大量のデータを使用するリリースのベストプラクティス

Salesforce, Spring '22

'22

@salesforcedocs

最終更新日: 2021/12/29

本書の英語版と翻訳版で相違がある場合は英語版を優先するものとします。

目次

大量のデータを使用するリリースのベストプラクティス

はじめに

対象読者

このドキュメントは、大量のデータを含むSalesforceリリースを操作する、経験豊富なアプリケーションアーキテクトを対象としています。

「大量のデータ」というのは不明確で幅広い意味を持つ言葉です。リリースに数万人のユーザ、数千万件のレコード、あるいは合計で数百ギガバイトものレコードストレージが含まれる場合、そのリリースには「大量のデータ」があります。これらのベストプラクティスは小規模のリリースにも役立ちます。

このドキュメントでSalesforceの実装の詳細について説明している部分を理解するには、https://developer.salesforce.com/page/Multi_Tenant_Architecture を参照してください。

概要

Salesforceを使用して、顧客は少量のデータから大量のデータにまで、アプリケーションを簡単に拡張することができます。この拡張は通常自動的に行われますが、データセットが大きくなるにつれ、特定の処理に必要な時間が増えます。アーキテクトがデータ構造と処理をどのように設計および設定するかによって、処理時間が桁違いに増加または減少することがあります。

アーキテクチャおよび設定が異なることで影響を受けるメインプロセスには、次のようなものがあります。

- 直接またはインテグレーションを使用したいずれかによる多数のレコードの読み込みまたは更新
- レポート、クエリ、またはビューからのデータの抽出

これらのメインプロセスの最適化方法は、次のとおりです。

- スキーマ変更およびデータベース対応アプリケーションでの操作に適応するための業界標準プラクティスに従う
- ビジネスルールおよび共有処理を延期または無視する
- タスクを達成するために最も効率的な処理を選択する

このドキュメントの内容

- 大量のデータを使用するアプリケーションのパフォーマンスを向上するための手法
- あまり明白ではない方法でパフォーマンスに影響を及ぼすSalesforceのメカニズムと実装
- 大量のデータを使用するシステムのパフォーマンスをサポートするために設計されたSalesforceのメカニズム

Salesforce Big Object

Salesforce では Big Object と呼ばれるビッグデータテクノロジを提供しています。Big Object は、Salesforce プラットフォーム上で大量のデータを保存して管理します。他のオブジェクトのデータをアーカイブしたり、外部システムの膨大なデータセットを Big Object に取り込んで顧客の全容を把握したりできます。Big Object では、100 万レコード、1 億レコード、10 億レコードでも一貫したパフォーマンスが提供されます。このスケールが Big Object を強力なものにして、その特徴を明確にします。

このドキュメントでは、Big Object ではなく、標準オブジェクトおよびカスタムオブジェクトに保存される大量のデータの最適化に焦点を当てています。さらに大きなデータセットのパフォーマンスの最適化および持続可能な長期保存を実現するには、Bulk API または Apex 一括処理を使用してデータを Big Object に移動します。

関連トピック:

Salesforce 開発者: Big Objects Implementation Guide (Big Object 実装ガイド)

基礎となる概念

このセクションでは、2 つの重要な概念、マルチテナンシーおよび検索アーキテクチャの概要を示し、Salesforce で次がどのように行われるのかについて説明します。

- 顧客のインスタンスおよび組織にアプリケーションを提供する
- サポートされるカスタマイズの安全、自己完結性、高パフォーマンスを維持する
- アプリケーションデータを追跡および保存する
- 検索を最適化するためにデータにインデックス付けする

このセクションの内容:

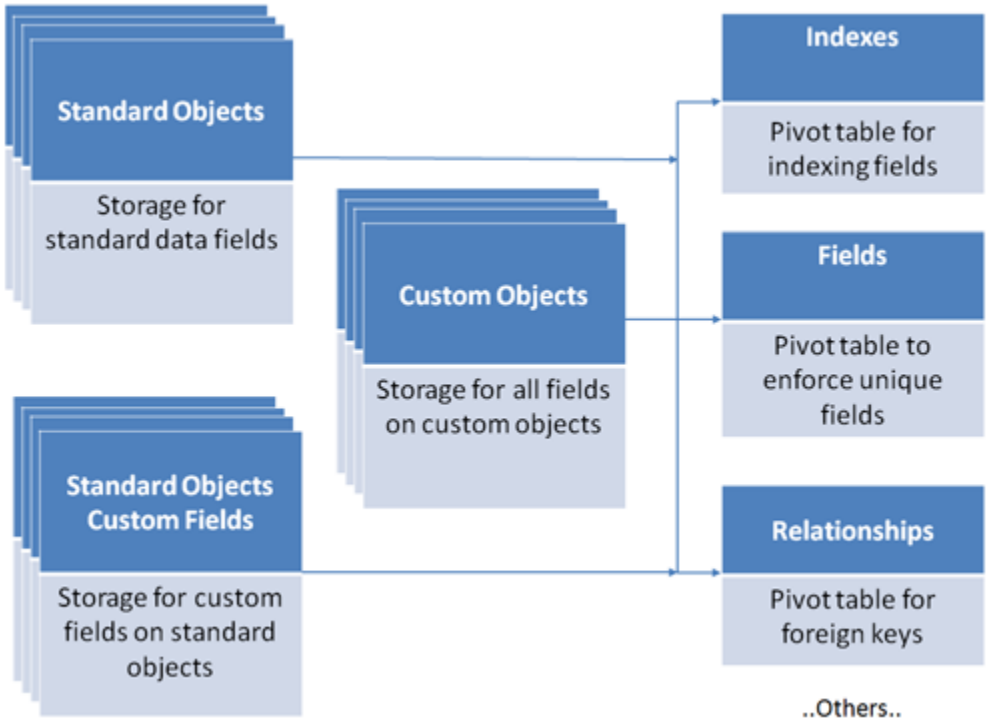
マルチテナンシーおよびメタデータの概要

マルチテナンシーとは、1 つのハードウェア-ソフトウェアスタックから、異なる会社または 1 つの会社内の異なる部門など、複数の組織に 1 つのアプリケーションを提供する手段です。各組織にハードウェアおよびソフトウェアリソースの完全なセットを提供する代わりに、Salesforce では、1 つのインスタンスと各組織のリリースの間にソフトウェアレイヤが挿入されます。このレイヤは組織には表示されません。組織には、組織のデータとスキーマのみが表示され、その間バックグラウンドで効率的に操作を実行するために Salesforce でデータが再整理されます。

マルチテナンシーでは、アーキテクトが Salesforce でサポートされるカスタマイズ (カスタムデータオブジェクトの作成、インターフェースの変更、ビジネスルールの定義など) を行っているときにも、アプリケーションが確実に動作する必要があります。テナント固有のカスタマイズが他のテナントのセキュリティを侵害したり、他のテナントのパフォーマンスに影響を及ぼさないように、Salesforce ではカスタマイズからアプリケーションコンポーネントを生成するランタイムエンジンが使用されます。基盤のアプリケーションのアーキテク

チャと各テナントのアーキテクチャの間に境界を維持することにより、各テナントのデータと操作の整合性が保護されます。

組織がカスタムオブジェクトを作成すると、オブジェクトと項目、リレーション、およびその他のオブジェクト定義特性に関するメタデータがプラットフォームにより追跡されます。Salesforce では、すべての仮想テーブルのアプリケーションデータがいくつかの大きなデータベーステーブルに格納されます。このデータベーステーブルは、テナントごとにパーティションで区切られ、ヒープストレージとして機能します。次にプラットフォームのエンジンで、対応するメタデータを考慮し、実行時に仮想テーブルデータが具体化されます。



膨大で常に変更される、各アプリケーションおよびテナントの実際のデータベース構造の管理を試みる代わりに、プラットフォームストレージモデルでは、メタデータ、データ、およびピボットテーブルを使用して、仮想データベース構造を管理します。したがって、組織のデータとスキーマに基づいて従来のパフォーマンス調整手法を適用する場合は、実際の基盤データ構造で期待した効果が得られない場合があります。

