SOQL および SOSL リファレンス

バージョン 62.0, Winter ‘25

最終更新日: 2024/01/02

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

第 1 章　SOQL および SOSL の概要

Salesforce のカスタム UI を作成している場合、Salesforce Object Query Language (SOQL) と Salesforce Object Search Language (SOSL) の API を使用して、組織の Salesforce データを検索できます。

このガイドでは、SOQL および SOSL をどのような場合に使用するかと、これらの言語の構文、句、制限、およびパフォーマンス上の考慮事項について説明します。このガイドは開発者を対象としており、API を使用したデータの操作に関して知識および経験があることを前提としています。

どちらを使用するかの決定

SOQL クエリは `SELECT` SQL ステートメントに相当し、組織のデータベースを検索します。SOSL は、検索インデックスに対してテキストベースの検索をプログラム的に実行する方法です。

SOQL または SOSL のどちらを使用するかは、検索するオブジェクトまたは項目を認識しているかどうかと、次の考慮事項によって決まります。

データがどのオブジェクトに存在しているかを認識しており、次の操作を行う場合は、SOQL を使用します。

- 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。
- 指定された条件を満たすレコードを数える。
- 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。

データがどのオブジェクトまたは項目に存在しているかを認識しておらず、次の操作を行う場合、SOSL を使用します。

- 項目内に存在することがわかっている、特定用語のデータを取得する。SOSL では項目内の複数の用語をトークン化して、そこから検索インデックスを構築できるため、SOSL 検索でより速く、より多くの関連結果を返すことができます。
- 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取得する。
- 中国語、日本語、韓国語、タイ語のデータを取得する。CJKT 用語の形態的トークン化によって、結果の正確性が確保されます。

メモ: SOSL では Big Object はサポートされません。

パフォーマンス上の考慮事項

クエリと検索の効率を高めるには、次の点を考慮してください。

- 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。
- 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。

詳細は、*「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」*を参照してください。

クエリおよび検索の送信

SOAP API、REST API、Apex などの環境のいずれかを使用して、クエリと検索を作成および実行します。クエリと検索をサポートする環境についての詳細は、「Salesforce Object Query Language (SOQL)」 (ページ 3)および「Salesforce Object Search Language (SOSL)」(ページ 98)を参照してください。

第 2 章 Salesforce Object Query Language (SOQL)

トピック:

Salesforce Object Query Language (SOQL) を使用して、組織の Salesforce データから特定の情報を検索できます。SOQL は、広く使用されている SQL (Structured Query Language) の SELECT ステートメントに似ていますが、Salesforce データ専用に設計されています。

SOQL を使用すると、複数の環境でシンプルながら強力なクエリ文字列を作成できます。

- SOAP API `query()` コールの `queryString` パラメーターの使用。 『*SOAP API 開発者ガイド*』の「query()」を参照してください。
- REST API クエリ要求の `q` パラメーターの使用。 『*REST API 開発者ガイド*』の「クエリ」を参照してください。
- Apex ステートメントの使用。 『*Apex 開発者ガイド*』の「SOQL および SOSL クエリ」を参照してください。
- Visualforce コントローラーと getter メソッドの使用。 『*Visualforce 開発者ガイド*』の「コントローラーメソッド」を参照してください。
- Salesforce CLI の使用。 『*Salesforce CLI コマンドリファレンス*』の「データコマンド」トピックにある「force:data:soql:query」を参照してください。
- Visual Studio Code の拡張機能の使用。 『*Salesforce Extensions for Visual Studio Code (Visual Studio Code 向け Salesforce 拡張機能)*』の「Write SOQL Queries (SOQL クエリの作成)」を参照してください。

Structured Query Language (SQL) の SELECT コマンドと同様に、SOQL では、ソースオブジェクト (Account など)、取得する項目のリスト、ソースオブジェクトから行を選択するための条件を指定できます。

メモ: SOQL では、SQL の SELECT コマンドの高度な機能のすべてはサポートされていません。たとえば、SOQL を使用して、任意の結合操作を実行したり、項目リストにワイルドカードを使用したり、計算式を使用したりはできません。

SOQL では、SELECT ステートメントを絞り込みステートメントと組み合わせて使用し、必要に応じて並び替えできるデータセットを返します。

```
SELECT one or more fields
FROM an object
WHERE filter statements and, optionally, results are ordered
```

たとえば、次の SOQL クエリは、`Name` の値が `Sandy` であるすべての取引先レコードの `Id` および `Name` 項目の値を返します。

```
SELECT Id, Name
FROM Account
WHERE Name = 'Sandy'
```

メモ: Apex では、使用しているステートメントで SOQL ステートメントや SOSL ステートメントを使うには、角括弧で囲む必要があります。前にコロン (:) がある場合は、Apex スクリプト変数と式を使用できます。

構文についての詳細は、「SOQL SELECT の構文」を参照してください。

SOQL を使用するケース

データがどのオブジェクトに存在しているかを認識しており、次の操作を行う場合は、SOQL を使用します。

- 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。
- 指定された条件を満たすレコードを数える。
- 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。

メモ: アーカイブデータと Big Object では、一部の SOQL 機能のみを使用できます。詳細は、「Big Object を使用する SOQL」 (ページ 91)を参照してください。

関連トピック:

Salesforce Developer の制限および割り当てクイックリファレンス: SOQL および SOSL の検索クエリの制限

このドキュメントの表記規則。

この SOQL リファレンスでは、カスタムの表記規則を使用します。

規則	説明
`SELECT Name FROM Account`	Courierフォントは表示どおりに入力する必要がある項目を示します。構文ステートメントでも、Courier フォントは、この表で説明する中括弧、角括弧、省略記号、その他の表記マーカーを除き、表示どおりに入力する必要がある項目を示します。
`SELECT` ***`fieldname`*** `FROM` ***`objectname`***	斜体は、変数またはプレースホルダーを表します。実際の値を入力してください。
`{ }`	中括弧は、曖昧さを排除するために要素をグループ化します。たとえば、`UPDATE {TRACKING|VIEWSTAT}[,...]` 句の場合、中括弧は、`UPDATE` の後に続く選択肢が、パイプで区切られた `TRACKING` または `VIEWSTAT` であることを示します (`UPDATE TRACKING` と `VIEWSTAT` ではない)。
`|`	パイプ文字は、選択肢となる要素を区切ります。たとえば、`UPDATE {TRACKING|VIEWSTAT}[,...]` 句の場合、`|` 文字は `UPDATE` の後に `TRACKING` または `VIEWSTAT` のどちらかを使用できることを示します。
`[ ]`	角括弧は、省略可能な要素を示します。たとえば、`[LIMIT rows]` は、`LIMIT` 句を指定しないか、1つ指定できることを示します。SOQL コマンドの一部として角括弧を入力しないでください。角括弧のネストは、要素が省略可能であることを示し、省略可能な親要素が存在する場合にのみ使用できます。たとえば、`[ORDER BY fieldOrderByList [ASC | DESC] [NULLS {FIRST | LAST}]]` 句の場合、`ASC`、`DESC`、または `NULLS` 句を使用するには `ORDER BY` 句が必要です。

`[...]` および `[,...]`	角括弧で囲まれた省略記号は、その前にある要素をその要素に設定されている上限まで繰り返せることを示します。カンマも含まれる場合は、繰り返す要素をカンマで区切る必要があります。要素が中括弧でグループ化された選択肢のリストである場合、リストの項目を任意の順序で使用できます。たとえば、`UPDATE {TRACKING|VIEWSTAT}[,...]` 句の場合、`[,...]` は、`TRACKING`、`VIEWSTAT`、またはその両方を使用できることを示します。
	`UPDATE TRACKING`
	`UPDATE VIEWSTAT`
	`UPDATE TRACKING, VIEWSTAT`

引用符で囲まれた文字列のエスケープシーケンス

SOQL では、クエリで有効な複数のエスケープシーケンスが定義されているため、クエリに特殊文字を含めることができます。改行、行頭復帰、タブ、引用符などをエスケープできます。SOQL のエスケープ文字はバックスラッシュ (\) 文字です。

SOQL で次のエスケープシーケンスを使用できます。

シーケンス	意味
`\n` または `\N`	改行
`\r` または `\R`	行頭復帰
`\t` または `\T`	タブ
`\b` または `\B`	バックスペース
`\f` または `\F`	フォームフィード
`\"`	1 つの二重引用符文字
`\'`	1 つの一重引用符文字
`\\`	バックスラッシュ
LIKE 式のみ: `\_`	1 つのアンダースコア文字 (_)
LIKE 式のみ: `\%`	1 つのパーセント記号文字 (%)
`\u`***XXXX***	*XXXX* がコードの Unicode 文字 (たとえば、`\u00e9` は é の文字を表します)。

その他のコンテキストでバックスラッシュ文字を使用すると、エラーが発生します。

エスケープ文字の例

`SELECT Id FROM Account WHERE Name LIKE 'Ter%'`
名前が 3 つの文字シーケンス「Ter」で始まるすべての取引先を選択します。

`SELECT Id FROM Account WHERE Name LIKE 'Ter\%'`
名前が 4 つの文字シーケンス「Ter%」に完全に一致するすべての取引先を選択します。

`SELECT Id FROM Account WHERE Name LIKE 'Ter\%%'`
名前が 4 つの文字シーケンス「Ter%」で始まるすべての取引先を選択します。

予約文字

単一引用符 (') およびバックスラッシュ (\) 文字は SOQL クエリで予約されており、適切に解釈されるためには直前にバックスラッシュを付ける必要があります。

SELECT 句で (一重引用符で囲んだ) 予約文字をリテラル文字列として指定する場合、予約文字をエスケープして (前にバックスラッシュ \ 文字を付けて) 予約文字が適切に解釈されるようにする必要があります。予約文字の前にバックスラッシュを付けないと、エラーが発生します。

次の文字が予約されています。

```
' (single quote)
\ (backslash)
```

たとえば、Account の Name 項目で「Bob's BBQ」を照会する場合、次の SELECT ステートメントを使用します。

```
SELECT Id
FROM Account
WHERE Name LIKE 'Bob\'s BBQ'
```

別名表記

SELECT クエリで別名表記を使用できます。

```
SELECT count()
FROM Contact c, c.Account a
WHERE a.name = 'MyriadPubs'
```

別名を設定するには、最初にオブジェクト (この例では取引先責任者) を特定してから別名 (この例では「c」) を指定します。残りの SELECT ステートメントでは、そのオブジェクトまたは項目名の代わりに別名を使用できます。

別名として使用できない SOQL キーワードは、AND、ASC、DESC、EXCLUDES、FIRST、FROM、GROUP、HAVING、IN、INCLUDES、LAST、LIKE、LIMIT、NOT、NULL、NULLS、OR、SELECT、USING、WHERE、WITH です。

SOQL SELECT の構文

SOQL クエリ構文は、必須の SELECT ステートメントとそれに続く 1 つ以上の省略可能な句 (TYPEOF、WHERE、WITH、GROUP BY、ORDER BY など) で構成されます。

SOQL SELECT ステートメントでは、次の構文を使用します。

```
SELECT fieldList [subquery][...]
[TYPEOF typeOfField whenExpression[...] elseExpression END][...]
FROM objectType[,...]
    [USING SCOPE filterScope]
[WHERE conditionExpression]
[WITH [DATA CATEGORY] filteringExpression]
[GROUP BY {fieldGroupByList|ROLLUP (fieldSubtotalGroupByList)|CUBE
(fieldSubtotalGroupByList)}
    [HAVING havingConditionExpression] ]
[ORDER BY fieldOrderByList {ASC|DESC} [NULLS {FIRST|LAST}] ]
[LIMIT numberOfRowsToReturn]
[OFFSET numberOfRowsToSkip]
[{FOR VIEW  | FOR REFERENCE} ]
```

```
[UPDATE {TRACKING|VIEWSTAT} ]
[FOR UPDATE]
```

構文	説明

fieldList subquery	指定した *object* から取得する、1 つ以上の項目のカンマ区切りのリストを指定します。次の例の太字の要素が *fieldlist* 値です。
	• SELECT **Id, Name, BillingCity** FROM Account
	• SELECT **count()** FROM Contact
	• SELECT **Contact.Firstname, Contact.Account.Name** FROM Contact
	• SELECT **FIELDS(STANDARD)** FROM Contact
	有効な項目名を使用し、指定項目ごとに参照レベルの権限を含めます。*fieldList* はクエリ結果内の項目の順序を定義します。
	クエリでリレーションをトラバースする場合、*fieldList* には、サブクエリを含めることができます。次に例を示します。
	`SELECT Account.Name, (SELECT Contact.LastName FROM Account.Contacts)`
	`FROM Account`
	fieldlist は、COUNT() や COUNT(***fieldName***) などの集計関数とすることも、返された結果を翻訳する toLabel() 関数でラップすることもできます。詳細は、「SELECT」を参照してください。

typeOfField	複数のオブジェクト種別を参照できる、***objectType*** の多態的なリレーション項目、または ***objectType*** の親の多態的な項目。たとえば、Task の Who リレーション項目には、Contact または Lead のいずれかを使用できます。***typeOfField*** は、***fieldList*** でも参照されているリレーション項目は参照できません。詳細は、「TYPEOF」を参照してください。
whenExpression	WHEN ***whenObjectType*** THEN ***whenFieldList*** という形式の句。TYPEOF 式内には、1 つ以上の ***whenExpression*** 句を使用できます。詳細は、「TYPEOF」を参照してください。
elseExpression	ELSE ***elseFieldList*** という形式の句。これは、TYPEOF 式内の省略可能な句です。詳細は、「TYPEOF」を参照してください。
objectType	query() の対象オブジェクトの種別を指定します。Account など、有効なオブジェクトを指定します。そのオブジェクトの参照レベルの権限を持っている必要があります。
filterScope	API バージョン 32.0 以降で使用できます。filterScope は、クエリの結果を制限するために指定します。

構文	説明
conditionExpression	WHERE が指定されている場合は、指定したオブジェクト (*objectType*) 内で絞り込みをする行と値が判断されます。指定されていない場合、query() はオブジェクト内でユーザーが参照可能なすべての行を取得します。
filteringExpression	WITH DATA CATEGORY が指定されている場合、query() は指定したデータカテゴリに関連付けられていて、ユーザーが参照可能な、条件に一致するレコードのみを返します。指定されていない場合、query() はユーザーが参照可能な、条件に一致するレコードを返します。WITH DATA CATEGORY 句は次の種別のオブジェクトのみを絞り込みます。 • Question — 質問を照会します。 • KnowledgeArticleVersion — 記事を照会します。 「WITH DATA CATEGORY」を参照してください。 WITH DATA CATEGORY が指定されていない場合、*filteringExpression* を使用して、項目およびオブジェクトレベルの権限を強制したり、Apex コードでユーザーまたはシステムモードのアクセスを指定したりできます。 **SECURITY_ENFORCED** WITH SECURITY_ENFORCED 句は、Apex コード内で照会する項目またはオブジェクトへのユーザーアクセス権限に基づいてレコードを絞り込むために使用します。『*Apex 開発者ガイド*』の「WITH SECURITY_ENFORCED を使用した SOQL クエリの絞り込み」を参照してください。 **USER_MODE または SYSTEM_MODE** Apex データベース操作のユーザーモードまたはシステムモードアクセスを指定します。デフォルトでは、Apex コードはシステムモードで実行されます。つまり、コードを実行しているユーザーよりも非常に上位の権限で実行されます。Apex のセキュリティコンテキストを強化するために、WITH USER_MODE を使用してユーザーモードアクセスを指定できます。システムモードとは異なり、ユーザーモードでは実行ユーザーの項目レベルセキュリティ (FLS) とオブジェクト権限が考慮され、共有ルールが適用されます。『*Apex 開発者ガイド*』の「ユーザーモードでのデータベース操作 (ベータ)」を参照してください。
fieldGroupByList	API バージョン 18.0 以降で使用できます。クエリ結果をグループ化するために使用される 1 つ以上の項目のカンマ区切りのリストを指定します。GROUP BY 句は、データを集計するために、集計関数と共に使用します。この句によって、コード内で個々のレコードを処理せずに、クエリ結果を積み上げ集計できます。「GROUP BY」を参照してください。
fieldSubtotalGroupByList	API バージョン 18.0 以降で使用できます。クエリ結果をグループ化するために使用される項目を最大 3 つまでカンマで区切って指定します。結果にはグループ化されたデータの小計行が含まれます。「GROUP BY ROLLUP」および「GROUP BY CUBE」を参照してください。

構文	説明
havingConditionExpression	API バージョン 18.0 以降で使用できます。クエリに GROUP BY 句が含まれている場合、この条件式では GROUP BY によって返されるレコードが絞り込まれます。「HAVING」を参照してください。

fieldOrderByList	クエリ結果を並び替えるために使用される 1 つ以上の項目のカンマ区切りのリストを指定します。たとえば、取引先責任者を照会し、結果を姓、次に名の順に並び替えることができます。
	`SELECT Id, LastName, FirstName`
	`FROM Contact`
	`ORDER BY LastName, FirstName`

SELECT ステートメントの実装のヒント

- ステートメントの文字数制限 — デフォルトでは、SOQL ステートメントの長さは 100,000 文字を超えることができません。この最大長を超える SOQL ステートメントでは、API は MALFORMED_QUERY 例外コードを返します。結果の行は返されません。

  メモ: 多数の数式項目を含むステートメントなど、長くて複雑な SOQL ステートメントでは、QUERY_TOO_COMPLICATED エラーが発生する場合があります。このエラーは、元の SOQL ステートメントが上限の 100,000 文字未満であっても、Salesforce によって処理されるときにステートメントが内部展開されるために発生します。このエラーを避けるには、SOQL ステートメントの複雑さを軽減します。

  Lightning のページレイアウトに 250 個を超える項目が含まれている場合も QUERY_TOO_COMPLICATED エラーが発生することがあります。Lightning ではレコードページレイアウトの項目を取得するために自動生成された SOQL が使用されるため、お客様が記述した SOQL がなくてもこのエラーが発生する場合があります。

  含まれている通貨項目が多すぎる場合にも、文字数の制限に達することがあります。通貨項目では、format メソッドを使用するために SOQL が必要であるため、各通貨項目の API 参照名の長さがおよそ 2 倍になります。

- ステートメントが多数の結果を返す場合 — SELECT 文がレコードごとに大量のデータを返す場合、SOQL によって自動的に結果の数が減らされます。結果の数は、SOQL クエリのコール方法やクエリの複雑さによっても異なります。リストビューを検索すると、リストの最初の 2,000 件のレコードのみが検索されます。オブジェクトに数式項目、派生項目、CLOB 項目、または BLOB 項目が含まれる場合は、大量のデータが返される場合もあります。結果のすべてのページを取得するには、次のいずれかの方法を使用します。
  - SOQL では、SOQL クエリで OFFSET と LIMIT を使用する場合、返されるレコード数が LIMIT よりも少なくなることがあります。返された結果の数を確認し、必要に応じて OFFSET を調整します。OFFSET を LIMIT で増分しないでください。
  - SOAP API では、`queryMore()` を使用します。
  - REST API では、`/query` および `/queryAll` で返される nextRecordsUrl を使用します。
  - Bulk API 2.0 では、ジョブの結果で返される Sforce-Locator 応答ヘッダーを使用します。

- **結果のローカライズ** — SELECT ステートメントには、ローカライズされた項目をサポートする toLabel()、`convertCurrency()`、および `FORMAT()` 関数を含めることができます。
- **Apex の動的 SOQL** — Apex では、使用しているステートメントで SOQL や SOSL ステートメントを使うには、角括弧で囲む必要があります。前にコロン (:) がある場合は、Apex スクリプト変数と式を使用できます。
- **結果の並び替え** — クエリで `ORDER BY` 句を使用しない限り、結果の順序は保証されません。
- **選択リスト値** — API バージョン 39.0 以降では、値の API 参照名で選択リスト値を照会するため、実際の値とは異なる可能性があります。
- **クエリタイムアウトの回避** — `WHERE` または `WITH` 句を使用して、クエリ操作時間を短縮することを検討してください。REST API では、クエリ処理に関する情報を得るために、`explain` パラメーター (ベータ) を試すことも可能です。詳細は、*『REST API 開発者ガイド』* の「クエリのパフォーマンスに関するフィードバックを取得する」を参照してください。

関連トピック:

Salesforce Developer の制限および割り当てクイックリファレンス: SOQL および SOSL の検索クエリの制限

SELECT

SOQL クエリの構文は、必須の `SELECT` ステートメントで構成されており、そこで照会する項目を指定します。`SELECT` ステートメントの ***`fieldList`*** では、取得する 1 つ以上の項目のカンマ区切りのリストを指定します。

```
SELECT fieldList [subquery][...]
```

`fieldList` で `FIELDS()` キーワードを使用すれば、事前に名前を把握していなくても項目のグループを選択できます。このキーワードによって SELECT ステートメントが簡単になり、複数の API コールも不要となります。このため、少ないコードで組織のデータを調査できます。このキーワードは、API バージョン 51.0 以降で使用できます。

取得すべき項目が多数ある場合は、SOQL SELECT ステートメントの項目リスト (`SELECT Id, Name FROM Account` など) の使用が複雑になる場合があります。オブジェクトの項目がわからない場合は、最初にオブジェクトの説明を取得する必要があります。通常は、最初にコールを使用してオブジェクトの説明を取得し、その説明を解析して項目を確認します。次に、項目を指定した SOQL クエリを作成して、別のコールで送信します。

この手順を簡略化できるように、`FIELDS()` キーワードでは、事前に名前を把握していなくても項目を簡単に選択できます。`FIELDS()` キーワードにより、SOQL ステートメントを準備するためのサーバーへのアクセスが不要になります。これにより、クエリステートメントを簡略化でき、オブジェクトの概要を簡単に把握できるようになります。また、SOQL クエリの文字制限を超えたり、REST コールの URI 長制限を超えたりすることも避けることができます。

項目リストに次の関数を含めることができます。

- `FIELDS(ALL)` — オブジェクトのすべての項目を選択します。
- `FIELDS(CUSTOM)` — オブジェクトのすべてのカスタム項目を選択します。
- `FIELDS(STANDARD)` — オブジェクトのすべての標準項目を選択します。

それぞれのケースにおいて、`FIELDS()` では、項目レベルのセキュリティが考慮されるため、自分にアクセス権がある項目しか表示されません。

使用方法

FIELDS() は、完全な項目リストとして使用できます。次に例を示します。

- SELECT FIELDS(ALL) FROM Account LIMIT 200
- SELECT FIELDS(CUSTOM) FROM Account LIMIT 200
- SELECT FIELDS(STANDARD) FROM Account

FIELDS() は、項目リストの他の項目名と共に使用することもできます。次に例を示します。

- SELECT Name, Id, FIELDS(CUSTOM) FROM Account LIMIT 200
- SELECT someCustomField__c, FIELDS(STANDARD) FROM Account

FIELDS() キーワードは、サブクエリでも使用できます。次に例を示します。

```
SELECT
   Account.Name,
   (SELECT FIELDS(ALL) FROM Account.Contacts LIMIT 200)
FROM Account
```

SELECT ステートメントで項目名の重複が生じた場合は、API でエラーが返されます。たとえば、次のクエリを実行したとします。

```
SELECT Id, FIELDS(ALL) FROM User LIMIT 200
```

この場合、次のエラーが発生します。

```
HTTP/1.1 400 Bad Request
[
   {
      "message" : "duplicate field selected: Id",
      "errorCode" : "INVALID_FIELD"
   }
]
```

FIELDS() のサポート

FIELDS() キーワードは、次のプラットフォーム機能でサポートされます。サポート内容の制限については、「制限クエリおよび無制限クエリ」を参照してください。

- Apex
- SOQL 言語 (query または queryAll 操作を実行できる場所であればどこでも)。
- Bulk API 2.0 のクエリジョブ。
- Salesforce CLI。
- Lightning Platform REST API の /query および /queryAll リソース。
- Lightning Platform SOAP API の query() および queryAll() 操作。

SOAP API インテグレーションを構築するときは、グループオプションが STANDARD、CUSTOM、ALL の FIELDS() を採用する場合に注意してください。組織のオブジェクトモデルに対する変更 (システム管理者による変更、機能の有効化、メジャーリリースによる更新など) は Enterprise WSDL ですぐに反映する必要があります。反映しなかった場合、クエリ操作でクライアントとの契約に一致しない結果が返されるため、エラーが発生します。

制限クエリおよび無制限クエリ

API では、*制限*クエリと*無制限*クエリが区別されます。制限クエリの一連の項目は明確に定義されていますが、無制限クエリの一連の項目は API が事前に判断できません。たとえば、オブジェクトのカスタム項目の数は事前に決定されていないため、`FIELDS(CUSTOM)` と `FIELDS(ALL)` は無制限とみなされます。次の表は、制限クエリと無制限クエリの `FIELDS()` のサポート状況を示しています。

	制限 – FIELDS(STANDARD)	無制限 – FIELDS(ALL) および FIELDS(CUSTOM)
Apex (インラインおよび動的)	サポート	未サポート
Bulk API 2.0	サポート	未サポート
CLI	サポート	結果行が制限されている場合にのみサポートされます。「結果行の制限」を参照してください。
SOAP API および REST API	サポート	結果行が制限されている場合にのみサポートされます。「結果行の制限」を参照してください。

結果行の制限

結果行を制限するには、クエリに次のいずれかの制限を追加します。

- `LIMIT` ***n*** (***n*** は 200 以下です)。次に例を示します。

```
SELECT FIELDS(ALL) FROM Contact LIMIT 200
```

- `WHERE Id IN` ***idList*** (***idList*** は ID のリストで最大 200 個指定できます)。次に例を示します。

```
SELECT FIELDS(ALL) FROM Contact WHERE Id IN ('003R000000ATjnCIAT', '003R000000AZFUIIA5',
 '003R000000DkYoFIAV')
```

- `WHERE Id IS` ***idList*** (***idList*** は Boolean 演算子と結合された ID テストのリストで最大 200 個指定できます)。次に例を示します。

```
SELECT FIELDS(ALL) FROM Contact WHERE Id = '003R000000ATjnCIAT' OR Id =
'003R000000AZFUIIA5' OR Id = '003R000000DkYoFIAV'
```

Salesforce CLI の例

次の例では、Salesforce CLI で `FIELDS()` が使用されています。

```
sf data query --target-org DevHub --query "SELECT FIELDS(STANDARD) FROM Account"
```

この例では、API のバージョン 51.0 以降で機能するように CLI が更新されているものと想定しています。

REST API の例

この要求では、REST API の /query で `FIELDS()` が使用されています。

```
GET https://yourInstance.salesforce.com/services/data/v60.0/query?
q=SELECT+FIELDS(STANDARD)+FROM+Account
```

SOAP API の例

この例では、SOAP API の query() で `FIELDS()` が使用されています。

```
POST https://yourInstance.salesforce.com/services/Soap/c/60.0

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:enterprise.soap.sforce.com">
   <soapenv:Header>
          <urn:SessionHeader>
         <urn:sessionId>sessionId</urn:sessionId>
      </urn:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <urn:query>
         <urn:queryString>SELECT FIELDS(STANDARD) FROM Account</urn:queryString>
      </urn:query>
   </soapenv:Body>
</soapenv:Envelope>
```

Bulk API 2.0 の例

この要求では、一括クエリジョブを作成するときに `FIELDS()` が使用されています。

```
POST https://yourInstance.salesforce.com/services/data/v60.0/jobs/query
{
  "operation": "query",
  "query": "SELECT FIELDS(STANDARD) FROM Account"
}
```

FIELDS() に関する考慮事項

`FIELDS()` キーワードを使用するときには、次の点を考慮してください。

- 取得する項目がすでにわかっている場合は、`FIELDS()` を使用して必要以上に項目を取得するのではなく、目的の項目を明示的に指定すれば、パフォーマンスが向上します。
- `FIELDS()` は、集計が必要な演算子と一緒に使用すると、エラーが発生する場合があります。
  - たとえば、次のクエリは、`FIELDS()` が含まれておらず、正常に機能します。

    ```
    SELECT Id, MIN(NumberOfEmployees) FROM Account GROUP BY Id
    ```

    ただし、クエリに `FIELDS()` を追加した場合、

    ```
    SELECT FIELDS(ALL), MIN(NumberOfEmployees) FROM Account GROUP BY Id LIMIT 200
    ```

「Field must be grouped or aggregated (項目をグループ化または集計する必要があります)」というエラーが発生します。これは、次のクエリに等しくなるためです。

`SELECT IsDeleted, <etc.>,MIN(NumberOfEmployees) FROM Account GROUP BY Id LIMIT 200`。

- `FIELDS()` が大量のデータを返す場合、次のいずれかの方法を使用して、結果のすべてのページを取得します。
  - SOQL では、クエリで OFFSET と LIMIT を使用する場合、返されるレコード数が LIMIT よりも少なくなることがあります。返された結果の数を確認し、必要に応じて OFFSET を調整します。OFFSET を LIMIT のみで増分しないでください。
  - Bulk API 2.0 では、ジョブの結果で返される Sforce-Locator 応答ヘッダーを使用します。
  - REST API では、/query および /queryAll で返される nextRecordsUrl を使用します。
  - SOAP API では、queryMore() を使用します。
- カスタム項目がないオブジェクトで `SELECT FIELDS(CUSTOM)` を使用すると、クエリの結果がエラーになります。

```
HTTP/1.1 400 Bad Request
[
   {
      "message": "'FIELDS(...)' expansion function must result in at least one field
being selected.",
      "errorCode" : "MALFORMED_QUERY"
   }
]
```

項目リスト内に項目が含まれていなければ、エラーは発生しません。たとえば、次のクエリは、オブジェクトにカスタム項目が含まれていない場合でも状況コード 200 を返します。

```
SELECT Id, FIELDS(CUSTOM) FROM User LIMIT 200
```

- `FIELDS()` で返される項目のリストは、組織のメタデータとデータモデルの現在の状態を反映しています。組織のメタデータやデータモデルへのいかなる変更も、クエリのパフォーマンスや結果に影響します。

TYPEOF

`TYPEOF` 句 (省略可能) を SOQL クエリの `SELECT` ステートメントで使用すると、多態的なリレーションが含まれるデータを照会できます。`TYPEOF` 式では、多態的な参照のランタイム型に依存する選択項目のセットを指定します。

`TYPEOF` は、API バージョン 46.0 以降で使用できます。これは、SOQL Polymorphism 機能の開発者プレビューの一部として、以前のバージョンでも利用可能です。

```
[TYPEOF typeOfField
        {WHEN whenObjectType THEN whenFieldList}[...]
        [ELSE elseFieldList]
    END][...]
```

複数の多態的なリレーション項目を照会する必要がある場合。単一の SELECT ステートメントで複数の TYPEOF 式を使用できます。

オブジェクト種別ごとに 1 つの WHEN 句を必要な数だけ指定できます。ELSE 句は省略可能で、現在のレコードの多態的なリレーション項目のオブジェクト種別が WHEN 句で指定されたどのオブジェクト種別にも一致しない場合に使用されます。TYPEOF に固有の構文は次のとおりです。

構文	説明
typeOfField	複数のオブジェクト種別を参照できる、***objectType*** の多態的なリレーション項目、または ***objectType*** の親の多態的な項目。***typeOfField*** は、***fieldList*** でも参照されるリレーション項目を参照できません。
whenObjectType	指定された WHEN 句のオブジェクト種別。SELECT ステートメントの実行時に、***typeOfField*** 式で指定された多態的なリレーション項目に関連付けられた各オブジェクト種別は、WHEN 句のオブジェクト種別と一致するかどうかが確認されます。
whenFieldList	指定された ***whenObjectType*** から取得する、カンマで区切られた 1 つ以上の項目のリスト。これらは参照されるオブジェクト種別の項目または関連オブジェクト項目へのパスで、SELECT ステートメントの主オブジェクト種別の項目ではありません。
elseFieldList	WHEN 句のすべてのオブジェクト種別が ***typeOfField*** で指定された多態的なリレーション項目に関連付けられたオブジェクト種別と一致しない場合に取得する、カンマで区切られた 1 つ以上の項目のリスト。このリストには、Name オブジェクト種別で有効な項目、または Name の関連オブジェクト項目へのパスのみが含まれる可能性があります。

TYPEOF の使用について、次の点に注意してください。

- namePointing 属性が false のリレーションで TYPEOF を使用することはできません。
- relationshipName 属性が false のリレーションで TYPEOF を使用することはできません。
- TYPEOF は、クエリの SELECT 句でのみ使用できます。WHERE 句で Type 修飾子を使用して、多態的なリレーションのオブジェクト種別を絞り込むことができます。詳細は、「多態的なリレーション項目の絞り込み」を参照してください。
- TYPEOF は、COUNT() など、オブジェクトを返さないクエリでは使用できません。
- TYPEOF は、ストリーミング API PushTopic のベースの SOQL クエリでは使用できません。
- TYPEOF は、Bulk API で使用される SOQL では使用できません。
- TYPEOF 式はネストできません。たとえば、TYPEOF 式の WHEN 句内で別の TYPEOF を使用することはできません。
- TYPEOF は、準結合クエリの SELECT 句では使用できません。準結合クエリを含む外側のクエリの SELECT 句では TYPEOF を使用できます。次の例は、TYPEOF が準結合クエリで使用されているため無効です。

```
SELECT Name FROM Account
WHERE CreatedById IN
```

```
    (
    SELECT
        TYPEOF Owner
            WHEN User THEN Id
            WHEN Group THEN CreatedById
        END
    FROM CASE
    )
```

TYPEOF が外側の SELECT 句のみで使用されているため、次の例は有効です。

```
SELECT
    TYPEOF What
        WHEN Account THEN Phone
        ELSE Name
    END
FROM Event
WHERE CreatedById IN
    (
    SELECT CreatedById
    FROM Case
    )
```

- TYPEOF は、SELECT 句で関数を含むクエリでは使用できません。次の例は、TYPEOF に FORMAT 関数が含まれているため無効です。

```
SELECT
 TYPEOF What
  WHEN Account THEN Id, FORMAT(LastModifiedDate) LastModifiedDate__f
  WHEN Oppty THEN Id
 END
FROM Task
```

代わりに、同じクエリを関数を使わずに実行して ID のリストを取得します。

```
SELECT
    TYPEOF What
        WHEN Account THEN Id, LastModifiedDate
        WHEN Opportunity THEN Id
    END
FROM Task
```

次に、結果の ID リストに対して、関数を使用する 2 番目のクエリを実行します。

```
SELECT
    FORMAT(LastModifiedDate) LastModifiedDate__f
FROM Account
WHERE Id in RetrievedIdList
```

- TYPEOF は、GROUP BY、GROUP BY ROLLUP、GROUP BY CUBE、および HAVING を含むクエリでは使用できません。

次の例では、Event の `What` 項目が Account を参照するか Opportunity を参照するかによって異なる特定の項目を選択します。

```
SELECT
  TYPEOF What
    WHEN Account THEN Phone, NumberOfEmployees
    WHEN Opportunity THEN Amount, CloseDate
    ELSE Name, Email
  END
FROM Event
```

