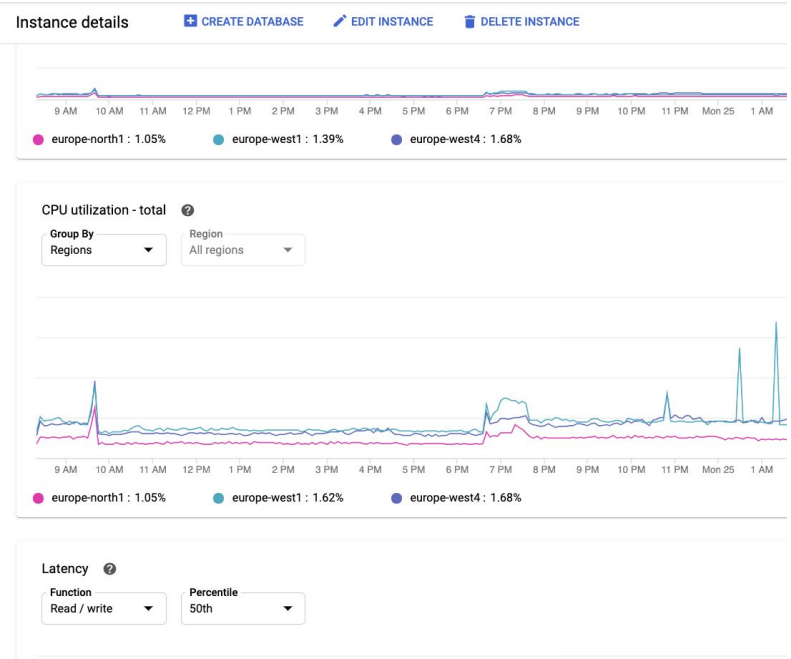
パフォーマンスの分析・調査
&
イントロスペクション

詳細は公式ドキュメントにて

- パフォーマンスの調整
  - SQL のベスト プラクティス
  - クエリ オプティマイザー
  - クエリ実行プラン
  - 効率的なデータ読み込み
  - CPU 使用率の指標
  - レイテンシの指標
- イントロスペクション ツール
  - 概要
  - クエリの統計情報
  - 最も古いアクティブなクエリ
  - 読み取りの統計情報
  - トランザクションの統計情報
  - ロックの統計情報

- イントロスペクション ツールはデータベースの問題を調査するための機能
- クエリ、トランザクション、読み取りなどについてより詳細な情報を取得する目的
- クエリを実行する対象となる一連の組み込みテーブルで構成されてる

Google Cloud のコンソールでモニタリング



特徴

- ノード数
- CPU 使用率 - 移動平均 24 時間
  高い優先度
  合計
- レイテンシ
  - リクエストを受信すると開始され、Cloud Spanner サービスがレスポンスの送信を開始すると終了します
  - レイテンシ指標は50 パーセンタイルと99 パーセンタイルで確認できます。
- 1 秒あたりのオペレーションの回数
  - 読み取りと書き込み
  - 読み取り専用（DML ステートメントも含まれます）
  - 書き込み専用（DML ステートメントは除外されます）
  - Cloud Spanner サーバーでのエラー
- スループット（MB/秒）
  - 読み取りまたは書き込みで処理された非圧縮データ量
  - APIs のリクエストとレスポンスが含まれます
- データベースストレージの合計
- バックアップストレージの合計

CPU 使用率の指標

	ユーザータスク	システムタスク
高い優先度	優先度の高いユーザータスク	優先度の高いシステムタスク
	アプリケーションによって開始され、Cloud Spanner によって高い優先度として処理されるタスク。 ほとんどの読み取りリクエストおよび commit リクエストが該当します。バッチ書き込みの一部の処理も該当しますが、バッチ読み取りは該当しません。	Cloud Spanner によって開始され、高い優先度として処理されるタスク。 インデックスのバックフィルやデータ分割が該当します。
低い優先度	優先度の低いユーザータスク	優先度の低いシステムタスク
	アプリケーションによって開始され、優先度の高いタスクほど早く完了する必要のないタスク。 データフロー ジョブ（Import / Exportを含む）から発行された読み取りと書き込みが含まれます。	Cloud Spanner によって開始され、優先度の高いタスクほど早く完了する必要のないタスク。 データベースの圧縮、スキーマ変更の検証、バックアップの作成が含まれます。

ポイント

- 優先度の高いタスクはすぐに優先度の低いタスクをプリエンプトします
- すべての優先度の低いタスクを停止して、100%のCPUを利用できます
- バックグラウンドでの使用率の急上昇は、問題の兆候ではありません。優先度の低いタスクは、ユーザータスクなどの優先度の高いタスクがあれば、ほぼ即座に後回しにされます。