メモ: 顧客は多数のアプリケーション操作の基盤である SQL を最適化することもできません。これは、各テナントが記述するのではなく、システムによって生成されるものだからです。

検索アーキテクチャ

*検索*とは、自由形式のテキストに基づいてレコードを照会する機能です。Salesforce の検索アーキテクチャは、独自のデータストアに基づき、そのテキストの検索用に最適化されています。

Salesforce には、次のような多数のアプリケーション領域の検索機能が用意されています。

- サイドバー
- 高度な検索とグローバル検索

- 検索ボックスと参照項目
- 推奨ソリューションと知識ベース
- Web-to-リードと Web-to-ケース
- 重複リード処理
- Apex および API 向けの Salesforce Object Search Language (SOSL)

データが検索対象となるには、インデックスが付けられている必要があります。インデックスは、検索インデックス化サーバを使用して作成されます。また、このサーバでは、新規作成または変更されたデータのキュー項目が生成および非同期で処理されます。検索可能なオブジェクトのレコードが作成または更新された後、更新されたテキストを検索できるようになるまで約 15 分またはそれ以上かかることがあります。

Salesforce では、最初に適切なレコードのインデックスを検索し、次にアクセス権限、検索制限、およびその他の検索条件に基づいて結果を絞り込むことで、インデックス付き検索が実行されます。このプロセスで結果セットが作成されます。通常これには最も関連性のある結果が含まれます。結果セットが事前に決めたサイズに達した後、残りのレコードは破棄されます。次に結果セットは、ユーザに表示される項目を取得するために、データベースのレコードのクエリに使用されます。

ヒント: 検索には、SOSL でもアクセスできます。これは、API または Apex を使用して起動できます。

大量のデータを使用するシステムのインフラストラクチャ

このセクションでは、次の事項について説明します。

- 大量のデータを使用するシステムのパフォーマンスを直接サポートする Salesforce コンポーネントと機能
- Salesforce でこれらのコンポーネントと機能が使用される状況
- Salesforce インフラストラクチャから得られる利点を最大化する方法

このセクションの内容:

Lightning Platform クエリオプティマイザ

Salesforceのマルチテナントアーキテクチャで基盤となるデータベースが使用される方法では、データベースシステムのオプティマイザが検索クエリを効率的に最適化できません。Lightning Platform クエリオプティマイザを使用すると、Salesforceでのデータアクセスが効率化されるため、データベースのオプティマイザは、効果的なクエリを生成できます。

重要: 可能な場合は、Equalityの会社の値に一致するように、含めない用語を変更しました。顧客の実装に対する影響を回避するために、一部の用語は変更されていません。

Lightning Platform クエリオプティマイザは、レポート、リストビュー、およびSOQLクエリを処理する、自動生成されたクエリを操作します。また、これらの生成されたクエリを利用する他のクエリも操作します。

オプティマイザは、具体的には以下のことを行います。

- クエリの検索条件に基づいて(可能な場合)、効果的なクエリを実行するのに最適なインデックスを判断する
- 適切なインデックスがない場合は、効果的なクエリを実行するのに最適なテーブルを判断する
- 残りのテーブルの順序を調整して、コストを最小限に抑える
- 効率的な結合パスを作成するために必要なカスタム外部キー値テーブルを挿入する
- 共有結合など、残りの結合の実行計画を調整して、データベース入出力(I/O)を最小限に抑える
- 統計情報を更新する

効果的なクエリの作成

大量のデータを操作するときは、効果的なSOQLクエリ、レポート、およびリストビューを作成することが重要です。これらはすべて選択度とインデックスに依存しています。Lightning Platform クエリオプティマイザは、SOQLクエリ、レポート、またはリストビューの検索条件の選択度を判断します。SOQLクエリがシンプルであると、必要な統計情報を簡単に取得して、特定の検索条件がセレクティブであるかどうかを判断できることがあります。セレクティブ項目のインデックスを作成すると、関連する検索条件が指定されたクエリがより効率的に実行できるため、ユーザの生産性が向上します。

実際に検索条件の選択度を測定する場合は、以下の考慮事項に留意してください。

検索条件の選択度の判断

選択度の理解を深めるため、例を使用しましょう。組織内で最大のオブジェクトの1つ、商談オブジェクトに対するSOQLクエリ、レポート、またはリストビューを作成します。使用する検索条件は、たとえば、オブジェクトから必要な行のみを取得する `WHERE` 句です。検索条件は、オプティマイザがインデックスを使用するのに十分なほどセレクティブですか?

SOQLクエリがシンプルであると、統計情報をすばやく取得して、特定の項目の値がセレクティブであるかどうかを理解できます。

SOQL の使用による検索条件の選択度の判断

基本的な単一の WHERE 句条件を使用するクエリを考えます。

```
SELECT Id, Name FROM Opportunity
WHERE Stagename = 'Closed Won'
```

任意のクエリツール、たとえば開発者コンソールのクエリエディタを使用して、以下のクエリを実行し、検索条件の選択度に関する統計情報を取得します。以下に示すのは、Stagename 項目に対するクエリの例です。

```
SELECT Stagename, COUNT(id) FROM Opportunity
GROUP BY ROLLUP (Stagename)
```

結果セットを見ると、オブジェクトのレコード総数を含め、Stagename 選択リスト項目のそれぞれの値のレコードの分布がわかります。これで必要な統計情報が得られたため、Stagename 項目を含む検索条件の選択度を判断できます。

より複雑な検索条件の選択度の判断

前述の例と似たクエリで GROUP BY ROLLUP を使用すると、必要な統計情報を簡単に取得して、各種の条件の選択度を評価できます。

以下に示すのは、検索条件がより複雑なクエリの例です。このクエリでは、日付項目 (CloseDate) が AND 演算子とともに使用されています。

```
SELECT Id, Name FROM Opportunity
WHERE Stagename = 'Closed Won'
AND CloseDate = THIS_WEEK
```

前のクエリから、Stagename 項目の統計情報はすでにわかっています。CloseDate 項目について、同じ統計情報を年ごとに週でグループ化して取得するには、以下のクエリを使用します。

```
SELECT WEEK_IN_YEAR(CloseDate), CALENDAR_YEAR(CloseDate), COUNT(id)
FROM Opportunity
GROUP BY ROLLUP(WEEK_IN_YEAR(CloseDate),CALENDAR_YEAR(CloseDate))
ORDER BY CALENDAR_YEAR(CloseDate), WEEK_IN_YEAR(CloseDate)
```

このクエリでは、すべての年のすべての週における CloseDate 別の商談レコードの分布に関する統計情報が返されます。

検索条件で 2 つ以上の条件が (AND を使用して) 組み合わされており、条件の対象が以下のしきい値未満である場合、クエリオプティマイザは、検索条件全体をセレクティブと判断します。

- 各条件の選択度しきい値の 2 倍
- これらの項目の論理積の選択度しきい値

このトピックの 3 番目の例では、以下を意味します。

- Status = 'Closed Won' がセレクティブ (49,899 < 150,000)
- CloseDate = THIS_WEEK がセレクティブ (~3000 < 150,000)

これらの両方の理由から、検索条件全体がセレクティブです。

検索条件の 1 つがセレクティブでない場合、たとえば Status='Closed Won' が 250,000 レコードに対応している場合でも、2 つの可能性を満たせば、検索条件全体がセレクティブになることができます。

- 各検索条件が 300,000 レコード未満に対応している (各条件の選択度しきい値の 2 倍)
- `Status='Closed Won' AND CloseDate = THIS_WEEK` の論理積が 150,000 レコード未満である。

この例の検索条件は 300,000 レコード未満であるため、条件全体はセレクティブです。

ヒント: `OR` 演算子を使用する場合は、各条件がしきい値に個々に一致する必要があります。

削除されたレコードが選択度に与える影響の理解

選択度の統計情報を収集する場合、削除されたレコードは、Boolean 項目 `IsDeleted` を使用して、含めることも、除外することもできます。この項目は、すべての標準およびカスタムオブジェクトで使用できます。