多態的なリレーションについての詳細およびその他の `TYPEOF` の例は、「リレーション項目および多態的な項目について」を参照してください。

USING SCOPE

SOQL クエリで `USING SCOPE` 句 (省略可能) を使用すると、指定した範囲内のレコードが返されます。たとえば、返されるレコードを、ユーザーが所有するオブジェクトのみ、またはユーザーのテリトリー内のレコードのみに制限できます。

API バージョン 32.0 以降では、`USING SCOPE` を使用して、指定した *`filterScope`* にクエリの結果を制限できます。

```
[USING SCOPE filterScope]
```

`filterScope` は、多数の列挙値のいずれかを取得できます。オブジェクトでサポートされる範囲のリストを取得するには、describeSObject() (SOAP API の場合) または sObject Describe (REST API の場合) をコールします。オブジェクトの説明の `supportedScopes` セクションでは、`name` に範囲の名前が、`label` にその範囲の説明が記述されます。この例は、取引先オブジェクトの説明を示しています。

```
 "supportedScopes" : [
      {
         "label" : "All accounts",
         "name" : "everything"
      },
      {
         "label" : "My accounts",
         "name" : "mine"
      },
      {
         "label" : "My team's accounts",
         "name" : "team"
      }
   ],
```

次の表は、`filterScope` が取得できる列挙値の例を示しています。

範囲	説明
`delegated`	アクションを実行するように別のユーザーに委任されたレコードに絞り込む検索条件。たとえば、クエリで委任された ToDo レコードのみに絞り込むことができます。

範囲	説明
everything	すべてのレコードの検索条件。
mine	クエリを実行しているユーザーが所有するレコードに絞り込む検索条件。
mine_and_my_groups	クエリを実行しているユーザーおよびユーザーのクエリに割り当てられたレコードに絞り込む検索条件。ユーザーがキューに割り当てられると、そのユーザーは、キュー内のレコードにアクセスできるようになります。この検索条件は、ProcessInstanceWorkItem オブジェクトにのみ適用されます。
my_territory	クエリを実行しているユーザーのテリトリー内のレコードに絞り込む検索条件。このオプションは、組織でテリトリー管理が有効になっている場合に使用できます。
my_team_territory	クエリを実行しているユーザーのチームのテリトリー内のレコードに絞り込む検索条件。このオプションは、組織でテリトリー管理が有効になっている場合に使用できます。
scopingRule	適用可能な範囲設定ルールに基づいて、レコードを絞り込みます。このオプションは、管理者がクエリ対象のオブジェクトに対して範囲設定ルールを少なくとも 1 つ有効にしている場合に使用できます。
team	取引先チームなど、チームに割り当てられたレコードに絞り込む検索条件。

WHERE

SOQL クエリの WHERE 句の条件式には、1 つ以上の項目式が含まれます。論理演算子を使用することで、複数の項目式を条件式で指定できます。

構文

```
WHERE conditionExpression
```

conditionExpression

conditionExpression では次の構文を使用します。

```
fieldExpression [logicalOperator fieldExpression2][...]
```

メモ: WHERE 句の文字列は 4,000 文字を超えることができません。

次の例では、SOQL の SELECT ステートメントの条件式が太字で表されています。

- `SELECT Name FROM Account WHERE Name LIKE 'A%'`
- `SELECT Id FROM Contact WHERE Name LIKE 'A%' AND MailingState='California'`

date、dateTime 値、または日付リテラルを使用できます。date 項目と dateTime 項目の形式は異なります。

- `SELECT Name FROM Account WHERE CreatedDate > 2011-04-26T10:00:00-08:00`

- `SELECT Amount FROM Opportunity` **`WHERE CALENDAR_YEAR(CreatedDate) = 2011`**

`CALENDAR_YEAR()` などの日付関数についての詳細は、「日付関数」を参照してください。

boolean 値 `TRUE` および `FALSE` を SOQL クエリで使用できます。boolean 項目を絞り込むには、次の構文を使用します。

```
WHERE BooleanField = TRUE

WHERE BooleanField = FALSE
```

fieldExpression

SOQL クエリに含まれる WHERE 句の項目式の構文は、項目名、比較演算子、および値で構成されます。クエリは、これらのコンポーネントを使用して、項目名の値と、検索対象のレコードとを比較します。

```
fieldName comparisonOperator value
```

構文	説明
fieldName	指定したオブジェクト内の項目の名前。名前の前後に一重または二重引用符を使用すると、エラーが発生します。項目に対する参照レベル以上の権限が必要です。ロングテキストエリア項目、暗号化されたデータ項目、または Base64 で符号化された項目以外の項目を指定できます。名前は、*fieldList* に含まれている項目である必要はありません。
comparisonOperator	値を比較する演算子。=、<=、IN、LIKE などがあります。演算子は、ほとんどの項目では大文字と小文字が区別されません。ただし、大文字と小文字が区別される項目では、大文字と小文字が区別されます。
value	*fieldName* の値と比較するために使用される値。指定した項目の型と一致するデータ型の値を指定します。値は、他の項目名や計算値ではなく、有効な値にする必要があります。引用符が必要な場合は、単一引用符を使用します。二重引用符を使用するとエラーになります。日付と数値には、引用符は不要です。

fieldExpression が評価される順序を定義するには、括弧を使用します。演算子をネストするときは、括弧を指定する必要があります。ただし、同じ種別の複数の演算子はネストする必要がありません。次の例の式は、***fieldExpression1*** が `true` で、***fieldExpression2*** または ***fieldExpression3*** のいずれかが `true` の場合、`true` です。

```
fieldExpression1 AND (fieldExpression2 OR fieldExpression3)
```

ただし、次の式は、***fieldExpression3*** が `true` であるか、***fieldExpression1*** と ***fieldExpression2*** の両方が `true` の場合、`true` です。

```
(fieldExpression1 AND fieldExpression2) OR fieldExpression3
```

比較演算子

SOQL クエリでは、SELECT ステートメントで使用する WHERE 句の項目式に =、<、>、IN、LIKE などの比較演算子を含めることができます。比較演算子では、準結合と反結合を使用して複雑なクエリを作成することもできます。

次の表に、***fieldExpression*** 構文で使用される ***comparisonOperator*** の値を示します。

メモ: 文字列の比較の場合、大文字と小文字が区別される一意の項目は大文字と小文字が区別され、他のすべての項目は大文字と小文字が区別されません。

演算子	名前	説明
=	Equals	*fieldName* の値が式の *value* に一致する場合、式は true です。
!=	Not equals	*fieldName* の値が指定した *value* に一致しない場合、式は true です。
<	Less than	*fieldName* の値が指定した *value* より小さい場合、式は true です。
<=	Less or equal	*fieldName* の値が指定した *value* 以下の式は true です。
>	Greater than	*fieldName* の値が指定した *value* より大きい場合、式は true です。
>=	Greater or equal	*fieldName* の値が指定した *value* 以上の場合、式は true です。

LIKE	Like	*fieldName* の値が指定した *value* のテキスト文字列の文字に一致する場合、式は true です。指定した value のテキスト文字列は、一重引用符で囲む必要があります。
		LIKE 演算子は、文字列項目でのみサポートされます。この演算子は、部分的なテキスト文字列を照合するメカニズムを提供し、次の使用がサポートされます。
		• % および _ ワイルドカード。
		– % ワイルドカードは、0 個以上の文字に一致します。
		– _ ワイルドカードは、1 文字のみに一致します。
		• 特殊文字 % または _ のエスケープ。
		メモ: 特殊文字をエスケープする場合を除き、検索ではバックスラッシュ (\) 文字を使用しないでください。「引用符で囲まれた文字列のエスケープシーケンス」を参照してください。
		次のクエリ例は Appleton、Apple、Appl と一致しますが、Bappl とは一致しません。
		`SELECT AccountId, FirstName, lastname`
		`FROM Contact`
		`WHERE lastname LIKE 'appl%'`

演算子	名前	説明

IN	IN	値が WHERE 句の値のいずれかに等しい場合、式は true です。IN の文字列値は括弧の中に入れて、一重引用符で囲む必要があります。
		IN を使用して、同じオブジェクトの別の項目に、指定された値のセットがある項目の値を照会できます。次に例を示します。
		`SELECT Name FROM Account`
		`WHERE BillingState IN ('California', 'New York')`
		IN と NOT IN は、ID (主キー) 項目または参照 (外部キー) 項目を照会するときに、準結合および反結合でも使用できます。

NOT IN	NOT IN	値が WHERE 句の値と等しくない場合、式は true です。NOT IN の文字列値は括弧の中に入れて、一重引用符で囲む必要があります。次に例を示します。
		`SELECT Name FROM Account`
		`WHERE BillingState NOT IN ('California', 'New York')`
		メモ: 論理演算子の NOT はこの比較演算子とは無関係です。

INCLUDES EXCLUDES		複数選択リストにのみ適用されます。「複数選択リストのクエリ」を参照してください。

IN を使用した準結合と NOT IN を使用した反結合

準結合は、返されるレコードを制限する、IN 句の別のオブジェクトのサブクエリです。反結合は、返されるレコードを制限する、NOT IN 句の別のオブジェクトのサブクエリです。IN または NOT IN 句内の値リストをサブクエリで置き換えることにより、より複雑なクエリを作成できます。サブクエリでは、ID (主キー) 項目または参照 (外部キー) 項目で絞り込むことができます。

準結合と反結合を使用する例としては、次のようなものがあります。

- 特定のレコード種別の商談がある取引先のすべての取引先責任者を取得する。
- 有効な契約がある取引先のすべての進行中の商談を取得する。
- 商談の意思決定者である取引先責任者のすべてのオープンケースを取得する。
- 進行中の商談がないすべての取引先を取得する。

ID 項目で絞り込む場合は、Account と Contact など、親-子の準結合または反結合を作成できます。参照項目で絞り込む場合は、Contact と Opportunity などの子-子の準結合か反結合、または Opportunity と Account などの子-親の準結合か反結合を作成できます。

ID 項目の準結合

WHERE 句には準結合を含めることができます。たとえば、次のクエリは、関連付けられている商談が不成立だった場合に取引先 ID を返します。

```
SELECT Id, Name
FROM Account
```

```
WHERE Id IN
  ( SELECT AccountId
    FROM Opportunity
    WHERE StageName = 'Closed Lost'
  )
```

この例は、Account と Opportunity の親-子の準結合です。IN 句の左のオペランド Id が ID 項目です。サブクエリは、比較対象の項目と同じ種別の項目を 1 つ返します。準結合クエリの不要な処理を防ぐための制限事項については、考慮事項を参照してください。

参照項目の準結合

次のクエリは、Twin Falls のすべての取引先責任者の ToDo ID を返します。

```
SELECT Id
FROM Task
WHERE WhoId IN
  (
    SELECT Id
    FROM Contact
    WHERE MailingCity = 'Twin Falls'
  )
```

IN 句の左のオペランド WhoId が参照項目です。このクエリの興味深い側面は、WhoId が、取引先責任者またはリードを参照できるため多態的な参照項目であるということです。サブクエリによって、結果は取引先責任者に制限されます。

ID 項目の反結合

次のクエリは、進行中の商談がないすべての取引先の取引先 ID を返します。

```
SELECT Id
FROM Account
WHERE Id NOT IN
  (
    SELECT AccountId
    FROM Opportunity
    WHERE IsClosed = false
  )
```

参照項目の反結合

次のクエリは、供給元が Web 以外のすべての取引先責任者の商談 ID を返します。

```
SELECT Id
FROM Opportunity
WHERE AccountId NOT IN
  (
    SELECT AccountId
    FROM Contact
    WHERE LeadSource = 'Web'
  )
```

この例は、Opportunity と Contact の子-子の反結合です。

複数の準結合または反結合

クエリでは、準結合句または反結合句を組み合わせることができます。たとえば、次のクエリは、取引先に関連付けられている取引先責任者の姓が「Apple」のような姓の場合、進行中の商談がある取引先 ID を返します。

```
SELECT Id, Name
FROM Account
WHERE Id IN
  (
    SELECT AccountId
    FROM Contact
    WHERE LastName LIKE 'apple%'
  )
  AND Id IN
  (
    SELECT AccountId
    FROM Opportunity
    WHERE isClosed = false
  )
```

メモ: 1 つの準結合クエリまたは反結合クエリでは、最大で 2 つのサブクエリを使用できます。また、複数の準結合および反結合クエリは、1 クエリあたりのサブクエリに対する既存の制限の対象となります。

リレーションクエリを評価する準結合または反結合

SELECT 句でリレーションクエリを評価する準結合または反結合を作成できます。たとえば次のクエリは、商談の品目の合計値が $10,000 を超える場合、商談 ID および関連品目を返します。

```
SELECT Id, (SELECT Id from OpportunityLineItems)
FROM Opportunity
WHERE Id IN
  (
    SELECT OpportunityId
    FROM OpportunityLineItem
    WHERE totalPrice > 10000
  )
```

準結合と反結合についての考慮事項

準結合および反結合クエリには多大な処理作業が必要となるため、Salesforce では可能な限り最良のパフォーマンスを維持するために次の制限を定めています。

- **基本制限:**
  - 1 つの WHERE 句では、IN または NOT IN ステートメントを 2 つまでしか使用できません。
  - 準結合および反結合と NOT 演算子は一緒に使用できません。併用すると、準結合が反結合に、反結合が準結合に変換されます。NOT 演算子を使用する代わりに、適切な準結合または反結合形式でクエリを記述します。
- **主クエリの制限:**

  次の制限は、準結合または反結合クエリの主 WHERE 句に適用されます。

  - 左のオペランドは、1 つの ID (主キー) 項目または参照 (外部キー) 項目を照会する必要があります。サブクエリで選択した項目は、参照項目にできます。次に例を示します。

```
SELECT Id
 FROM Idea
 WHERE (Id IN (SELECT ParentId FROM Vote WHERE CreatedDate > LAST_WEEK AND
Parent.Type='Idea'))
```

  - 左のオペランドにはリレーションを使用できません。たとえば、次の準結合クエリは `Account.Id` リレーション項目があるため無効です。

```
SELECT Id
FROM Contact
WHERE Account.Id IN
  (
   SELECT ...
  )
```

- サブクエリの制限:
  - サブクエリは、主クエリと同じオブジェクト種別を参照する項目を照会する必要があります。
  - サブクエリ内で一致したレコードの数に制限はありません。主クエリには標準の SOQL クエリの制限が適用されます。
  - サブクエリで選択した列は、外部キー項目であることが必要で、リレーションをトラバースすることはできません。つまり、この制限により、サブクエリの選択した項目でドット表記は使用できません。たとえば、次のクエリは有効です。

```
SELECT Id, Name
FROM Account
WHERE Id IN
  (
    SELECT AccountId
    FROM Contact
    WHERE LastName LIKE 'Brown_%'
  )
```

    AccountId の代わりに `Account.Id` (ドット表記) を使用することはサポートされていません。同様に、`Contact.AccountId FROM Case` のようなサブクエリは無効です。

  - サブクエリで主クエリのオブジェクトと同じオブジェクトは照会できません。そのような*自己準結合クエリ*は、準結合または反結合を使用せずに記述できます。たとえば、次の自己準結合クエリは無効です。

```
SELECT Id, Name
FROM Account
WHERE Id IN
  (
    SELECT ParentId
    FROM Account
    WHERE Name = 'myaccount'
  )
```

次のような有効な形式でクエリを記述し直します。

```
SELECT Id, Name
FROM Account
WHERE Parent.Name = 'myaccount'
```

- 準結合または反結合ステートメントを別の準結合または反結合ステートメント内にネストできません。
- 準結合と反結合は HAVING 句では使用できません。
- 主 WHERE ステートメントで準結合と反結合を使用できますが、サブクエリの WHERE ステートメントでは使用できません。たとえば、次のクエリは有効です。

```
SELECT Id
 FROM Idea
 WHERE (Idea.Tit'sle LIKE 'Vacation%')
AND (Idea.LastCommentDate > YESTERDAY)
AND (Id IN (SELECT ParentId FROM Vote
            WHERE CreatedById = '005x0000000sMgYAAU'
             AND Parent.Type='Idea'))
```

次のクエリは、ネストされたクエリが 1 つ下のレベルなので無効です。

```
SELECT Id
 FROM Idea
 WHERE
  ((Idea.Title LIKE 'Vacation%')
  AND (CreatedDate > YESTERDAY)
  AND (Id IN (SELECT ParentId FROM Vote
              WHERE CreatedById = '005x0000000sMgYAAU'
               AND Parent.Type='Idea')
  )
  OR (Idea.Title like 'ExcellentIdea%'))
```

- サブクエリと OR を組み合わせて使用することはできません。
- COUNT、FOR UPDATE、ORDER BY、LIMIT はサブクエリではサポートされていません。
- 現在、サブクエリでは次のオブジェクトはサポートされていません。
  - ActivityHistory
  - Attachments
  - Event
  - Note
  - OpenActivity
  - Tags (AccountTag、ContactTag、その他すべてのタグオブジェクト)
  - Task

論理演算子

論理演算子を SOQL クエリの WHERE 句内の項目式に使用できます。AND、OR、NOT 演算子を使用できます。

次の表に、***fieldExpression*** 構文で使用される論理演算子の値を示します。

演算子	構文	説明
AND	***fieldExpressionX*** AND ***fieldExpressionY***	*fieldExpressionX* と *fieldExpressionY* の両方が true の場合に true。
OR	***fieldExpressionX*** OR ***fieldExpressionY***	*fieldExpressionX* または *fieldExpressionY* のいずれかが true の場合に true。 OR を使用する WHERE 句では、レコードの外部キーの値が null の場合でも、レコードが返されます。 `SELECT Id FROM Contact WHERE LastName = 'Young' or Account.Name = 'Quarry'`
NOT	not ***fieldExpressionX***	*fieldExpressionX* が false の場合に true。 この論理演算子とは異なる比較演算子 NOT IN もあります。

WHERE での日付形式と日付リテラル

WHERE 句に日付値または日付リテラルを指定して、SOQL クエリの結果を絞り込むことができます。日付は特定の日や時間を表します。一方、日付リテラルは相対的な時間の範囲、たとえば先月、今週、来年などを表します。

Salesforce から返される日付と時間の書式設定については、「FORMAT ()」および「convertTimezone()」を参照してください。

日付を使用したクエリ結果の絞り込み

WHERE 句の *fieldExpression* は、date および dateTime 項目に基づくクエリ結果の絞り込みをサポートしています。項目名と指定の日付値の間で比較演算子を使用して、条件に一致する結果を絞り込むことができます。たとえば、次のクエリは、指定された日時より後に作成された Account レコードを絞り込みます。

```
SELECT Id
FROM Account
WHERE CreatedDate > 2005-10-08T01:02:03Z
```

特定の日付に基づいてクエリ結果を絞り込むには、date 形式を使用して値を設定します。特定の日時に基づいてクエリ結果を絞り込むには、dateTime 形式を使用して値を設定します。

次の表は、SOQL クエリの WHERE 句で使用可能なサポートされる date と dateTime の形式を示しています。

項目の型	表示形式	例
date	YYYY-MM-DD	1999-01-01
dateTime	• YYYY-MM-DDThh:mm:ss+hh:mm • YYYY-MM-DDThh:mm:ss-hh:mm • YYYY-MM-DDThh:mm:ssZ	• 1999-01-01T23:01:01+01:00 • 1999-01-01T23:01:01-08:00 • 1999-01-01T23:01:01Z

dateTime 項目の値は協定世界時 (UTC) として Salesforce に保存されますが、タイムゾーンのオフセットを使用すれば、別のタイムゾーンの dateTime 値を得ることもできます。タイムゾーンのオフセットは必ずUTCを基準とします。

dateTime 形式とタイムゾーンのオフセットについての詳細は、次を参照してください。

- http://www.w3.org/TR/xmlschema-2/#isoformats
- http://www.w3.org/TR/NOTE-datetime

日付リテラルを使用したクエリ結果の絞り込み

WHERE 句の *fieldExpression* で日付リテラルを使用して、日付の範囲を基準にクエリ結果を絞り込むことができます。たとえば、過去3か月以内に作成されたクエリ結果を絞り込んだり、期限が次の会計年度より後の結果を絞り込んだりできます。

各日付リテラルは、本日を基準にした時間の範囲を表します。範囲の正確な開始と終了は、日付リテラルと、クエリを送信するユーザーのロケールによって決まります。ユーザーが個人のロケールを設定していない場合、範囲は組織のロケールによって決まります。詳細は、*Salesforce* ヘルプの「言語、ロケール、通貨の選択」を参照してください。

範囲内の結果を絞り込むには、比較演算子の = を使用します。範囲の一方の側の結果を絞り込むには、比較演算子の > または < を使用します。次の表は、日付リテラルとその範囲、および例の一覧を示しています。

日付リテラル	範囲	例
YESTERDAY	昨日の 12:00:00 AM から、その 24 時間後までが指定されます。	SELECT Id FROM Account WHERE CreatedDate = YESTERDAY
TODAY	本日の 12:00:00 AM から、その 24 時間後までが指定されます。	SELECT Id FROM Account WHERE CreatedDate > TODAY
TOMORROW	翌日の 12:00:00 AM から、その 24 時間後までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = TOMORROW
LAST_WEEK	先週の最初の日の 12:00:00 AM から、その 7 日後までが指定されます。	SELECT Id FROM Account WHERE CreatedDate > LAST_WEEK
THIS_WEEK	今週の最初の日の 12:00:00 AM から、その 7 日後までが指定されます。	SELECT Id FROM Account WHERE CreatedDate < THIS_WEEK
NEXT_WEEK	来週の最初の日の 12:00:00 AM から、その 7 日後までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = NEXT_WEEK
LAST_MONTH	先月の最初の日の 12:00:00 AM から、その月のすべての日が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_MONTH
THIS_MONTH	今月の最初の日の 12:00:00 AM から、その月のすべての日が指定されます。	SELECT Id FROM Account WHERE CreatedDate < THIS_MONTH
NEXT_MONTH	来月の最初の日の 12:00:00 AM から、その月のすべての日が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = NEXT_MONTH

日付リテラル	範囲	例
LAST_90_DAYS	本日の90日前の12:00:00 AMから、現在までが指定されます。(範囲には本日も含まれます。この日付値を使用すると、91日前から本日までのレコードが含まれます)。	SELECT Id FROM Account WHERE CreatedDate = LAST_90_DAYS
NEXT_90_DAYS	レポートの実行日の12:00:00 AMから、その90日後までが指定されます。(範囲には本日も含まれます)。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_90_DAYS
LAST_N_DAYS:*n*	本日の *n* 日前の12:00:00 AMから、現在までが指定されます。(範囲には本日も含まれます。この日付値を使用すると、*n + 1* 日前から本日までのレコードが含まれます)。標準の検索条件では、*n* を7、30、60、90、または120にできます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_DAYS:365
NEXT_N_DAYS:*n*	標準の日付検索条件の場合、レポートの実行日の12:00:00 AMから、その *n* 日後までが指定されます。(範囲には本日も含まれます)。標準の検索条件では、*n* を7、30、60、90、または120にできます。 カスタム項目検索条件の場合、翌日の12:00:00 AMから、翌 *n* 日間が指定されます。(範囲には本日は含まれません)。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_DAYS:15
N_DAYS_AGO:*n*	*n* 日前の12:00:00 AMから24時間が指定されます。(範囲には本日は含まれません)。	SELECT Id FROM Opportunity WHERE CloseDate = N_DAYS_AGO:25
NEXT_N_WEEKS:*n*	翌週の最初の日の12:00:00 AMから *n*×7日間が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_WEEKS:4
LAST_N_WEEKS:*n*	*n* 週間前の週の最初の日の12:00:00 AMから先週の最終日の11:59 PMまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_WEEKS:52
N_WEEKS_AGO:*n*	*n* 週間前の週の最初の日の12:00:00 AMから7日間が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = N_WEEKS_AGO:3
NEXT_N_MONTHS:*n*	来月1日の12:00:00 AMから、*n* か月後の月末日までのすべての日が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_MONTHS:2
LAST_N_MONTHS:*n*	*n* か月前の月の最初の日の12:00:00 AMから先月の最終日の11:59 PMまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_MONTHS:12

日付リテラル	範囲	例
N_MONTHS_AGO:*n*	*n* か月前の月の最初の日の 12:00:00 AM から、その月のすべての日が指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = N_MONTHS_AGO:6
THIS_QUARTER	当カレンダー四半期最初の日の 12:00:00 AM から、その四半期の終わりまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate = THIS_QUARTER
LAST_QUARTER	前カレンダー四半期最初の日の 12:00:00 AM から、その四半期の終わりまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_QUARTER
NEXT_QUARTER	翌カレンダー四半期最初の日の 12:00:00 AM から、その四半期の終わりまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate < NEXT_QUARTER
NEXT_N_QUARTERS:*n*	翌カレンダー四半期最初の日の 12:00:00 AM から、*n* 期後のカレンダー四半期の終わりまでが指定されます。(範囲には当期は含まれません)。	SELECT Id FROM Account WHERE CreatedDate < NEXT_N_QUARTERS:2
LAST_N_QUARTERS:*n*	*n* 四半期前のカレンダー四半期最初の日の 12:00:00 AM から、前カレンダー四半期の終わりまでが指定されます。(範囲には当期は含まれません)。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_QUARTERS:2
N_QUARTERS_AGO:*n*	*n* カレンダー四半期前のカレンダー四半期最初の日の 12:00:00 AM から、その四半期の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = N_QUARTERS_AGO:3
THIS_YEAR	今年の 1 月 1 日 12:00:00 AM から、その年の 12 月 31 日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = THIS_YEAR
LAST_YEAR	昨年の 1 月 1 日 12:00:00 AM から、その年の 12 月 31 日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_YEAR
NEXT_YEAR	来年の 1 月 1 日 12:00:00 AM から、その年の 12 月 31 日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate < NEXT_YEAR
NEXT_N_YEARS:*n*	来年の 1 月 1 日 12:00:00 AM から、*n* 年後の 12 月 31 日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = NEXT_N_YEARS:5

日付リテラル	範囲	例
`LAST_N_YEARS:n`	*n+1* 年前の 1 月 1 日 12:00:00 AM から開始します。範囲は今年より 1 年前の 12 月 31 日に終了します。	`SELECT Id FROM Opportunity WHERE CloseDate = LAST_N_YEARS:5`
`N_YEARS_AGO:n`	*n* カレンダー年前の 1 月 1 日 12:00:00 AM から、その年の 12 月 31 日の終わりまでが指定されます。	`SELECT Id FROM Opportunity WHERE CloseDate = N_YEARS_AGO:2`
`THIS_FISCAL_QUARTER`	現在の会計四半期の最初の日の 12:00:00 AM から、その会計四半期の最終日の終わりまでが指定されます。会計四半期は、[設定] の [会計年度] ページで定義されます。	`SELECT Id FROM Account WHERE CreatedDate = THIS_FISCAL_QUARTER`
`LAST_FISCAL_QUARTER`	直前の会計四半期の最初の日の 12:00:00 AM から、その会計四半期の最終日までが指定されます。会計四半期は、[設定] の [会計年度] ページで定義されます。 メモ: モバイルカスタムビューを作成するとき、会計日付項目のリテラル値はいずれもサポートされません。	`SELECT Id FROM Account WHERE CreatedDate > LAST_FISCAL_QUARTER`
`NEXT_FISCAL_QUARTER`	直後の会計四半期の最初の日の 12:00:00 AM から、その会計四半期の最終日までが指定されます。(範囲には当期は含まれません)。会計四半期は、[設定] の [会計年度] ページで定義されます。	`SELECT Id FROM Account WHERE CreatedDate < NEXT_FISCAL_QUARTER`
`NEXT_N_FISCAL_QUARTERS:n`	直後の会計四半期の最初の日の 12:00:00 AM から、*n* 期後の会計四半期の最終日の終わりまでが指定されます。(範囲には現在の会計四半期は含まれません)。会計四半期は、[設定] の [会計年度] ページで定義されます。	`SELECT Id FROM Account WHERE CreatedDate = NEXT_N_FISCAL_QUARTERS:6`
`LAST_N_FISCAL_QUARTERS:n`	*n* 期前の会計四半期の最初の日の 12:00:00 AM から、当期の直前の会計四半期の最終日の終わりまでが指定されます。(範囲には現在の会計四半期は含まれません)。会計四半期は、[設定] の [会計年度] ページで定義されます。	`SELECT Id FROM Account WHERE CreatedDate > LAST_N_FISCAL_QUARTERS:6`

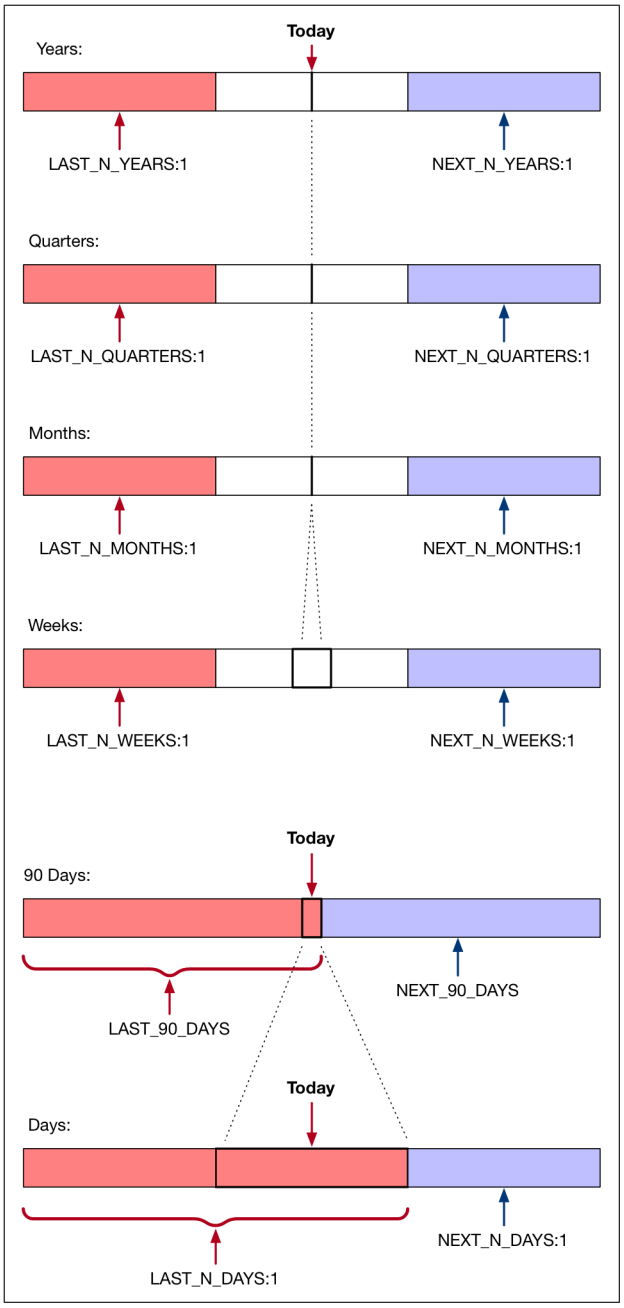
日付リテラル	範囲	例
N_FISCAL_QUARTERS_AGO:*n*	*n* 会計四半期前の会計四半期最初の日の 12:00:00 AM から、その会計四半期の最終日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = N_FISCAL_QUARTERS_AGO:6
THIS_FISCAL_YEAR	現在の会計年度の最初の日の 12:00:00 AM から、その会計年度の最終日の終わりまでが指定されます。会計四半期は、[設定]の[会計年度]ページで定義されます。	SELECT Id FROM Opportunity WHERE CloseDate = THIS_FISCAL_YEAR
LAST_FISCAL_YEAR	直前の会計年度の最初の日の 12:00:00 AM から、その会計年度の最終日の終わりまでが指定されます。会計四半期は、[設定]の[会計年度]ページで定義されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_FISCAL_YEAR
NEXT_FISCAL_YEAR	直後の会計年度の最初の日の 12:00:00 AM から、その会計年度の最終日の終わりまでが指定されます。会計四半期は、[設定]の[会計年度]ページで定義されます。	SELECT Id FROM Opportunity WHERE CloseDate < NEXT_FISCAL_YEAR
NEXT_N_FISCAL_YEARS:*n*	直後の会計年度の最初の日の 12:00:00 AM から、*n* 期後の会計年度の最終日の終わりまでが指定されます。(範囲には現在の会計年度は含まれません)。会計四半期は、[設定] の [会計年度] ページで定義されます。	SELECT Id FROM Opportunity WHERE CloseDate = NEXT_N_FISCAL_YEARS:3
LAST_N_FISCAL_YEARS:*n*	*n* 年前の会計年度の最初の日の 12:00:00 AM から、現在の直前の会計年度の最終日の終わりまでが指定されます。(範囲には現在の会計年度は含まれません)。会計四半期は、[設定] の [会計年度] ページで定義されます。	SELECT Id FROM Opportunity WHERE CloseDate = LAST_N_FISCAL_YEARS:3
N_FISCAL_YEARS_AGO:*n*	*n* 会計年度前の会計年度最初の日の 12:00:00 AM から、その会計年度の最終日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate = N_FISCAL_YEARS_AGO:3

FISCAL 日付リテラルを使用するには、最初に Salesforce でカスタム会計年度を定義する必要があります。FISCAL 日付リテラルを使用する場合、定義されている会計年度の範囲を超えた時間範囲を指定すると、日付が無効であるというエラーが返されます。

本日が日付リテラルの範囲に含まれるかどうかは、使用する日付リテラルによって決まります。

- LAST_N_*UNIT*:*n* の *UNIT* が、DAYS を除くいずれかの単位である場合、日付リテラルに本日は含まれません。したがって、たとえば、LAST_N_WEEKS:1 に本日は含まれません。

- `LAST_N_DAYS:n` および `LAST_90_DAYS` の 2 つの日付リテラルには、本日が含まれます。したがって、たとえば、`LAST_N_DAYS:1` には、昨日と本日が含まれます。また、`LAST_90_DAYS` には、90 日ではなく、91 日が含まれます。



日付の最小値と最大値

特定の範囲内の日付のみが有効です。最も早い有効な日付は 1700-01-01T00:00:00Z GMT、つまり、1700 年 1 月 1 日の午前 0 時です。最も遅い有効な日付は 4000-12-31T00:00:00Z GMT、つまり、4000 年 12 月 31 日の午前 0 時です。これらの値は、タイムゾーンごとのオフセットとなります。たとえば、太平洋タイムゾーンでは、最も早い有効な日付は 1699-12-31T16:00:00、つまり 1699 年 12 月 31 日の午後 4 時です。

多態的なリレーション項目の絞り込み

SOQL クエリで多態的なリレーション項目を検索できます。多態的なリレーションとは、現在のオブジェクトを複数のオブジェクト種別のいずれかにできるリレーションです。

多態的なリレーション項目を絞り込むには、`Type` 修飾子を使用します。

```
WHERE polymorphicRelationship.Type comparisonExpression
```

構文	説明
polymorphicRelationship	複数のオブジェクト種別を参照できる、クエリ対象オブジェクト内の多態的なリレーション項目。たとえば、Task の `Who` リレーション項目には、Contact または Lead のいずれかを使用できます。
comparisonExpression	多態的なリレーションのオブジェクト種別を対象とする比較。詳細は、*「fieldExpression」* の構文を参照してください。`Type` で返される種別名は「User」などの文字列値です。