CPU 使用率の指標

指標	シングルリージョン インスタンスの場合の最大値	マルチリージョン インスタンスの場合のリージョンあたりの最大値
高い優先度の合計	65%	45%
24 時間の平滑化された集計	90%	90%

ポイント

- 高い優先度CPU 使用率と24 時間の平均CPU 使用率を追跡するアラートを作成
- CPU 使用率は、リクエストのレイテンシに影響する可能性があります

対応

- SQL クエリを特定し、それらを最適化
- 推奨されるCPU 使用率の最大値を超える場合は、インスタンスにノードを追加

クエリの統計情報

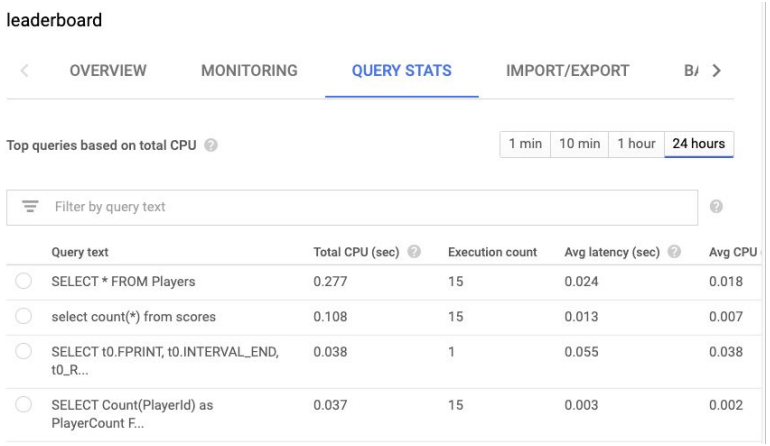
CPU を最も多く使用したクエリおよびすべてのクエリの総計について、多数の統計情報を保持する組み込みテーブルが提供

- クエリごとにグループ化された CPU 使用率

SPANNER_SYS.QUERY_STATS_TOP_[1分/10分/1時間]

- 集計した統計のテーブル

SPANNER_SYS.QUERY_STATS_TOTAL_[1分/10分/1時間]



列名	型	説明
INTERVAL_END	TIMESTAMP	その時間間隔に含まれているクエリが実行された時間間隔の終わり。
TEXT	STRING	SQL クエリテキスト。約 64 KB に切り捨てられます。
TEXT_TRUNCATED	BOOL	クエリテキストが切り捨てられたかどうか。
TEXT_FINGERPRINT	INT64	クエリテキストのハッシュ。
EXECUTION_COUNT	INT64	Cloud Spanner が一定期間内にクエリを検出した回数。
AVG_LATENCY_SECONDS	FLOAT64	データベース内での各クエリ実行の平均時間（秒単位）。この平均からは、オーバーヘッドだけでなく、結果セットのエンコードおよび伝送時間も除外されます。
AVG_ROWS	FLOAT64	クエリから返された平均行数。
AVG_BYTES	FLOAT64	送信エンコードのオーバーヘッドを除いた、クエリから返されたデータの平均バイト数。
AVG_ROWS_SCANNED	FLOAT64	クエリでスキャンされた平均行数（削除された値を除く）。
AVG_CPU_SECONDS	FLOAT64	Cloud Spanner がクエリを実行するためにすべての操作に費やした CPU 時間の平均秒数。

最も古いアクティブなクエリ

最も古いアクティブなクエリとは？

最も古いアクティブなクエリ（長時間実行中のクエリとも呼ばれる）は、データベース内で現在アクティブなクエリのリストであり、クエリの実行時間順に並べられています。これらのクエリを分析して、システムのレイテンシと CPU 使用率が高い場合の原因を特定できます。

テーブル

- 実行中クエリ（DML ステートメントも含む）開始時間別に昇順で表示

OLDEST_ACTIVE_QUERIES

- すべての実行中のクエリの統計情報

ACTIVE_QUERIES_SUMMARY

```
SELECT start_time,
       text_fingerprint,
       text,
       text_truncated,
       session_id
FROM spanner_sys.oldest_active_queries
ORDER BY start_time ASC;
```

start_time	text_fingerprint	text	text_truncated	session_id
2020-07-18T07:52:28.225877Z	-3426560921851907385	SELECT a.SingerId, a.AlbumId, a.TrackId, b.SingerId as b_id, b.AlbumId as b_albumid, b.TrackId as b_trackId FROM Songs as a CROSS JOIN Songs as b;	False	ACjbPvYsucrtcffHrR

```
SELECT active_count,
       oldest_start_time,
       count_older_than_1s,
       count_older_than_10s,
       count_older_than_100s
FROM spanner_sys.active_queries_summary;
```