`ROLLUP` 関数を使用する以前のクエリは、すべての商談レコードのデータを収集します。`IsDeleted` が `true` と `false` のいずれであるかは関係ありません。`Opportunity.StageName` の選択度の統計情報を収集し、削除されたレコードを明示的に除外するには、以下のクエリを試してください。

```
SELECT Stagename, COUNT(id) FROM Opportunity WHERE IsDeleted=false GROUP BY Stagename
```

ヒント: ワークベンチを使用している場合は、[削除済みおよびアーカイブ済みのレコード] オプションリストで *[除外]* を選択してください。

項目にインデックスがあることの確認

クエリ、レポート、またはリストビューを本番環境で使用する前に、指定された検索条件がセレクティブであることを確認してください。ただし、条件をセレクティブにする項目では、インデックスを有効にする必要があります。必要なインデックスがない場合、クエリオプティマイザは、完全スキャンを実行して、目的の行を取得する必要があります。インデックスがあると、クエリの実行速度が高速化され、組織のユーザの生産性が向上します。

「Database Query & Search Optimization Cheat Sheet (データベースのクエリおよび検索最適化のための早見表)」には、`Id`、`Name`、`OwnerId`、`CreatedDate`、`SystemModstamp`、`RecordType` などのデフォルトでインデックスがある標準項目と、すべての主従関係項目および参照項目が示されています。

検索条件にカスタム項目が含まれる場合は、カスタマーサポートに連絡し、検索で使用される項目のカスタムインデックスを作成してください。すべての項目にインデックスを作成できるわけではありません。たとえば、非決定性の数式項目には、インデックスを作成できません。

データベースの統計情報

現代のデータベースでは、格納されているデータの量と種類の統計情報を収集し、この情報を利用してクエリが効率的に実行されます。Salesforce ではソフトウェアアーキテクチャにマルチテナント方法を採用しているため、最善のデータアクセス方法をデータベースが認識できるように、プラットフォームでは独自の統計情報を保持する必要があります。その結果、API を使用して大量のデータが作成、更新、または削除される場合、アプリケーションが効率的にデータにアクセスできるようにするには、データベースで統計情報を収集する必要があります。現在、この統計情報収集プロセスは夜間に実行されています。

スキニーテーブル

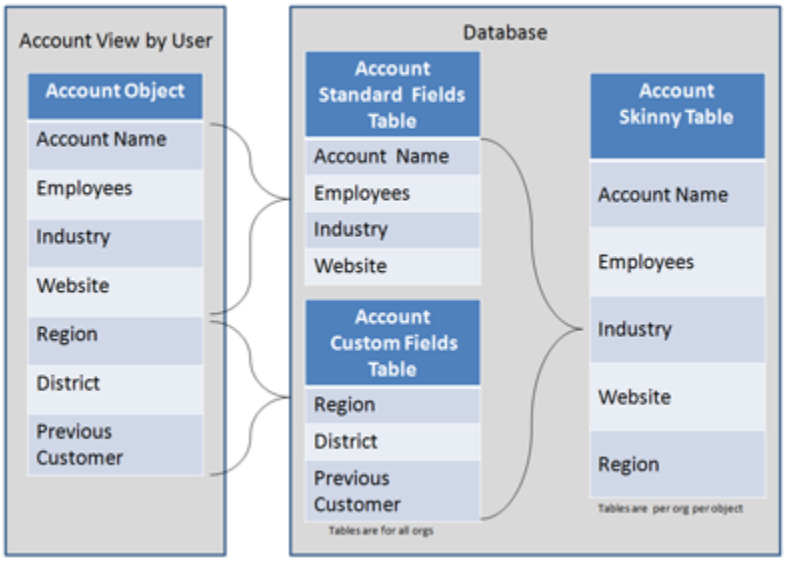
Salesforceでは、頻繁に使用される項目を含めるようにして、さらに結合を避けるためにスキニーテーブルを作成できます。これにより、特定の参照のみの操作のパフォーマンスが改善されます。参照元テーブルが変更されたときに、スキニーテーブルと参照元テーブルの同期が保たれます。

スキニーテーブルを使用する場合は、Salesforceカスタマーサポートにお問い合わせください。スキニーテーブルが有効になっていると、必要に応じて自動的に作成されて使用されます。スキニーテーブルを自分で作成、アクセス、変更することはできません。最適化しているレポート、リストビュー、またはクエリを変更(新規項目の追加など)する場合、Salesforceにスキニーテーブル定義の更新を依頼する必要があります。

スキニーテーブルでパフォーマンスが改善される理由

参照可能な各オブジェクトテーブルでは、データベースレベルで標準項目およびカスタム項目用に他の別個のテーブルが保持されます。この分離は顧客に表示されるため、通常、クエリに両方の種別の項目が含まれる場合は結合が必要になります。スキニーテーブルには、両方の種別の項目が含まれ、論理削除されたレコードは除外されます。

このテーブルは、取引先ビュー、対応するデータベーステーブル、および取引先クエリの処理時間を短縮できるスキニーテーブルを示しています。



スキニーテーブル内の項目のみを参照する参照のみの操作は追加結合が不要なため、パフォーマンスが向上する可能性があります。スキニーテーブルは、レポートなどの参照のみの操作のパフォーマンスを向上させるために大量のレコードを含むテーブルに使用すると最も有益です。

重要: スキニーテーブルは、パフォーマンスの問題を解決する魔法の杖ではありません。個別のテーブルには本番データのコピーが格納されていますが、これらのテーブルを管理するためのオーバーヘッドが発生します。不適切なコンテキストで使用すれば、パフォーマンスの向上ではなく低下を招く可能性があります。

スキニーテーブルは、カスタムオブジェクト、および取引先、取引先責任者、商談、リード、ケースの各オブジェクトに作成できます。レポート、リストビュー、およびSOQLのパフォーマンスを向上できます。

スキニーテーブルには、次の種別の項目を含めることができます。

- チェックボックス
- 日付
- 日付と時刻
- メール
- 数値
- パーセント
- 電話
- 選択リスト (複数選択)
- テキスト
- テキストエリア
- ロングテキストエリア
- URL

スキニーテーブルとスキニーインデックスには暗号化されたデータを含めることもできます。

スキニーテーブルでクエリの処理時間が短縮される例を次に示します。年間レポートまたは年度累計レポートを作成する場合に、コストのかかる反復計算を伴う *01/01/11* ～ *12/31/11* のような日付の範囲を使用する代わりに、スキニーテーブルを使用すると、`Year` 項目を表示して *`Year = '2011'`* で絞り込むことができます。

考慮事項

- スキニーテーブルには、最大で 100 列を含めることができます。
- スキニーテーブルには、別のオブジェクトの項目を含めることができません。
- Full Sandbox の場合: スキニーテーブルは Full Sandbox 組織にコピーされます。

  他の種類の Sandbox では、スキニーテーブルは Sandbox 組織にコピーされません。Full Sandbox 以外の Sandbox で本番用スキニーテーブルを有効にするには、Salesforce カスタマーサポートにお問い合わせください。

インデックス

重要: 可能な場合は、Equality の会社の値に一致するように、含めない用語を変更しました。顧客の実装に対する影響を回避するために、一部の用語は変更されていません。

Salesforce は、クエリの処理速度を短縮するためにカスタムインデックスをサポートしています。カスタムインデックスを作成する場合は、Salesforce カスタマーサポートにお問い合わせください。

メモ: Salesforce カスタマーサポートで本番環境用に作成するカスタムインデックスは、その本番環境から作成するすべての Sandbox にコピーされます。

プラットフォームでは、ほとんどのオブジェクトの次の項目のインデックスが保持されます。

- `RecordTypeId`
- `Division`
- `CreatedDate`

- Systemmodstamp (LastModifiedDate)
- Name
- Email (取引先責任者とリード)
- 外部キーリレーション (参照と主従関係)
- 各オブジェクトの主キーである、一意の Salesforce レコード ID