次の例では、Event の `What` 項目が Account または Opprotunity を参照しているレコードのみが返されます。Event レコードが `What` 項目で Campaign を参照している場合は、この `SELECT` の一部としては返されません。

```
SELECT Id
FROM Event
WHERE What.Type IN ('Account', 'Opportunity')
```

多態的なリレーションについての詳細およびその他の絞り込みの例は、「リレーション項目および多態的な項目について」を参照してください。

複数選択リストのクエリ

クライアントアプリケーションでよく使用される複数選択リスト内の個々の値を検索できます。

クライアントアプリケーションでは、複数選択リストのクエリに、複数の項目を選択可能な特定の構文を使用します。API バージョン 39.0 以降では、値の API 参照名で選択リスト値を照会するため、実際の値とは異なる可能性があります。

複数選択リストのクエリでは、次の演算子がサポートされます。

演算子	説明
`=`	指定した文字列と一致する。
`!=`	指定した文字列と一致しない。
`includes`	指定した文字列を含む。
`excludes`	指定した文字列を含まない。

演算子	説明
;	複数の文字列に AND (かつ) 条件を指定する。複数の項目が選択されているときには、複数選択リストで ; を使用します。次に例を示します。 `'AAA;BBB'`

例

次のクエリでは、選択されている AAA および BBB に一致する (完全一致の) MSP1__c 項目の値に絞り込まれます。

```
SELECT Id, MSP1__c FROM CustObj__c WHERE MSP1__c = 'AAA;BBB'
```

次のクエリの場合

```
SELECT Id, MSP1__c from CustObj__c WHERE MSP1__c includes ('AAA;BBB','CCC')
```

次の値のいずれかを含む MSP1__c 項目の値で絞り込まれます。

- 選択されている AAA および BBB。
- 選択されている CCC。

条件に一致するのは、「AAA」および「BBB」を含む項目値、または「CCC」を含む項目になります。たとえば、次が一致します。

- 「AAA;BBB」と一致する

```
            'AAA;BBB'
            'AAA;BBB;DDD'
```

- 「CCC」と一致する

```
            'CCC'
            'CCC;EEE'
                    'AAA;CCC'
```

WHERE での null の使用

SOQL クエリでは、null キーワードを使用して null 値を検索できます。

たとえば、次のクエリ例では活動日が null 以外のすべての行動の取引先 ID が返されます。

```
SELECT AccountId
FROM Event
WHERE ActivityDate != null
```

Boolean 項目に対してクエリを実行する場合、null は FALSE 値に対応します。たとえば、Test_c が Boolean 項目の場合、次のクエリでは、Test_c が false であるすべてのアカウントレコードが返されます。

```
SELECT Id, Name Test_c
FROM Account
WHERE Test_c = null
```

`WHERE Test_c = null` 句は、`WHERE Test_c = false` と同等です。`WHERE Test_c != null` 句は、`WHERE Test_c = true` と同等です。

`WHERE` 句は、リレーションクエリの親項目の null 値を処理するときに、バージョンに応じて 2 通りの動作をします。親項目の値をチェックする `WHERE` 句では、親が存在しない場合でもレコードが返されます。

```
SELECT Id
FROM Case
WHERE Contact.LastName = null
```

ケースレコード `Id` 値が返されます。

関連トピック:

参照関係と外部結合での null 値

WITH

レコードを項目値に基づいて絞り込むことができます。たとえば、カテゴリによって絞り込んだり、ユーザーのプロファイルフィードで追跡されている変更を照会して取得したりするには、`WITH` ***filteringExpression*** を使用します。この句 (省略可能) は、SOQL クエリの `SELECT` ステートメントに追加できます。

`FROM` 句で指定されたオブジェクトの項目のみをサポートする `WHERE` 句とは異なり、`WITH` ではその他の関連条件で絞り込めます。たとえば、`WITH` 句を使用して、1 つ以上のデータカテゴリグループの分類に基づいて記事を絞り込めます。`WITH` 句は、次の場合にのみ使用できます。

- カテゴリに基づいてレコードを絞り込む場合。「`WITH DATA CATEGORY`」を参照してください。
- ユーザープロファイルフィードで追跡されるレコードの変更を照会して取得する場合。『Salesforce のオブジェクトリファレンス』の「Custom Object__Feed」を参照してください。
- Apex コード内で照会する項目またはオブジェクトへのユーザーアクセス権限に基づいてレコードを絞り込む場合。『*Apex 開発者ガイド*』の「WITH SECURITY_ENFORCED を使用した SOQL クエリの絞り込み」を参照してください。
- Apex データベース操作のユーザーモードアクセスを指定する場合。デフォルトでは、Apex コードはシステムモードで実行されます。つまり、コードを実行しているユーザーよりも非常に上位の権限で実行されます。Apex のセキュリティコンテキストを強化するために、`WITH USER_MODE` を使用することによってユーザーモードアクセスを指定できます。システムモードとは異なり、ユーザーモードでは実行ユーザーの項目レベルセキュリティ (FLS) とオブジェクト権限が考慮されます。ユーザーモードでは、共有ルールが常に適用されますが、システムモードではクラスの共有キーワードによって適用を制御できます。『*Apex 開発者ガイド*』の「ユーザーモードでのデータベース操作 (ベータ)」を参照してください。

`WITH` が指定されている場合、クエリは絞り込み条件に一致し、ユーザーが参照可能なレコードのみを返します。指定されていない場合、クエリは条件に一致し、ユーザーが参照可能なレコードのみを返します。

以下のステートメントでは、絞り込み条件式が太字で強調表示されています。

- `SELECT Title FROM KnowledgeArticleVersion WHERE PublishStatus='online' **WITH DATA CATEGORY Geography__c ABOVE usa__c**`
- `SELECT Id FROM UserProfileFeed **WITH UserId='005D0000001AamR'** ORDER BY CreatedDate DESC, Id DESC LIMIT 20`

WITH DATA CATEGORY

SOQL クエリでは、Salesforce ナレッジの記事と質問をデータカテゴリで検索できます。SELECT ステートメントで WITH DATA CATEGORY 句 (省略可能) を使用すると、1 つ以上のデータカテゴリに関連付けられていてユーザーが参照可能なレコードを検索できます。

```
[WITH [DATA CATEGORY] filteringExpression]
```

WITH DATA CATEGORY が指定されている場合、query() は、条件に一致するレコードのうち、指定されたデータカテゴリに関連付けられており、ユーザーが参照可能なレコードのみを返します。指定されていない場合、query() は、条件に一致するレコードのうち、ユーザーが参照可能なレコードのみを返します。

重要: CategoryData はオブジェクトで、DATA CATEGORY は SOQL WITH 句の構文です。WITH DATA CATEGORY は有効な構文ですが、WITH CategoryData はサポートされていません。

WITH DATA CATEGORY 句を使用する SOQL ステートメントには、*ObjectTypeName* が次と一致する FROM ***ObjectTypeName*** 句も含める必要があります。

- KnowledgeArticleVersion (すべての記事タイプを照会する場合)
- 記事タイプの API Name (特定の記事タイプを照会する場合)
- Question (質問を照会する場合)

ObjectTypeName が FROM 句の KnowledgeArticleVersion または記事タイプの API Name と一致する場合、次のいずれかのパラメーターを使用して WHERE 句を指定する必要があります。

- PublishStatus (公開サイクルの状況に応じて記事を照会する場合)
  - WHERE PublishStatus='online' (公開記事の場合)
  - WHERE PublishStatus='archived' (アーカイブ済み記事の場合)
  - WHERE PublishStatus='draft' (ドラフト記事の場合)
- Id (ID に基づいて記事を照会する場合)

記事タイプまたは質問については、Salesforce ヘルプの「ナレッジ記事タイプ」を参照してください。

メモ: WITH DATA CATEGORY 句はバインド変数をサポートしていません。この句に指定できるデータカテゴリ条件は 3 つまでです。

filteringExpression

WITH DATA CATEGORY 句の *filteringExpression* では次の構文を使用します。

```
dataCategorySelection [AND [dataCategorySelection][...]
```

このセクションの例は、次のデータカテゴリグループに基づいています。

```
Geography__c
    ww__c
        northAmerica__c
            usa__c
            canada__c
            mexico__c
        europe__c
            france__c
```

```
            uk__c
        asia__c
```

下記のステートメントでは、カテゴリの絞り込みが太字で強調表示されています。

- `SELECT Title FROM KnowledgeArticleVersion WHERE PublishStatus='online'` **`WITH DATA CATEGORY Geography__c ABOVE usa__c`**
- `SELECT Title FROM Question WHERE LastReplyDate > 2005-10-08T01:02:03Z` **`WITH DATA CATEGORY Geography__c AT (usa__c, uk__c)`**
- `SELECT UrlName FROM KnowledgeArticleVersion WHERE PublishStatus='draft'` **`WITH DATA CATEGORY Geography__c AT usa__c AND Product__c ABOVE_OR_BELOW mobile_phones__c`**

`AND` 論理演算子のみを使用できます。`OR` はサポートされていないため、次の構文は不正です。

```
WITH DATA CATEGORY Geography__c ABOVE usa__c OR Product__c AT mobile_phones__c
```

dataCategorySelection

SOQL クエリの `WITH DATA CATEGORY` 句のデータカテゴリ選択構文には、絞り込み条件として使用するカテゴリグループの名前、絞り込みセレクター、および絞り込みに使用するカテゴリの名前が含まれます。

dataCategorySelection では次の構文を使用します。

```
dataCategoryGroupName filteringSelector dataCategoryName
```

構文	説明
dataCategoryGroupName	絞り込み条件として使用するデータカテゴリグループの名前。次の例では、`Geography__c` がデータカテゴリグループです。 クエリで同じデータカテゴリグループを複数回使用することはできません。たとえば、次のコマンドは不正です。 `WITH DATA CATEGORY Geography__c ABOVE usa__c AND Geography__c BELOW europe__c`
filteringSelector	指定されたデータカテゴリでデータを絞り込むために使用するセレクター。有効なセレクターのリストは、「絞り込みセレクター」を参照してください。
dataCategoryName	絞り込むデータカテゴリの名前。指定したカテゴリを表示するように設定する必要があります。 複数のデータカテゴリに絞り込みの演算子を適用するには、括弧を使用します。各データカテゴリをカンマで区切る必要があります。 例: `WITH DATA CATEGORY Geography__c AT (usa__c,france__c,uk__c)` 複数のデータカテゴリの指定に括弧の代わりに `AND` 演算子を使用することはできません。次の構文は機能しません。`WITH DATA CATEGORY Geography__c AT usa__c AND france__c`

絞り込みセレクター

SOQL クエリの WITH DATA CATEGORY 句に絞り込み条件を指定する場合、指定されたカテゴリを選択するには AT、カテゴリとそのすべての親カテゴリを選択するには ABOVE、カテゴリとそのすべてのサブカテゴリを選択するには BELOW、カテゴリとその親カテゴリおよびサブカテゴリを選択するには ABOVE_OR_BELOW を使用できます。

このセクションの例は、次のデータカテゴリグループに基づいています。

```
Geography__c
    ww__c
        northAmerica__c
            usa__c
            canada__c
            mexico__c
        europe__c
            france__c
            uk__c
        asia__c
```

次の表に、*dataCategorySelection* 構文で使用される *filteringSelector* の値を示します。

セレクター	説明
AT	指定されたデータカテゴリを選択します。 たとえば、次の構文は asia__c を選択します。 `WITH DATA CATEGORY Geography__c AT asia__c`
ABOVE	指定されたデータカテゴリとそのすべての親カテゴリを選択します。 たとえば、次の構文は usa__c、northAmerica__c、および ww__c を選択します。 `WITH DATA CATEGORY Geography__c ABOVE usa__c`
BELOW	指定されたデータカテゴリとそのすべてのサブカテゴリを選択します。 たとえば、次の構文は northAmerica__c、usa__c、canada__c、および mexico__c を選択します。 `WITH DATA CATEGORY Geography__c BELOW northAmerica__c`
ABOVE_OR_BELOW	指定されたデータカテゴリと次のカテゴリを選択します。 • そのすべての親カテゴリ • そのすべてのサブカテゴリ たとえば、次の構文は ww__c、europe__c、france__c、および uk__c を選択します。 `WITH DATA CATEGORY Geography__c ABOVE_OR_BELOW europe__c`

WITH DATA CATEGORY 句を SOQL クエリの SELECT ステートメントで使用するその他の例を次に示します。

検索タイプ	例
Product__c データカテゴリグループの mobile_phones__c データカテゴリで分類されたすべての質問からタイトルを選択	`SELECT Title FROM Question WHERE LastReplyDate < 2005-10-08T01:02:03Z WITH DATA CATEGORY Product__c AT mobile_phones__c`
次で分類されたすべての公開ナレッジ記事からタイトルと概要を選択 • Geography__c データカテゴリグループの europe__c の親カテゴリとサブカテゴリ • Product__c データカテゴリグループの allProducts__c のサブカテゴリ	`SELECT Title, Summary FROM KnowledgeArticleVersion WHERE PublishStatus='Online' AND Language = 'en_US' WITH DATA CATEGORY Geography__c ABOVE_OR_BELOW europe__c AND Product__c BELOW All__c`
次で分類された「Offer__kav」タイプのドラフト記事から ID とタイトルを選択 • Geography__c データカテゴリグループの france__c または usa__c データカテゴリ • Product__ データカテゴリグループの dsl__c データカテゴリの親カテゴリ	`SELECT Id, Title FROM Offer__kav WHERE PublishStatus='Draft' AND Language = 'en_US' WITH DATA CATEGORY Geography__c AT (france__c,usa__c) AND Product__c ABOVE dsl__c`

WITH RecordVisibilityContext

RecordVisibilityContext を使用して WITH 句を絞り込み、1 つ以上のレコードの表示を決定する属性を照会します。この機能は、API バージョン 48.0 以降で使用できます。

次の項目は、パラメーターとして使用できます。

項目	説明	デフォルト
maxDescriptorPerRecord	レコードごとに返される記述子の最大数。特定レコードの記述子の実際の数がこの値を超えると、そのレコードの「tooManyDescriptors」表示属性が返されます。	• CRM Analytics を使用している組織: 150 • その他の組織: 400
supportsDomains	表示サービスに「domain」表示属性の生成を許可するかどうか。	true

項目	説明	デフォルト
supportsDelegates	表示サービスに「delegate」表示属性の生成を許可するかどうか。	true

次に例を示します。

```
SELECT Id, RecordVisibility.VisibilityAttribute FROM Account WHERE Id = 'xxx'
   WITH RecordVisibilityContext (maxDescriptorPerRecord=100, supportsDomains=true,
supportsDelegates=true)
```

```
SELECT recordId, VisibilityAttribute FROM RecordVisibility WHERE recordId = 'xxx'
   WITH RecordVisibilityContext (maxDescriptorPerRecord=100, supportsDomains=true,
supportsDelegates=true)
```

1 つ以上の項目が必要です。次に示すクエリは*無効*です。

```
SELECT Id, RecordVisibility.VisibilityAttribute FROM Account WHERE Id = 'xxx'
   WITH RecordVisibilityContext
```

次に示すクエリは有効です。

```
SELECT Id, RecordVisibility.VisibilityAttribute FROM Account WHERE Id = 'xxx'
   WITH RecordVisibilityContext (maxDescriptorPerRecord=100)
```

詳細は、*『オブジェクトリファレンス』*の「RecordVisibility」を参照してください。

GROUP BY

SOQL クエリで GROUP BY オプションを使用すると、個々のクエリ結果の反復処理を避けることができます。つまり、多くのレコードを個別に処理する代わりにレコードのグループを指定します。

GROUP BY を SUM() や MAX() などの集計関数と共に使用することで、コードで個々のレコードを処理することなく、データを集計してクエリ結果をロールアップできます。

```
[GROUP BY fieldGroupByList]
```

fieldGroupByList では、グループ化する項目のカンマ区切りのリストを指定します。SELECT 句の項目リストに集計関数が含まれている場合は、GROUP BY 句に集計関数以外のすべての項目を含める必要があります。

たとえば、GROUP BY を使用せずに、各 LeadSource 値に関連付けられているリードの数を調べるには、次のクエリを実行できます。

```
SELECT LeadSource FROM Lead
```

次に、クエリ結果を反復するコードを記述し、各 LeadSource 値のカウンターを増分します。GROUP BY を使用すると、追加のコードを記述せずに同じ結果を取得できます。次に例を示します。

```
SELECT LeadSource, COUNT(Name)
FROM Lead
GROUP BY LeadSource
```

オブジェクトのすべての個別値(null を含む)を照会するには、集計関数を使用せずに GROUP BY 句を使用できます。次のクエリは、LeadSource 項目に保存されている個別の値セットを返します。

```
SELECT LeadSource
FROM Lead
GROUP BY LeadSource
```

GROUP BY ROLLUP

クエリ結果の集計データについて小計を追加するには、SOQL クエリで GROUP BY ROLLUP 句(省略可能)を使用します。このアクションによりクエリで小計を計算できるため、コード内にそのロジックを含める必要はありません。

GROUP BY ROLLUP は、SUM() や COUNT(***fieldName***) などの集計関数と共に使用できます。

```
[GROUP BY ROLLUP (fieldName[,...])]
```

GROUP BY ROLLUP 句を使用するクエリは、GROUP BY 句を使用する同等のクエリと同じ集計データを返します。また、複数レベルの小計行も返します。GROUP BY ROLLUP 句には、カンマ区切りのリストで 3 つの項目まで含めることができます。

GROUP BY ROLLUP 句は、グルーピング列のリスト全体を右から左に集計し、異なるレベルの小計を追加します。積み上げ集計項目の順序は重要です。3 つの積み上げ集計項目を含むクエリは次の合計行を返します。

- *fieldName1* と *fieldName2* の各組み合わせの第 1 レベルの小計。結果は *fieldName3* でグループ化されます。
- *fieldName1* の各値の第 2 レベルの小計。結果は *fieldName2* と *fieldName3* でグループ化されます。
- 1 つの総計行。

メモ:

- 同じステートメント内で GROUP BY と GROUP BY ROLLUP 構文を組み合わせることはできません。たとえば、グループ化されたすべての項目は括弧で囲む必要があるため、GROUP BY ROLLUP(field1), field2 は無効です。
- GROUP BY 句で項目の考えられるすべての組み合わせの小計を含むクロス表形式レポートを作成する場合は、代わりに GROUP BY CUBE を使用します。

1 つの積み上げ集計項目によるグループ化

この単純な例では、1 つの項目で結果を積み上げ集計します。

```
SELECT LeadSource, COUNT(Name) cnt
FROM Lead
GROUP BY ROLLUP(LeadSource)
```

次の表に、クエリ結果を示します。集計された結果には、すべてのグルーピングの総計を返す、LeadSource が null 値の追加行が含まれます。積み上げ集計項目は 1 つのみのため、その他の小計はありません。

LeadSource	cnt
Web	7

LeadSource	cnt
Phone Inquiry	4
Partner Referral	4
Purchased List	7
null	22

2 つの積み上げ集計項目によるグループ化

この例では、2 つの項目で結果を積み上げ集計します。

```
SELECT Status, LeadSource, COUNT(Name) cnt
FROM Lead
GROUP BY ROLLUP(Status, LeadSource)
```

次の表は、第1レベルの小計と総計の行を含むクエリ結果を示しています。これらの例では COUNT(***fieldName***) **集計関数を使用していますが、この構文には任意の集計関数を使用できます。また、3 つの積み上げ集計項目でグループ化してさらに多くの小計行を返すこともできます。**

Status	LeadSource	cnt	コメント
Open - Not Contacted	Web	1	Status = Open - Not Contacted で LeadSource = Web のリードは 1 件
Open - Not Contacted	Phone Inquiry	1	Status = Open - Not Contacted で LeadSource = Phone Inquiry のリードは 1 件
Open - Not Contacted	Purchased List	1	Status = Open - Not Contacted で LeadSource = Purchased List のリードは 1 件
Open - Not Contacted	null	3	Status = Open - Not Contacted のすべてのリードの第 1 レベルの小計
Working - Contacted	Web	4	Status = Working - Contacted で LeadSource = Web のリードは 4 件
Working - Contacted	Phone Inquiry	1	Status = Working - Contacted で LeadSource = Phone Inquiry のリードは 1 件
Working - Contacted	Partner Referral	3	Status = Working - Contacted で LeadSource = Partner Referral のリードは 3 件
Working - Contacted	Purchased List	4	Status = Working - Contacted で LeadSource = Purchased List のリードは 4 件
Working - Contacted	null	12	Status = Working - Contacted のすべてのリードの第 1 レベルの小計
Closed - Converted	Web	1	Status = Closed - Converted で LeadSource = Web のリードは 1 件

Status	LeadSource	cnt	コメント
Closed - Converted	Phone Inquiry	1	Status = Closed - Converted で LeadSource = Phone Inquiry のリードは 1 件
Closed - Converted	Purchased List	1	Status = Closed - Converted で LeadSource = Purchased List のリードは 1 件
Closed - Converted	null	3	Status = Closed - Converted のすべてのリードの第 1 レベルの小計
Closed - Not Converted	Web	1	Status = Closed - Not Converted で LeadSource = Web のリードは 1 件
Closed - Not Converted	Phone Inquiry	1	Status = Closed - Not Converted で LeadSource = Phone Inquiry のリードは 1 件
Closed - Not Converted	Partner Referral	1	Status = Closed - Not Converted で LeadSource = Partner Referral のリードは 1 件
Closed - Not Converted	Purchased List	1	Status = Closed - Not Converted で LeadSource = Purchased List のリードは 1 件
Closed - Not Converted	null	4	Status = Closed - Not Converted のすべてのリードの第 1 レベルの小計
null	null	22	リードの総計は 22 件

GROUP BY CUBE

クエリ結果のグループ化項目のすべての組み合わせについて小計を追加するには、SOQL クエリで GROUP BY CUBE 句を使用します。このアクションは、データのクロス表形式レポートを作成するときに役立ちます。たとえば、クロス表クエリを作成して、合計、平均、または別の集計関数の計算を行ってから、2 つの値セット (横方向と縦方向) で結果をグループ化できます。

GROUP BY CUBE は、SUM() や COUNT(***fieldName***) などの集計関数と共に使用できます。

```
[GROUP BY CUBE (fieldName[,...])]
```

GROUP BY CUBE 句を使用するクエリは、GROUP BY 句を使用する同等のクエリと同じ集計データを返します。さらに、カンマ区切りのグルーピングリストで指定された項目の各組み合わせの追加小計行および総計行を返します。GROUP BY CUBE 句には、3 つの項目まで含めることができます。

メモ:

- 同じステートメント内で GROUP BY と GROUP BY CUBE 構文を組み合わせることはできません。たとえば、グループ化されたすべての項目は括弧で囲む必要があるため、GROUP BY CUBE(field1), field2 は無効です。
- グループ化された項目の組み合わせのサブセットで小計のみが必要な場合は、代わりに GROUP BY ROLLUP を使用します。

次のクエリは、Type と BillingCountry の各組み合わせでの取引先の小計を返します。

```
SELECT Type, BillingCountry,
    GROUPING(Type) grpType, GROUPING(BillingCountry) grpCty,
    COUNT(id) accts
FROM Account
GROUP BY CUBE(Type, BillingCountry)
ORDER BY GROUPING(Type), GROUPING(BillingCountry)
```

次の表に、クエリ結果を示します。このクエリは、集計データ行の後に小計行と総計行が返されるように ORDER BY GROUPING(Type), GROUPING(BillingCountry) を使用しています。これは必須ではありませんが、コードのクエリ結果を反復処理するときに役立ちます。

Type	BillingCountry	grpType	grpCty	accts	コメント
Customer - Direct	null	0	0	6	Type = Customer - Direct で BillingCountry = null の取引先は 6 件
Customer - Channel	USA	0	0	1	Type = Customer - Channel で BillingCountry = USA の取引先は 1 件
Customer - Channel	null	0	0	2	Type = Customer - Channel で BillingCountry = null の取引先は 2 件
Customer - Direct	USA	0	0	1	Type = Customer - Direct で BillingCountry = USA の取引先は 1 件
Customer - Channel	France	0	0	1	Type = Customer - Channel で BillingCountry = France の取引先は 1 件
null	USA	0	0	1	Type = null で BillingCountry = USA の取引先は 1 件
Customer - Channel	null	0	1	4	Type = Customer - Channel の取引先の小計は 4 件 (grpCty = 1 は結果が BillingCountry 項目でグループ化されていることを示す)
Customer - Direct	null	0	1	7	Type = Customer - Direct の取引先の小計は 7 件 (grpCty = 1 は結果が BillingCountry 項目でグループ化されていることを示す)
null	null	0	1	1	Type = null の取引先の小計は 1 件 (grpCty = 1 は結果が BillingCountry 項目でグループ化されていることを示す)
null	France	1	0	1	BillingCountry = France の取引先の小計は 1 件 (grpType = 1 は結果が Type 項目でグループ化されていることを示す)
null	USA	1	0	3	BillingCountry = USA の取引先の小計は 3 件 (grpType = 1 は結果が Type 項目でグループ化されていることを示す)

Type	BillingCountry	grpType	grpCty	accts	コメント
null	null	1	0	8	BillingCountry = null の取引先の小計は 8 件 (grpType = 1 は結果が Type 項目でグループ化されていることを示す)
null	null	1	1	12	取引先の総計は 12 件 (grpType = 1 と grpCty = 1 はこれが総計であることを示す)

これらのクエリ結果を使用して、結果のクロス表形式レポートを表示できます。

Type/BillingCountry	USA	France	null	合計
Customer - Direct	1	0	6	7
Customer - Channel	1	1	2	4
null	1	0	0	1
Total	3	1	8	12

GROUP BY での別名の使用

SOQL クエリの SELECT ステートメントで任意の項目または集計項目に別名を使用できます。コードでクエリ結果を処理するときに、項目の別名を使用して項目を識別します。

関連付けられた項目の直後に別名を指定します。たとえば、次のクエリでは Name 項目に n、MAX(Amount) 集計項目に max の 2 つの別名が指定されています。

```
SELECT Name n, MAX(Amount) max
FROM Opportunity
GROUP BY Name
```

別名のない SELECT リストの集計項目は、形式が expr*i* の暗黙的別名を自動的に取得します。*i* は、明示的な別名のない集計項目の順序を示します。*i* の値は 0 から始まり、明示的な別名のない集計項目ごとに増えます。

次の例では、MAX(Amount) の暗黙的別名は expr0 で、MIN(Amount) の暗黙的別名は expr1 です。

```
SELECT Name, MAX(Amount), MIN(Amount)
FROM Opportunity
GROUP BY Name
```

次のクエリでは、MIN(Amount) の暗黙的別名は min、MAX(Amount) の暗黙的別名は expr0 で、SUM(Amount) の暗黙的別名は expr1 です。

```
SELECT Name, MAX(Amount), MIN(Amount) min, SUM(Amount)
FROM Opportunity
GROUP BY Name
```

GROUP BY による小計の識別

SOQL クエリで GROUP BY ROLLUP または GROUP BY CUBE を使用して小計を追加する場合、GROUPING(fieldName) 関数を使用して、行が項目の小計であるかどうかを識別できます。

データのレポートまたはグラフを作成するためにクエリ結果を反復処理する場合、集計データ行と小計行を区別する必要があります。GROUPING(fieldName) を使用してこれを実行できます。GROUPING(fieldName) の使用は、GROUP BY ROLLUP 句または GROUP BY CUBE 句で複数の項目を使用しているときに、結果を解釈する場合に特に重要です。これは集計データと小計を区別するのに最適な方法です。

このクエリ例は、LeadSource 項目と Rating 項目の組み合わせの小計を返します。GROUPING(LeadSource) は、行が LeadSource 項目の集計行であるかどうかを示し、GROUPING(Rating) は行が Rating 項目の集計行であるかどうかを示します。

```
SELECT LeadSource, Rating,
    GROUPING(LeadSource) grpLS, GROUPING(Rating) grpRating,
    COUNT(Name) cnt
FROM Lead
GROUP BY ROLLUP(LeadSource, Rating)
```

次の表に、クエリ結果を示します。

LeadSource	Rating	grpLS	grpRating	cnt	コメント
Web	null	0	0	5	LeadSource = Web で Rating なしのリードは 5 件
Web	Hot	0	0	1	LeadSource = Web で Rating = Hot のリードは 1 件
Web	Warm	0	0	1	LeadSource = Web で Rating = Warm のリードは 1 件
Web	null	0	1	7	LeadSource = Web のリードの小計は 7 件 (grpRating = 1 は結果が Rating 項目でグループ化されていることを示す)
Phone Inquiry	null	0	0	4	LeadSource = Phone Inquiry で Rating なしのリードは 4 件
Phone Inquiry	null	0	1	4	LeadSource = Phone Inquiry のリードの小計は 4 件 (grpRating = 1 は結果が Rating 項目でグループ化されていることを示す)
Partner Referral	null	0	0	4	LeadSource = Partner Referral で Rating なしのリードは 4 件
Partner Referral	null	0	1	4	LeadSource = Partner Referral のリードの小計は 4 件 (grpRating = 1 は結果が Rating 項目でグループ化されていることを示す)
Purchased List	null	0	0	7	LeadSource = Purchased List で Rating なしのリードは 7 件

LeadSource	Rating	grpLS	grpRating	cnt	コメント
Purchased List	null	0	1	7	LeadSource = Purchased List のリードの小計は 7 件 (grpRating = 1 は結果が Rating 項目でグループ化されていることを示す)
null	null	1	1	22	リードの総計は 22 件 (grpRating = 1 と grpLS = 1 はこれが総計であることを示す)

ヒント: GROUP BY ROLLUP 句で指定されている項目の順序は重要です。たとえば、各 LeadSource よりも各 Rating の小計を取得する方が重要な場合、項目の順序を GROUP BY ROLLUP(Rating, LeadSource) に変更します。

GROUP BY に関する考慮事項

SOQL クエリで GROUP BY 句を使用する場合には、特殊な動作や制限について理解する必要があります。

- 一部のオブジェクト項目には、グループ化がサポートされないデータ型があります。GROUP BY 句には、これらのデータ型を使用する項目を含めることができません。

  項目でグループ化がサポートされるかどうかを確認するには、SOAP API または REST API を使用して、その項目を含むオブジェクトに対して describe コールを実行します。レスポンスには、項目ごとに GROUP BY 句に項目を含めることができるかどうかが定義された、Boolean の groupable 属性が含まれています。項目で標準オブジェクトのグループ化がサポートされるかどうかを確認するには、『Salesforce および Lightning プラットフォームのオブジェクトリファレンス』で項目のプロパティも確認してください。

- クエリに LIMIT 句と集計関数がある場合には、GROUP BY 句を使用する必要があります。たとえば、次のクエリは有効です。

  ```
  SELECT Name, Max(CreatedDate)
  FROM Account
  GROUP BY Name
  LIMIT 5
  ```

  次のクエリは、GROUP BY 句がないので無効です。

  ```
  SELECT MAX(CreatedDate)
  FROM Account LIMIT 1
  ```

- GROUP BY 句を使用するクエリでは、__r 構文を使用する子リレーションの式は使用できません。詳細は、「リレーション名、カスタムオブジェクトおよびカスタム項目について」を参照してください。
- SOAP API では、GROUP BY 句を含むクエリで queryMore() を使用してより多くの結果を得ることはできません。
- REST API では、GROUP BY 句を含むクエリでクエリロケーターを使用してより多くの結果を得ることはできません。

HAVING

HAVING 句 (省略可能) を SOQL クエリに追加すると、集計関数が返した結果を絞り込むことができます。

HAVING 句は、GROUP BY 句と一緒に使用することで、SUM() などの集計関数によって返される結果を絞り込めます。HAVING 句は WHERE 句と類似しています。HAVING 句には集計関数を含めることができますが、WHERE 句には集計関数を含めることはできない点が異なります。

WHERE 句と同様に、HAVING 句でも条件式で = などのすべての比較演算子がサポートされており、論理演算子 AND、OR、および NOT を使用して複数の条件を含めることができます。

```
[HAVING havingConditionExpression]
```

havingConditionExpression には、クエリ結果を絞り込む集計関数を使用して 1 つ以上の条件式を指定します。

たとえば、次のクエリで GROUP BY 句を使用して、各 LeadSource 値に関連付けられたリード数を判断できます。

```
SELECT LeadSource, COUNT(Name)
FROM Lead
GROUP BY LeadSource
```

ただし、100 件を超えるリードを生成した LeadSource 値のみが必要な場合、HAVING 句を使用して結果を絞り込めます。次に例を示します。

```
SELECT LeadSource, COUNT(Name)
FROM Lead
GROUP BY LeadSource
HAVING COUNT(Name) > 100
```

次のクエリ例は、名前が重複する取引先を返します。

```
SELECT Name, Count(Id)
FROM Account
GROUP BY Name
HAVING Count(Id) > 1
```

次のクエリ例は、City が GROUP BY 句に含まれていないため無効です。

```
SELECT LeadSource, COUNT(Name)
FROM Lead
GROUP BY LeadSource
HAVING COUNT(Name) > 100 and City LIKE 'San%'
```

メモ: HAVING 句に準結合または反結合サブクエリを含めることはできません。

ORDER BY

SOQL クエリの SELECT ステートメントで ORDER BY (省略可能) を使用すると、クエリ結果の順序を制御できます (アルファベットの降順など)。レコードが null の場合、ORDER BY を使用して空のレコードを最初か最後に表示できます。

```
[ORDER BY fieldOrderByList {ASC|DESC} [NULLS {FIRST|LAST}] ]
```

構文	説明
`ASC` または `DESC`	結果を昇順 (`ASC`) で並び替えるか降順 (`DESC`) で並び替えるかを指定します。デフォルトの並び替え順序は昇順です。
`NULLS FIRST` または `NULLS LAST`	null のレコードを結果の先頭 (`NULLS FIRST`) または最後 (`NULLS LAST`) に並び替えます。デフォルトでは、null の値が最初に並び替えられます。

クエリで `ORDER BY` 句を使用しない場合、結果の順序は保証されません。

`ORDER BY` 句を使用したで場合も、`ORDER BY` 句で使用している項目の値に重複がある場合には、結果の順序が変わる可能性があります。たとえば、同じ Industry (業界) の複数の Account (取引先) レコードがある場合、このクエリの結果の順番が変わることがあります。

```
SELECT Name, Industry FROM Account ORDER BY Industry
```

この問題を回避するには、`ORDER BY` 句に ID (または結果で一意となるその他の項目) を追加します。次に例を示します。

```
SELECT Name, Industry FROM Account ORDER BY Industry, Id
```

次のクエリ例は Account レコードの名をアルファベットの降順で並び替えて返します。名が null の取引先は末尾に並べられます。

```
SELECT Name
FROM Account
ORDER BY Name DESC NULLS LAST
```

次の要素が `ORDER BY` を使用して返される結果に影響します。

- `ORDER BY` はリレーションクエリの構文と互換性があります。
- 複数の ***`fieldExpression`*** 句を使用した複数列の並び替えがサポートされています。
- `ORDER BY` 句で外部キーの値を使用したリレーションクエリの動作は、Lightning Platform API のバージョンによって異なります。`ORDER BY` 句でレコードの外部キーの値が null の場合でも、レコードが返されます。

  ```
  SELECT Id, CaseNumber, Account.Id, Account.Name
  FROM Case
  ORDER BY Account.Name
  ```