Query output

active_count	oldest_start_time	count_older_than_1s	count_older_than_10s	count_older_than_100s
22	2020-07-18T07:52:28.225877Z	21	21	1

トランザクションの統計情報

トランザクションに関する統計情報を保存する組み込みテーブル

- トランザクション別にグループ化されたレイテンシ統計

SPANNER_SYS.TXN_STATS_TOP_[1分/10分/1時間]

```
SELECT fprint,
       read_columns,
       write_constructive_columns,
       write_delete_tables,
       avg_total_latency_seconds,
       avg_commit_latency_seconds,
       avg_bytes
FROM spanner_sys.txn_stats_top_minute
WHERE interval_end =
  (SELECT MAX(interval_end)
   FROM spanner_sys.txn_stats_top_minute);
```

Query output

fprint	read_columns	write_constructive_columns	write_delete_tables	avg_total_latency_seconds	avg_commit_latenc
40015598317	[]	["Routes"]	["Users"]	0.006578737	0.006547737
20524969030	["id", "no"]	[]	[]	0.001732442	0.000247442
77848338483	[]	[]	["Cars", "Routes"]	0.033467418	0.000251418

列名	型	説明
INTERVAL_END	TIMESTAMP	発生したトランザクションの実行を含む時間間隔の終了。
FPRINT	INT64	フィンガープリントは、トランザクションに含まれるオペレーションに基づいて計算されるハッシュです。INTERVAL_END と FPRINT は、これらのテーブルの一意のキーとして機能します。
READ_COLUMNS	ARRAY<STRING>	トランザクションが読み取った列のセット。
WRITE_CONSTRUCTIVE_COLUMNS	ARRAY<STRING>	トランザクションによって建設的に書き込まれた（つまり、新しい値に割り当てられた）列のセット。
WRITE_DELETE_TABLES	ARRAY<STRING>	トランザクションによって行が削除または置換されたテーブルのセット。
COMMIT_ATTEMPT_COUNT	INT64	トランザクションに対する commit の合計回数。
COMMIT_FAILED_PRECONDITION_COUNT	INT64	トランザクションの前提条件の失敗（FAILED_PRECONDITION）の合計数。
COMMIT_ABORT_COUNT	INT64	トランザクションの commit が中止された回数。
AVG_PARTICIPANTS	FLOAT64	各 commit 試行の平均参加者数。参加者の詳細については、Cloud Spanner の読み取りと書き込みのライフサイクルをご覧ください。
AVG_TOTAL_LATENCY_SECONDS	FLOAT64	トランザクションの最初のオペレーションから commit の中断までにかかった平均秒数。
AVG_COMMIT_LATENCY_SECONDS	FLOAT64	commit オペレーションの実行にかかった平均秒数。
AVG_BYTES	FLOAT64	トランザクションが書き込んだ平均バイト数。

パフォーマンス分析の最新情報

- **クエリの統計情報 (追加)**

Cloud Spanner のクエリ統計情報のテーブルスキーマに、
「失敗」「キャンセル」「タイムアウト」
したクエリの情報が追加されました。

大量のリソースを消費しているにもかかわらず完了に失敗したり、ユーザーまたはアプリケーションによって中止されたクエリを識別できます。

- **ロックの統計情報**

Cloud Spanner のロックに関する統計情報が確認できるようになりました。ロックの統計を使用して、特定の期間中にデータベースでトランザクションロックの競合の主な原因となった row key と table column(s) を特定できます。

ロックの統計情報

- 行キーによるロックの統計情報
  SPANNER_SYS.LOCK_STATS_TOP [1分/10分/1時間]
- 統計情報の集計
  SPANNER_SYS.LOCK_STATS_TOTAL [1分/10分/1時間]

```
SELECT
      CAST(s.row_range_start_key AS STRING) AS
      row_range_start_key,
      T.total_lock_wait_seconds,
      S.lock_wait_seconds,
      s.lock_wait_seconds/t.total_lock_wait_seco
      nds frac_of_total,
      s.sample_lock_requests
FROM
      spanner_sys.lock_stats_top_10minute s,
      spanner_sys.lock_stats_total_10minute t
WHERE
      t.interval_end = "2020-11-12T22:50:00Z"
      and s.interval_end = t.interval_end;
```

row_range_start_key	total_lock_wait_seconds	lock_wait_seconds	frac_of_total	sample_lock_requests
Singers(32)	780.193	780.193	1	LOCK_MODE: WriterShared COLUMN: Singers.SingerInfo LOCK_MODE: ReaderShared COLUMN: Singers.SingerInfo

Thank you!