Salesforce では、カスタム項目のカスタムインデックスもサポートされています。ただし、複数選択リスト、ロングテキストエリア、リッチテキストエリア、非決定性数式項目、および暗号化されたテキスト項目を除きます。

外部 ID を使用すると、その項目にインデックスが作成されます。これらの項目はクエリオプティマイザによって考慮されます。

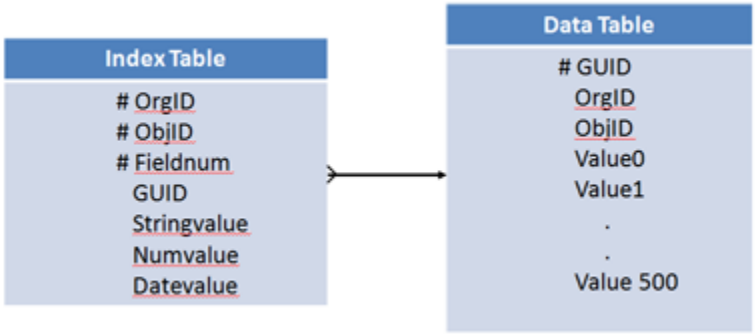
外部 ID は、次の項目にのみ作成できます。

- 自動採番
- メール
- 番号
- テキスト

標準項目など、その他の項目種別にカスタムインデックスを作成するには、Salesforce カスタマーサポートにお問い合わせください。

インデックステーブル

Salesforce のマルチテナントアーキテクチャでは、インデックス付けに適さないカスタム項目の基盤のデータテーブルが作成されます。この制限を克服するために、プラットフォームでは、データのコピーとデータ型に関する情報を含むインデックステーブルが作成されます。



プラットフォームでは、このインデックステーブルに標準のデータベースインデックスが構築されます。インデックステーブルでは、効率的にインデックス付き検索によって返されるレコード数に上限が設定されます。

デフォルトでは、インデックステーブルには null のレコード (空の値を持つレコード) は含まれません。null 行を含むカスタムインデックスを作成するには、Salesforce カスタマーサポートをご利用ください。カスタム項目にカスタムインデックスがすでに存在する場合でも、カスタムインデックスを明示的に有効化および再構築して、空の値の行をインデックス付けする必要があります。

標準インデックス付き項目およびカスタムインデックス付き項目

クエリオプティマイザでは、各インデックスのデータ分配に関する統計情報を含むテーブルが保持されます。このテーブルを使用して、インデックスを使用することでクエリの処理時間を短縮できるかどうかを判断するために、事前クエリを実行します。

たとえば、取引先オブジェクトに `Account_Type` という項目があるとします。この項目は、`Large`、`Medium`、または `Small` の値をとることができ、項目にはカスタムインデックスが付けられています。

たとえば、Salesforce で次のようなクエリが生成されたとします。

```
SELECT *
FROM Account
WHERE Account_Type__c = 'Large'
```

クエリオプティマイザでは、`Account_Type` 項目の値が `Large` であるレコードの数を判断するために、内部統計情報テーブルに対して事前クエリが実行されます。この数がオブジェクトの合計レコード数の 10% または 333,333 レコードを超える場合、クエリではカスタムインデックスが使用されません。

クエリオプティマイザでは、どのインデックスを使用するかは、次のように判断されます。

標準インデックス付き項目
: 条件の一致率が、最初の 100 万件のレコードで 30% 未満、その他のレコードで 15% 未満である場合に使用されます (最大で 100 万件)。

  たとえば、標準インデックスは次の場合に使用されます。

  - 200 万件のレコードを含むテーブルに対してクエリが実行され、検索条件が 450,000 件以下のレコードに一致する。
  - 500 万件のレコードを含むテーブルに対してクエリが実行され、検索条件が 900,000 件以下のレコードに一致する。

カスタムインデックス付き項目
: 検索条件が合計レコード数の 10% 未満 (最大で 333,333 件) に一致する場合に使用されます。

  たとえば、カスタムインデックスは次の場合に使用されます。

  - 500,000 件のレコードを含むテーブルに対してクエリが実行され、検索条件が 50,000 件以下のレコードに一致する。
  - 500 万件のレコードを含むテーブルに対してクエリが実行され、検索条件が 333,333 件以下のレコードに一致する。

インデックス付き項目の条件が満たされない場合は、そのインデックスのみがクエリから除外されます。インデックスが `WHERE` 句にあり、レコードのしきい値を満たす場合は、その他のインデックスが使用されることがあります。

クエリオプティマイザでは、`WHERE` 句に `AND`、`OR`、または `LIKE` が含まれている場合にインデックスを使用するかどうかを判断するために、同様の考慮事項を使用します。

- `AND` の場合、クエリオプティマイザでは、インデックスのいずれかから返されるレコード数が、オブジェクトのレコード数の 20 % または合計レコード数 666,666 を超えない限り、インデックスが使用されます。
- `OR` の場合、クエリオプティマイザでは、すべてのインデックスから返されるレコード数が、オブジェクトのレコード数の 10 % または合計レコード数 333,333 を超えない限り、インデックスが使用されます。

メモ: インデックスが使用されるには、OR 句のすべての項目がインデックス付けされている必要があります。

- LIKE の場合、クエリオプティマイザでは内部統計情報テーブルは使用されません。代わりに、最大で 100,000 件の実際のデータのレコードを抽出し、カスタムインデックスを使用するかどうかが決定されます。

カスタムインデックスは、決定性数式項目に作成できます。時間の経過に伴って値が変化したり、トランザクションにより関連エンティティが更新されたときに値が変更されることがあるため、プラットフォームで非決定性数式をインデックス付けすることはできません。

次に、数式項目を非決定性にする例を示します。

非決定性の数式項目では、次の操作を実行できます。

- 他のエンティティ (参照項目を使用してアクセスできる項目など) を参照する
- 他のエンティティにまたがる他の数式項目を含める
- 動的な日時関数 (TODAY や NOW など) を使用する

次のような数式項目も非決定性であると見なされます。

- 所有者、自動採番、ディビジョン、監査項目 (CreatedDate および CreatedByID 項目は除く)
  - Lightning Platform でインデックス付けできない項目の参照
  - 複数選択リスト
  - マルチ通貨組織の通貨項目
  - ロングテキストエリア項目
  - バイナリ項目 (blob、ファイル、または暗号化されたテキスト)
- 特殊な機能を持つ標準項目
  - 商談: Amount、TotalOpportunityQuantity、ExpectedRevenue、IsClosed、IsWon
  - ケース: ClosedDate、IsClosed
  - 商品: ProductFamily、IsActive、IsArchived
  - ソリューション: Status
  - リード: Status
  - 活動: Subject、TaskStatus、TaskPriority

メモ: インデックスが作成された後に数式が変更されると、インデックスは無効にされます。インデックスを再度有効にするには、Salesforce カスタマーサポートにお問い合わせください。

次の例に示すように、クロスオブジェクト表記を使用して指定されている場合は、クロスオブジェクトインデックスが一般的に使用されます。

```
SELECT Id
FROM Score__c
WHERE CrossObject1__r.CrossObject2__r.IndexedField__c
```

この方法は、他のオブジェクトを参照するためにカスタムインデックス付けできない数式項目の置換に使用できます。参照される項目がインデックス付けされている限り、クロスオブジェクト表記に複数のレベルを使用できます。

2 列のカスタムインデックス

2 列のカスタムインデックスは、Salesforce プラットフォームの特殊機能です。これは、表示するレコードを選択するために 1 つの項目を使用し、それらのレコードを並び替えるために別の項目を使用するようなリストビューやその他の状況で便利です。たとえば、`State` で選択し、`City` で並び替える取引先リストビューでは、最初の列に `State`、2 番目の列に `City` を使用する 2 列インデックスを使用できます。