  `AccountId` が空白のケースレコードは、返されます。
- 並び替え順は、ユーザーのロケール設定によって異なります。英語ロケールの場合、SOQL は、並び替え順を作成する際に、大文字の UTF-8 値を使用します。つまり、英語のロケールの並び替えは、大文字と小文字が区別されません。

  英語以外のロケールの場合、SOQL は、事前定義済みの並び替えのうち、指定されたロケールで自然であるものを使用します。これは、同じ文字であっても文化が異なれば並び替えの方法も異なる場合があることや、このニュアンスを UTF-8 値単独ではとらえることができないことが理由です。

`ORDER BY` を使用する場合、データ型に次の制限事項が適用されます。

- 複数選択リスト、リッチテキストエリア、ロングテキストエリア、暗号化 (有効な場合)、およびデータカテゴリグループの参照 (Salesforce ナレッジが有効な場合) のデータ型はサポートされていません。

- その他すべてのデータ型はサポートされていますが、次の点に注意してください。
  - マスター通貨は、使用可能な場合は常にマスター通貨値を使用して並び替えられます。
  - phone データの特殊な書式は並び替えに含まれません。たとえば、ダッシュや括弧などの数値以外の文字は並び替えには含まれません。
  - picklist の並び替えは、設定時に決定された選択リストの並び替えで定義されます。

外部オブジェクトの場合、ORDER BY 句に次の制限があります。

SELECT ステートメントで ORDER BY を省略可能な LIMIT 修飾子と共に使用できます。

```
SELECT Name
FROM Account
WHERE industry = 'media'
ORDER BY BillingPostalCode ASC NULLS LAST LIMIT 125
```

LIMIT

LIMIT 句 (省略可能) を SOQL クエリの SELECT ステートメントに追加すると、返される最大行数を指定できます。

LIMIT の構文は次のとおりです。

```
SELECT fieldList
FROM objectType
[WHERE conditionExpression]
  [LIMIT numberOfRows]
```

次に例を示します。

```
SELECT Name
FROM Account
WHERE Industry = 'Media' LIMIT 125
```

このクエリは、Industry が Media の最初の 125 件の Account レコードを返します。

fieldList として count() と共に LIMIT を使用して、指定された最大数まで数えられます。

集計関数を使用して GROUP BY 句を使用しないクエリでは LIMIT 句を使用できません。たとえば、次のクエリは無効です。

```
SELECT MAX(CreatedDate)
FROM Account LIMIT 1
```

OFFSET

クエリ結果に大量のレコードが含まれると予想される場合、SOQL クエリに OFFSET 句を使用して結果を複数ページに表示できます。たとえば、OFFSET を使用して 51 ～ 75 番目のレコードを表示した後、スキップして 301 ～ 350 番目のレコードを表示できます。OFFSET を使用すると、大きな結果セットを効率よく処理できます。

クエリによって返される結果セットに開始行を指定するには、OFFSET を使用します。オフセットの計算はサーバーで実行されて結果サブセットのみが返されるため、完全な結果セットを取得して結果をローカルで絞り込むよりも OFFSET を使用した方が効率的です。OFFSET は、API バージョン 24.0 以降で使用できます。

```
SELECT fieldList
FROM objectType
[WHERE conditionExpression]
ORDER BY fieldOrderByList
LIMIT numberOfRowsToReturn
OFFSET numberOfRowsToSkip
```

たとえば、SOQL クエリが通常は50行を返す場合、クエリで OFFSET 10 を使用することで最初の10行をスキップできます。クエリ例の結果セットは、完全な結果セットの行 11 ～ 110 が返されたサブセットです。

```
SELECT Name
FROM Merchandise__c
WHERE Price__c > 5.0
ORDER BY Name
LIMIT 100
OFFSET 10
```

OFFSET を使用するときの考慮事項

クエリで OFFSET を使用する場合、次のいくつかの事項を考慮してください。

- 最大オフセットは2,000行です。2,000 より大きいオフセットを要求すると NUMBER_OUTSIDE_VALID_RANGE エラーが発生します。
- OFFSET 句は、SOAP API、REST API、および Apex で使用される SOQL で許可されます。Bulk API またはストリーミング API 内で使用される SOQL では許可されません。
- OFFSET は最上位のクエリでのみ使用可能で、ほとんどのサブクエリでは使用できないため、次のクエリは無効であり、MALFORMED_QUERY エラーを返します。

  ```
  SELECT Name, Id
  FROM Merchandise__c
  WHERE Id IN
     (
       SELECT Id
       FROM Discontinued_Merchandise__c
       LIMIT 100
       OFFSET 20
     )
  ORDER BY Name
  ```

  サブクエリは、親クエリに LIMIT 1 句がある場合に限り OFFSET を使用できます。次に、クエリでの OFFSET の有効な使用を示します。

  ```
  SELECT Name, Id,
      (
          SELECT Name FROM Opportunities LIMIT 10 OFFSET 2
      )
  FROM Account
  ```

```
ORDER BY Name
LIMIT 1
```

OFFSET は、親クエリに LIMIT 1 句がある場合でも WHERE 句のサブクエリとして使用できません。

> メモ: サブクエリでの OFFSET の使用は、今後のリリースで変更される可能性のあるパイロット機能で、本番設定で使用することを想定していません。このパイロット機能に関連するサポートはありません。詳細は、Salesforce にお問い合わせください。

- OFFSET を使用する場合、結果セットの順序付けの一貫性を保つため ORDER BY 句を使用することをお勧めします。ORDER BY 句を使用しない結果セットの行の順序は一定ですが、順序付けキーは変更される可能性があるため信頼しないようにしてください。
- 同じ結果セットの後続のサブセットを取得する必要がある場合は、LIMIT 句と OFFSET を使用することをお勧めします。たとえば、次を使用してクエリの最初の 100 行を取得できます。

```
SELECT Name, Id
FROM Merchandise__c
ORDER BY Name
LIMIT 100
OFFSET 0
```

その後、以下のクエリを使用して次の 100 行 (101 ～ 201) を取得できます。

```
SELECT Name, Id
FROM Merchandise__c
ORDER BY Name
LIMIT 100
OFFSET 100
```

- OFFSET は、クエリの時点で返された結果セットに適用されます。その後の OFFSET クエリで完全な結果セットをキャッシュするサーバー側のカーソルやクエリロケーターは作成されません。OFFSET を使用して同じ結果セットに複数のクエリを実行しているときに、基になるデータが変更された場合、ページ結果が変更される可能性があります。たとえば、次のクエリは通常は50行の完全な結果セットを返し、OFFSET 句を使用して最初の 10 行がスキップされるとします。

```
SELECT Name
FROM Merchandise__c
ORDER BY Name
OFFSET 10
```

クエリが実行された後に、並び替え順で先頭になる Name 値を含む新しい 10 行が Merchandise__c に挿入されます。同じ OFFSET 値でクエリを再度実行すると、異なる行のセットがスキップされます。複数のページのレコードを照会し、データベース内の特定のレコードを一貫して検出する必要がある場合は、queryMore() コールを使用します。

- 最大オフセットサイズとサーバー側カーソルの欠如を考慮すると、オフセットは queryMore() の代わりに使用するようには意図されていません。大きな結果セットへのオフセットを含む複数のクエリは、サーバー側カーソルに対して queryMore() を使用するよりもパフォーマンス上の影響が大きくなります。
- OFFSET を使用する場合、所定のクエリではレコードの最初のバッチのみが返されます。次のバッチを取得する場合、オフセット値を高くしたクエリを再実行する必要があります。

FOR VIEW と FOR REFERENCE

Salesforce では、インターフェースでのレコード表示に関する情報が保存され、その情報を使用して、サイドバーや検索のオートコンプリートオプションなどで、最近表示および参照したレコードのリストが生成されます。取得されたオブジェクトの最近の利用状況データを更新するには、FOR VIEW 句を FOR REFERENCE 句と共に使用します。

FOR VIEW

SOQL クエリで FOR VIEW 句 (省略可能) を使用することにより、オブジェクトをその最終参照時刻に関する情報で更新できます。

FOR VIEW 句をクエリで使用すると、次の 2 つの処理が行われます。

- 取得されたレコードの LastViewedDate 項目が更新されます。
- レコードが RecentlyViewed オブジェクトに追加され、取得されたレコードに最近参照したデータが反映されます。

次の SOQL クエリの例は、取引先責任者を取得し、FOR VIEW を使用して、取得した取引先責任者の最終参照日を更新します。同じステートメントで、レコードの取得と最終参照日の更新の両方の処理が行われます。

```
SELECT Name, ID FROM Contact  LIMIT 1 FOR VIEW
```

FOR REFERENCE

FOR REFERENCE 句 (省略可能) を SOQL クエリで使用すると、モバイルアプリケーションなどのカスタムインターフェースまたはカスタムページからレコードが参照されたときに Salesforce に通知できます。

レコードは、関連レコードが表示されるたびに参照されます。たとえば、ユーザーが取引先レコードを表示すると、すべての関連レコード (取引先責任者、所有者、リード、商談など) が参照されます。

FOR REFERENCE 句をクエリで使用すると、次の 2 つの処理が行われます。

- 取得されたレコードの LastReferencedDate 項目が更新されます。
- レコードが RecentlyViewed オブジェクトに追加され、取得されたレコードごとに最近参照したデータが反映されます。

次の SOQL クエリの例は、取引先責任者を取得し、FOR REFERENCE を使用して、取得した取引先責任者の LastReferencedDate 項目を更新します。

```
SELECT Name, ID FROM Contact  LIMIT 1 FOR REFERENCE
```

メモ:

- FOR VIEW および FOR REFERENCE 句は、取得されたレコードをログインユーザーが確実に参照する場合にのみ使用してください。参照されないと、レコードの使用状況の情報が、この句により誤って更新されます。また、最近使ったデータや、グローバル検索のオートコンプリートリストで誤って更新されたレコードが表示されるとき、ユーザーはそれを識別できません。
- RecentlyViewed オブジェクトは、ログインユーザーがレコードを表示または参照するたびに更新されます。また、SOQL クエリで FOR VIEW または FOR REFERENCE 句を使用してレコードを取得した場合にも更新されます。最新のデータを確実に使用できるようにするには、1 オブジェクトにつきレコード

が 200 件までになるよう、RecentlyViewed データを定期的に切り捨てます。RecentlyViewed データは 90 日間保持され、90 日が経過すると定期的に削除されます。

UPDATE

UPDATE TRACKING 句 (省略可能) を使用すると、Salesforce ナレッジ記事の検索で使用されるキーワードを追跡できます。Salesforce ナレッジ記事が表示された回数を判別するには、UPDATE VIEWSTAT 句 (省略可能) を使用します。

SOQL を使用して記事のキーワード追跡を更新する

記事の検索と表示数に関するレポートを作成するには、UPDATE TRACKING 句 (省略可能) を SOQL クエリの SELECT ステートメントに追加します。開発者は UPDATE TRACKING を使用して、Salesforce ナレッジ記事の検索で使用されるキーワードを追跡できます。

```
SELECT Title FROM FAQ__kav
WHERE Keyword='Apex' and
Language = 'en_US' and
KnowledgeArticleVersion = 'ka230000000PCiy'
UPDATE TRACKING
```

SOQL を使用して記事の参照統計を更新する

UPDATE VIEWSTAT 句は、Salesforce ナレッジの記事の検索および参照についてレポートするために SELECT ステートメントで使用します。オンラインでのアクセス権のあるすべての記事について参照数を取得できます。開発者は、UPDATE VIEWSTAT を使用して、記事の参照統計を更新できます。

次の構文を使用して、オンラインでアクセスできるすべての記事の参照数を増加できます。

```
SELECT Title FROM Knowledge__kav
   WHERE PublishStatus='online' and
   Language = 'en_US' and
   KnowledgeArticleVersion = 'ka230000000PCiy'
   UPDATE VIEWSTAT
```

FOR UPDATE

Apex の FOR UPDATE では、レコードの更新中に sObject レコードをロックして、競合の条件やスレッドの安全性の問題の発生を回避できます。

sObject レコードがロックされると、他のすべてのクライアントとユーザーは、コードまたは Salesforce ユーザーインターフェースを使用して更新を行えません。レコードをロックしているクライアントは、レコードに対してロジックを実行し、更新を行うことができます。ロック中は、ロックされたレコードが別のクライアントによって変更されることはありません。トランザクションが完了するとロックが解除されます。

メモ: ロックを使用する SOQL クエリでは、ORDER BY 句を使用できません。

Apex の一連の sObject レコードをロックするには、インライン SOQL ステートメントの後に FOR UPDATE キーワードを埋め込みます。たとえば、2 つの取引先を照会すると共に、返された取引先をロックします。

```
Account [] accts = [SELECT Id FROM Account LIMIT 2 FOR UPDATE];
```

警告: Apex コードにロックを設定する場合は、慎重に行ってください。詳細は、『*Apex 開発者ガイド*』の「レコードのロック」を参照してください。

SOQL SELECT の例

SOQL を使用するテキスト検索の例を次に示します。

検索タイプ	例
単純なクエリ	`SELECT Id, Name, BillingCity FROM Account`
WHERE	`SELECT Id FROM Contact WHERE Name LIKE 'A%' AND MailingCity = 'California'`
ORDER BY	`SELECT Name FROM Account ORDER BY Name DESC NULLS LAST`
LIMIT	`SELECT Name FROM Account WHERE Industry = 'media' LIMIT 125`
ORDER BY と LIMIT	`SELECT Name FROM Account WHERE Industry = 'media' ORDER BY BillingPostalCode ASC NULLS LAST LIMIT 125`
count()	`SELECT COUNT() FROM Contact`
GROUP BY	`SELECT LeadSource, COUNT(Name) FROM Lead GROUP BY LeadSource`
HAVING	`SELECT Name, COUNT(Id) FROM Account GROUP BY Name HAVING COUNT(Id) > 1`
OFFSET と ORDER BY	`SELECT Name, Id FROM Merchandise__c ORDER BY Name OFFSET 100`
OFFSET と ORDER BY、LIMIT の併用	`SELECT Name, Id FROM Merchandise__c ORDER BY Name LIMIT 20 OFFSET 100`
リレーションクエリ: 子-親	`SELECT Contact.FirstName, Contact.Account.Name FROM Contact` `SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'`
リレーションクエリ: 親-子	`SELECT Name, (SELECT LastName FROM Contacts) FROM Account` `SELECT Account.Name, (SELECT Contact.LastName FROM Account.Contacts) FROM Account`

検索タイプ	例
WHERE を使用するリレーションクエリ	`SELECT Name, (SELECT LastName FROM Contacts WHERE CreatedBy.Alias = 'x') FROM Account WHERE Industry = 'media'`
リレーションクエリ:カスタムオブジェクトを使用する子-親	`SELECT Id, FirstName__c, Mother_of_Child__r.FirstName__c FROM Daughter__c WHERE Mother_of_Child__r.LastName__c LIKE 'C%'`
リレーションクエリ:カスタムオブジェクトを使用する親-子	`SELECT Name, (SELECT Name FROM Line_Items__r) FROM Merchandise__c WHERE Name LIKE 'Acme%'`

多態的なキーを使用するリレーションクエリ	`SELECT Id, Owner.Name FROM Task WHERE Owner.FirstName like 'B%'`
	`SELECT Id, Who.FirstName, Who.LastName FROM Task WHERE Owner.FirstName LIKE 'B%'`
	`SELECT Id, What.Name FROM Event`

TYPEOF を使用する多態的なリレーションクエリ	`SELECT TYPEOF What WHEN Account THEN Phone, NumberOfEmployees WHEN Opportunity THEN Amount, CloseDate ELSE Name, Email END FROM Event`
集計を使用するリレーションクエリ	`SELECT Name, (SELECT CreatedBy.Name FROM Notes) FROM Account` `SELECT Amount, Id, Name, (SELECT Quantity, ListPrice, PricebookEntry.UnitPrice, PricebookEntry.Name FROM OpportunityLineItems) FROM Opportunity`
単純なクエリ:各ユーザーの UserId と LoginTime	`SELECT UserId, LoginTime from LoginHistory`
特定時間範囲のユーザーあたりのログイン回数を使用するリレーションクエリ	`SELECT UserId, COUNT(Id) from LoginHistory WHERE LoginTime > 2010-09-20T22:16:30.000Z AND LoginTime < 2010-09-21T22:16:30.000Z GROUP BY UserId`
ユーザーあたりの RecordType を取得するクエリ	`SELECT Id, Name, IsActive, SobjectType, DeveloperName, Description FROM RecordType` メモ: ユーザーにレコードタイプが属するオブジェクトに対する「参照」アクセス権がなければ、クエリの結果にそのレコードタイプは返されません。

メモ: Apex では、使用しているステートメントで SOQL ステートメントや SOSL ステートメントを使うには、角括弧で囲む必要があります。前にコロン (:) がある場合は、Apex スクリプト変数と式を使用できます。

SOQL SELECT 関数

SOQL クエリでは関数を使用することで、分析用のレポートを作成したり、標準およびカスタム項目にローカライズした書式を適用したり、日付の期間でデータをグループ化したり、絞り込んだりできます。

関数	説明
集計関数	分析用のレポートを生成するには、SOQL クエリの GROUP BY 句に集計関数を使用します。
convertCurrency()	通貨項目をユーザーの通貨に変換するには、SOQL クエリに convertCurrency() を使用します。
convertTimezone()	dateTime 項目をユーザーのタイムゾーンに変換するには、SOQL クエリの日付関数で convertTimezone() を使用します。
日付関数	SOQL クエリで日付関数を使用すると、日、カレンダー月、会計年度などの日付の期間によってデータのグループ化または絞り込みができます。
FORMAT ()	SOQL クエリで FORMAT() を使用することで、標準およびカスタムの数値、日付、時間、通貨項目にローカライズされた書式を適用できます。
GROUPING(fieldName)	SOQL クエリで GROUPING(fieldName) を使用することにより小計を識別できます。
toLabel()	toLabel() メソッドを使用することにより、SOQL クエリの結果をクエリ送信者の言語に翻訳できます。

集計関数

分析用のレポートを生成するには、SOQL クエリの GROUP BY 句に集計関数を使用します。集計関数には、AVG()、COUNT()、MIN()、MAX()、SUM() などがあります。

集計関数は、GROUP BY 句を使用しなくても使用できます。たとえば、AVG() 集計関数を使用して、すべての商談の平均［金額］を調べることができます。

```
SELECT AVG(Amount)
FROM Opportunity
```

ただし、これらの関数は GROUP BY 句と共に使用すると、より強力なレポートを生成するツールとなります。たとえば、キャンペーンごとにすべての商談の平均［金額］を調べることができます。

```
SELECT CampaignId, AVG(Amount)
FROM Opportunity
GROUP BY CampaignId
```

SOQL でサポートされるすべての集計関数を次の表に示します。

メモ: すべての集計関数では、null 値が無視されます。ただし、COUNT() と COUNT(Id) は除きます。COUNT(***fieldname***) は、COUNT() とは異なります。COUNT(***fieldname***) では、null 値が無視されます。

集計関数	説明

AVG()	数値項目の平均値を返します。次に例を示します。
	`SELECT CampaignId, AVG(Amount)`
	`FROM Opportunity`
	`GROUP BY CampaignId`

COUNT() および COUNT(***fieldName***)	クエリ条件に一致する行数を返します。COUNT() を使用した例:
	`SELECT COUNT()`
	`FROM Account`
	`WHERE Name LIKE 'a%'`
	COUNT(***fieldName***) を使用した例:
	`SELECT COUNT(Id)`
	`FROM Account`
	`WHERE Name LIKE 'a%'`
	メモ: SOQL の COUNT() と COUNT(Id) は、SQL の COUNT(*) に相当します。

COUNT_DISTINCT()	クエリ条件に一致する、null 以外の個別の項目値の数を返します。次に例を示します。
	`SELECT COUNT_DISTINCT(Company)`
	`FROM Lead`
	メモ: SOQL の COUNT_DISTINCT(***fieldName***) は、SQL の COUNT(DISTINCT ***fieldName***) に相当します。

MIN()	項目の最小値を返します。次に例を示します。
	`SELECT MIN(CreatedDate), FirstName, LastName`
	`FROM Contact`
	`GROUP BY FirstName, LastName`
	選択リスト項目で MIN() または MAX() 関数を使用すると、アルファベット順の代わりに、選択リスト値の並び替え順が使用されます。

MAX()	項目の最大値を返します。次に例を示します。
	`SELECT Name, MAX(BudgetedCost)`
	`FROM Campaign`
	`GROUP BY Name`
	API バージョン 18.0 以降で使用できます。

SUM()	数値項目の合計を返します。次に例を示します。
	`SELECT SUM(Amount)`
	`FROM Opportunity`
	`WHERE IsClosed = false AND Probability > 60`

集計関数を使用して GROUP BY 句を使用しないクエリでは LIMIT 句を使用できません。たとえば、次のクエリは無効です。

```
SELECT MAX(CreatedDate)
FROM Account LIMIT 1
```

COUNT() および COUNT(*fieldName*)

クエリが返す行数を取得するには、集計関数 COUNT() を SOQL クエリの SELECT ステートメントで使用します。

COUNT()

COUNT() は絞り込み条件に一致する行数を返します。

次に例を示します。

```
SELECT COUNT()
FROM Account
WHERE Name LIKE 'a%'
```

```
SELECT COUNT()
FROM Contact, Contact.Account
WHERE Account.Name = 'MyriadPubs'
```

COUNT() の場合、QueryResult オブジェクトの size 項目は、クエリで取得された行数を返します。records 項目は null を返します。

COUNT() を使用するときは、次の点に注意してください。

- COUNT() は SELECT リストの唯一の要素であることが必要です。
- COUNT() が返す行数には、クエリの絞り込み条件に一致する null 値が含まれます。
- COUNT() と LIMIT 句を一緒に使用できます。
- COUNT() と ORDER BY 句は一緒に使用できません。

COUNT(*fieldName*)

COUNT(***fieldName***) は絞り込み条件に一致し、*fieldName* の値が null 以外の行数を返します。

次に例を示します。

```
SELECT COUNT(Id)
FROM Account
WHERE Name LIKE 'a%'
```

メモ: SOQL の COUNT() と COUNT(Id) は、SQL の COUNT(*) に相当します。

COUNT(***fieldName***) の場合、records 項目の オブジェクトが行数を返します。size 項目にはこの数が反映されません。次に例を示します。

```
SELECT COUNT(Id)
FROM Account
WHERE Name LIKE 'a%'
```

この数は、集計項目の暗黙的別名の `expr0` で返されます。

`SELECT` 句には複数の `COUNT(fieldName)` 項目を含めることができます。たとえば次のクエリは、商談の数およびキャンペーンに関連付けられている商談の数を返します。

```
SELECT COUNT(Id), COUNT(CampaignId)
FROM Opportunity
```

`GROUP BY` 句と `COUNT(fieldName)` を一緒に使用して、レコードを分析し、サマリーレポート情報を返すことができます。たとえば、次のクエリは各 `LeadSource` の値のリード数を返します。

```
SELECT LeadSource, COUNT(Name)
FROM Lead
GROUP BY LeadSource
```

関連トピック:

Apex リファレンスガイド: countQuery(query)

集計関数のデータ型のサポート

SOQL クエリで集計関数を使用すると、レコードを分析する強力な手段となりますが、すべてのデータ型に有効なわけではありません。たとえば、base64項目では、集計関数を使用しても意味のあるデータは生成されないため、集計関数はサポートされません。

集計関数は、複数のプリミティブデータ型とデータ型でサポートされます。集計関数でサポートされるプリミティブデータ型を次の表に示します。

データ型	AVG()	COUNT()	COUNT_DISTINCT()	MIN()	MAX()	SUM()
base64	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
boolean	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
byte	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
date	いいえ	はい	はい	はい	はい	いいえ
dateTime	いいえ	はい	はい	はい	はい	いいえ
double	はい	はい	はい	はい	はい	はい
int	はい	はい	はい	はい	はい	はい
string	いいえ	はい	はい	はい	はい	いいえ
time	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ

プリミティブデータ型に加えて、API では、オブジェクト項目の拡張されたデータ型が使用されます。集計関数でサポートされるこれらのデータ型を次の表に示します。

データ型	AVG()	COUNT()	COUNT_DISTINCT()	MIN()	MAX()	SUM()
address	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
anyType	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
calculated	データ型によって異なる*	データ型によって異なる*	データ型によって異なる*	データ型によって異なる*	データ型によって異なる*	データ型によって異なる*
combobox	いいえ	はい	はい	はい	はい	いいえ
currency**	はい	はい	はい	はい	はい	はい
DataCategoryGroupReference	いいえ	はい	はい	はい	はい	いいえ
email	いいえ	はい	はい	はい	はい	いいえ
encryptedstring	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
location	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
ID	いいえ	はい	はい	はい	はい	いいえ
masterrecord	いいえ	はい	はい	はい	はい	いいえ
multipicklist	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
percent	はい	はい	はい	はい	はい	はい
phone	いいえ	はい	はい	はい	はい	いいえ
picklist	いいえ	はい	はい	はい	はい	いいえ
reference	いいえ	はい	はい	はい	はい	いいえ
textarea	いいえ	はい	はい	はい	はい	いいえ
url	いいえ	はい	はい	はい	はい	いいえ

* calculated 項目は、他の項目、式、または値から値を取得するアルゴリズムである数式で定義されるカスタム項目です。そのため、集計関数のサポートは、calculated 項目の種別によって異なります。

** currency 項目の集計関数の結果は、デフォルトのシステムの通貨に設定されます。

ヒント: 一部のオブジェクト項目には、グループ化がサポートされないデータ型があります。GROUP BY 句には、これらのデータ型を使用する項目を含めることができません。

convertCurrency()

通貨項目をユーザーの通貨に変換するには、SOQL クエリの SELECT ステートメントで convertCurrency() を使用します。このアクションを使用するには、組織でマルチ通貨が有効化されている必要があります。

SELECT 句で convertCurrency() を使用するには、次の構文を使用します。

```
convertCurrency(field)
```

次に例を示します。

```
SELECT Id, convertCurrency(AnnualRevenue)
FROM Account
```

ユーザーの地域に従って通貨の書式を設定するには、SELECT() ステートメントで FORMAT() を使用します。この例では、convertedCurrency は返される金額の別名です。ユーザーインターフェースでは適切な書式で表示されます。

```
SELECT Amount, FORMAT(amount) Amt, convertCurrency(amount) convertedAmount,
FORMAT(convertCurrency(amount)) convertedCurrency FROM Opportunity where id =
'006R00000024gDtIAI'
```

組織で高度な通貨管理を有効にしている場合は、商談、商談品目名、商談履歴の通貨項目を変換するときに期間指定換算レートが使用されます。高度な通貨管理では、convertCurrency は特定の項目([CloseDate on opportunities] など)に対応する換算レートを使用します。高度な通貨管理が有効になっていない場合は、入力された最新の換算日が使用されます。

考慮事項と回避策

convertCurrency() 関数は WHERE 句では使用できません。使用すると、エラーが返されます。組織の有効な通貨からユーザーの通貨に数値を換算するには、次の構文を使用します。

```
WHERE Object_name Operator ISO_CODEvalue
```

次に例を示します。

```
SELECT Id, Name
FROM Opportunity
WHERE Amount > USD5000
```

この例では、レコードの通貨の Amount 値が USD 5000 相当より大きい場合、商談レコードが返されます。たとえば、金額が USD5001 の商談は返されますが JPY7000 の商談は返されません。

組織で有効になっている、アクティブな ISO コードを使用してください。ISO コードを入力しないと、比較金額の代わりに数値が使用されます。前の例を使用すると、JPY5001、EUR5001、USD5001 の商談レコードが返されます。WHERE 句で IN を使用する場合は、ISO コード値と ISO 以外のコード値を混在させて使用できません。

convertCurrency() 関数をコールして集計関数の結果をユーザーの通貨に換算することはできません。クエリに GROUP BY 句または HAVING 句が含まれる場合、SUM() や MAX() などの集計関数を使用して返される通貨データは、組織のデフォルトの通貨で表示されます。

次に例を示します。

```
SELECT Name, MAX(Amount)
FROM Opportunity
GROUP BY Name
HAVING MAX(Amount) > 10000
```

集計関数を使用するときには、USD などの特定の通貨で値を表すために *ISO_CODEvalue* を使用することはできません。たとえば、次のクエリは機能しません。

```
SELECT Name, MAX(Amount)
FROM Opportunity
GROUP BY Name
HAVING MAX(Amount) > USD10000
```

convertCurrency() と ORDER BY は一緒に使用できません。順序は、レポートの場合と同様に、必ず換算後の通貨の値に基づきます。

convertTimezone()

クライアントアプリケーションの SOQL クエリは、dateTime 項目の値を協定世界時 (UTC) の値で返します。dateTime 項目をユーザーのタイムゾーンに変換するには、日付関数で convertTimezone() を使用します。

たとえば、convertTimezone(***dateTimeField***) 関数を使用して、1 時間ごとのすべての商談の Amount の値の合計を確認できます。この場合の時間は、ユーザーのタイムゾーンに変換されます。

```
SELECT HOUR_IN_DAY(convertTimezone(CreatedDate)), SUM(Amount)
FROM Opportunity
GROUP BY HOUR_IN_DAY(convertTimezone(CreatedDate))
```

日付関数では convertTimezone() のみを使用できます。次のクエリは、日付関数がないため機能しません。

```
SELECT convertTimezone(CreatedDate)
FROM Opportunity
```

日付関数

SOQL クエリで日付関数を使用すると、日、カレンダー月、会計年度などの期間によってデータのグループ化または絞り込みができます。

たとえば、CALENDAR_YEAR() 関数を使用して、各カレンダー年のすべての商談の［金額］の合計値を調べることができます。

```
SELECT CALENDAR_YEAR(CreatedDate), SUM(Amount)
FROM Opportunity
GROUP BY CALENDAR_YEAR(CreatedDate)
```

メモ: クライアントアプリケーションの SOQL クエリは、dateTime 項目の値を協定世界時 (UTC) の値で返します。dateTime 項目の値をデフォルトのタイムゾーンに変換するには、「convertTimezone()」を参照してください。

SOQL でサポートされるすべての日付関数を次の表に示します。

日付関数	説明	例
CALENDAR_MONTH()	date 項目のカレンダー月を表す数値を返します。	• 1 月は 1 • 12 月は 12

日付関数	説明	例
CALENDAR_QUARTER()	date 項目の四半期を表す数値を返します。	• 1 は 1 月 1 日～ 3 月 31 日 • 2 は 4 月 1 日～ 6 月 30 日 • 3 は 7 月 1 日～ 9 月 30 日 • 4 は 10 月 1 日～ 12 月 31 日
CALENDAR_YEAR()	date 項目のカレンダー年を表す数値を返します。	2009
DAY_IN_MONTH()	日付項目の月の何日目かを表す数値を返します。	2 月 20 日の場合は 20
DAY_IN_WEEK()	date 項目の曜日を表す数値を返します。	• 日曜日は 1 • 土曜日は 7
DAY_IN_YEAR()	date 項目の年の何日目かを表す数値を返します。	2 月 1 日の場合は 32
DAY_ONLY()	dateTime 項目の日付部分を表す日付を返します。	2009 年 9 月 22 日は 2009-09-22 DAY_ONLY() は dateTime 項目でのみ使用できます。
FISCAL_MONTH()	date 項目の会計年度の月を表す数値を返します。これは、組織でグレゴリオ暦と一致しない会計年度が使用されている場合は、CALENDAR_MONTH() と異なります。 この関数は、組織がカスタム会計年度を有効にしている場合には、サポートされません。	会計年度の期首月が 3 月とすると • 3 月の場合は 1 • 2 月の場合は 12
FISCAL_QUARTER()	date項目の会計年度の四半期を表す数値を返します。これは、組織でグレゴリオ暦と一致しない会計年度が使用されている場合は、CALENDAR_QUARTER() と異なります。 この関数は、組織がカスタム会計年度を有効にしている場合には、サポートされません。	会計年度の期首月が 7 月とすると • 7 月 15 日の場合は 1 • 6 月 6 日の場合は 4
FISCAL_YEAR()	date項目の会計年度を表す数値を返します。これは、組織でグレゴリオ暦と一致しない会計年度が使用されている場合は、CALENDAR_YEAR() と異なります。 この関数は、組織がカスタム会計年度を有効にしている場合には、サポートされません。	2009

日付関数	説明	例
HOUR_IN_DAY()	dateTime 項目の時間を表す数値を返します。	18:23:10 の場合は 18 HOUR_IN_DAY() は dateTime 項目でのみ使用できます。
WEEK_IN_MONTH()	date 項目の月の何週目かを表す数値を返します。	4 月 10 日の場合は 2 第 1 週は月の 1 日～ 7 日です。
WEEK_IN_YEAR()	date 項目の年の何週目かを表す数値を返します。	1 月 3 日の場合は 1 第 1 週は 1 月 1 日～ 1 月 7 日です。

日付関数を使用する場合、次の点に注意してください。

- クエリに GROUP BY 句が含まれていない場合でも、WHERE 句で日付関数を使用して結果を絞り込むことができます。次のクエリは 2009 年のデータを返します。

```
SELECT CreatedDate, Amount
FROM Opportunity
WHERE CALENDAR_YEAR(CreatedDate) = 2009
```

- 日付関数の結果を WHERE 句の日付リテラルと比較することはできません。次のクエリは機能しません。

```
SELECT CreatedDate, Amount
FROM Opportunity
WHERE CALENDAR_YEAR(CreatedDate) = THIS_YEAR
```

- GROUP BY 句にも含めない限り、SELECT 句で日付関数を使用することはできません。日付関数で使用される項目が date 項目の場合は例外があります。GROUP BY 句では日付関数の代わりに、date 項目を使用できます。これは、dateTime 項目に対しては機能しません。次のクエリは、CALENDAR_YEAR(CreatedDate) が GROUP BY 句に含まれていないため機能しません。

```
SELECT CALENDAR_YEAR(CreatedDate), Amount
FROM Opportunity
```

次のクエリは、CloseDate という date 項目が GROUP BY 句に含まれているため機能します。これが、CreatedDate などの dateTime 項目の場合には機能しません。

```
SELECT CALENDAR_YEAR(CloseDate)
FROM Opportunity
GROUP BY CALENDAR_YEAR(CloseDate)
```

FORMAT ()

SELECT 句で FORMAT を使用して、標準およびカスタムの数値、日付、時間、通貨項目にローカライズされた書式を適用できます。

FORMAT 関数が適用されると、これらの項目に特定のユーザーロケールに適した書式が反映されます。項目書式は、Salesforce Classic のユーザーインターフェースの表示と同じになります。たとえば、2015 年 12 月 28 日は、

組織のロケール設定に応じて 2015-12-28、28-12-2015、28/12/2015、12/28/2015、または 28.12.2015 の数値で表示されます。

組織でマルチ通貨が有効化されている場合の例を次に示します。

```
SELECT FORMAT(amount) Amt, format(lastModifiedDate) editDate FROM Opportunity
editDate = "7/2/2015 3:11 AM"
Amt = "AED 1,500.000000 (USD 1,000.00)"
```