2 つの項目の組み合わせがクエリ文字列の一般的な検索条件であるとき、レコードの並び替えと表示に 2 列インデックスが一般的に役立ちます。たとえば、擬似コードに見られる次の SOQL の場合、`f1__c,f2__c` の 2 列インデックスは、`f1__c and f2__c` の単一列インデックスよりも効率的です。

```
SELECT Name
FROM Account
WHERE f1__c = 'foo'
     AND f2__c = 'bar'
```

メモ: 2 列インデックスには、単一列インデックスと同じ制限が適用されます。ただし、例外が 1 つあります。2 列インデックスの 2 番目の列に null を使用できますが、単一列インデックスでは null を含めるオプションを Salesforce カスタマーサポートで明示的に有効にしていない限り使用できません。

ディビジョン

ディビジョンとは、クエリやレポートで返されるレコード数を削減するために、大規模なリリースのデータをパーティションで区分する手段です。たとえば、多数の顧客レコードを含むリリースでは、顧客を相互関係がほとんどないと思われる小さなグループに分割するために、`US`、`EMEA`、`APAC` などのディビジョンが作成される場合があります。

Salesforce には、ディビジョンでデータをパーティション区分するための特別なサポートが用意されています。このサポートを有効にするには、Salesforce カスタマーサポートにご連絡ください。

パフォーマンス最適化の手法

このセクションでは、次の事項について説明します。

- Salesforce のパフォーマンスを最適化するための手法
- これらの手法を支える配置、機能、メカニズム、およびオプション
- これらの手法を使用する状況およびニーズに合わせたカスタマイズ

このセクションの内容:

マッシュアップの使用

Salesforce のデータ量を削減する方法の 1 つは、大きなデータセットを別のアプリケーションで維持し、必要に応じてそのアプリケーションを Salesforce で使用できるようにすることです。このような配置は、2 つのアプリケーションの迅速な疎結合インテグレーションを提供するため、Salesforce ではこれをマッシュアップと呼んでいます。マッシュアップでは、Salesforce プレゼンテーションを使用して Salesforce でホストされたデータと外部でホストされたデータが表示されます。

Salesforce では、次のマッシュアップ設計がサポートされます。

外部 Web サイト
Salesforce UI に外部 Web サイトが表示され、情報を渡したり、情報を要求したりします。この設計では、Web サイトを Salesforce UI の一部のように見せることができます。

コールアウト
Apex コードでは、Web サービスを使用して Salesforce がリアルタイムで外部システムと情報交換できるようにします。

リアルタイムの制限があるため、マッシュアップは短時間のやり取りと少量のデータに限られます。

『Apex 開発者ガイド』を参照してください。

マッシュアップを使用した場合の利点

- データが期限切れにならない。
- 2 つのシステムを統合するために、独自の方法を開発する必要がない。

マッシュアップを使用した場合の欠点

- データへのアクセスに時間がかかる。
- 機能性が低下する。たとえば、レポートとワークフローは外部データでは機能しません。

共有適用の延期

場合によっては、*共有適用の延期*という機能の使用が適切なことがあります。ユーザはこの機能を使用して、新しいユーザ、ルール、およびその他のコンテンツが読み込まれるまで共有ルールの処理を延期できます。

組織のシステム管理者は、「共有の適用を延期」権限を使用して、共有適用をサスペンドおよび再開し、グループメンバーの適用と共有ルールの適用の2つのプロセスを管理できます。システム管理者は、共有ルールの評価に長時間かかったり、タイムアウトが発生する可能性がある多数の設定変更を行うときに、これらの適用をサスペンドし、組織のメンテナンス期間に適用を再開できます。この延期により、ユーザは勤務時間中に多数の共有関連の設定変更を迅速に処理でき、再適用プロセスを営業日を挟んだ夜間または週末に実行できます。

SOQL および SOSL の使用

SOQL クエリは SQL `SELECT` ステートメントに相当し、SOSL クエリはテキストベース検索をプログラム的に実行する方法です。

	SOQL	SOSL
実行対象	データベース	検索インデックス
使用するコール	`query()` コール	`search()` コール

次の場合に SOQL を使用します。

- データの保存先のオブジェクトまたは項目がわかっている。
- 次を実行する場合:
  - 1 つのオブジェクト、または相互に関連する複数のオブジェクトからデータを取得する
  - 指定した条件を満たすレコード数を数える
  - クエリの一部として結果を並び替える
  - 数値、日付、またはチェックボックス項目からデータを取得する

次の場合に SOSL を使用します。

- データの保存先のオブジェクトまたは項目が不明であるため、最も効率的な方法で確認したい。
- 次を実行する場合:
  - 相互に関連している、または関連していない複数のオブジェクトおよび項目を効率的に取得する
  - ディビジョン機能を使用して、組織の特定のディビジョンのデータを最も効率的な方法で検索する

SOQL または SOSL を使用するときは、次の点に留意してください。

- SOQL の `WHERE` 検索条件と SOSL の検索クエリの両方とも、検索するテキストを指定できます。特定の検索にどちらの言語も使用できる場合、検索表現に `CONTAINS` 用語を使用するときは、通常 SOSL のほうが SOQL より処理時間が短くなります。
- SOSL は、項目内の複数の単語 (たとえば、スペースで区切られた複数の単語など) をトークン化でき、これを基に検索インデックスを構築します。探している特定の用語が項目内に存在することがわかっている場合は、SOQL よりも SOSL のほうが短時間で検索できることがあります。たとえば、「Paul and John Company」などの値が含まれる項目で「John」を検索する場合は、SOSL の使用を検討します。
- ときには、複数の `WHERE` 検索条件が SOQL で使用されていると、`WHERE` 句の項目にインデックスが付けられている場合でも、インデックスが使用できないことがあります。このような状況では、1つのクエリを複数のクエリに分解して、各クエリに 1 つの `WHERE` 検索条件を使用し、検索結果を結合します。
- 選択リストまたは外部キー項目に対し、null値を使用する `WHERE` 検索条件でクエリを実行すると、インデックスは使用されないため、避けてください。

  たとえば、次の顧客のクエリのパフォーマンスは良くありません。

```
SELECT Contact__c, Max_Score__c, CategoryName__c, Category__Team_Name__c
FROM Interest__c
WHERE Contact__c != null
    AND Contact__c IN :contacts
    AND override__c != 0
    AND (
            (override__c != null AND override__c > 0)
            OR
```

```
            (score__c != null AND score__c > 0)
        )
    AND Category__c != null
    AND (
            (Category_Team_IsActive__c = true OR CategoryName__c IN :selectvalues)
            AND
            (
                Category_Team_Name__c != null
                AND
                Category_Team_Name__c IN :selectTeamValues
            )
        )
```

(`:contacts` のように前にコロンがある項目は、Apex 変数です。『*Apex 開発者ガイド*』の「SOQL および SOSL クエリでの Apex 変数の使用」を参照してください)。条件に `Nulls` が含まれているためにインデックスが使用できず、一部の条件が冗長だったため、実行時間が長くなりました。有効な項目値として `nulls` に依存しないようにデータモデルを設計してください。

クエリは次のように記述し直すことができます。

```
SELECT Contact__c, Max_Score__c, CategoryName__c, Category_Team_Name__c
FROM Interest__c
WHERE Contact__c IN :contacts
    AND (override__c > 0 OR score__c > 0)
    AND Category__c != 'Default'
    AND (
            (Category_Team_IsActive__c = true OR CategoryName__c IN :selectvalues)
            AND
            Category_Team_Name__c IN :selectTeamValues
        )
```

項目 `Category__c` で値 `Default` が `NULL` から置き換えられ、この項目にインデックスが使用できるようになりました。

もう 1 つの例としては、`WHERE` 項目に動的な値が使用されていて、null 値を渡すことができる場合は、レコードがないことを判断するためにクエリを実行せずに、代わりに null 値がないかどうかをチェックし、可能であればクエリは避けてください。

外部キー取引先番号で取引先を取得するクエリは、次のように記述できます (模擬コード)。

```
SELECT Name
   FROM Account
   WHERE Account_ID___c = :acctid;

if (rows found == 0) return "Not Found"
```

`acctid` が `null` の場合、すべてのデータを調べるまで、取引先テーブル全体が 1 行ずつスキャンされます。

コードは次のように書き換えたほうが効率が良くなります。

```
if (acctid != null) {
   SELECT Name
      FROM Account
      WHERE Account_Id___c = :acctid
```

```
}
else {
    return "Not Found"
}
```

- カスタムクエリ検索のユーザインターフェースを設計するときには、次の点が重要です。
  - 検索またはクエリ対象の項目数を最小限に抑えます。多数の項目を使用すると、多数の順列が発生し、調整が難しくなる場合があります。
  - SOQL、SOSL、または 2 つの組み合わせが検索に適切かどうかを判断します。

データの削除

Salesforce のデータ削除のメカニズムは、大量のデータのパフォーマンスに大きな影響を及ぼすことがあります。Salesforce では、ユーザが削除したデータに対してごみ箱メタファーを使用します。Salesforce では、データを削除する代わりに、データに削除済みのフラグを付け、ごみ箱に表示できるようにします。このプロセスを*論理削除*と呼びます。データが論理削除された場合、データはまだ存在しているためデータベースのパフォーマンスに影響するので、削除されたデータはクエリから除外する必要があります。

データはごみ箱に 15 日間、またはごみ箱が特定のサイズになるまで保存されます。データをごみ箱に移動した 15 日後、またはごみ箱がサイズ制限に達したとき、あるいは UI、API、Apex のいずれかを使用してごみ箱を空にすると、データはデータベースから*物理削除*されます。

さらに、Bulk API および Bulk API 2.0 では、ごみ箱をスキップし、レコードをただちに削除の対象にできる*物理削除*オプションがサポートされています。大量のデータを削除するには、Bulk API 2.0 物理削除機能を使用することをお勧めします。

Sandbox 組織のカスタムオブジェクトのレコードをただちに削除する場合は、カスタムオブジェクトの*切り捨て*機能を使用します。この作業のサポートが必要な場合は、Salesforce カスタマーサポートにお問い合わせください。

検索

大量のデータを追加または変更する場合、すべてのユーザがその情報を検索できるようにするために、検索システムでその情報にインデックスを付ける必要があります。このプロセスには、長時間かかることがあります。

「検索アーキテクチャ」 (ページ 3)を参照してください。

ベストプラクティス

このセクションでは、大量のデータを使用したリリースで優れたパフォーマンスを実現するためのベストプラクティスについて説明します。

大規模な Salesforce リリースでパフォーマンスを調整する場合の主なアプローチは、システムが処理する必要のあるレコード数を減らすことに依存します。取得されるレコード数が十分に少なければ、プラットフォームでは、データの取得時間を短縮するためにインデックスや非正規化などの標準データベース構造が使用されます。

レコード数の削減には、次のようなアプローチがあります。

- 絞り込んだクエリ、つまりセレクティブクエリを記述することで範囲を縮小する

  たとえば、取引先オブジェクトにすべての都道府県に均等に配分された取引先が含まれている場合、1都道府県の市区郡別に取引先を集計するレポートは、1 都道府県の 1 市区郡の取引先を集計するレポートよりも、範囲がかなり広くなり、実行にも時間がかかります。

- 有効状態にしておくデータの量を削減する

  たとえば、データ量が増大している場合は、時間の経過と共にパフォーマンスが低下する可能性があります。システムに入ってくるデータと同じだけのデータをアーカイブまたは削除する方針を取ることにより、この影響を回避できます。

以下の表に、主要な目的とその目的を達成するためのベストプラクティスを示します。

このセクションの内容:

レポート

目的	ベストプラクティス
レポートパフォーマンスを次の方法で最大化する • 想定される用途に合わせてデータをパーティション区分する • オブジェクトあたりのレコード数を最小限に抑える	クエリ対象のレコード数を削減します。データ内の値を使用してクエリを分類します。たとえば、すべての都道府県の代わりに、1 つの都道府県のみを照会します。(「ディビジョン」 (ページ 13)を参照してください)。
結合数を削減する	• 次の数を最小限に抑えます。 – レポートで照会されるオブジェクト – レポート生成に使用するリレーション • 実用的な場合はデータを非正規化します。データの「過剰な非正規化」はオーバーヘッドを増やすことになります。レポートには親レコードに保存されている集計データを使用します。この方法は、

目的	ベストプラクティス
	レポートで子レコードを集計するよりも効率的です。
返されるデータ量を削減する	照会される項目数を削減します。項目は必要なレポート、リストビュー、または SOQL クエリにのみ追加します。
クエリ対象のレコード数を削減する	• 未使用のレコードをアーカイブしてデータ量を削減します。未使用のレコードをカスタムオブジェクトテーブルに移動して、レポートオブジェクトのサイズを縮小します。 • 標準またはカスタムインデックス付き項目の使用に重点を置くレポート検索条件を使用します。可能な場合は、レポート検索条件でインデックス項目を使用します。

API からのデータ読み込み

目的	ベストプラクティス
パフォーマンスを向上する	数十万件を超えるレコードがある場合は、Salesforce Bulk API 2.0 を使用します。
最も効率的な操作を使用する	• 使用可能な最速の操作を使用します。最速なのは `insert()` で、次に `update()`、`upsert()` の順に続きます。可能な場合は、`upsert()` を `create()` と `update()` の 2 つの操作に分割します。 • Bulk API 2.0 を使用するときには、読み込む前にデータがクリーンであることを確認します。バッチにエラーがある場合は、そのバッチの単一行処理がトリガされ、この処理はパフォーマンスに大きく影響します。
転送および処理対象データを削減する	更新するときには、変更された項目のみを送信します (デルタのみ読み込み)。
送信時間および中断回数を削減する	カスタムインテグレーションの場合: • レコードごとではなく、読み込みごとに 1 回認証します。

目的	ベストプラクティス
	• GZIP 圧縮と HTTP キープアライブを使用して、長時間にわたる保存操作中に接続が切断されるのを回避します。
不要なオーバーヘッドを避ける	カスタムインテグレーションの場合は、レコードごとではなく、読み込みごとに 1 回認証します。
計算を避ける	初回読み込み時に公開/参照・更新可能セキュリティを使用して、共有適用のオーバーヘッドを回避します。
計算を削減する	• 初回読み込みで可能な場合は、共有ルールを入力する前にロールを入力します。 1. ユーザをロールに読み込みます。 2. レコードデータと所有者を読み込み、ロール階層での適用をトリガします。 3. 公開グループとキューを設定し、それらの計算結果を伝搬します。 4. 共有ルールを 1 つずつ追加し、各ルールの計算が終了してから次のルールを追加します。 • 可能であれば、グループとキューの作成と割り当てをする前に、ユーザとデータを追加します。 1. 新規ユーザと新規レコードデータを読み込みます。 2. 必要に応じて、新規公開グループおよびキューを読み込みます。 3. 共有ルールを 1 つずつ追加し、各ルールの計算が終了してから次のルールを追加します。
計算を延期し、読み込みスループットの時間を短縮する	Apex トリガ、ワークフロールール、および読み込み時の検証を無効にします。読み取り完了後にレコードを処理する Apex 一括処理の使用について調べます。
効率的なバッチサイズとタイムアウトの可能性のバランスをとる	次の場合、SOAP API を使用するときには、ネットワークタイムアウトを回避しながら、できる限り多くのバッチ (最大で 200 個まで) を使用します。 • レコードが大きい。 • 保存操作に、延期できない多数の処理が伴う。
Salesforce を使用するために Lightning Platform Web Service Connector (WSC) を最適化する	Axis などの Java API ではなく WSC を使用します。

目的	ベストプラクティス
親レコードのロック競合を最小限に抑える	子レコードを変更するときには、親別にグループ化します。ロック競合を最小限に抑えるために、同じバッチ内で項目 `ParentId` 別にレコードをグループ化します。
共有適用を延期する	「共有適用を延期」権限を使用して、すべてのデータが読み込まれるまで、共有適用を延期します (「共有適用の延期」 (ページ 14)を参照してください)。
Salesforce へのデータ読み込みを避ける	マッシュアップを使用して、アプリケーションの連動インテグレーションを作成します (「マッシュアップの使用」 (ページ 14)を参照してください)。