FORMAT 関数では別名指定がサポートされます。さらに、クエリに同じ項目が複数回含まれるときは、別名指定が必要です。次に例を示します。

```
SELECT Id, LastModifiedDate, FORMAT(LastModifiedDate) formattedDate FROM Account
```

集計関数または convertCurrency() 関数でネストすることもできます。次に例を示します。

```
SELECT amount, FORMAT(amount) Amt, convertCurrency(amount) editDate,
FORMAT(convertCurrency(amount)) convertedCurrency FROM Opportunity where id = '12345'
SELECT FORMAT(MIN(closedate)) Amt FROM opportunity
```

GROUPING(*fieldName*)

GROUPING(***fieldName***) 関数は SELECT 句、HAVING 句、および ORDER BY 句で使用できます。

SOQL クエリで GROUP BY ROLLUP または GROUP BY CUBE を使用する場合、GROUPING(***fieldName***) 関数を使用して、行が小計か項目かを判別できます。

GROUPING(***fieldName***) は、その行が項目の小計の場合は 1、それ以外の場合は 0 を返します。

```
SELECT LeadSource, Rating,
    GROUPING(LeadSource) grpLS, GROUPING(Rating) grpRating,
    COUNT(Name) cnt
FROM Lead
GROUP BY ROLLUP(LeadSource, Rating)
```

詳細は、「GROUP BY による小計の識別」を参照してください。

toLabel()

toLabel 関数を使用して、SOQL クエリの結果をクエリ送信者の言語に翻訳します。

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。

toLabel() 関数は、どの組織でも使用できます。

toLabel では、次の構文を使用します。

```
toLabel(object.field)
```

この関数を使用すると、次の検索結果が翻訳されて返されます。

- 関連する describe API コールによって返される値を含む、通常、multiselect、ディビジョン、または通貨コードの選択リスト項目。
- データカテゴリグループとデータカテゴリの一意の名前の項目。

- Recordtype の名前。
- オブジェクト履歴。`HISTORY` は制限付きでサポートされます。追跡項目が変更されると、句 `SELECT toLabel(Field) FROM *History` は、変更された項目の主表示ラベル、または既存の翻訳済み表示ラベルを返します。履歴オブジェクトの OldValue 項目と NewValue 項目も `toLabel` メソッドをサポートしますが、クエリで翻訳済みの値が返されるのは、サポートされる項目種別の変更追跡のみです。

次に例を示します。

```
SELECT Company, toLabel(Recordtype.Name) FROM Lead
```

返されるレコードは、クエリを発行したユーザーの言語に翻訳されます。

メモ: レコードタイプを翻訳された名前の値で絞り込むことはできません。レコードタイプは、常にオブジェクトのマスター値または ID で絞り込みます。

`toLabel()` では、クエリに同じ項目が複数回含まれるときに必要となる別名指定がサポートされています。次に例を示します。

```
SELECT Company, Status, toLabel(Status) translatedStatus FROM Lead
```

`toLabel()` メソッドを使用して、翻訳された選択リスト値を使用するレコードを絞り込めます。次に例を示します。

```
SELECT Company, toLabel(Status)
FROM Lead
WHERE toLabel(Status) = 'le Draft'
```

このクエリでは、Status の選択リスト値が「le Draft」のリードレコードが返されます。ユーザーの言語での値が比較されます。選択リストの翻訳がない場合は、マスター値に対して比較が実行されます。

翻訳された選択リスト値には、次の制限事項が適用されます。

- LIKE 演算子がクエリを実行できるのは、その API 名ではなく、選択リストの表示ラベルのみです。
- `toLabel()` 関数は、演算子 ORDER BY では使用できません。SOQL では、選択リストの定義された順序が常に使用されます。
- ディビジョンまたは通貨の ISO コード選択リストで、`WHERE` 句に `toLabel()` 関数を使用することはできません。

リレーションクエリ

クライアントアプリケーションは、一度に複数のオブジェクト種別へのクエリを実行できる必要があります。これらのタイプのクエリをサポートするために、SOQL は、標準オブジェクトとカスタムオブジェクトに対して、*リレーションクエリ*と呼ばれる構文を提供します。リレーションクエリは、結果を絞り込んで返すために、オブジェクト間の親-子のリレーションおよび子-親のリレーションをトラバースします。

リレーションクエリは、SQL 結合に似ています。ただし、任意の SQL 結合を実行することはできません。このセクションの残り部分で定義されるように、SOQL のリレーションクエリは、有効なリレーションをトラバースする必要があります。

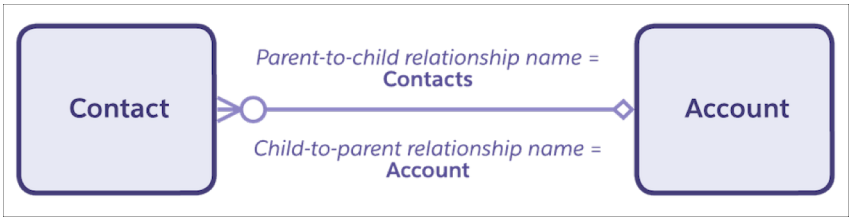
たとえば、リレーションクエリを使用すると、ある種別のオブジェクトを別の種別のオブジェクトに適用される条件に基づいて返すことができます。たとえば、「Bob Jones によって作成されたすべての取引先と、その取引先に関連付けられた取引先責任者を返す」ことができます。オブジェクトを接続している親-子または子-親

リレーションがなければなりません。「Bob Jonesによって作成された取引先とユーザーをすべて返す」のような、任意のクエリを記述することはできません。

このあとは、以下のようなトピックを取り上げ、SOQL のリレーションクエリの概要と使用方法について解説します。

リレーション名について

親-子および子-親のリレーションは、多くのオブジェクト種別の間に存在しています。たとえば、取引先は取引先責任者の親オブジェクトです。



標準オブジェクトではこれらのリレーションをトラバースできるようにするために、各リレーションにリレーション名が与えられています。リレーションの方向に応じて、名前の形式は次のように異なります。

- 子-親リレーションにとって、親へのリレーション名は外部キーの名前であり、親オブジェクトへの参照を保持する `relationshipName` プロパティがあります。たとえば、Contact 子オブジェクトは、Account オブジェクトへの子-親リレーションを持っていて、Contact 内の `relationshipName` の値は、`Account` です。これらのリレーションは、たとえば次のように、クエリ内でドット表記法を使用して親を指定することによってトラバースされます。

  ```
  SELECT Contact.FirstName, Contact.Account.Name from Contact
  ```

  このクエリは、組織内のすべての取引先責任者の名 (ファーストネーム) と、各取引先責任者に対して関連付けられた取引先 (親) の取引先名を返します。

- 親-子リレーションに関しては、親オブジェクトは、親に固有の子リレーションのための名前 (子オブジェクト名の複数個) を持っています。たとえば、Account は他のオブジェクトの間に Assets、Cases、および Contact への子リレーションを持っていて、それぞれには、`Assets`、`Cases`、および `Contacts` という

relationshipName があります。これらのリレーションは、ネスト化したSOQLクエリを使用して、SELECT 句内のみをトラバースすることができます。次に例を示します。

```
SELECT Account.Name, (SELECT Contact.FirstName, Contact.LastName FROM Account.Contacts)
 FROM Account
```

このクエリは、すべての取引先を返し、各取引先に対して関連付けられた取引先責任者 (子) の姓と名を返します。

警告: そのリレーションの方向のための正しい命名規則と SELECT 構文を使用する必要があります。組織のWSDLまたは describeSObjects() を使ったリレーション名の特定方法については、「親子リレーションの識別」を参照してください。リレーションの方向によって、リレーションクエリに制限があります。詳細は、「リレーションクエリ制限について」を参照してください。

リレーション名は、SELECT 構文が同じでも、カスタムオブジェクトに関しては異なります。詳細は、「親子リレーションの識別」を参照してください。

リレーションクエリの使用

SOQL を使用して、複数のリレーション種別を照会できます。

子-親リレーションの照会

子-親リレーション (多くの場合、多対一) を照会する場合は、ドット (.) 演算子を使用します。SELECT 句、FROM 句、または WHERE 句内で、これらのリレーションを直接指定してください。

```
SELECT Id, Name, Account.Name
FROM Contact
WHERE Account.Industry = 'media'
```

このクエリは、関連付けられた取引先の業種が「メディア」である取引先責任者に関してのみ、ID と名前を返し、そして返された各取引先責任者に関しては取引先名を返します。

親-子リレーションの照会

親-子リレーション (多くの場合は一対多) を照会する場合は、括弧で囲まれたサブクエリを使用します。サブクエリ内の FROM 句の初期メンバーが外部クエリの FROM 句の初期メンバーである場合、サブクエリ内でこれらのリレーションを指定してください。標準オブジェクトのサブクエリでは、リレーション名は子オブジェクトの複数形の名前になります。

API バージョン 58.0 以降では、SOQL のクエリに最大 5 レベルの親-子リレーションを含めることができます。5 レベルの親-子リレーションを照会する機能は、標準およびカスタムオブジェクトに対する REST および SOAP クエリコールを介した SOQL クエリでのみ使用できます。

各リレーションで、親はクエリの第 1 レベルと見なされ、子リレーションは親ルートから最大 4 レベルの深さになります。SOQL クエリに 6 レベル以上の親-子リレーションが含まれると、エラーが発生します。

この SOQL クエリの例には、5 レベルの親-子リレーションが含まれています。

```
SELECT Name,
    (SELECT LastName,
```

```
        (SELECT AssetLevel,
            (SELECT Description,
                (SELECT LineItemNumber FROM WorkOrderLineItems)
            FROM WorkOrders)
        FROM Assets)
    FROM Contacts)
FROM Account
```

サブクエリに WHERE 句を含めると、クエリの親ルートから到達できる現在の範囲内の任意のオブジェクトで、親リレーション経由で絞り込みが実行されます。

このクエリの例では、業種がメディアであるすべての取引先の名前が返されます。返された取引先ごとに、作成者の別名が「x」のすべての取引先責任者の姓が返されます。

```
SELECT Name,
  (
    SELECT LastName
    FROM Contacts
    WHERE CreatedBy.Alias = 'x')
 FROM Account WHERE Industry = 'media'
```

親-子リレーションをトラバースする SOQL クエリのバージョン管理された動作については、「リレーションクエリ制限について」を参照してください。

リレーションクエリのトラバース

親-子リレーションと子-親リレーションの両方をトラバースするリレーションクエリの例をいくつかご紹介します。

この例のクエリでは、組織のすべての取引先の名前が返されます。また取引先ごとに、取引先の各メモを作成したユーザーの名前が返されます。取引先のメモがない場合は、結果セットには何も含まれません。

```
SELECT Name,
  (
    SELECT CreatedBy.Name
    FROM Notes
  )
FROM Account
```

以下は、親-子リレーションと子-親リレーションをトラバースするもう 1 つのクエリの例です。

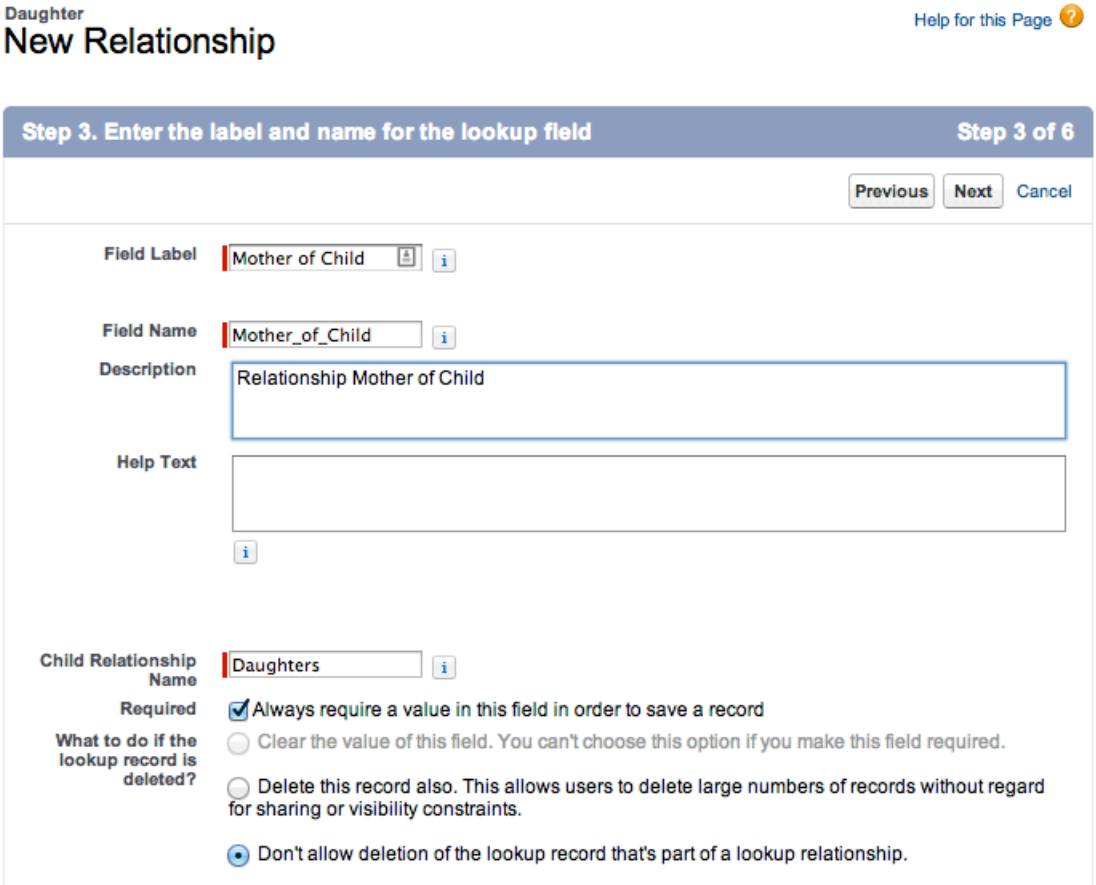
```
SELECT Amount, Id, Name, (SELECT Quantity, ListPrice,
  PriceBookEntry.UnitPrice, PricebookEntry.Name,
  PricebookEntry.product2.Family FROM OpportunityLineItems)
  FROM Opportunity
```

リレーション名、カスタムオブジェクトおよびカスタム項目について

カスタムオブジェクトは、リレーションクエリに含めることができます。Salesforce では、カスタムオブジェクト名と同じ名前の標準オブジェクトが現在または将来に利用可能である場合でも、カスタムオブジェクト名、カスタム項目名、およびそれらと関連付けられたリレーション名が常に一意であることが保証されるようになっています。オブジェクト、項目、およびリレーション名を使用するリレーションをクエリでトラバースする場合、リレーションクエリが一意であることが重要です。

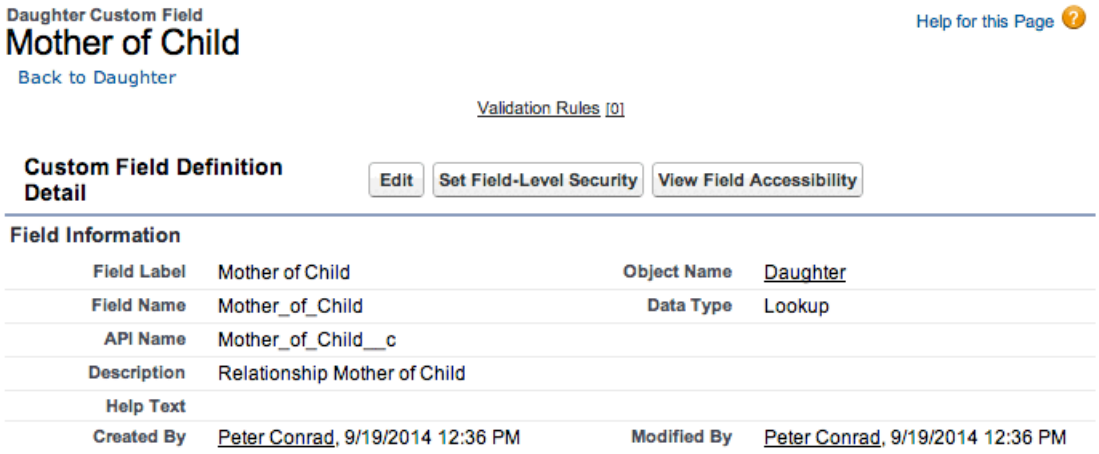
このトピックでは、どのようにカスタムオブジェクトとカスタム項目のリレーション名が作成され、使用されるかを説明します。

Salesforce ユーザーインターフェース内で新しいカスタムリレーションを作成する場合、オブジェクト名の (リレーションクエリのために使用する) 複数のバージョンを指定するよう要求されます。



[子リレーション名] (親-子) が子オブジェクト名の複数形になります (この場合は、Daughters)。

リレーションが作成されると、作成したカスタム項目の名前に __c (アンダースコア 2 つの後に c) が追加された [API 参照名] 参照名が割り当てられます。


Daughter Custom Field
Mother of Child
Back to Daughter
Help for this Page
Validation Rules [0]
Custom Field Definition Detail
Edit | Set Field-Level Security | View Field Accessibility
Field Information

Field Label	Mother of Child	Object Name	Daughter
Field Name	Mother_of_Child	Data Type	Lookup
API Name	Mother_of_Child__c		
Description	Relationship Mother of Child		
Help Text			
Created By	Peter Conrad, 9/19/2014 12:36 PM	Modified By	Peter Conrad, 9/19/2014 12:36 PM

この項目を API 経由で参照する場合には、この特殊な形式の名前を使用する必要があります。これによって、Salesforce がカスタム項目と同名の標準オブジェクトを作成可能な場合に、あいまいさを防止できます。同様のプロセスは、カスタムオブジェクトにもあてはまります。カスタムオブジェクトが作成される場合、[API 名] 参照名 (__c が追加されたオブジェクト名) が割り当てられ、それを使用する必要があります。

クエリ内のリレーション名を使用する場合には、__c ではなくリレーション名を使用する必要があります。代わりに、__r (アンダースコア 2 つの後に r) を追加してください。

次に例を示します。

- 子-親リレーションを使用する場合には、次のようにドット表記法を使用できます。

```
SELECT Id, FirstName__c, Mother_of_Child__r.FirstName__c
FROM Daughter__c
WHERE Mother_of_Child__r.LastName__c LIKE 'C%'
```

  このクエリは、親 (Mother) オブジェクトの姓が「C」から始まる場合、子 (Daughter) オブジェクトの ID、名と、親 (Mother) オブジェクトの名の値を返します。

- 親-子リレーションクエリは、次のようにドット表記法を使用しません。

```
SELECT LastName__c,
  (
    SELECT LastName__c
    FROM Daughters__r
  )
FROM Mother__c
```

  上の例では、すべての親 (Mother) オブジェクトの姓と子 (Daughter) オブジェクトの姓を返します。

クエリ結果について

クエリ結果は、ネスト化されたオブジェクトとして返されます。SOQL クエリのメイン SELECT ステートメントで処理される主なオブジェクトは、サブクエリのクエリ結果を含みます。

たとえば、次のように親-子または子-親構文のいずれかを使用して、クエリを作成できます。

- 子-親:

```
SELECT Id, FirstName, LastName, AccountId, Account.Name
FROM Contact
WHERE Account.Name LIKE 'Acme%'
```

このクエリは、WHERE 句の条件を満たすすべての取引先責任者に関して、(返されるレコードが多すぎない場合) 1 行ごとに 1 つのクエリ結果を返します。

- 親-子:

```
SELECT Id, Name,
  (
    SELECT Id, FirstName, LastName
    FROM Contacts
  )
FROM Account
  WHERE Name like 'Acme%'
```

このクエリは、取引先のセットを返します。そして、各取引先内では、サブクエリからの取引先責任者情報を含む Contact 項目のクエリ結果セットを返します。

次のサンプルは、サブクエリ結果を処理する方法を示しています。

```
private void querySample() {
  QueryResult qr = null;
  try {
    qr = connection.query("SELECT a.Id, a.Name, " +
      "(SELECT c.Id, c.FirstName, " +
      "c.LastName FROM a.Contacts c) FROM Account a");
    boolean done = false;
    if (qr.getSize() > 0) {
      while (!done) {
        for (int i = 0; i < qr.getRecords().length; i++) {
        Account acct = (Account) qr.getRecords()[i];
        String name = acct.getName();
        System.out.println("Account " + (i + 1) + ": " + name);
        printContacts(acct.getContacts());
        }
        if (qr.isDone()) {
          done = true;
        } else {
          qr = connection.queryMore(qr.getQueryLocator());
        }
      }
    } else {
      System.out.println("No records found.");
    }
    System.out.println("\nQuery succesfully executed.");
  } catch (ConnectionException ce) {
    System.out.println("\nFailed to execute query successfully, error message " +
    "was: \n" + ce.getMessage());
  }
}
```

```
private void printContacts(QueryResult qr) throws ConnectionException {
  boolean done = false;
  if (qr.getSize() > 0) {
    while (!done) {
    for (int i = 0; i < qr.getRecords().length; i++) {
      Contact contact = (Contact) qr.getRecords()[i];
      String fName = contact.getFirstName();
      String lName = contact.getLastName();
      System.out.println("Child contact " + (i + 1) + ": " + lName
      + ", " + fName);
    }
    if (qr.isDone()) {
      done = true;
    } else {
      qr = connection.queryMore(qr.getQueryLocator());
    }
    }
  } else {
    System.out.println("No child contacts found.");
  }
}
```

参照関係と外部結合での null 値

リレーション SOQL クエリは、外部結合の場合と同様に、関連付けられた外部キー項目が null 値であってもレコードを返します。

- ORDER BY 句でレコードの外部キーの値が null の場合でも、レコードが返されます。次に例を示します。

  ```
  SELECT Id, CaseNumber, Account.Id, Account.Name
  FROM Case
  ORDER BY Account.Name
  ```

  AccountId が空白のケースレコードは、返されます。

  次の例はカスタムオブジェクトを使用します。

  ```
  SELECT ID, Name, Parent__r.id, Parent__r.name
  FROM Child__c
  ORDER BY Parent__r.name
  ```

  このクエリは、子オブジェクトの Id と Name 値、各 Child 内で参照される親オブジェクトの Id と名前を使用し、親の名前で順番を付けます。Parent__r.id または Parent__r.name が null の場合でも、レコードを返します。

- OR を使用する WHERE 句では、レコードの外部キーの値が null の場合でも、レコードが返されます。たとえば、組織に、LastName 項目の値が Young に一致し、AccountId 項目の値が null の取引先責任者が存在し、かつ、LastName 項目の値が Young とは異なり、Quarry という名前の親取引先を持つ別の取引先責任者が存在するとします。この場合に次のクエリを実行すると、両方の取引先責任者が返されます。

  ```
  SELECT Id FROM Contact WHERE LastName = 'Young' or Account.Name = 'Quarry'
  ```

- 親項目の値をチェックする WHERE 句では、親が存在しない場合でもレコードが返されます。次に例を示します。

```
SELECT Id
FROM Case
WHERE Contact.LastName = null
```

  ケースレコード Id 値が返されます。

- Boolean 項目を使用する WHERE 句では、Boolean 項目に null 値が含まれることはありません。代わりに、null は false として処理されます。外部結合オブジェクトの Boolean 項目は、クエリに一致するレコードがない場合は false として処理されます。

親子リレーションの識別

親子リレーションを識別するには、エンティティリレーションダイアグラム (ERD) を確認するか、組織のエンタープライズ WSDL を調べます。

www.salesforce.com/us/developer/docs/object_reference/index.htm にある『*Salesforce* オブジェクトリファレンス』の「データモデル」セクションの ERD ダイアグラムを確認することによって、親子リレーションを識別できます。ただし、すべての親子リレーションが SOQL 内に表示されるわけではないため、念のため適切な記述用の API コールを発行することによって親子リレーション内でクエリを行うこともできます。結果は、親子リレーション情報を含みます。

組織について Enterprise WSDL を調べることもできます。

- 子リレーション名を見つけるために、子オブジェクトの複数形を含み、type="tns:QueryResult" で終わる項目を探してください。Account の例を次に示します。

```
<complexType name="Account">
   <complexContent>
     <extension base="ens:sObject">
       <sequence>
         ...
         <element name="Contacts" nillable="true" minOccurs="0"
                        type="tns:QueryResult"/>
         ...
       </sequence>
     </extension>
   </complexContent>
 </complexType>
```

  上記の例では、子リレーション名 Contacts は、親の Account のエントリに含まれています。

- オブジェクトの親の場合は、AccountId と Account などのエントリのペアを探します。この場合、ID 項目はその ID によって参照される親オブジェクトを表し、Account 項目はレコードのコンテンツを表します。親エントリには、非プリミティブ型 type="ens:Account" があります。

```
<complexType name="Opportunity">
  <complexContent>
    <extension base="ens:sObject">
      <sequence>
        ...
```

```
                <element name="Account" nillable="true" minOccurs="0"
                        type="ens:Account"/>
                <element name="AccountId" nillable="true" minOccurs="0"
                        type="tns:ID"/>
                ...
              </sequence>
            </extension>
          </complexContent>
        </complexType>
```

メモ: すべてのリレーションがAPIで公開されるわけではありません。リレーションを識別する最も確実な方法は、`describeSObjects()` コールを実行することです。AJAX Toolkit を使用すると、テストコールをすばやく実行できます。

- カスタムオブジェクトの場合は、リレーションのサフィックス `__r` が使用されているエントリのペアを探します。

```
<complexType name="Mother__c">
 <complexContent>
  <extension base="ens:sObject">
   <sequence>
     ...
     <element name="Daughters__r" nillable="true" minOccurs="0"
           type="tns:QueryResult"/>
     <element name="FirstName__c" nillable="true" minOccurs="0"
           type="xsd:string"/>
     <element name="LastName__c" nillable="true" minOccurs="0"
           type="xsd:string"/>
     ...
    </sequence>
   </extension>
  </complexContent>
  </complexType>
```

```
<complexType name="Daughter__c">
 <complexContent>
  <extension base="ens:sObject">
   <sequence>
     ...
     <element name="Mother_of_Child__c" nillable="true" minOccurs="0"
           type="tns:ID"/>
     <element name="Mother_of_Child__r" nillable="true" minOccurs="0"
           type="xsd:string"/>
     <element name="LastName__c" nillable="true" minOccurs="0"
           type="ens:Mother__c"/>
     ...
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

リレーション項目および多態的な項目について

多態的なリレーションでは、リレーションの参照オブジェクトを複数の異なる種別のオブジェクトのいずれかにすることができます。

いくつかの項目は*リレーション項目*で、関連オブジェクトの情報を取得するために使用できます。これらのリレーション項目のいくつかは、*多態的*な項目です。多態的な項目とは、関連オブジェクトを複数の異なる種別のオブジェクトのいずれかにできる項目です。たとえば、Task の `Who` リレーション項目には、Contact または Lead のいずれかを使用できます。

項目の種類を確認するには、オブジェクトに対して `describeSObjects()` をコールし、その項目のプロパティを調べます。

1. `relationshipName` が null でない場合、項目はリレーション項目です。
2. また、`namePointing` が true、`polymorphicForeignKey` が true、`referenceTo` の参照されるオブジェクト種別が複数ある場合、項目は多態的な項目です。

リレーション項目の例

`Account` オブジェクトの `OwnerId` 項目には、次のプロパティがあります。

- `relationshipName` = Owner
- `namePointing` = false
- `referenceTo` = User

これは、リレーション項目であることを示しています。`relationshipName` は、関連オブジェクトに関する情報を取得するための擬似項目として使用します。この種別は、`referenceTo` 項目で指定されます。たとえば、次の SOQL クエリを使用できます。

```
SELECT Id, Owner.Name FROM Account
```

メモ: *項目*の名前 (`OwnerId`) ではなく、*リレーション*の名前 (`Owner`) を使用してください。

多態的な項目の例

`Event` オブジェクトの `OwnerId` 項目には、次のプロパティがあります。

- `relationshipName` = Owner
- `namePointing` = true
- `polymorphicForeignKey` = true
- `referenceTo` = Calendar, User

これは、多態的な項目であることを示しています。`Owner` は、Calendar または User にすることができます。たとえば、次の SOQL クエリを使用できます。

```
SELECT Id, Owner.Name FROM Event WHERE Owner.Type = 'User'
```

多態的な項目の使用

多態的な項目は、複数の方法で使用できます。

- リレーションの多態的な項目を使用する。
- 多態的な項目で `Type` 修飾子を使用する。

- クエリで TYPEOF 句を使用する。

複雑なクエリの場合は、これらの方法を組み合わせることもできます。

よく使用される多態的な項目

よく使用される多態的な項目には、次のものがあります。

- Owner: この項目は、レコードの親を表します。次に例を示します。

```
SELECT Id, Owner.Name
FROM Task
WHERE Owner.FirstName like 'B%'
```

このクエリ例は、Calendar または User のいずれかを所有者とする ToDo レコードに対して機能します。

- Who: この項目は、レコードに関連付けられている個人を表します。次に例を示します。

```
SELECT Id, Who.FirstName, Who.LastName
FROM Task
WHERE Owner.FirstName LIKE 'B%'
```

このクエリ例は、所有者が Calendar または User のいずれかで、Who 項目が Contact または Lead のいずれかである ToDo レコードに対して機能します。

クエリで返されるオブジェクトの種別を確認するには、Who.Type を使用します。次に例を示します。

```
SELECT Id, Who.Id, Who.Type
FROM Task
```

この例を使用して、Contact に関連付けられたすべての ToDo を照会できます。

```
SELECT Id, Who.Id, Who.Type
FROM Task
WHERE Who.Type='Contact'
```

- What: この項目は、レコードに関連付けられている、人以外のオブジェクトを表します。次に例を示します。

```
SELECT Id, What.Name
FROM Event
```

このクエリ例は、What が Account もしくは Solution、または別の番号の任意のオブジェクト種別になる場合がある Event に対して機能します。

Type 修飾子の使用

Type 修飾子を項目で使用すると、多態的なリレーションの参照対象のオブジェクト種別を判別できます。参照対象のオブジェクト種別に応じてクエリから返される内容を条件に基づいて制御するには、SELECT ステー

トメントの WHERE 句で Type 修飾子を使用します。次の SELECT ステートメントでは、Type が使用され、Event の What 項目に基づいてクエリが絞り込まれます。

```
SELECT Id
FROM Event
WHERE What.Type IN ('Account', 'Opportunity')
```

実行時に、この SELECT ステートメントは What 項目で Account または Opportunity を参照する Event の ID を返します。Event が What 項目で Campaign を参照している場合は、この SELECT の一部としては返されません。TYPEOF 式とは異なり、オブジェクト種別は文字列として Type から返されます。オブジェクト種別の文字列には、= (次の文字列と一致する) や LIKE など、任意の WHERE 比較演算子を適用できます。

TYPEOF の使用

SOQL では、SELECT ステートメントで TYPEOF 式を使用して、多態的なリレーションをサポートします。TYPEOF は、API バージョン 46.0 以降で使用できます。(これは、開発者プレビューの一部として API バージョン 26.0 以降で使用することもできます。)

多態的なリレーションの各オブジェクト種別に対して照会する項目を制御するには、SELECT ステートメントで TYPEOF を使用します。次の SELECT ステートメントは、Event の多態的な What リレーション項目に関連付けられたオブジェクト種別に応じて、さまざまな項目のセットを返します。

```
SELECT
  TYPEOF What
    WHEN Account THEN Phone, NumberOfEmployees
    WHEN Opportunity THEN Amount, CloseDate
    ELSE Name, Email
  END
FROM Event
```

実行時に、この SELECT ステートメントは、Event の What 項目で参照されるオブジェクト種別を確認します。オブジェクト種別が Account の場合、参照対象の Account の Phone 項目と NumberOfEmployee 項目が返されます。オブジェクト種別が Opportunity の場合、参照対象の Opportunity の Amount 項目と CloseDate 項目が返されます。オブジェクト種別がこれら以外の種別である場合、Name 項目と Email 項目が返されます。ELSE 句が指定されていなくて、オブジェクト種別が Account でも Opportunity でもない場合、その Event には null が返されます。

TYPEOF の考慮事項について、次の点を注意してください。

- namePointing 属性が false のリレーションで TYPEOF を使用することはできません。
- relationshipName 属性が false のリレーションで TYPEOF を使用することはできません。
- TYPEOF は、クエリの SELECT 句でのみ使用できます。WHERE 句で Type 修飾子を使用して、多態的なリレーションのオブジェクト種別を絞り込むことができます。詳細は、「多態的なリレーション項目の絞り込み」を参照してください。
- TYPEOF は、COUNT() など、オブジェクトを返さないクエリでは使用できません。
- TYPEOF は、ストリーミング API PushTopic のベースの SOQL クエリでは使用できません。
- TYPEOF は、Bulk API で使用される SOQL では使用できません。
- TYPEOF 式はネストできません。たとえば、TYPEOF 式の WHEN 句内で別の TYPEOF を使用することはできません。

- TYPEOF は、準結合クエリの SELECT 句では使用できません。準結合クエリを含む外側のクエリの SELECT 句では TYPEOF を使用できます。次の例は、TYPEOF が準結合クエリで使用されているため無効です。

```
SELECT Name FROM Account
WHERE CreatedById IN
    (
    SELECT
        TYPEOF Owner
            WHEN User THEN Id
            WHEN Group THEN CreatedById
        END
    FROM CASE
    )
```

TYPEOF が外側の SELECT 句のみで使用されているため、次の例は有効です。

```
SELECT
    TYPEOF What
        WHEN Account THEN Phone
        ELSE Name
    END
FROM Event
WHERE CreatedById IN
    (
    SELECT CreatedById
    FROM Case
    )
```

- TYPEOF は、SELECT 句で関数を含むクエリでは使用できません。次の例は、TYPEOF に FORMAT 関数が含まれているため無効です。

```
SELECT
 TYPEOF What
  WHEN Account THEN Id, FORMAT(LastModifiedDate) LastModifiedDate__f
  WHEN Oppty THEN Id
 END
FROM Task
```

代わりに、同じクエリを関数を使わずに実行して ID のリストを取得します。

```
SELECT
    TYPEOF What
        WHEN Account THEN Id, LastModifiedDate
        WHEN Opportunity THEN Id
    END
FROM Task
```

次に、結果の ID リストに対して、関数を使用する 2 番目のクエリを実行します。

```
SELECT
    FORMAT(LastModifiedDate) LastModifiedDate__f
FROM Account
WHERE Id in RetrievedIdList
```

- TYPEOF は、GROUP BY、GROUP BY ROLLUP、GROUP BY CUBE、および HAVING を含むクエリでは使用できません。

TYPEOF と Type の併用

SELECT ステートメントでは、TYPEOF と Type を組み合わせて使用できます。次の SELECT ステートメントでは、TYPEOF と Type の両方が使用され、Event の What 項目に基づいてクエリが絞り込まれ、返された項目セットがさらに絞り込まれます。

```
SELECT Id,
  TYPEOF What
    WHEN Account THEN Phone
    WHEN Opportunity THEN Amount
  END
FROM Event
WHERE What.Type IN ('Account', 'Opportunity')
```

実行時に、この SELECT ステートメントは必ず Event の ID を返し、次に Event の What 項目で参照されるオブジェクト種別に応じて Account.Phone または Opportunity.Amount のいずれかを返します。ELSE 句は指定されていません。このステートメントは、WHERE 句の What 項目に基づいて絞り込みを行うため、Account か Opportunity のいずれかを参照する Event のみが返されます。そのため、ELSE 句は必要ありません。この場合に ELSE 句が含まれていると、実行時に無視されます。

WSDL のオブジェクト種別

Enterprise および Tooling API WSDL の場合、多態的な項目のオブジェクト種別は、API のバージョンによって異なります。

1. API バージョン 46.0 以降 (および SOQL Polymorphism 機能の開発者プレビュー部分が有効であるバージョン) の場合、オブジェクト種別は sObject です。次に例を示します。

```
<complexType name="Task">
  <complexContent>
    <extension base="ens:sObject">
      <sequence>
      ...
      <element name="Owner" nillable="true" minOccurs="0" type="ens:sObject"/>
      ...
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

2. その他のバージョンの場合、種別は Name です。次に例を示します。

```
<complexType name="Task">
  <complexContent>
    <extension base="ens:sObject">
      <sequence>
      ...
      <element name="Owner" nillable="true" minOccurs="0" type="ens:Name"/>
      ...
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

これは、これらの WSDL から生成された Java コードに影響を与えます。たとえば、`Task.java` の場合、Owner 項目は、現在次のように定義されています。

```
private com.sforce.soap.enterprise.sobject.SObject Owner;
```

WSDL ファイルから Java コードを生成する方法についての詳細は、「Java 開発者環境の設定」を参照してください。

関連トピック:

SOQL および SOSL リファレンス: リレーションクエリ制限について

Apex リファレンスガイド: SOQL クエリの多態的なリレーションの処理

Apex リファレンスガイド: getSObject(field)

リレーションクエリ制限について

SOQL リレーションクエリを設計する場合、複数の制限を考慮する必要があります。

- リレーションクエリは SQL 結合と同じではありません。SOQL 内で結合を作成するためには、オブジェクト間のリレーションを持つ必要があります。
- 1 回のクエリに指定できる子-親リレーションは、55 個以下です。カスタムオブジェクトには最大 40 個のリレーションが許可されているため、1 回のクエリでカスタムオブジェクトのすべての子-親リレーションを参照できます。
- 多態的な項目への 1 回のクエリは、子から親へのリレーションの制限に対して複数回反映される可能性があります。たとえば、次のクエリには制限の対象となる 3 つのリレーションがあります。

  ```
  SELECT
    TYPEOF What
      WHEN Account THEN Phone, NumberOfEmployees
      WHEN Opportunity THEN Amount, CloseDate
      ELSE Name, Email
    END
  FROM Event
  ```

  `What`、`Account`、`Opportunity` はこの制限の対象となります。

  次のクエリでは、ID を使用して 1 つのレコードを指定しているため、制限の対象となるリレーションは 1 つです。指定されたレコードのみが照会されます。

  ```
  SELECT
    TYPEOF What
      WHEN Account THEN Phone, NumberOfEmployees
      WHEN Opportunity THEN Amount, CloseDate
      ELSE Name, Email
    END
  FROM Event WHERE ID="someId"
  ```

  クエリで同じリレーションが複数回使用されている場合も、1 つのリレーションとして計数されます。
- 1 回のクエリに指定できる親-子リレーションは、20 個以下です。

- 指定された各リレーションで、1 つの子-親リレーションに指定できるレベルは 5 つ以下です。たとえば、`Contact.Account.Owner.FirstName` は 3 レベルです。
- API バージョン 57.0 以前の場合、1 つのクエリ内で 2 レベルまでの親-子リレーションしか指定できません。
- API バージョン 58.0 以降では、REST および SOAP クエリコールにより、標準およびカスタムオブジェクトの最大 5 レベルの親-子リレーションを照会できます。5 レベルの親-子リレーションについての SOQL クエリは、Big Object、外部オブジェクト、または Apex、Bulk API、Bulk API 2.0 ではサポートされていません。
- 情報を得るために、メモと添付ファイルを照会することは可能ですが、メモと添付ファイルの内容を絞り込むことはできません。オブジェクト内の textarea 項目、BLOB、または Scontrol コンポーネントの内容に対しての絞り込みはできません。たとえば、次のクエリは有効で、取引先と関連付けられたあらゆるメモに関して、すべての取引先名と所有者 ID を返します。

  ```
  SELECT Account.Name, (SELECT Note.OwnerId FROM Account.Notes) FROM Account
  ```

  ただし、次のクエリは、メモの本文に保存された情報の評価を試みているため、有効でありません。

  ```
  SELECT Account.Name, (SELECT Note.Body FROM Account.Notes WHERE Note.Body LIKE 'D%')
  FROM Account
  ```

  `WHERE` 句を削除すると、クエリは有効になり、メモの本文の内容が返されます。

  ```
  SELECT Account.Name, (SELECT Note.Body FROM Account.Notes) FROM Account
  ```

- `USING SCOPE` 句を使用して親-子リレーションクエリの結果を指定の範囲に制限することはできません。
- 外部オブジェクトについては、次の制限を考慮します。
  - 外部オブジェクト、または親の外部オブジェクトの検索条件を含むサブクエリで取得できるデータは、最大 1,000 行です。
  - 各 SOQL クエリ内の結合は、外部オブジェクトとその他の種別のオブジェクト全体で最大 4 個です。

    クエリの実行時、結合ごとに外部システムへの往復処理が必要です。クエリ内の各結合に対して長めの応答時間を想定してください。
  - 外部オブジェクトは、リレーションクエリの `ORDER BY` 句をサポートしません。この制限は、Salesforce Connect の OData 2.0 アダプターを介して外部データにアクセスする場合にのみ適用されます。
  - `SELECT` ステートメントの主オブジェクト (「主導」オブジェクト) が外部オブジェクトの場合、`queryMore()` は主オブジェクトのみをサポートし、サブクエリをサポートしません。

履歴オブジェクトとリレーションクエリの使用

カスタムオブジェクトといくつかの標準オブジェクトは、オブジェクトレコードへの変更を追跡する関連付けられた履歴オブジェクトを持っています。その親オブジェクトに対する履歴オブジェクトをトラバースするために、SOQL リレーションクエリを使用できます。

たとえば、次のクエリは `Foo__c` のすべての履歴行を返し、`Foo` の名前項目とカスタム項目を表示します。

```
SELECT OldValue, NewValue, Parent.Id, Parent.name, Parent.customfield__c
FROM foo__history
```

このクエリ例は、ネスト化されたサブクエリ内の対応する履歴行と共に、すべてのFooオブジェクト行を返します。

```
SELECT Name, customfield__c, (SELECT OldValue, NewValue FROM Histories)
FROM foo__c
```

データカテゴリ選択に関するオブジェクトとリレーションクエリの使用

データカテゴリは、レコードの分類に使用します。SOQLでは、*Article*__DataCategorySelectionまたはQuestionDataCategorySelectionオブジェクトを使用できます。`FROM`句で`DataCategorySelections`リレーション名を使用して、リレーションクエリを作成することもできます。

たとえば、`Offer`記事タイプがあるとします。次のクエリは、Offer(提示)に関連付けられたカテゴリのIDと分類された記事のIDを返します。

```
SELECT Id,ParentId
FROM Offer__DataCategorySelection
```

次の例では、`DataCategorySelections`リレーション名を使用して、公開された提示のIDとこれらの提示に関連付けられているすべてのカテゴリのIDを返すリレーションクエリを作成します。

```
SELECT Id, Title,
  (
    SELECT Id
    FROM DataCategorySelections
  )
FROM Offer__kav WHERE PublishStatus='online'
```

Partner WSDLとリレーションクエリの使用

Enterprise WSDLにはSOQLリレーションクエリに必要な詳細な型情報が含まれているのに対し、Partner WSDLにはそうした情報は含まれていません。最初に`describeSObjects()`コールを実行し、その結果からリレーションクエリを作成するために必要な情報を収集する必要があります。

- 一対多リレーションの`relationshipName`値。たとえば、Accountオブジェクトでは、納入商品の子のリレーション名は`Assets`になります。
- 関連オブジェクトに利用可能な参照項目。たとえばLeadオブジェクト、Caseオブジェクト、またはカスタムオブジェクトの`whoId`、`whatId`、または`ownerId`。

リレーションクエリでPartner WSDLを使用する例は、developer.salesforce.com **の例**を参照してください(ログインが必要です)。

クエリのバッチサイズの変更

クエリ結果で返される行数は、クエリのバッチサイズで決まります。REST APIおよびSOAP APIを使用して実行されたクエリで返されるバッチサイズを変更できます。

クエリ結果のバッチサイズのデフォルトおよび最大値は 2,000 レコードです。ただし、パフォーマンスを最適化するために、返されるバッチサイズは、照会されたレコードのサイズと複雑さに基づいて、最大値またはリクエストで設定された値よりも少なくすることができます。

REST API では、Query Options ヘッダーの `batchSize` 項目を使用して、1 回のクエリで返される結果の件数を変更します。詳細は、『*REST API 開発者ガイド*』の「Query」を参照してください。

SOAP API では、`query()` コールを呼び出す前に、`QueryOptions` ヘッダーのバッチサイズを変更します。Salesforce Web Service Connector (WSC) クライアントでバッチサイズを設定するには、接続オブジェクトに対して `setQueryOptions()` をコールします。詳細は、『*SOAP API 開発者ガイド*』の「query()」を参照してください。

メモ: SOQL ステートメントでロングテキスト型のカスタム項目を 2 つ以上選択した場合、バッチサイズが 200 レコードを超えることはできません。この制限により、大きな SOAP メッセージが返されることが防止されます。

バッチサイズを決定する WSC (Java) の例

次の Java WSC (Java) コードの例では、バッチサイズが 250 レコードに設定されます。

```
public void queryOptionsSample() {
  connection.setQueryOptions(250);
}
```

バッチサイズを決定する C# (.NET) の例

次の C# (.NET) コードの例では、バッチサイズが 250 レコードに設定されます。

```
private void queryOptionsSample()
{
    binding.QueryOptionsValue = new QueryOptions();

    binding.QueryOptionsValue.batchSize = 250;
    binding.QueryOptionsValue.batchSizeSpecified = true;
}
```

オブジェクトに対する SOQL の制限

SOQL では、特定の制限が検索結果のオブジェクトと状況に適用されます。SOQL 制限は、ContentDocumentLink オブジェクト、ContentHubItem オブジェクト、Big Object、外部オブジェクト、NewsFeed、KnowledgeArticleVersion、RecentlyViewed、TopicAssignment、UserRecordAccess、UserProfileFeed、Vote に対して定義されています。

一部のオブジェクトや状況では、SOQL に特定の制限があります。

オブジェクト	説明
ContentDocumentLink	SOQL クエリの絞り込みでは、`Id`、`ContentDocumentId`、`LinkedEntityId` のいずれかを条件にする必要があります。

オブジェクト	説明
ContentHubItem	SOQL クエリの絞り込みでは、Id、ExternalId、ContentHubRepositoryId のいずれかを条件にする必要があります。

カスタムメタデータ型	カスタムメタデータ型は、次の SOQL クエリ構文をサポートします。
	`SELECT fieldList [...]`
	`FROM objectType`
	`[USING SCOPE filterScope]`
	`[WHERE conditionExpression]`
	`[ORDER BY field {ASC|DESC} [NULLS {FIRST|LAST}] ]`
	• *fieldList* および *conditionExpression* にメタデータリレーション項目を使用できます。
	• FROM に追加できるのは 1 つのオブジェクトのみです。
	• 次の演算子を使用できます。
	– IN および NOT IN
	– =、>、>=、<、<=、および !=
	– LIKE (ワイルドカードを含む)
	– AND
	– LIKE 演算子および = 演算子がある同じ列にある場合は OR
	メモ: 子検索条件が異なる 2 つの列にある場合、OR を複合検索条件として使用することはできません。
	• リレーション以外の項目にのみ ORDER BY を使用できます。
	• リレーション以外の複数の項目で ORDER BY、ASC、および DESC を使用できます。
	• メタデータリレーション項目では、すべての標準リレーションクエリがサポートされています。

Big Object	• SOQL クエリは、Big Object のインデックスに定義された順序で欠落のない項目でのみ絞り込むことができます。 • クエリの最後の項目では次の演算子のみを使用できます。 – =、<、>、<=、>=、および IN – クエリのそれより前の項目では = 演算子のみを使用できます。 • Big Object では、次の演算子はサポートされていません。 – !=、LIKE、NOT IN、EXCLUDES、および INCLUDES
外部オブジェクト	• 外部オブジェクトが含まれるサブクエリが取得できるデータは、最大 1,000 行です。

オブジェクト	説明
	• 各SOQLクエリ内の結合は、外部オブジェクトとその他の種別のオブジェクト全体で最大4個です。 クエリの実行時、結合ごとに外部システムへの往復処理が必要です。クエリ内の各結合に対して長めの応答時間を想定してください。 • 外部オブジェクトでは、次の集計関数と句をサポートしていません。 – `AVG()` 関数 – `COUNT(fieldName)` 関数(ただし、`COUNT()` はサポートされている) – `HAVING` 句 – `GROUP BY` 句 – `MAX()` 関数 – `MIN()` 関数 – `SUM()` 関数 • 外部オブジェクトでは、以下もサポートしていません。 – `EXCLUDES` 演算子 – `FOR VIEW` 句 – `FOR REFERENCE` 句 – `INCLUDES` 演算子 – `LIKE` 演算子 – `toLabel()` 関数 – `TYPEOF` 句 – `WITH` 句 次の制限は、Salesforce Connect の OData 2.0 および 4.0 アダプターにのみ適用されます。 • 外部オブジェクトの場合、`ORDER BY` 句に次の制限があります。 – `NULLS FIRST` と `NULLS LAST` は無視されます。 – 外部オブジェクトは、リレーションクエリの `ORDER BY` 句をサポートしません。 • `COUNT()` 集計関数は、外部データソースで［要求の行数］が有効になっている外部オブジェクトでのみサポートされます。特に、外部システムからの応答には、結果セットの行の合計数を含める必要があります。 次の制限は、Salesforce Connect のカスタムアダプターにのみ適用されます。 • 外部オブジェクトの位置情報に基づくSOQLクエリはサポートされていません。 • 外部オブジェクトのSOQLクエリに次の要素が含まれている場合、クエリは失敗します。

オブジェクト	説明
	- `convertCurrency()` 関数 - `UPDATE TRACKING` 句 - `UPDATE VIEWSTAT` 句 - `USING SCOPE` 句 • `ORDER BY` 句の次の構文は無視されます。 - `NULLS FIRST` 構文 - `NULLS LAST` 構文 • Apex テストでは、動的 SOQL を使用して外部オブジェクトを照会します。外部オブジェクトの静的 SOQL クエリを実行するテストは失敗します。

KnowledgeArticleVersion	• 最良の結果を得るには、`PublishStatus` の 1 つの値で絞り込みます。各記事のすべてのバージョンを検索するには、`PublishStatus` フィルターを省略し、1 つ以上のプライマリキー ID で絞り込みます。特定の記事のすべてのアーカイブバージョンを取得するには、SOQL フィルターで `IsLatestVersion` を `false` に設定します。
	• API バージョン 46.0 以前の場合、デフォルトでは、クエリの実行時に `PublishStatus` で絞り込まないと、公開記事が返されます。API バージョン 47.0 以降の場合、Lightning Knowledge が有効のときは、ドラフト記事、公開記事、およびアーカイブ済み記事が返されます。
	• セキュリティをサポートするために、`PublishStatus` 値が `Draft` の記事は「ドラフト記事の表示」権限を持つユーザーにのみ表示されます。同様に、`PublishStatus` 値が `Archived` の記事は「アーカイブ済み記事の表示」権限を持つユーザーにのみ表示されます。
	• アーカイブ済み記事のバージョンは、***`Knowledge`***`__kav` オブジェクトに保存されます。アーカイブ済み記事のバージョンを照会するには、記事の `Id` を指定し、`IsLatestVersion='0'` を設定します。
	• KnowledgeArticleVersion オブジェクトでは、Apex SOQL ステートメントでバインド変数を使用できません。たとえば、次の SOQL ステートメントでは、コンパイルエラーが発生します。
	`final String PUBLISH_STATUS_ONLINE = 'Online';`
	`List<Knowledge__kav> articles = [`
	`SELECT Id FROM Knowledge__kav`
	`WHERE PublishStatus = :PUBLISH_STATUS_ONLINE`
	`];`
	代わりに、次の動的 SOQL を使用します。『*Apex 開発者ガイド*』の「動的 SOQL」を参照してください。
	`final String PUBLISH_STATUS_ONLINE = 'Online';`
	`final String q = 'SELECT Id, PublishStatus FROM`
	`Knowledge__kav`

オブジェクト	説明
	`WHERE PublishStatus = :PUBLISH_STATUS_ONLINE';` `List<Knowledge__kav> articles = Database.query(q);`
NewsFeed	• ログインしたユーザーに「すべてのデータの参照」権限がある場合、SOQL の制限はありません。この権限がない場合は、LIMIT 句に 1,000 レコード以下を指定してください。 • リレーションを使用する項目に対して SOQL ORDER BY は使用できません。SOQL クエリでは、ORDER BY はルートオブジェクトの項目に対して使用してください。
RecentlyViewed	RecentlyViewed オブジェクトは、ログインユーザーがレコードを表示または参照するたびに更新されます。また、SOQL クエリで `FOR VIEW` または `FOR REFERENCE` 句を使用してレコードを取得した場合にも更新されます。最新のデータを確実に使用できるようにするには、1 オブジェクトにつきレコードが 200 件までになるよう、RecentlyViewed データを定期的に切り捨てます。RecentlyViewed データは 90 日間保持され、90 日が経過すると定期的に削除されます。
TopicAssignment	ログインしたユーザーに「すべてのデータの参照」権限がある場合、SOQL の制限はありません。そうでない場合は、次のいずれかの操作を実行します。 • LIMIT 句に 1,100 件以下のレコードを指定する。 • 「=」を指定した `WHERE` 句を使用する場合に、`Id` または `Entity` を絞り込む。
UserRecordAccess	• 必ず『SOAP API 開発者ガイド』で指定されたクエリ形式を使用してください。 • `ORDER BY` 句を含めることができます。`SELECT HasAccess` の場合は `ORDER BY HasAccess`、`SELECT MaxAccessLevel` の場合は `ORDER BY MaxAccessLevel` を使用する必要があります。 • クエリ可能な最大レコード数は 200 件です。
UserProfileFeed	• ログインしたユーザーに「すべてのデータの参照」権限がある場合、SOQL の制限はありません。この権限がない場合は、LIMIT 句に 1,000 レコード以下を指定してください。 • リレーションを使用する項目に対して SOQL ORDER BY は使用できません。SOQL クエリでは、ORDER BY はルートオブジェクトの項目に対して使用してください。 また、SOQL クエリには `WITH UserId = {userId]` を含める必要があります。

オブジェクト	説明
Vote	• `ParentId = [単一の ID]` • `Parent.Type = [単一型]` • `Id = [単一の ID]` • `Id IN = [ID のリスト]`

他の SOQL 制限については、『*Salesforce Developer* の制限および割り当てクイックリファレンス』の「SOQL と SOSL の制限」を参照してください。

Big Object を使用する SOQL

標準の SOQL コマンドのサブセットを使用して、Big Object のインデックスの項目を照会できます。

インデックスクエリを構築するには、インデックスに定義された最初の項目から開始し、欠落が生じないよう最初から最後の項目までクエリに指定します。クエリの項目では = または `IN` を使用できますが、`IN` を使用できるのは 1 回のみです。範囲演算子 <、>、<=、または >= はクエリの最後の項目でのみ使用できます。

ヒント: `FirstName IN('Charlie')` のように引数が 1 つしかない `IN` 句を使用した場合は、`FirstName='Charlie'` のように = を使用した場合と同等です。明確化するために、このケースでは = 形式を使用することをお勧めします。

サブクエリはサポートされていません。クエリに複数のステートメントを含めないでください。たとえば、次のクエリはサポートされません。

```
Select CreatedById, CreatedDate, Created_Date__c, Id, Legacy_Record_ID__c, Parent_Case__c,
 SystemModstamp, Text_Body__c FROM Archived_Email_Message__b WHERE Parent_Case__c IN(select
 id from case where owner.id in ('00580000008BBVUAA4'))
```

システム項目 `CreatedById`、`CreatedDate`、`SystemModstamp` をクエリに含めることができます。

クエリ結果の順序を保証するには、`ORDER BY` 句を使用します。

次のクエリは、テーブル内で `LastName__c`、`FirstName__c`、`PhoneNumber__c` によってインデックスが定義されていると仮定します。

次のクエリは、インデックスに 3 つすべての項目を指定します。このケースでは、`PhoneNumber__c` の検索条件で範囲演算子を使用できます。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c='Kelly' AND FirstName__c='Charlie' AND PhoneNumber__c='2155555555'
```

次のクエリは、インデックスに最初の2つの項目のみを指定します。このケースでは、`FirstName__c` の検索条件で範囲演算子を使用できます。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c='Kelly' AND FirstName__c='Charlie'
```

次のクエリは、インデックスに最初の項目のみを指定します。LastName__c の検索条件で範囲演算子を使用できます。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c='Kelly'
```

次のクエリでは、インデックスの最初の項目で IN 演算子を使用しています。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c IN ('Kelly','Jones','Capulet','Montague') AND FirstName__c='Charlie'
```

次のクエリは、欠落があるため機能しません。FirstName__c が必要です。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c='Kelly' AND PhoneNumber__c='2155555555'
```

次のクエリも IN 演算子を 2 回使用しているため機能しません。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c IN ('Kelly','Jones') AND FirstName__c IN ('Charlie','Lisa')
```

次のクエリは WHERE 句に 2 つの IN 演算子があるように見えますが、機能します。ただし、2 番目の IN には引数が 1 つしかないため、等号演算子と同等であるために使用できます。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c IN ('Kelly','Jones') AND FirstName__c IN ('Charlie')
```

明確化するために、上記の SOQL ステートメントを次のように書き換えることをお勧めします。

```
SELECT LastName__c, FirstName__c, PhoneNumber__c
FROM Phone_Book__b
WHERE LastName__c IN ('Kelly','Jones') AND FirstName__c='Charlie'
```

Big Object で許可されない SOQL 操作

- インデックスクエリを構築するときは、クエリ内の最初の項目から最後の項目までの間に欠落が残らないようにしてください。
- 演算子 !=、LIKE、NOT IN、EXCLUDES、INCLUDES はクエリでは無効です。
- 集計関数は、どのクエリでも無効です。
- 結果のリストを取得するには、クエリに Id 項目は使用しないでください。クエリに Id を含めると、空の ID (000000000000000 または 000000000000000AAA) の結果のみが返されます。

  メモ: 開発者コンソールを使用してリソースからクエリを生成すると、Id 項目が自動的に含まれます。開発者コンソールで Big Object を照会するには、生成されたクエリから Id を削除します。

シンジケーションフィード SOQL と対応付けの構文

シンジケーションフィードサービスでは、アプリケーションでオブジェクトセットと個々のオブジェクトの参照、およびオブジェクト間のリレーションをトラバースすることを可能にする SOQL クエリおよび対応付けの仕様を使用します。データの絞り込みやデータがどのように表示されるかを制御するクエリ文字列パラメーターとして、オプションを追加できます。シンジケーションフィードは、公開サイトに対して定義できます。

クエリフィード定義での SOQL の制限事項についての詳細は、シンジケーションフィードに関する Salesforce オンラインヘルプを参照してください。

位置情報に基づく SOQL クエリ

位置情報に基づく SOQL クエリを使用すると、Salesforce に保存されている位置情報の値を比較したり照合したりできます。倉庫と店舗の間の距離など、2 つの位置情報値の間の距離を計算できます。また、位置情報の値と固定経度・緯度座標の間の距離を計算することもできます。たとえば、倉庫とサンフランシスコが位置する緯度・経度 (37.775°, -122.418°) の間の距離を計算できます。

地理位置情報カスタム項目種別を使用すると、位置情報の値を保存する項目を作成できます。地理位置情報項目では、緯度と経度に基づいて場所を識別します。Salesforce オブジェクトの標準住所にも地理位置情報項目が含まれており、(値が入力されていれば) 同じように使用できます。ただし、いくつかの制限があります。どちらの種別でも、場所を比較したり照合したりできます。たとえば、最も近い 10 件の取引先を見つけることができます。

詳細と考慮事項については、『*Salesforce オブジェクトリファレンス*』の「複合項目」を参照してください。

位置情報に基づく SOQL クエリに対応している項目種別

SOQL では、GEOLOCATION 関数を使用して、緯度と経度に単純な数値を指定できます。これらの値は、標準数値項目、ユーザー入力、計算値などから取得できます。地理位置情報項目の個々のコンポーネントから取得することもできます。この項目では、緯度と経度の両方を 1 つの論理項目に格納しています。地理コードサービスによって標準住所の地理位置情報項目が入力される場合は、住所から直接、緯度と経度の値を使用することもできます。

SOAP API および REST API を使用して作成された SOQL では、地理位置情報コンポーネントを含む住所項目など、地理位置情報項目を、SOQL ステートメント内で直接使用することもできます。これにより多くの場合、SOQL ステートメントが簡素化されます。複合項目は、SOAP API および REST API を使用して作成された SOQL クエリのみで使用できます。

SELECT 句

場所が地理位置情報項目または住所項目に保存されているレコードを、個々の緯度および経度の値として取得するには、通常の「__c」の代わりに「__latitude__s」または「__longitude__s」を項目名に追加します。次に例を示します。

```
SELECT Name, Location__latitude__s, Location__longitude__s
FROM Warehouse__c
```

このクエリは、カスタムオブジェクト Warehouse に保存されているすべての倉庫を検索します。検索結果には、各倉庫の緯度と経度の値が表示されます。緯度および経度のコンポーネントを個々に選択するには、`Location__c` で個別の項目コンポーネントを使用します。

SOAP API および REST API を使用して実行される SOQL は、個々の要素ではなく複合項目を選択できます。複合項目は、プリミティブ値ではなく構造化データとして返されます。次に例を示します。

```
SELECT Name, Location__c
FROM Warehouse__c
```

このクエリは、前のクエリと同じデータを取得します。ただし *Location__c* 項目は地理位置情報の複合項目であるため、その結果は 2 つのプリミティブ値で構成されます。以下に示すのは、REST API 要求の結果のサンプルです。

```
{
  "totalSize" : 10,
  "done" : true,
  "records" : [ {
    "attributes" : {
      "type" : "Warehouse__c",
      "url" : "/services/data/v30.0/sobjects/Warehouse__c/a06D00000017O4nIAE"
    },
    "Name" : "Ferry Building Depot",
    "Location__c" : {
      "latitude" : 37.79302,
      "longitude" : -122.394507
    }
  }, {
    "attributes" : {
      "type" : "Warehouse__c",
      "url" : "/services/data/v30.0/sobjects/Warehouse__c/a06D00000017O4oIAE"
    },
    "Name" : "Aloha Warehouse",
    "Location__c" : {
      "latitude" : 37.786108,
      "longitude" : -122.430152
    }
  },
  ...
  ]
}
```

WHERE 句

クエリの `WHERE` 句に距離条件を指定し、場所が特定の半径内または半径外にあるレコードを取得します。適切な距離条件を作成するには、以下の関数を使用します。

DISTANCE

2 つの場所の距離をマイル単位またはキロ単位で計算します。

使用方法: `DISTANCE(`***mylocation1***`,` ***mylocation2***`, 'unit')`。*mylocation1* と *mylocation2* に、2 つの場所項目、または 1 つの場所項目と GEOLOCATION 関数で返された値を指定します。*unit* を mi (マイル) または km (キロ) に置き換えます。

メモ: 場所項目と GEOLOCATION 値を使用する場合、場所項目は最初の変数、GEOLOCATION は 2 番目の変数でなければなりません。GEOLOCATION が最初の変数の場合、クエリは `MALFORMED_QUERY` エラーになります。

GEOLOCATION

指定された緯度と経度に基づいて地理位置情報を返します。DISTANCE 関数と併用する必要があります。

使用方法: `GEOLOCATION(latitude, longitude)`。*latitude* と *longitude* に、対応する地理位置情報の数値コード値を指定します。

2 つの項目値を比較するか、1 つの項目値と固定の場所を比較します。次に例を示します。

```
SELECT Name, Location__c
FROM Warehouse__c
WHERE DISTANCE(Location__c, GEOLOCATION(37.775,-122.418), 'mi') < 20
```

ORDER BY 句

`ORDER BY` 句で距離条件を使用して、距離を基準にレコードを並べ変えます。次に例を示します。

```
SELECT Name, StreetAddress__c
FROM Warehouse__c
WHERE DISTANCE(Location__c, GEOLOCATION(37.775,-122.418), 'mi') < 20
ORDER BY DISTANCE(Location__c, GEOLOCATION(37.775,-122.418), 'mi')
LIMIT 10
```

このクエリは、カスタムオブジェクト Warehouse にある倉庫のうち、地理位置情報 37.775°, -122.418° (サンフランシスコ) から 20 マイル以内にある倉庫を最大 10 件見つけます。結果には、各倉庫の名前と住所が含まれますが、地理座標は含まれません。最も近い倉庫がリストの一番上に表示され、最も遠い倉庫が一番下に表示されます。

SOQL が Null の位置情報の値を処理する方法

地理位置情報項目は、地球上のある地点を示すために緯度と経度の値を組み合わせる複合項目です。null 値は、緯度と経度の*両方*が null の場合にのみ有効です。

地理位置情報項目値が無効なレコードでは、緯度と経度の両方の値が、SOQL の `WHERE DISTANCE()` と `ORDER BY` の句で使用された場合に Null として処理されます。緯度または経度のどちらかが Null である地理位置情報項目を持つレコードは、項目が設定されていないものとして処理されます。

複合地理位置情報項目を `SELECT` 句で使用する場合、無効な地理位置情報の値には null が返されます。次に例を示します。

```
SELECT Name, Location__c
FROM Warehouse__c
LIMIT 5
```

この表にある値は API コールから返されます。

名前	Location__c
Ferry Building Depot	*null*

名前	Location__c
Aloha Warehouse	(37.786108,-122.430152)
Big Tech Warehouse	*null*
S H Frank & Company	*null*
San Francisco Tech Mart	(37.77587,-122.399902)

この結果には、地理位置情報の値として 3 つの null が含まれています。どの値が実際に null で、どの値が無効なデータであるかを判断することはできません。

同じ地理位置情報項目の個別の項目コンポーネントを SELECT 句で使用すると、これまでと同様に保存された値が返されます。null 以外の場合はその値で返され、null の場合は null 値として返されます。次に例を示します。

```
SELECT Name, Location__latitude__s, Location__longitude__s
FROM Warehouse__c
LIMIT 5
```

これらの値はクエリ結果のサンプルです。

名前	Location__latitude__s	Location__longitude__s
Ferry Building Depot	*null*	-122.394507
Aloha Warehouse	37.786108	-122.430152
Big Tech Warehouse	*null*	*null*
S H Frank & Company	37.763662	*null*
San Francisco Tech Mart	37.77587	-122.399902

この結果では、実際に null であるのは 1 つの地理位置情報項目のみです。他の 2 つは部分的に null であるため、無効です。

DISTANCE の計算に使用する数式項目を作成するときは、[空白項目の処理] セクションの [空白項目を空白として処理] を選択します。[空白項目を 0 として処理] を選択した場合、地理位置情報項目に Null 値があると、赤道がグリニッジ子午線と交差する位置 (0°,0°) を基準に距離が計算されます。レコード詳細ページでは、[空白項目を 0 として処理] に設定されている DISTANCE 数式項目に Null 値の地理位置情報の値があると、数式項目が空白になります。

SOQL による距離の計算と比較の方法

DISTANCE 関数は、半正矢 (大円) の距離計算を 0.0002% 以内に近似します。この数式では、地球は完全な球体だと仮定していますが、地球は実際には楕円形、つまり不規則な球体です。この仮定の誤差は、赤道を横断するときに最大 0.55% になる可能性があります。ただし緯度と進行方向にもよりますが、通常は 0.3% 以下です。

顧客の現在の場所に最も近い 10 件の店舗を計算する場合は、DISTANCE 関数を使用しても問題ありません。ただし、この関数に基づいて、サンフランシスコからシドニーまでのフライトの飛行機に燃料を補給するのはやめましょう。

また、この近似は地理位置情報と距離に「等しい」という概念がないことを意味します。位置情報や距離が等しいかどうかは確認することができません。ある場所が別の場所より遠いか近いか、またはある距離が別の距離より長いか短いかしか判断できません。2 つの場所が「同じ」であることを確認するには、その距離を浮動小数点数として扱い、その差異と許容値を比較します。たとえば、この WHERE 句は、testLocation から 25 フィート以内にある別のレコードを検索します。

```
WHERE ( DISTANCE(Location__c, testLocation) < 0.05 )
```

距離がほぼ同じ場合は小さな誤差ですみますが、誤差によって、期待した場所が位置情報クエリの結果に含まれたり、含まれなかったりすることがあります。アプリケーションで正確な距離の計算と比較を行う必要がある場合は、自身で計算することをお勧めします。

位置情報に基づく SOQL クエリの考慮事項

位置情報に基づくクエリは、Apex の SOQL と、SOAP API および REST API でサポートされています。次の考慮事項に留意します。

- DISTANCE と GEOLOCATION は、SOQL の WHERE 句および ORDER BY 句ではサポートされますが、GROUP BY 句ではサポートされません。DISTANCE は SELECT 句でサポートされます。
- DISTANCE は、論理演算子 > および < のみをサポートし、指定された半径内 (<) または外 (>) の値を返します。
- SOQL クエリで GEOLOCATION 関数を使用する場合、地理位置情報項目を緯度および経度座標よりも前に入力する必要があります。たとえば、`DISTANCE(warehouse_location__c, GEOLOCATION(37.775,-122.418), 'km')` は機能しますが、`DISTANCE(GEOLOCATION(37.775,-122.418), warehouse_location__c, 'km')` は機能しません。
- Apex バインド変数は、DISTANCE 関数の単位パラメーターではサポートされません。次のクエリは機能しません。

```
String units = 'mi';
List<Account> accountList =
    [SELECT ID, Name, BillingLatitude, BillingLongitude
     FROM Account
     WHERE DISTANCE(My_Location_Field__c, GEOLOCATION(10,10), :units) < 10];
```

詳細は、『*Salesforce オブジェクトリファレンス*』に記載されている「**複合項目の考慮事項と制限**」を参照してください。

第 3 章　Salesforce Object Search Language (SOSL)

トピック:

Salesforce Object Search Language (SOSL) を使用して、検索インデックスにテキストベースの検索クエリを作成できます。

効率的な SOSL クエリを作成する際には、選択性の高い条件を作成します。検索は、インデックス内のすべてのレコードを分析し、デフォルトでは、関連性に基づいて上位 2,000 件の一致したレコードのみを返します。この API では、さらなる一致を取得できるようにページ分割がサポートされています。共有は、結果セットが検索スタックから返された後に適用されます。検索条件が選択的でなく、検索語が 2,000 を超えるレコードと一致してしまう場合には、検索ヒット数が上限を超える可能性があります。