API からのデータ抽出

目的	ベストプラクティス
最も効率的な操作を使用する	• `getUpdated()` および `getDeleted()` の SOAP API を使用して、外部システムと Salesforce を 5 分を超える間隔で同期します。これより頻繁に同期する場合は、発信メッセージ機能を使用します。 • 100 万を超える結果が返される可能性があるクエリを使用するときには、Bulk API 2.0 のクエリ機能を使用することを検討してください。この機能のほうがより適している場合があります。

検索

目的	ベストプラクティス
返すレコード数を削減する	可能な場合は、検索を明確に指定し、ワイルドカードの使用を避けます。たとえば、*`「Mi*」`* の代わりに、*`「Michael」`* を使用して検索します。
結合数を削減する	検索速度および精度を高めるために単一オブジェクト検索を使用します。
効率を向上する	検索の設定領域を使用して言語の最適化を有効にし、高度なルックアップ検索とオートコンプリートをオンにして参照項目のパフォーマンスを向上します。

目的	ベストプラクティス
検索パフォーマンスを向上する	場合によっては、ディビジョンを使用してデータをパーティション区分します(「ディビジョン」(ページ 13)を参照してください)。
大量のデータの挿入および更新のインデックス付けにかかる時間を短縮する	「検索アーキテクチャ」(ページ 3)を参照してください。

SOQL と SOSL

GOAL	ベストプラクティス
複数の WHERE 条件を使用した SOQL クエリでインデックスを使用できない場合にインデックス付き検索を許可する	クエリを分解します。WHERE 句で OR で結合した 2 つのインデックス付き項目を使用していて、検索のインデックスしきい値を超えた場合は、クエリを2つのクエリに分割し、結果を結合します。
リアルタイムで計算される数式項目のクエリの実行を避ける	数式項目に対してクエリする必要がある場合は、数式を使用してください。動的な非決定性参照を含む数式項目で条件検索することは避けます。「標準インデックス付き項目およびカスタムインデックス付き項目」(ページ 11)を参照してください。
特定の検索に SOQL または SOSL の最も適切な言語を使用する	「SOQL および SOSL の使用」(ページ 14)を参照してください。
選択リストまたは外部キー項目の WHERE 検索条件で null 値を使用したクエリを実行する	NA などの値を使用して NULLS オプションを置き換えます(「SOQL および SOSL の使用」(ページ 14)を参照してください)。
効率的なカスタムクエリおよび検索ユーザインターフェイスを設計する	適切な場合は SOQL および SOSL を使用し、クエリを絞り込み、クエリまたは検索されるデータ量を最小限に抑えます。(「SOQL および SOSL の使用」(ページ 14)を参照してください)。
効率的な SOQL および SOSL クエリを構築する	クエリでは検索条件と対象を特定する語句を使用してください。 SOQL では、次の点に留意してください。 • 的を絞った検索条件を使用することで、Query Optimizer でスキャンが必要な行数を減らせます。たとえば、インデックス付きの項目を参照したり、さまざまな有効値がある項目を参照する検索条件を使用します。さほど絞り込んでない検索条件の場合、オプティマイザはインデックス付きの列を使用しません。

GOAL	ベストプラクティス
	• FirstName と LastName についての検索条件を使用する場合は、代わりに Name 項目を使用します。たとえば、Select id, Email from Lead where Name='Sam Kennedy' のように記述します。 • 否定の検索条件は使用しないでください。たとえば、status !='failed' や status != NULL です。 • 大量の OR ステートメントのリストの代わりに IN を使用します。たとえば、id in ('001xxx','001xxy', '001xxz') のように記述します。 • クロスオブジェクト参照の数式項目を使用することは避けてください。クロスオブジェクトを検索対象にしないでください。クロスオブジェクトはインデックス化できません。 SOSL の場合: • SOSL の場合、検索条件を絞り込むことにより、無関係な結果の数を減らせます。検索条件が選択的でなく、検索語が2,000レコード以上と一致してしまう場合には、結果が検索の上限超過の影響を受ける可能性があります。 • 検索しないカスタムオブジェクトはインデックス化しないでください。検索可能なレコードの数が増えるため、検索の上限超過が発生する可能性があります。 • 検索しないオブジェクトは検索条件によって除外してください。 • 具体的な検索語を使用してください。 • 検索対象を指定する検索グループを使用してください。検索グループには、NAME、EMAIL、および PHONE項目が含まれます。たとえば、FIND 'Avery Smith' IN NAME FIELDS RETURNING Account(Id,Name), Lead(Id,Name) のように記述します。
大きい SOQL クエリのタイムアウトを回避する	クエリ範囲を縮小したり、セレクティブ検索条件を使用したりして、SOQL クエリを調整します。さらに、Bulk API 2.0 と Bulk API 2.0 クエリを組み合わせて使用することを検討します。前述の提案を試してもまだタイムアウトする場合は、LIMIT 句 (100,000 レコードから開始)

GOAL	ベストプラクティス
	をクエリに追加することを検討します。クエリにApex一括処理を使用している場合は、チェーニングを使用してレコードのセットを取得するか(LIMITを使用)、検索条件ロジックを実行メソッドに移動することを検討します。

データの削除

目的	ベストプラクティス
大量のデータを削除する	100万以上のレコードが削除されるような、大量のデータの削除を行うときには、Bulk APIまたはBulk API 2.0の物理削除オプションを使用します。大量のデータを削除すると、削除プロセスの複雑さにより、処理にかなりの時間がかかる場合があります(「データの削除」(ページ 24)を参照してください)。
データ削除プロセスの効率を向上する	多数の子レコードがあるレコードを削除するときには、子レコードを先に削除します。

一般情報

目的	ベストプラクティス
共有計算を避ける	ユーザが10,000件を超えるレコードを所有しないようにします。
パフォーマンスの向上	複数のオブジェクト間でデータを分散するデータ階層化方法を使用して、別のオブジェクトまたは外部ストアからオンデマンドでデータを取り込みます。
大量のデータを使用する本番Sandboxのフルコピーの作成にかかる時間を短縮する	本番Sandboxのコピーを作成するときには、必要がなければ項目の履歴を除外し、Sandboxのコピーが作成されるまで大量のデータを変更しないようにします。
リリースの効率を向上する	親が10,000件を超える子レコードを持たないように、子レコードを分配します。たとえば、取引先責任者が多数いるけれども、取引先を使用しないリリースの場合は、ダミーの取引先をいくつか設定し、これらの取引先間で取引先責任者を分配します。

大量のデータの事例

このセクションには、次の事項についての説明が含まれます。

- 顧客が経験した大量のデータ関連の問題
- それらの問題を解決するために顧客が使用した、または使用できたはずであるソリューション

同様の問題を認識して解決するには、次の事例を参考にしてください。

このセクションの内容:

データ集計

状況

顧客は、標準レポートを使用して、毎月および毎年の総計値を集計する必要がありました。顧客の毎月および毎年の詳細は、それぞれ 400 万件と 900 万件のレコードを含むカスタムオブジェクトに保存されていました。レポートは、これら 2 つのオブジェクトにまたがる数百万件のレコードを集計していたため、パフォーマンスは最適ではありませんでした。

解決方法

この問題は、毎月および毎年の値を必要なレポートの必要な形式に集計する集計カスタムオブジェクトを作成することで解決されました。レポートは、この集計カスタムオブジェクトから実行され、集計オブジェクトは Apex 一括処理を使用して入力されました。

カスタム検索機能

状況

顧客は、特定の値とワイルドカードを使用して、複数のオブジェクトにまたがって大量のデータ内を検索する必要がありました。そのため、顧客は、ユーザが 1 ～ 20 個の異なる項目を入力し、それらの項目の組み合わせに対して SOQL を使用して検索ができるカスタム Visualforce ページを作成しました。