次の環境では、カスタムオブジェクトも含め、アクセス権のある複数のオブジェクトのテキスト、メール、電話項目を 1 つのクエリで検索できます。

- SOAP API search() コール。 『*SOAP API 開発者ガイド*』の「search()」を参照してください。
- REST API 検索コール。 『*REST API 開発者ガイド*』の「検索」を参照してください。
- Apex ステートメント。 『*Apex 開発者ガイド*』の「SOQL および SOSL クエリ」を参照してください。
- Visualforce コントローラーと getter メソッド。 『*Visualforce 開発者ガイド*』の「コントローラーメソッド」を参照してください。
- Eclipse Toolkit の Schema Explorer。 『*Force.com IDE 開発者ガイド*』の「Schema Explorer」を参照してください。

メモ: 組織でリレーションクエリが有効になっている場合、SOSL で SOQL リレーションクエリがサポートされます。

SOSL を使用するケース

データがどのオブジェクトまたは項目に存在しているかを認識しておらず、次の操作を行う場合、SOSL を使用します。

- 項目内に存在することがわかっている、特定用語のデータを取得する。SOSL では項目内の複数の用語をトークン化して、そこから検索インデックスを構築できるため、SOSL 検索でより速く、より多くの関連結果を返すことができます。
- 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取得する。
- 中国語、日本語、韓国語、タイ語のデータを取得する。CJKT 用語の形態的トークン化によって、結果の正確性が確保されます。

パフォーマンス上の考慮事項

検索内容が一般的すぎると、検索に時間がかかり膨大な結果が返されます。次の句を使用して、効率的なテキスト検索を定義します。

- `IN`: メール、名前、電話など、検索する項目の種別を制限します。
- `LIMIT`: 返す行の最大数を指定します。
- `OFFSET`: 検索結果を複数のページに表示します。
- `RETURNING`: 返すオブジェクトと項目を制限します。
- `WITH DATA CATEGORY`: 返すデータカテゴリを指定します。
- `WITH DivisionFilter`: 返すディビジョン項目を指定します。
- `WITH NETWORK`: 返す Experience Cloud サイト ID を指定します。
- `WITH PricebookId`: 返す価格表 ID を指定します。

このドキュメント内の移動

- 使用可能なリソースの一覧を表示するには、「SOSL の構文」を参照してください。
- SOSL の使用を開始するには、「テキスト検索の例」を参照してください。

このドキュメントの表記規則。

この SOSL リファレンスでは、特定の表記規則を使用します。

次の表記規則を使用します。

規則	説明
`FIND Name IN Account`	Courierフォントは表示どおりに入力する必要がある項目を示します。構文ステートメントでも、Courier フォントは、この表で説明する中括弧、角括弧、省略記号、その他の表記マーカーを除き、表示どおりに入力する必要がある項目を示します。
`FIND` ***`fieldname`*** `IN` ***`objectname`***	斜体は、変数またはプレースホルダーを表します。実際の値を入力してください。
`|`	パイプ文字は、選択肢となる要素を区切ります。たとえば、`UPDATE TRACKING|VIEWSTAT[,...]` 句の場合、`|` 文字は `UPDATE` の後に `TRACKING` または `VIEWSTAT` のどちらかを使用できることを示します。
`[LIMIT n]`	角括弧は、省略可能な要素を示します。たとえば、`[LIMIT n]` は、`LIMIT` 句を指定できることを示します。SOSL コマンドの一部として角括弧を入力しないでください。角括弧のネストは、要素が省略可能であることを示し、省略可能な親要素が存在する場合にのみ使用できます。たとえば、`[ORDER BY fieldname [ASC | DESC] [NULLS {FIRST | LAST}]]` 句の場合、`ASC`、`DESC`、または `NULLS` 句を使用するには `ORDER BY` 句が必要です。

`[...]` および `[,...]`	角括弧で囲まれた省略記号は、その前にある要素をその要素に設定されている上限まで繰り返せることを示します。カンマも含まれる場合は、繰り返す要素をカンマで区切る必要があります。要素が中括弧でグループ化された選択肢のリストである場合、リストの項目を任意の順序で使用できます。たとえば、`[ [ [ UPDATE [TRACKING|VIEWSTAT][,...]]` 句の場合、`[,...]` は、`TRACKING`、`VIEWSTAT`、またはその両方を使用できることを示します。
	`UPDATE TRACKING`
	`UPDATE VIEWSTAT`
	`UPDATE TRACKING, VIEWSTAT`

検索結果に対する SOSL の制限

検索エンジンは、検索プロセスの各フェーズで分析するレコード数を制限します。この制限のために、一致するレコードがユーザーの結果から除外される場合があります。

次の図は、検索エンジンがどのように SOSL 検索を処理し、結果を制限しているかを示しています。各色はオブジェクトを表し、各雨滴はレコードを表します。数字は次のフローに対応しています。

1. 検索エンジンは、最大 2,000 件のレコード内で検索語との一致を探します (この制限は API バージョン 28.0 以降に適用されます)。
2. SOSL は、特定のオブジェクトまたは状況に異なる制限を適用します。単一オブジェクトを検索する場合、完全なレコード制限が適用されます。複数のオブジェクトでのグローバル検索の場合、各オブジェクトには合計 2,000 レコードの制限が個別に適用されます。
3. システム管理者 (「すべてのデータの参照」権限を持つユーザー) には、返された結果セットのすべてが表示されます。
4. その他すべてのユーザーには、SOSL によってユーザー権限検索条件が適用されます。個々のユーザーには、参照権限を持つレコードのみが表示されます。結果セットと順序は検索を実行するユーザーによって異なり、インデックスからのレコードの追加または削除に応じて 1 日の間に変化する可能性があります。

例: Acme, Inc. の営業担当役員である Joe Smith が、「Industrial Computing」の取引先レコードを検索します。検索バーに「Industrial」と入力します。「Industrial」という検索語に一致するレコードは多数あるため、結果に制限が適用されます。残念ながら、Joe が探しているレコードは制限の枠内にはありませんでした。この概念は、画像ではじょうごの外側にある 1 つの雨滴で示されています。Joe はグローバル検索を使用したため、各オブジェクト種別に 2,000 件のレコード制限が適用されます。この図では、5 つの青い雨滴がじょうごに入っていますが、そのうち 3 つのみが次のフェーズに進んでいます。検索を 1 つのオブジェクトのみに絞り込んでいれば、制限はそのオブジェクトにのみ適用され、探しているレコードが返される確率は高くなります。Joe は「Industrial Computing San Francisco」と入力して検索を再試行します。検索語がより具体的になったため、同じ制限が適用される場合でも検索エンジンはより適切な一致を返すことができます。このシナリオでは、Joe が探しているレコードは、じょうごの最上部から Joe の検索結果ページまで通過している青い雨滴のうちの 1 つです。

他の SOSL 制限については、『*Salesforce Developer の制限および割り当てクイックリファレンス*』の「SOQL と SOSL の制限」を参照してください。

外部オブジェクトの検索結果に対する SOSL の制限

SOSL では、特定の制限が検索結果の外部オブジェクトに適用されます。

- SOSL および Salesforce 検索に外部オブジェクトを含めるには、外部のオブジェクトと外部のデータソースの両方で検索を有効にします。ただし、同期すると外部オブジェクトの検索状況が常に上書きされて、外部のデータソースの検索状況と一致します。
- 検索できるのは、外部オブジェクトのテキスト、テキストエリア、およびロングテキストエリア項目のみです。外部オブジェクトに検索可能項目がない場合、そのオブジェクトに対する検索ではレコードは返されません。
- 外部オブジェクトでは、以下をサポートしていません。
  - `INCLUDES` 演算子
  - `LIKE` 演算子
  - `EXCLUDES` 演算子
  - `toLabel()` 関数
- 外部オブジェクトは、次のような Salesforce ナレッジ固有の句もサポートしていません。
  - `UPDATE TRACKING` 句
  - `UPDATE VIEWSTAT` 句
  - `WITH DATA CATEGORY` 句
- 検索結果に返すには、外部オブジェクトを `RETURNING` 句で明示的に指定する必要があります。次に例を示します。

```
FIND {MyProspect} RETURNING MyExternalObject, MyOtherExternalObject
```

- テキスト文字列は 100 文字以内にする必要があります。

次の制限は、Salesforce Connect の OData 2.0 および 4.0 アダプターにのみ適用されます。

- Salesforce Connect の OData アダプターでは、`FIND` 句で論理演算子をサポートしていません。外部システムには、検索クエリ文字列全体が、ハイフン (-) を除く ASCII の句読文字をすべて削除した後に大文字と小文字が区別される 1 つの句として送信されます。たとえば、`FIND {MyProspect OR "John Smith"}` の場合、「MyProspect OR John Smith」と完全一致する語句が検索されます。

次の制限は、Salesforce Connect のカスタムアダプターにのみ適用されます。

- 外部オブジェクトの SOSL クエリでは、`convertCurrency()` 関数はサポートされていません。
- 外部オブジェクトの SOSL クエリでは、`WITH` 句はサポートされていません。

SOSL の構文

SOSL クエリは、必須の FIND 句で始まります。次に任意の句を追加して、オブジェクト種別、項目、データカテゴリなどによってクエリを絞り込むことができます。返される内容を決定することもできます。たとえば、結果の順序を指定したり、返す行の数を指定したりできます。

必須の FIND 句の後には、1 つ以上の任意の句を次の順序で追加できます。

```
FIND {SearchQuery}
[ IN SearchGroup ]
[ RETURNING FieldSpec [[ toLabel(fields)] [convertCurrency(Amount)] [FORMAT()]] ]
[ WITH DivisionFilter ]
[ WITH DATA CATEGORY DataCategorySpec ]
[ WITH SNIPPET[(target_length=n)] ]
[ WITH NETWORK NetworkIdSpec ]
[ WITH PricebookId ]
[ WITH METADATA ]
[ LIMIT n ]

[ UPDATE [TRACKING], [VIEWSTAT] ]
```

メモ: OFFSET n および WHERE は RETURNING FieldSpec 内に含まれます。

各項目は次のとおりです。

構文	説明
convertCurrency()	省略可能。組織でマルチ通貨が有効になっている場合、通貨項目をユーザーの通貨に換算します。
FIND {SearchQuery}	必須。検索するテキスト(単語または語句)を指定します。検索クエリは中括弧で囲みます。 SearchQuery 文字列が 10,000 字を超えると、結果行は返されません。SearchQuery が 4,000 文字を超えると、論理演算子はすべて削除されます。たとえば、4,001 文字の SearchQuery を含むステートメント内の AND 演算子は、デフォルトの OR 演算子になるため、予想よりも多くの結果が返される場合があります。
FORMAT()	省略可能。FIND 句で FORMAT を使用して、標準およびカスタムの数値、日付、時刻、通貨項目にローカライズされた書式を適用できます。FORMAT 関数では別名指定がサポートされます。さらに、クエリに同じ項目が複数回含まれるときは、別名指定が必要です。
IN SearchGroup	省略可能。検索する項目範囲。次のいずれかの値になります。 • ALL FIELDS • NAME FIELDS

構文	説明
	• EMAIL FIELDS • PHONE FIELDS • SIDEBAR FIELDS 指定されていない場合、デフォルトは ALL FIELDS です。RETURNING *FieldSpec* 句で、検索するオブジェクトのリストを指定できます。 メモ: この句は、記事、ドキュメント、フィードコメント、フィード項目、ファイル、商品、およびソリューションには適用されません。RETURNING句にこれらのオブジェクトが指定されている場合、検索は特定の項目に制限されたうえで、すべての項目が検索されます。
LIMIT *n*	省略可能。テキストクエリで返される行の最大数を2,000以下に指定します。指定されていない場合、返される行の最大数である2,000がデフォルトです。この制限は、APIバージョン28.0以降に適用されます。それ以前のバージョンでは最大200行までがサポートされます。
OFFSET n	省略可能。クエリ結果に大量のレコードが含まれると予想される場合、SOSLクエリに OFFSET 句を使用して結果を複数ページに表示できます。たとえば、OFFSET を使用して51～75番目のレコードを表示した後、スキップして301～350番目のレコードを表示できます。OFFSET を使用すると、大きな結果セットを効率よく処理できます。
ORDER BY	省略可能。ORDER BY 句を使用して、返される検索結果の順序を指定します。また、この句を使用して、結果の先頭または最後に空のレコードを表示することもできます。
RETURNING *FieldSpec*	省略可能。検索結果で返す情報。1つ以上のオブジェクトのリスト、および各オブジェクト内の1つ以上の項目のリスト(省略可能な絞り込み対象の値を含む)。指定されていない場合、検索結果には見つかったすべてのオブジェクトのIDが含まれます。
toLabel(fields)	省略可能。クエリの結果がユーザーの言語に翻訳されて返されます。
USING ListView=	1つの特定のオブジェクトのリストビュー内で検索するために使用される省略可能な句。1つのリストビュー

構文	説明
	のみを指定できます。ユーザーがリストビューに設定した並び替え順に従って、リストビューの最初の2,000レコードのみが検索されます。
`[ UPDATE [TRACKING | VIEWSTAT][,...] ]`	省略可能。組織でSalesforceナレッジを使用する場合、次の処理を行います。 • `UPDATE TRACKING` は、Salesforce ナレッジ記事検索で使用するキーワードを追跡します。 • `Update an Article's Viewstat with SOSL` は、記事の参照統計を更新します。 • `UPDATE TRACKING, VIEWSTAT` は両方を行います。
`WHERE`	省略可能。デフォルトで、オブジェクトに対するSOSLクエリでは、ユーザーに表示されているすべての行が取得されます。検索を限定するために、特定の項目値で検索結果を絞り込みます。
`WITH DATA CATEGORY` ***`DataCategorySpec`***	省略可能。組織でSalesforceナレッジの記事または回答を使用する場合、1つ以上のデータカテゴリに基づいてすべての検索結果を絞り込みます。
`WITH` *`DivisionFilter`*	省略可能。組織でディビジョンを使用する場合、`Division` 項目の値に基づいてすべての検索結果を絞り込みます。
`WITH HIGHLIGHT`	省略可能。特定のオブジェクトの検索結果で、検索語に一致する語を強調表示します。
`WITH METADATA = MetadataSpec`	省略可能。応答でメタデータが返されるかどうかを指定します。デフォルト設定は no であり、メタデータは返されません。
`WITH NETWORK NetworkIdSpec`	省略可能。検索結果をコミュニティ ID で絞り込みます。
`WITH PricebookId`	省略可能。1つの価格表IDで商品検索結果を絞り込みます。
`WITH SNIPPET [(target_length=`*`n`*`)]`	省略可能。組織でSalesforceナレッジ記事を使用する場合、コンテキストスニペットを表示し、検索結果で各記事の検索語を強調表示します。デフォルトでは、各スニペットには最大で約300文字が表示されます。これは、標準のブラウザーウィンドウでは通常3行分のテキストに相当します。対象の長さに別の値 (100 ～

構文	説明
	500 文字) を指定するには、target_length パラメーター (省略可能) を追加します。
WITH SPELL_CORRECTION	省略可能。true に設定すると、スペル修正をサポートする検索のスペル修正が有効になります。false に設定されていると、スペル修正は有効になりません。デフォルト値は true です。

メモ: SOSL ステートメントの文字数制限は、組織で定義されている SOQL ステートメントの文字数制限に関連付けられます。デフォルトでは、SOQL クエリおよび SOSL クエリは 100,000 文字を超えることはできません。この最大長を超える SOSL ステートメントでは、API が MALFORMED_SEARCH 例外コードを返します。結果の行は返されません。

テキスト検索の例

SOSL を使用するテキスト検索の例を次に示します。

システム内で joe を検索します。joe が見つかったレコードの ID を返します。

```
FIND {joe}
```

システム内で大文字と小文字を区別せずに名前 Joe Smith を検索します。Joe Smith が見つかったレコードの ID を返します。

```
FIND {Joe Smith}
```

リードの名前項目で名前 Joe Smith を検索し、見つかったレコードの ID 項目を返します。

```
FIND {Joe Smith}
IN Name Fields
RETURNING lead
```

リードの名前項目で名前 Joe Smith を検索し、見つかったレコードの名前と電話番号を返します。

```
FIND {Joe Smith}
IN Name Fields
RETURNING lead(name, phone)
```

リードの名前項目で名前 Joe Smith を検索します。一致したレコードのうち現在の会計四半期に作成されたレコードの名前と電話番号を返します。

```
FIND {Joe Smith}
IN Name Fields
RETURNING lead (name, phone Where createddate = THIS_FISCAL_QUARTER)
```

リードまたは取引先責任者の名前項目で名前 `Joe Smith` または `Joe Smythe` を検索し、見つかったレコードの名前と電話番号を返します。人レコードの名前が `Joe Smith` または `Joe Smythe` の場合、そのレコードは返されません。

```
FIND {"Joe Smith" OR "Joe Smythe"}
IN Name Fields
RETURNING lead(name, phone), contact(name, phone)
```

ワイルドカード。

```
FIND {Joe Sm*}
FIND {Joe Sm?th*}
```

「and」と「or」が単独で使用される場合はリテラルとして区切ります。

```
FIND {"and" or "or"}
FIND {"joe and mary"}
FIND {in}
FIND {returning}
FIND {find}
```

特殊文字 `&|!(){}[]^"~*?:\'+-` をエスケープします。

```
FIND {right brace \}}
FIND {asterisk \*}
FIND {question \?}
FIND {single quote  \'}
FIND {double quote  \"}
```

メモ: Apex では、使用しているステートメントで SOQL ステートメントや SOSL ステートメントを使うには、角括弧で囲む必要があります。前にコロン (:) がある場合は、Apex スクリプト変数と式を使用できます。

convertCurrency()

通貨項目をユーザーの通貨に変換するには、SOSL クエリに `convertCurrency()` を使用します。このアクションを使用するには、組織でマルチ通貨が有効化されている必要があります。

この構文を `RETURNING` 句で使用します。

`convertCurrency(`***`Amount`***`)`

次に例を示します。

```
FIND {test} RETURNING Opportunity(Name, convertCurrency(Amount))
```

組織で高度な通貨管理を有効にしている場合は、商談、商談品目名、商談履歴の通貨項目を変換するときに期間指定換算レートが使用されます。高度な通貨管理では、`convertCurrency` は特定の項目 (`[CloseDate on opportunities]` など) に対応する換算レートを使用します。高度な通貨管理が有効になっていない場合は、入力された最新の換算日が使用されます。

convertCurrency() 関数は WHERE 句では使用できません。使用すると、エラーが返されます。組織の有効な通貨からユーザーの通貨に数値を換算するには、次の構文を使用します。

```
WHERE Object_name Operator ISO_CODEvalue
```

次に例を示します。

```
FIND {test} IN ALL FIELDS RETURNING Opportunity(Name WHERE Amount>USD5000)
```

この例では、レコードの通貨の Amount 値が USD 5000 相当より大きい場合、商談レコードが返されます。たとえば、金額が USD5001 の商談は返されますが JPY7000 の商談は返されません。

組織で有効になっている、アクティブな ISO コードを使用してください。ISO コードを入力しないと、比較金額の代わりに数値が使用されます。前の例を使用すると、JPY5001、EUR5001、USD5001 の商談レコードが返されます。WHERE 句で IN を使用する場合は、ISO コード値と ISO 以外のコード値を混在させて使用できません。

メモ: 順序は、レポートの場合と同様に、必ず換算後の通貨の値に基づきます。したがって、convertCurrency() と ORDER BY 句は一緒に使用できません。

convertCurrency() 関数では別名指定がサポートされます。さらに、クエリに同じ項目が複数回含まれるときは、別名指定が必要です。次に例を示します。

```
FIND {Acme} RETURNING Account(AnnualRevenue, convertCurrency(AnnualRevenue) AliasCurrency)
```

FIND *{SearchQuery}*

検索する単語または語句を指定するには、SOSL クエリで FIND 句 (必須) を使用します。検索クエリには、リテラルの単語または語句が含まれます。また、ワイルドカードと論理演算子 (AND、OR、AND NOT) も含めることができます。

検索クエリには次が含まれます。

- 検索するリテラルテキスト (単語または語句)
- ワイルドカード (省略可能)
- グルーピングの括弧を含む論理演算子 (省略可能)

検索は左から右に評価され、Unicode (UTF-8) 文字コードを使用します。テキスト検索は大文字と小文字を区別しません。たとえば、Customer、customer、CUSTOMER の検索はすべて同じ結果を返します。

実行時に評価される特殊な種類のテキスト式 (マクロ、関数、正規表現など) は、FIND 句に含めることはできません。

メモ: テキスト検索で検索式を他の句と明確に区別するには、SearchQuery を中括弧で囲みます。

検索語

SearchQuery には、次を含めることができます。

- 単語: test や hello など
- 語句: 二重引用符で囲まれた単語と空白 ("john smith" など)

検索エンジンによって、スペースまたは句読点で区切られたレコード情報が別個のトークンに分割されます。検索エンジンは形態的トークン化を使用して、単語間にスペースを含まない東アジア言語の検索で正確な検索結果を返します。

例: 日本語で語「東京都」のインデックスを作成し、その後で「*京都*」を検索した場合の問題について考えてみましょう。

インデックス	検索
東京都 東京都	京都 京都

形態的トークン化は、語「東京都」を 2 つのトークンに分割します。

東京 東京	都 都

このトークン化形式により、「京都」を検索すると、「京都」を含む結果のみが返され、「東京都」を含む結果は返されません。

ワイルドカード

検索内のテキストパターンと一致させるために、次のワイルドカード文字を指定できます。

ワイルドカード	説明
*	検索語の途中または末尾で、0 個以上の文字の代わりにアスタリスクを使用できます。たとえば、「*太**」を検索すると、「*太一*」、「*太郎*」、「*太次郎*」などの「*太*」で始まるデータが表示されます。ただし、中国語、日本語、韓国語、またはタイ語で検索する場合は、検索語の中間にアスタリスクまたは疑問符のワイルドカードは使用できません。 単語または語句内のリテラルアスタリスクを検索する場合、アスタリスクをエスケープします (\ 文字をそれの前に付けます)。
?	疑問符は、検索語の途中または末尾にある 1 つのみの文字の代わりに使用できます。たとえば、「*jo?n*」を検索すると、「*john*」や「*joan*」を含むデータが表示されます。ただし、中国語、日本語、韓国語、またはタイ語で検索する場合は、検索語の中間にアスタリスクまたは疑問符のワイルドカードは使用できません。また、検索キーワードの先頭にワイルドカードの疑問符を使用しても機能しません。ルックアップ検索では ? は使用できません。

ワイルドカードを使用する場合には、以下の点に注意してください。

- ワイルドカード検索の条件を絞り込むほど、検索結果はより速く返され、期待する結果が返される可能性が高まります。たとえば、単語 `prospect` (または複数形 `prospects`) のすべての発生を検索するには、無

関係の一致(prosperity など)を返す可能性のある制限のより少ないワイルドカード検索(prosp* など)を指定するよりも、検索文字列内で prospect* を指定する方がより効率的です。

- 単語のすべてのバリエーションを見つけるために、検索を調整します。たとえば、property と properties を見つけるには、propert* を指定します。
- 句読点にはインデックスを付けます。語句内で * または ? を見つけるためには、検索文字列を引用符で囲む必要があり、特殊文字をエスケープする必要があります。たとえば、"where are you\?" は、語句 where are you? を見つけます。エスケープ文字(\)は、この検索が正しく機能するために必要です。

演算子

複数の単語を論理演算子およびグルーピング演算子と組み合わせて、より複雑なクエリを作成できます。次の特殊演算子を使用して、テキスト検索を絞り込めます。演算子のサポートでは大文字と小文字を区別しません。

演算子	説明
" "	入力した検索語の順序で一致を見つけるには、検索語を引用符で囲みます。"monday meeting" では、monday meeting をこの順番で含む項目が検索されます。 検索結果に語「and」、「or」、「and not」を含めるには、これらの語を二重引用符で囲みます。囲まないと、それぞれ対応する演算子として解釈されます。
AND	すべての検索語に一致する項目を検索します。たとえば、john AND smith は john と smith の両方の単語を含む項目を検索します。通常、演算子が指定されていない場合は AND がデフォルトの演算子です。記事、ドキュメント、およびソリューションを検索する場合は OR がデフォルトの演算子であるため AND は明示的に指定する必要があります。
OR	検索キーワードを最低 1 つ含むデータを検索します。たとえば、john OR smith は john か smith、またはその両方を含む項目を検索します。
AND NOT	検索語を含まない項目を検索します。たとえば、john AND NOT smith は単語 john を含み、単語 smith を含まない項目を検索します。
()	括弧で囲んだ検索語と論理演算子を使用して、検索語をグループ化します。たとえば、次の検索を実行できます。 • ("Bob" and "Jones") OR ("Sally" and "Smith") — Bob Jones または Sally Smith を検索します。 • ("Bob") and ("Jones" OR "Thomas") and Sally Smith — Bob Jones と Sally Smith、または Bob Thomas と Sally Smith を含むドキュメントを検索します。

SearchQuery の文字制限

SearchQuery 文字列が10,000字を超えると、結果行は返されません。SearchQuery が4,000文字を超えると、論理演算子はすべて削除されます。たとえば、4,001 文字の SearchQuery を含むステートメント内の AND 演算子は、デフォルトの OR 演算子になるため、予想よりも多くの結果が返される場合があります。

複数の演算子を組み合わせて 1 つの検索文字列として指定した場合は、次の順序で計算されます。

1. 括弧
2. AND および AND NOT (右から左に計算されます)
3. OR

予約文字

次の文字が予約されています。

```
? & | ! { } [ ] ( ) ^ ~ * : \ " ' + -
```

テキスト検索で予約文字を指定する場合、予約文字をエスケープして (前にバックスラッシュ \ 文字を付けて) 予約文字が適切に解釈されるようにする必要があります。予約文字の前にバックスラッシュを付けないと、エラーが発生します。これは SearchQuery を二重引用符で囲んだ場合にも該当します。

たとえば、次のテキストを検索するとします。

```
{1+1}:2
```

各予約文字の前にバックスラッシュを挿入します。

```
\{1\+1\}\:2
```

FIND 句の例

検索タイプ	例
単語の例	FIND {MyProspect} FIND {mylogin@mycompany.com} FIND {FIND} FIND {IN} FIND {RETURNING} FIND {LIMIT}
単一語句	FIND {John Smith}
単語 OR 単語	FIND {MyProspect OR MyCompany}
単語 AND 単語	FIND {MyProspect AND MyCompany}
単語 AND 語句	FIND {MyProspect AND "John Smith"}

検索タイプ	例
単語 OR 語句	`FIND {MyProspect OR "John Smith"}`
AND/OR を使用した複雑なクエリ	`FIND {MyProspect AND "John Smith" OR MyCompany}` `FIND {MyProspect AND ("John Smith" OR MyCompany)}`
AND NOT を使用した複雑なクエリ	`FIND {MyProspect AND NOT MyCompany}`
ワイルドカード検索	`FIND {My*}`
エスケープシーケンス	`FIND {Why not\?}`
無効または不完全な語句 (成功しません)	`FIND {"John Smith}`

Apex の FIND 句

Apex の `FIND` 句の構文は、SOAP API および REST API の `FIND` 句の構文と異なります。

- Apex の場合、`FIND` 句の値は単一引用符で区画されます。次に例を示します。

```
FIND 'map*' IN ALL FIELDS RETURNING Account (Id, Name), Contact, Opportunity, Lead
```

  メモ: システムモードで実行される Apex では、`IN ALL FIELDS` を使用して一致をスキャンする際に、項目レベルセキュリティが無視されます。

- API の場合、`FIND` 句の値は中括弧で区画されます。次に例を示します。

```
FIND {map*} IN ALL FIELDS RETURNING Account (Id, Name), Contact, Opportunity, Lead
```

Apex での SOSL と SOQL の使用についての詳細は、*『Apex 開発者ガイド』*を参照してください。

FORMAT()

`FIND` 句で `FORMAT` を使用して、標準およびカスタムの数値、日付、時刻、通貨項目にローカライズされた書式を適用できます。

`FORMAT` 関数が適用されると、これらの項目に特定のユーザーロケールに適した書式が反映されます。項目書式は、Salesforce Classic のユーザーインターフェースの表示と同じになります。たとえば、*2015* 年 *12* 月 *28* 日は、組織のロケール設定に応じて、*2015-12-28*、*28-12-2015*、*28/12/2015*、*12/28/2015*、*28.12.2015* の数値で表示されます。

`FORMAT` 関数では別名指定がサポートされます。さらに、クエリに同じ項目が複数回含まれるときは、別名指定が必要です。次に例を示します。

```
FIND {Acme} RETURNING Account(Id, LastModifiedDate, FORMAT(LastModifiedDate) FormattedDate)
```

集計関数または `convertCurrency()` 関数でネストすることもできます。

```
FIND {Acme} RETURNING Account(AnnualRevenue, FORMAT(convertCurrency(AnnualRevenue))
convertedCurrency)
```

IN *SearchGroup*

SOSL クエリでは、IN *SearchGroup* 句 (省略可能) を使用して検索するテキスト項目の種別を指定できます。*SearchGroup* は、検索の範囲を定義します。たとえば、名前、電子メール、電話番号、サイドバー、またはすべての項目を検索できます。

次の表は、SearchGroup の有効な値を示しています。値を指定しない場合、デフォルトの動作では、検索可能なオブジェクトのすべてのテキスト項目が検索されます。数値項目は検索できません。

メモ: この句は、記事、ドキュメント、フィードコメント、フィード項目、ファイル、商品、およびソリューションには適用されません。RETURNING 句にこれらのオブジェクトが指定されている場合、検索は特定の項目に制限されず、すべての項目が検索されます。

有効な SearchGroup の設定

SearchGroup	説明
`ALL FIELDS`	検索可能なすべての項目を検索します。`IN` 句が指定されていない場合、`ALL FIELDS` がデフォルト設定です。
`EMAIL FIELDS`	メール項目のみを検索します。
`NAME FIELDS`	標準オブジェクトの名前項目のみを検索します。 これらの標準オブジェクトに対して `IN NAME FIELDS` を使用するときは、名前項目に加えて次の項目も検索されます。 • Account: Website、Site、NameLocal • Asset: SerialNumber • Case: SuppliedName、SuppliedCompany、Subject • Contact: AssistantName、FirstNameLocal、LastNameLocal、AccountName • Event: Subject • Lead: Company、CompanyLocal、FirstNameLocal、LastNameLocal • Note: Title • PermissionSet: Label • Report: Description • TagDefinition: NormName • Task: Subject • User: CommunityNickname カスタムオブジェクトでは、「Name Field」として定義された項目が検索されます。標準およびカスタムオブジェクトでは、名前項目は `nameField` プロパティ

SearchGroup	説明
	が `true` に設定されています。(詳細は、DescribeSObjectResult の `fields` パラメーターの Field 配列を参照)。
`PHONE FIELDS`	電話番号項目のみを検索します。
`SIDEBAR FIELDS`	サイドバーのドロップダウンリストに表示される有効なレコードを検索します。アプリケーションでの検索とは異なり、アスタリスク (*) ワイルドカードは検索文字列の最後に追加されません。

`IN` 句は省略可能ですが、検索範囲が制限されるため、検索の効率が向上します。たとえば、メールアドレスのみを検索するには、`IN EMAIL FIELDS` と指定します。

IN 句の例

検索タイプ	例
検索グループなし	`FIND {MyProspect}`
`ALL FIELDS`	`FIND {MyProspect} IN ALL FIELDS`
`EMAIL FIELDS`	`FIND {mylogin@mycompany.com} IN EMAIL FIELDS`
`NAME FIELDS`	`FIND {MyProspect} IN NAME FIELDS`
`PHONE FIELDS`	`FIND {MyProspect} IN PHONE FIELDS`
`SIDEBAR FIELDS`	`FIND {MyProspect} IN SIDEBAR FIELDS`
無効な検索	`FIND {MyProspect} IN Accounts`

LIMIT *n*

テキストクエリで返される最大行数 (2,000 行まで) を指定するには、`LIMIT` 句 (省略可能) を SOSL クエリに追加します。指定しない場合、デフォルトは最大 2,000 件です。

デフォルトの 2,000 件は、API バージョン 28.0 以降で返される行の最大数です。以前のバージョンでは 200 件までの結果が返されます。

個別のオブジェクト、またはクエリ全体に制限を設定できます。

クエリ全体に制限を設定した場合、返されるオブジェクト間で結果が均等に分散されます。たとえば、クエリ全体に 20 件の制限を設定し、個別のオブジェクトには制限を定義しないとします。19 件の結果が取引先で 35 件の結果が取引先責任者の場合、10 件の取引先と 10 件の取引先責任者のみが返されます。

```
FIND {test} RETURNING Account(id), Contact LIMIT 20
```

個別のオブジェクトに制限を設定することで、他のオブジェクトが返される前に1つのオブジェクトが最大クエリ制限を使い切ってしまうことを回避できます。たとえば、次のクエリを発行する場合、返される取引先レコードは最大で 20 件で、残りのレコード数を取引先責任者に割り当てることができます。

```
FIND {test} RETURNING Account(id LIMIT 20), Contact LIMIT 100
```

制限に 0 を指定した場合、そのオブジェクトのレコードは返されません。

OFFSET *n*

クエリ結果に大量のレコードが含まれると予想される場合、SOSL クエリに OFFSET 句を使用して結果を複数ページに表示できます。たとえば、OFFSET を使用して 51 ～ 75 番目のレコードを表示した後、スキップして 301 ～ 350 番目のレコードを表示できます。OFFSET を使用すると、大きな結果セットを効率よく処理できます。

クエリによって返される結果セットへの開始行オフセットを指定するには、OFFSET (省略可能)を使用します。オフセットの計算はサーバーで実行されて結果サブセットのみが返されるため、完全な結果セットを取得して結果をローカルで絞り込むよりも OFFSET を使用した方が効率的です。OFFSET は、1 つのオブジェクトを照会するときにのみ使用できます。OFFSET 句は、クエリの最後に指定する必要があります。OFFSET は、API バージョン 30.0 以降で使用できます。

```
FIND {conditionExpression} RETURNING objectType(fieldList ORDER BY fieldOrderByList
LIMIT number_of_rows_to_return
OFFSET number_of_rows_to_skip)
```

例として、クエリが通常は 50 行を返す場合、クエリで OFFSET 10 を使用して最初の 10 行をスキップできます。

```
FIND {test} RETURNING Account(id LIMIT 10 OFFSET 10)
```

前述の例の結果セットは、完全な結果セットの行 11 ～ 20 が返されたサブセットです。

OFFSET を使用するときの考慮事項

クエリで OFFSET を使用する場合、次の点を考慮してください。

- 最大オフセットは 2,000 行です。2,000 より大きいオフセットを要求すると System.SearchException: SOSL offset should be between 0 to 2000 エラーが発生します。
- 同じ結果セットの後続のサブセットを取得する必要がある場合は、LIMIT 句を OFFSET と組み合わせて使用することをお勧めします。たとえば、次を使用してクエリの最初の 100 行を取得できます。

  ```
  FIND {test} RETURNING Account(Name, Id ORDER BY Name LIMIT 100)
  ```