その結果、検索の最適化が次の理由で困難になりました。

- 多数の値が入力されると、WHERE 句が大きくなり調整が難しい。ワイルドカードが使用されると、クエリの処理時間が長くなる。
- 検索クエリ全体の結果を返すために、複数のオブジェクトにまたがるクエリの実行が必要になる場合がある。このやり方では複数のクエリが発生することになり、検索が拡張される。
- SOQL は、すべてのクエリタイプに使用するのには必ずしも適切ではない。

解決方法

この問題は、次のように解決されました。

- 不可欠な検索項目のみを使用して、検索対象の項目数を削減する。1回の検索で同時に使用できる項目を一般的な使用例に制限することで、Salesforce がインデックスを使用して調整できるようにしました。
- 複数のオブジェクトのデータを単一のカスタムオブジェクトに非正規化し、複数のクエリコールを実行する必要が生じないようにする。
- SOQL または SOSL の使用を動的に判断し、検索される項目数と入力された値の型の両方に基づいて検索を実行する。たとえば、明確に指定した値 (つまり、ワイルドカードを使用しない) では、SOQL を使用してクエリが実行されます。この場合、インデックスを使用できるためパフォーマンスが向上します。

null を使用したインデックス付け

状況

顧客は、項目で null を許可し、その項目を照会できるようにする必要がありました。選択リストおよび外部キー項目の単一列インデックスでは、インデックス列が null の行が除外されるため、null クエリにインデックスを使用できませんでした。

解決方法

この場合、最初から null 値を使用しないことがベストプラクティスです。同じような状況にある場合は、NULL の代わりに、N/A など他の文字列を使用してください。null 値を含むオブジェクトにレコードがすでに存在するなどの理由で他の文字列を使用できない場合は、null の代わりにテキストを表示する数式項目を作成し、この数式項目にインデックス付けします。

たとえば、[Status] 項目にインデックスが付けられていて、null が含まれているとします。

次のような SOQL クエリを発行すると、インデックスは使用されなくなります。

```
SELECT Name
FROM Object
WHERE Status__c = ''
```

代わりに、Status_Value という数式を作成できます。

```
Status_Value__c = IF(ISBLANK(Status__c), "blank", Status__c)
```

この数式項目は、インデックス付けが可能で、null 値を照会するときに使用できます。

```
SELECT Name
FROM Object
WHERE Status_Value__c = 'blank'
```

この概念を拡張し、複数の項目を含めることができます。

```
SELECT Name
FROM Object
WHERE Status_Value__c = '' OR Email = ''
```

大量のデータを含む関連リストの表示

重要: 可能な場合は、Equality の会社の値に一致するように、含めない用語を変更しました。顧客の実装に対する影響を回避するために、一部の用語は変更されていません。

状況

顧客には、数十万件の取引先レコードと 1,500 万件の請求書があり、これらは取引先と主従関係にあるカスタムオブジェクト内に保存されていました。請求書関連リストの表示時間が長いため、各取引先レコードが表示されるのに時間がかかっていました。

解決方法

請求書関連リストの表示の遅れは、データスキューに関連していました。ほとんどの取引先レコードの請求書レコードは数件でしたが、なかには数千件もの請求書を含むレコードがありました。

遅延を軽減するために、顧客はこれらの親の請求書レコード数を減らし、子オブジェクトのデータスキューを最小限に抑えることを試みました。[関連リストの別途読み込みを有効化] 設定を使用することで、関連リストのクエリが完了するのを待つ間に取引先詳細が表示できるようになりました。「ユーザインターフェースの設定」を参照してください。

API パフォーマンス

状況

顧客は、外部の顧客のアプリケーションと Salesforce データを同期するカスタムインテグレーションを設計しました。

インテグレーションプロセスは、次の手順で実行されました。

- 特定のオブジェクトのすべてのデータを対象に Salesforce でクエリを実行する
- このデータを外部システムに読み込む
- どのデータが Salesforce から削除されたのかをインテグレーションプロセスで判断できるように、Salesforce で再度クエリを実行してすべてのデータの ID を取得する

オブジェクトには数百万のレコードが含まれていました。このインテグレーションでは、取得されるレコード数を制限するために、共有階層に含まれる特定の API ユーザも使用しました。クエリは数分で完了しました。

Salesforce では、共有は特定のレコードを特定のユーザに表示されるようにする非常に強力なメカニズムであり、UI 操作に使用すると効果的です。ただし、SOQL クエリで大量のデータの検索条件として使用する場合、共有を検索条件として使用するとデータアクセスの処理がより複雑で困難であるため、パフォーマンスが低下する可能性があります。これは大量のデータを使用している状況で、レコードを除外しようとする場合に特に当てはまります。

解決方法

この問題は、クエリにすべてのデータへのアクセス権を与え、セレクティブ検索条件を使用して適切なレコードを取得することで解決しました。たとえば、システム管理者を API ユーザとして使用すると、すべてのデータへのアクセス権が与えられ、クエリで共有が考慮されなくなります。

また、デルタ抽出を作成し、処理が必要なデータ量を削減することでも解決できたはずです。

共有がパフォーマンスに与える影響については、*『共有モデルの仕組み』*を参照してください。

クエリの並び替えの最適化

状況

顧客は次のようなクエリを使用していました。

```
SELECT Id,Product_Code__c
FROM Customer_Product__c
WHERE CreatedDate = Last_N_Days:3
```

過去 3 日間に作成されたすべてのレコードをクエリで検出しましたが、オブジェクトのデータ量が、合計レコード数の 30% (最大で 100 万まで) という標準インデックスのしきい値を超えてしまいました。クエリのパフォーマンスはよくありませんでした。

解決方法

クエリは次のように書き換えられました。

```
SELECT Id,Product_Code__c
FROM Customer_Product__c
WHERE CreatedDate = Last_N_Days:3
ORDER BY CreatedDate LIMIT 99999
```

このクエリでは、しきい値チェックを実行せず、レコードの検出には `CreatedDate` インデックスを使用しました。このようなクエリでは、過去 3 日間に作成されたレコード数が 99,999 以下であると仮定して、最大で 99,999 件のレコードが過去 3 日以内に作成された順に返されます。

メモ: 通常、`Last_N_Days` の間に追加されたデータを照会するときには、レコード数を 100,000 件未満に制限してインデックス付き項目に `ORDER BY` クエリを指定すると、クエリの実行には `ORDER BY` インデックスが使用されます。

複数結合レポートのパフォーマンス

状況

顧客は、取引先 (314,000)、販売注文 (769,000)、販売詳細 (230 万)、取引先会社形態 (120 万) という 4 つの関連オブジェクトを使用してレポートを作成しました。このレポートでは条件検索があまり行われず、最適化する必要がありました。

解決方法

レポートを最適化するために、顧客は次を行いました。

- 検索条件を追加してクエリの選択度を高め、できるだけ多くの検索条件にインデックス付けができるようにした。
- 可能な場合には、各オブジェクトのデータ量を削減した。
- ごみ箱を空の状態に保った。ごみ箱にデータが入っていると、クエリのパフォーマンスに影響します。
- 4つの関連オブジェクトに複雑な共有ルールが存在しないようにした。複雑な共有ルールは、パフォーマンスに顕著な影響を及ぼす可能性があります。

まとめ

Salesforceプラットフォームは、適切なパフォーマンスを維持しながら、ネイティブアプリケーションとカスタムアプリケーションを大量のデータに迅速に拡張できる堅牢な環境です。

次を行うことにより、これらの機能の利点を最大化できます。

- セレクティブクエリを使用する。レポート、リストビュー、およびSOQLで適切な検索条件を使用します。
- 有効なデータの量を削減する。アーカイブ、マッシュアップ、その他の方法を使用して、Salesforce に保存されるデータの量を削減します。

上記の 2 つの原則とこれらをサポートするベストプラクティスに従うことによって、Salesforce アプリケーションのパフォーマンスに大量のデータが及ぼす影響を軽減することができます。