  その後、以下のクエリを使用して次の 100 行 (101 ～ 200) を取得できます。

  ```
  FIND {test} RETURNING Account(Name, Id ORDER BY Name LIMIT 100 OFFSET 100)
  ```

- OFFSET を使用する場合、所定のクエリではレコードの最初のバッチのみが返されます。次のバッチを取得する場合、オフセット値を高くしたクエリを再実行する必要があります。

- 同じ検索語での連続する SOSL 要求で異なる OFFSET を使用すると、前回の要求以降に検索対象データが更新されている場合、同じデータの異なるサブセットが返される保証はありません。
- OFFSET 句は、SOAP API、REST API、および Apex で使用される SOSL で許可されます。

ORDER BY 句

ORDER BY 句を使用して、SOSL クエリから返される検索結果の順序を指定できます。また、この句を使用して、結果の先頭または最後に空のレコードを表示することもできます。

SOSL ステートメントで 1 つ以上の ORDER BY 句を使用します。

構文

```
ORDER BY fieldname [ASC | DESC] [NULLS [first | last]]
```

構文	説明
ASC または DESC	結果を昇順または降順に並び替えます。デフォルトは昇順です。複数の ORDER BY 句を指定できます。
NULLS [first | last]	null のレコードを結果の先頭 (NULLS FIRST) または最後 (NULLS LAST) に並び替えます。デフォルトでは、null の値が最初に並び替えられます。

例

この例では、取引先名を ID で昇順に並び替えます。

```
FIND {MyName} RETURNING Account(Name, Id ORDER BY Id)
```

この例では、複数の ORDER BY 句が表示されており、取引先責任者を名前および取引先の説明で昇順に並び替えます。

```
FIND {MyContactName} RETURNING Contact(Name, Id ORDER BY Name), Account(Description, Id
ORDER BY Description)
```

次の検索では、名前でアルファベットの降順に並び替えられた取引先レコードが返されます。このとき、名前が null の取引先が最後に表示されます。

```
FIND {MyAccountName} IN NAME FIELDS RETURNING Account(Name, Id ORDER BY Name DESC NULLS
last)
```

この検索では、すべての項目に「San Francisco」が含まれていて、緯度座標 37、経度座標 122 の場所から 500 マイル内にある地理位置情報項目または住所項目が含まれているカスタムオブジェクトが返されます。結果は、座標からその場所への距離での降順に並び替えられます。

```
FIND {San Francisco} RETURNING My_Custom_Object__c (Name, Id WHERE
DISTANCE(My_Location_Field__c,GEOLOCATION(37,122),'mi') < 500 ORDER BY
DISTANCE(My_Location_Field__c,GEOLOCATION(37,122),'mi') DESC)
```

RETURNING *FieldSpec*

テキスト検索結果で返される情報を指定するには、RETURNING 句 (省略可能) を SOSL クエリに追加します。

この句を指定しない場合、デフォルトの動作では、高度な検索で検索可能なすべてのオブジェクトの ID が最大制限の数まで返されます。この最大制限は、LIMIT *n* 句で指定されている値か、2,000 (API バージョン 28.0 以降) のいずれか小さい方となります。返される ID には、カスタムタブがない場合でも、カスタムオブジェクトが組み込まれます。検索結果にはオブジェクトのリストがその句で指定された順序で表示されます。API バージョン 27.0 以前では、最大 200 件の結果をサポートしています。

メモ: 外部オブジェクト、記事、ドキュメント、フィードコメント、フィード項目、ファイル、商品、およびソリューションが検索結果で返されるようにするには、RETURNING 句で明示的に指定する必要があります。次に例を示します。

```
FIND {MyProspect} RETURNING MySampleExternalObject, KnowledgeArticleVersion, Document,
 FeedComment, FeedItem, ContentVersion, Product2, Solution
```

RETURNING 句を使用して、search() コールから返される結果データを制限できます。ID についての詳細は、「ID データ型」を参照してください。

構文

次の構文ステートメントでは、角括弧[]は省略可能な要素を表します。カンマは、そのセグメントを複数回指定できることを示します。

```
RETURNING ObjectTypeName
[(FieldList [WHERE] [USING Listview=listview name] [ORDER BY Clause] [LIMIT n] [OFFSETn])]

        [, ObjectTypeName [(FieldList [WHERE] [ORDER BY Clause] [LIMITn] [OFFSETn])]]
```

RETURNING には次の要素を含めることができます。

名前	説明
ObjectTypeName	返されるオブジェクト。指定されている場合、search() コールは指定されたオブジェクトに一致するすべての検出されたオブジェクトの ID を返します。有効な sObject 種別である必要があります。複数のオブジェクトをカンマで区切って指定できます。複数の *ObjectTypeName* を指定する場合、各オブジェクトは固有である必要があります。1 つの RETURNING 句の中で *ObjectTypeName* を繰り返すことはできません。search() コールは、RETURNING 句で指定されたオブジェクトのみを返します。
FieldList	特定のオブジェクトに対して返す、カンマで区切られた 1 つ以上の項目のリスト (省略可能)。1 つ以上の項目を指定している場合、検出されたすべてのオブジェクトに対してその項目が返されます。
USING ListView=	1 つの特定のオブジェクトのリストビュー内で検索するために使用される省略可能な句。1 つのリストビューのみを指定できます。ユーザーがリストビューに設定した並び替え順に従って、リストビューの最初の 2,000 レコードのみが検索さ

名前	説明
	れます。ListView=Recent は、現在のユーザーが表示または参照した、最後にアクセスされた項目を検索します。

WHERE	特定のオブジェクトの検索結果を個別の項目値に基づいて絞り込む方法の説明(省略可能)。指定されていない場合、ユーザーに表示されるオブジェクトのすべての行が検索で取得されます。
	WHERE 句を指定する場合、1 つ以上の指定された項目を含む *FieldList* を含める必要があります。次の例は、適切な構文ではありません。
	`RETURNING Account (WHERE name like 'test')`
	正しい構文は次のとおりです。
	`RETURNING Account (Name, Industry WHERE Name like 'test')`
	詳細は、「*conditionExpression*」を参照してください。

ORDER BY *句*	昇順、降順、および null の表示順序を含む、返される結果の順序付け方法の説明(省略可能)。複数の ORDER BY 句を指定できます。
	ORDER BY 句を指定する場合、*FieldList* を 1 つ以上の項目と共に指定する必要があります。次の例は、適切な構文ではありません。
	`RETURNING Account (ORDER BY id)`
	正しい構文は次のとおりです。
	`RETURNING Account (Name, Industry ORDER BY Name)`

LIMIT*n*	指定されたオブジェクトで返されるレコードの最大数を設定する句 (省略可能)。指定されていない場合、クエリ全体に設定された制限までの一致するすべてのレコードが返されます。
	LIMIT 句を指定する場合、*FieldList* を 1 つ以上の項目と共に指定する必要があります。次の例は、適切な構文ではありません。
	`RETURNING Account (LIMIT 10)`
	正しい構文は次のとおりです。
	`RETURNING Account (Name, Industry LIMIT 10)`

OFFSET*n*	クエリによって返される結果セットへの開始行オフセットを指定するために使用する句 (省略可能) です。OFFSET は、1 つのオブジェクトを照会するときにのみ使用できます。OFFSET 句は、クエリの最後に指定する必要があります。 OFFSET 句を指定する場合、*FieldList* を 1 つ以上の項目と共に指定する必要があります。次の例は、適切な構文ではありません。 `RETURNING Account (OFFSET 25)`

名前	説明
	正しい構文は次のとおりです。 `RETURNING Account (Name, Industry OFFSET 25)`

メモ: `RETURNING` **句は、外部オブジェクトが検索されるかどうかに影響します。他のオブジェクトの場合、** `RETURNING` **句は、どのデータが検索されるかではなく、どのデータが返されるかを指定します。** `IN` **句は、検索されるデータに影響します。**

RETURNING 句の例

検索タイプ	例
項目指定なし	`FIND {MyProspect}`
1 **つの** sObject**、項目なし**	`FIND {MyProspect} RETURNING Contact`
複数の sObject **オブジェクト、項目なし**	`FIND {MyProspect} RETURNING Contact, Lead`
1 **つの** sObject**、**1 **つ以上の項目**	`FIND {MyProspect} RETURNING Account(Name)`
	`FIND {MyProspect} RETURNING Contact(FirstName, LastName)`
カスタム sObject	`FIND {MyProspect} RETURNING CustomObject__c`
	`FIND {MyProspect} RETURNING CustomObject__c(CustomField__c)`
複数の sObject **オブジェクト、**1 **つ以上の項目、制限**	`FIND {MyProspect} RETURNING Contact(FirstName, LastName LIMIT 10), Account(Name, Industry)`
複数の sObject **オブジェクト、不定数の項目**	`FIND {MyProspect} RETURNING Contact(FirstName, LastName), Account, Lead(FirstName)`
検索不可の sObject **オブジェクト**	`FIND {MyProspect} RETURNING RecordType`
	`FIND {MyProspect} RETURNING Pricebook`
無効な sObject **オブジェクト**	`FIND {MyProspect} RETURNING FooBar`
無効な sObject **項目**	`FIND {MyProspect} RETURNING Contact(fooBar)`
1 **つのオブジェクトの制限**	`FIND {MyProspect} RETURNING Contact(FirstName, LastName LIMIT 10)`
複数のオブジェクトの制限とクエリ制限	`FIND {MyProspect} RETURNING Contact(FirstName, LastName LIMIT 20), Account(Name, Industry LIMIT 10), Opportunity LIMIT 50`
1 **つのオブジェクトのオフセット**	`FIND {MyProspect} RETURNING Contact(FirstName, LastName OFFSET 10)`

検索タイプ	例
リストビュー	`FIND {MyAcccount} IN ALL FIELDS RETURNING Account(Id, Name USING ListView=ListViewName)`

メモ: Apex では、使用しているステートメントで SOQL ステートメントや SOSL ステートメントを使うには、角括弧で囲む必要があります。前にコロン (:) がある場合は、Apex スクリプト変数と式を使用できます。

toLabel(fields)

SOSL クエリの結果をユーザーの言語に翻訳するには、toLabel(*fields*) を使用します。

`toLabel()` メソッドは、どの組織でも使用できます。これはトランスレーションワークベンチを有効にしている組織で特に役立ちます。

```
toLabel(object.field)
```

次に例を示します。

```
FIND {Joe} RETURNING Lead(company, toLabel(Recordtype.Name))
```

返されるレコードは、クエリを発行したユーザーの言語に翻訳されます。

メモ: レコードタイプを翻訳された名前の値で絞り込むことはできません。レコードタイプは、常にオブジェクトのマスター値または ID で絞り込みます。

`toLabel()` メソッドを使用して、翻訳された選択リスト値を使用するレコードを絞り込めます。次に例を示します。

```
FIND {test} RETURNING Lead(company, toLabel(Status) WHERE toLabel(Status) = 'le Draft'
 )
```

このクエリでは、Status の選択リスト値が「le Draft」のリードレコードが返されます。ユーザーの言語での値が比較されます。選択リストの翻訳がない場合は、マスター値に対して比較が実行されます。

メモ: `toLabel()` メソッドは ORDER BY *句*では使用できません。Salesforce では、定義された順序が選択リストで常に使用されます (レポートと同様)。

`toLabel` 関数では別名指定がサポートされます。さらに、クエリに同じ項目が複数回含まれるときは、別名指定が必要です。次に例を示します。

```
FIND {Joe} RETURNING Lead(company, toLabel(Recordtype.Name) AliasName)
```

SOSL を使用して記事のキーワード追跡を更新する

SOSL クエリで `UPDATE TRACKING` 句 (省略可能) を使用すると、Salesforce ナレッジ記事の検索で使用されるキーワードを追跡できます。言語属性を使用してロケールで検索できます。

UPDATE TRACKING 句は、Salesforce ナレッジの記事の検索および参照についてレポートするために使用します。開発者は、Salesforce ナレッジの記事の検索で使用するキーワードを追跡できます。また、言語属性を使用して、特定の言語 (ロケール) で検索することもできます。ただし、1 つのクエリで指定できるのは 1 つの言語のみです。目的の言語ごとに、個別のクエリを行います。ロケールを指定するには、アンダースコアを使用する Java 形式 (fr_FR、jp_JP など) を使用します。サポートされているロケールのリストを取得するには、Web で「java ロケールコード」を検索してください。

次の構文を使用して、Salesforce ナレッジの記事の検索で使用するキーワードを追跡できます。

```
FIND {Keyword}
RETURNING KnowledgeArticleVersion (Title WHERE PublishStatus="Online" and language="en_US")
UPDATE TRACKING
```

SOSL を使用して記事の参照統計を更新する

Salesforce ナレッジ記事が表示された回数を判別するには、SOSL クエリで UPDATE VIEWSTAT 句 (省略可能) を使用します。言語属性を使用してロケールで検索できます。

UPDATE VIEWSTAT 句 (省略可能) は、Salesforce ナレッジの記事の検索および参照についてレポートするために使用します。開発者は、記事の参照統計を更新できます。また、言語属性を使用して、特定の言語 (ロケール) で検索することもできます。ただし、1 つのクエリで指定できるのは 1 つの言語のみです。目的の言語ごとに、個別のクエリを行います。ロケールを指定するには、アンダースコアを使用する Java 形式 (fr_FR、jp_JP など) を使用します。サポートされているロケールのリストを取得するには、Web で「java ロケールコード」を検索してください。

次の構文を使用して、オンラインで英語でアクセスできるすべての記事の参照数を増加できます。

```
FIND {Title}
RETURNING FAQ__kav (Title WHERE PublishStatus="Online" and
language="en_US" and
KnowledgeArticleVersion = 'ka230000000PCiy')
UPDATE VIEWSTAT
```

USING Listview=

1 つの特定のオブジェクトのリストビュー内で検索するために使用される省略可能な句。1 つのリストビューのみを指定できます。ユーザーがリストビューに設定した並び替え順に従って、リストビューの最初の 2,000 レコードのみが検索されます。この句は、API バージョン 41 以降で使用できます。

例: 次の SOSL ステートメントは *MVP Customers* リストビューで *Acme* の Account オブジェクトを検索しています。

```
FIND {Acme} IN ALL FIELDS RETURNING Account(Id, Name USING ListView=MVPCustomers)
```

例: 次の SOSL ステートメントは最近表示されたリストビューで *Acme* の Account オブジェクトを検索しています。

```
FIND {Acme} IN ALL FIELDS RETURNING Account(Name USING LISTVIEW = Recent ORDER BY Name
 ASC NULLS FIRST, Id ASC NULLS FIRST LIMIT 51)
```

サポートされる API

SOSL 内の句は、SOAP API、REST API、および Apex でサポートされます。

WHERE

デフォルトでは、オブジェクトに対する SOSL クエリを実行すると、アーカイブされた行を含めユーザーに表示されているすべての行が取得されます。検索を限定するために、特定の項目値で検索結果を絞り込むことができます。

conditionExpression

`WHERE` 句の *conditionExpression* では次の構文を使用します。

```
fieldExpression [logicalOperator fieldExpression2 ... ]
```

論理演算子を使用して、複数の項目式を条件式に追加できます。

SOSL FIND ステートメントの条件式は、これらの例では太字で表示されます。

- `FIND {test} RETURNING Account (id WHERE createddate = THIS_FISCAL_QUARTER)`
- `FIND {test} RETURNING Account (id WHERE cf__c includes('AAA'))`

fieldExpression が評価される順序を定義するには、括弧を使用します。演算子をネストするときは、括弧を指定する必要があります。ただし、同じ種別の複数の演算子はネストする必要がありません。次の例の式は、*fieldExpression1* が `true` で、*fieldExpression2* または *fieldExpression3* のいずれかが `true` の場合、`true` です。

```
fieldExpression1 AND (fieldExpression2 OR fieldExpression3)
```

ただし、次の式は、*fieldExpression3* が `true` であるか、*fieldExpression1* と *fieldExpression2* の両方が `true` の場合、`true` です。

```
(fieldExpression1 AND fieldExpression2) OR fieldExpression3
```

fieldExpression

fieldExpression では次の構文を使用します。

```
fieldName comparisonOperator value
```

各項目は次のとおりです。

構文	説明
fieldName	指定したオブジェクト内の項目の名前。名前の前後に一重または二重引用符を使用すると、エラーが発生します。項目に対する参照レベル以上の権限が必要です。ロングテキストエリア項目、暗号化されたデータ項目、または Base64 で符号化された項目以外の項目を指定できます。名前は、*fieldList* に含まれている項目である必要はありません。

構文	説明
comparisonOperator	値を比較する演算子。=、<=、IN、LIKE などがあります。演算子は、ほとんどの項目では大文字と小文字が区別されません。ただし、大文字と小文字が区別される項目では、大文字と小文字が区別されます。
value	*fieldName* の値と比較するために使用される値。指定した項目の型と一致するデータ型の値を指定します。値は、他の項目名や計算値ではなく、有効な値にする必要があります。引用符が必要な場合は、単一引用符を使用します。二重引用符を使用するとエラーになります。日付と数値には、引用符は不要です。

比較演算子

次の表に、*fieldExpression* 構文で使用される *comparisonOperator* の値を示します。文字列の比較では、大文字と小文字は区別されません。

演算子	名前	説明
=	Equals	*fieldName* の値が式の *value* に一致する場合、式は true です。
!=	Not equals	*fieldName* の値が指定した *value* に一致しない場合、式は true です。
<	Less than	*fieldName* の値が指定した *value* より小さい場合、式は true です。
<=	Less or equal	*fieldName* の値が指定した *value* 以下の式は true です。
>	Greater than	*fieldName* の値が指定した *value* より大きい場合、式は true です。
>=	Greater or equal	*fieldName* の値が指定した *value* 以上の場合、式は true です。
LIKE	Like	*fieldName* の値が指定した *value* のテキスト文字列の文字に一致する場合、式は true です。指定した value のテキスト文字列は、一重引用符で囲む必要があります。 LIKE 演算子は、文字列項目でのみサポートされます。この演算子は、部分的なテキスト文字列を照合するメカニズムを提供し、次の使用がサポートされます。 • % および _ ワイルドカード。 – % ワイルドカードは、0 個以上の文字に一致します。 – _ ワイルドカードは、1 文字のみに一致します。 • 特殊文字 % または _ のエスケープ。 メモ: 特殊文字をエスケープする場合を除き、検索ではバックスラッシュ (\) 文字を使用しないでください。「引用符で囲まれた文字列のエスケープシーケンス」を参照してください。

演算子	名前	説明

		次のクエリ例は Appleton、Apple、Appl と一致しますが、Bappl とは一致しません。
		`SELECT AccountId, FirstName, lastname`
		`FROM Contact`
		`WHERE lastname LIKE 'appl%'`

IN	IN	値が WHERE 句の値のいずれかに等しい場合、式は true です。IN の文字列値は括弧の中に入れて、一重引用符で囲む必要があります。
		IN を使用して、同じオブジェクトの別の項目に、指定された値のセットがある項目の値を照会できます。次に例を示します。
		`SELECT Name FROM Account`
		`WHERE BillingState IN ('California', 'New York')`
		IN と NOT IN は、ID (主キー) 項目または参照 (外部キー) 項目を照会するときに、準結合および反結合でも使用できます。

NOT IN	NOT IN	値が WHERE 句の値と等しくない場合、式は true です。NOT IN の文字列値は括弧の中に入れて、一重引用符で囲む必要があります。次に例を示します。
		`SELECT Name FROM Account`
		`WHERE BillingState NOT IN ('California', 'New York')`
		メモ: 論理演算子の NOT はこの比較演算子とは無関係です。

INCLUDES EXCLUDES		複数選択リストにのみ適用されます。「複数選択リストのクエリ」を参照してください。

論理演算子

次の表に、*fieldExpression* 構文で使用される論理演算子の値を示します。

演算子	構文	説明
AND	***fieldExpressionX*** AND ***fieldExpressionY***	*fieldExpressionX* と *fieldExpressionY* の両方が true の場合に true。

OR	***fieldExpressionX*** OR ***fieldExpressionY***	*fieldExpressionX* または *fieldExpressionY* のいずれかが true の場合に true。
		OR を使用する WHERE 句では、レコードの外部キーの値が null の場合でも、レコードが返されます。
		`SELECT Id FROM Contact WHERE LastName = 'Young'`
		`or Account.Name = 'Quarry'`

演算子	構文	説明
NOT	not ***fieldExpressionX***	*fieldExpressionX* が false の場合に true。 この論理演算子とは異なる比較演算子 NOT IN もあります。

引用符で囲まれた文字列のエスケープシーケンス

SOSL で次のエスケープシーケンスを使用できます。

シーケンス	意味
\n または \N	改行
\r または \R	行頭復帰
\t または \T	タブ
\b または \B	バックスペース
\f または \F	フォームフィード
\"	1 つの二重引用符文字
\'	1 つの一重引用符文字
\\	バックスラッシュ
LIKE 式のみ: _	1 つのアンダースコア文字 (_)
LIKE 式のみ: \%	1 つのパーセント記号文字 (%)
\u***XXXX***	*XXXX* がコードの Unicode 文字 (たとえば、\u00e9 は é の文字を表します)。

その他のコンテキストでバックスラッシュ文字を使用すると、エラーが発生します。

WHERE 句の例

例

```
FIND {test}
    RETURNING Account (id WHERE createddate = THIS_FISCAL_QUARTER)
```

```
FIND {test}
    RETURNING Account (id WHERE cf__c includes('AAA'))
```

例

```
FIND {test}
    RETURNING Account (id), User(Field1,Field2 WHERE Field1 = 'test' order by id ASC,
Name DESC)
```

```
FIND {test} IN ALL FIELDS
    RETURNING Contact(Salutation, FirstName, LastName, AccountId WHERE Name = 'test'),
        User(FirstName, LastName),
        Account(id WHERE BillingState IN ('California', 'New York'))
```

```
FIND {test}
    RETURNING Account (id WHERE (Name = 'New Account')
        or (Id = '001z00000008Vq7'
        and Name = 'Account Insert Test')
        or (NumberOfEmployees < 100 or NumberOfEmployees = null)
        ORDER BY NumberOfEmployees)
```

ID で Salesforce ナレッジの記事を検索する

```
FIND {tourism}
    RETURNING KnowledgeArticleVersion (Id, Title WHERE id = 'ka0D0000000025eIAA')
```

ID で Salesforce ナレッジの複数の記事を検索する

```
FIND {tourism}
    RETURNING KnowledgeArticleVersion
        (Id, Title WHERE id IN ('ka0D0000000025eIAA', 'ka0D000000002HCIAY'))
```

緯度座標 37、経度座標 122 の場所から 500 マイル以内にある地理位置情報項目または住所項目が含まれているすべての My_Custom_Object__c オブジェクトの全項目で「San Francisco」を検索する

```
FIND {San Francisco}
    RETURNING My_Custom_Object__c (Id
        WHERE DISTANCE(My_Location_Field__c,GEOLOCATION(37,122),'mi') < 100)
```

WITH DATA CATEGORY *DataCategorySpec*

WITH DATA CATEGORY 句 (省略可能) を SOSL クエリに追加すると、1 つ以上のデータカテゴリに関連付けられていてユーザーが参照可能なすべての検索結果を絞り込むことができます。この句は、Salesforce ナレッジの記事と質問の検索で使用されます。

WITH DATA CATEGORY 句は、API バージョン 18.0 以降で使用できます。

構文

WITH DATA CATEGORY 構文は、次のようになります。

```
WITH DATA CATEGORY DataCategorySpec [logicalOperator DataCategorySpec2 ... ]
```

DataCategorySpec は *groupName*、*Operator*、および *category* で構成されます。

名前	説明
groupName	絞り込むデータカテゴリグループの名前。カテゴリグループについては、Salesforce ヘルプの「カテゴリグループの作成と編集」を参照してください。
Operator	次のいずれかの演算子を使用します。 • AT — 指定されたデータカテゴリを照会します。 • ABOVE — 指定されたデータカテゴリとそのすべての親カテゴリを照会します。 • BELOW — 指定されたデータカテゴリとそのすべてのサブカテゴリを照会します。 • ABOVE_OR_BELOW — 指定されたデータカテゴリ、そのすべての親カテゴリ、およびそのすべてのサブカテゴリを照会します。
category	絞り込むカテゴリの名前。複数のデータカテゴリを含めるには、括弧で囲み、カンマで区切ります。カテゴリについては、Salesforce ヘルプの「カテゴリグループへのデータカテゴリの追加」を参照してください。

論理演算子 AND を使用して、複数のデータカテゴリ指定子を追加できます。OR や AND NOT などの他の演算子はサポートされていません。

WITH DATA CATEGORY 句を使用した SOSL ステートメントには、PublishStatus 項目で絞り込む WHERE 句と共に RETURNING ***ObjectTypeName*** 句も含める必要があります。

RETURNING 句では、*ObjectTypeName* に次のいずれかを指定します。

- *__kav* の付いた記事タイプ名 (特定の記事タイプを検索する場合)
- KnowledgeArticleVersion (すべての記事タイプを検索する場合)
- Question (質問を検索する場合)

記事タイプについては、Salesforce ヘルプの「ナレッジ記事タイプ」を参照してください。

WHERE 句は、次のいずれかの公開状況を使用する必要があります。

- `WHERE PublishStatus='online'` (公開記事の場合)
- `WHERE PublishStatus='archived'` (アーカイブ済み記事の場合)
- `WHERE PublishStatus='draft'` (ドラフト記事の場合)

例

検索タイプ	例
1つのカテゴリグループ内のカテゴリで、すべての公開(オンライン)Salesforceナレッジ記事を検索します。	`FIND {tourism} RETURNING KnowledgeArticleVersion (Id, Title WHERE PublishStatus='online') WITH DATA CATEGORY Location__c AT America__c`
2つのカテゴリグループ内のそれぞれのカテゴリで、オンラインFAQ記事を検索します。	`FIND {tourism} RETURNING FAQ__kav (Id, Title WHERE PublishStatus='online') WITH DATA CATEGORY Geography__c ABOVE France__c AND Product__c AT mobile_phones__c`
1つのカテゴリグループからアーカイブ済みFAQ記事を検索します。	`FIND {tourism} RETURNING FAQ__kav (Id, Title WHERE PublishStatus='archived') WITH DATA CATEGORY Geography__c AT Iceland__c`
1つのカテゴリグループからすべてのドラフトSalesforceナレッジ記事を検索します。	`FIND {tourism} RETURNING KnowledgeArticleVersion (Id, Title WHERE PublishStatus='draft') WITH DATA CATEGORY Geography__c BELOW Europe__c`

`WITH DATA CATEGORY` 句については、「*WITH DATA CATEGORY filteringExpression*」を参照してください。

ヒント: `WITH DATA CATEGORY` 句を使用せずに、IDによって記事を検索することもできます。詳細は、「WHERE 句の例」を参照してください。

WITH *DivisionFilter*

すべての検索結果をディビジョン項目に基づいて絞り込むには、`WITH DivisionFilter` 句(省略可能)をSOSLクエリに追加します。他の検索条件を適用する前に、ディビジョンに基づいてすべてのレコードが事前に絞り込まれます。ディビジョンをIDではなく名前で指定することもできます。

次に例を示します。

```
FIND {test} RETURNING Account (id where name like '%test%'),
                      Contact (id where name like '%test%')
            WITH DIVISION = 'Global'
```

メモ:

- ユーザーは「ディビジョンの使用」権限を持っているかどうかに関わらず、ディビジョンに基づく検索を実行できます。

- 特定のディビジョン内のすべての検索には、グローバルディビジョンも含まれます。たとえば、「西部ディビジョン」というディビジョン内で検索すると、西部ディビジョンとグローバルディビジョンの両方で見つかったレコードが検索結果に含まれます。

WITH HIGHLIGHT

`WITH HIGHLIGHT` 句 (省略可能) を法人取引先、キャンペーン、取引先責任者、カスタムオブジェクト、リード、商談、見積、およびユーザー検索の SOSL クエリに追加できます。これにより、検索結果内で検索クエリに一致する語が強調表示されるため、関連するコンテンツを容易に特定できます。`WITH HIGHLIGHT` 句は、API バージョン 39.0 以降で使用できます。カスタム項目およびオブジェクトの `WITH HIGHLIGHT` 句は、API バージョン 40.0 以降で使用できます。

強調表示される検索語は、次の標準およびカスタム項目種別から生成されます。

- 自動採番
- メール
- テキスト
- テキストエリア
- ロングテキストエリア

強調表示される検索語は、次の項目種別からは生成されません。

- チェックボックス
- 複合項目
- 通貨
- 日付
- 日付/時間
- ファイル
- 数式
- 参照関係
- 数値
- パーセント
- 電話
- 選択リスト
- 選択リスト (複数選択)
- リッチテキストエリア
- URL

例: 次の SOSL ステートメントでは、検索語「*salesforce*」を強調表示した検索結果が返されます。

```
FIND {salesforce} IN ALL FIELDS RETURNING Account(Name,Description) WITH HIGHLIGHT
```

一致する語は `<mark>` タグによって強調表示されます。スペルミスにより元の検索語では結果が生成されない場合、検索語の修正されたスペルが結果内で強調表示されます。

例:

```
{
  "searchRecords" : [ {
    "attributes" : {
      "type" : "Account",
     "url" : "/services/data/v39.0/sobjects/Account/001xx000003DpxkAAC"
    },
    "Name" : "salesforce",
    "Description" : "Salesforce.com",
    "highlight.Description" : "<mark>salesforce</mark>.com",
    "highlight.Name" :
     "<mark>salesforce</mark>"
  } ]
}
```

例: 次の SOSL ステートメントでは、カスタムオブジェクト `Building` のカスタム項目 `BuildingDescription` について、検索語 *「salesforce west」* を強調表示した検索結果が返されます。

```
FIND {Salesforce West} IN ALL FIELDS RETURNING Building__c(Name, BuildingDescription__c)
 WITH HIGHLIGHT
```

使用方法

ワイルドカードを含む検索語は強調表示されません。

`WITH HIGHLIGHT` 付きの検索に含まれるその他のオブジェクトは、強調表示された検索語を返しません。

1 つの SOSL クエリの 1 エンティティあたり最大 25 のレコードが強調表示されます。

サポートされる API

SOSL 内の `WITH HIGHLIGHT` 句は、SOAP API および REST API でサポートされます。

WITH METADATA

応答でメタデータが返されるかどうかを指定します。省略可能です。

デフォルトではメタデータは返されません。応答にメタデータを含めるには、検索結果で返される項目の表示ラベルを返す `LABELS` 値を使用します。次に例を示します。

```
FIND {Acme} RETURNING Account(Id, Name) WITH METADATA='LABELS'
```

WITH NETWORK *NetworkIdSpec*

Experience Cloud サイトユーザーとフィードを検索するには、`WITH NETWORK` 句 (省略可能) を SOSL クエリで使用します。検索結果を Experience Cloud サイトで絞り込む場合、各サイトは `NetworkId` で表されます。

次の構文を使用できます。

- `WITH NETWORK IN (’NetworkId1', ’NetworkId2', ...)` は、1 つ以上の Experience Cloud サイトによる絞り込みをサポートします。
- `WITH NETWORK = ’NetworkId’` は、1 つの Experience Cloud サイトによる絞り込みのみサポートします。

ユーザーおよびフィード以外のオブジェクトの場合、クエリでネットワーク絞り込みを使用しても、すべての Experience Cloud サイトおよび内部会社データ全体での一致内容が検索結果に含まれます。

- 同じ Experience Cloud サイトでは、複数のオブジェクトに対して検索を実行できます。
- 範囲が指定されている検索と指定されていない検索を同じクエリで実行することはできません。たとえば、特定の Experience Cloud サイトのユーザーと組織全体の取引先は、一緒に検索できません。

グループまたはトピックの検索結果を Experience Cloud サイト別に絞り込むには、`WHERE` 句で NetworkId 値を使用します。内部組織を検索する場合は、NetworkId にすべてゼロの値を使用します。

WITH NETWORK *NetworkIdSpec* 句の例

複数の Experience Cloud サイトの「テスト」という文字列を含むユーザーおよびフィード項目を検索する、およびフィード項目を最も新しいものから最も古いものの順に並び替えるには、次の構文を使用します。

```
FIND {test} RETURNING User (id),
                      FeedItem (id, ParentId WHERE CreatedDate =
                                THIS_YEAR Order by CreatedDate DESC)
            WITH NETWORK IN ('NetworkId1', 'NetworkId2', 'NetworkId3')
```

NetworkId の「テスト」という文字列を含むユーザーおよびフィード項目を検索する、およびフィード項目を最も新しいものから最も古いものの順に並び替えるには、次の構文を使用します。

```
FIND {test} RETURNING User (id),
                      FeedItem (id, ParentId WHERE CreatedDate =
                                THIS_YEAR Order by CreatedDate DESC)
            WITH NETWORK = 'NetworkId'
```

内部組織の「テスト」という文字列を含むユーザーおよびフィード項目を検索する、およびフィード項目を最も新しいものから最も古いものの順に並び替えるには、次の構文を使用します。

```
FIND {test} RETURNING User (id),
                      FeedItem (id, ParentId WHERE CreatedDate =
                                THIS_YEAR Order by CreatedDate DESC)
            WITH NETWORK = '00000000000000'
```

WITH PricebookId

1 つの価格表 ID で商品検索結果を絞り込みます。

Product2 オブジェクトにのみ適用可能です。価格表 ID は、検索する商品に関連付けられている必要があります。次に例を示します。

```
Find {laptop} RETURNING Product2 WITH PricebookId = '01sxx0000002MffAAE'
```

WITH SNIPPET

`WITH SNIPPET` 句 (省略可能) を記事、ケース、フィード、およびアイデアの検索の SOSL クエリに追加できます。検索結果ページの記事タイトルの下の抜粋で、周囲のテキストのコンテキスト内で検索クエリに一致する語が強調表示されます。スニペットにより、ユーザーは探しているコンテンツを容易に特定できます。

メモ: スニペットを生成せずに、強調表示された一致を含む検索結果を生成するには、`WITH HIGHLIGHT` を使用してください。

検索のスニペットと強調表示は、次の項目種別から生成されます。

- メール
- テキスト
- テキストエリア
- ロングテキストエリア
- リッチテキストエリア

検索のスニペットと強調表示は、次の項目種別からは生成されません。

- チェックボックス
- 通貨
- 日付
- 日付/時間
- ファイル
- 数式
- 参照関係
- 数値
- パーセント
- 電話
- 選択リスト
- 選択リスト (複数選択)
- URL

例: 次の SOSL ステートメントでは、検索語 *「San Francisco」* に一致する記事のスニペットを返します。

```
FIND {San Francisco} IN ALL FIELDS RETURNING KnowledgeArticleVersion(id, title WHERE
PublishStatus = 'Online' AND Language = 'en_US') WITH
      SNIPPET (target_length=120)
```

一致する語は、スニペットの結果のコンテキスト内で `<mark>` タグによって強調表示されます。検索語の語幹処理された形式および定義されているシノニムも強調表示されます。

例:

```
[ {
  "attributes" : {
    "type" : "KnowledgeArticleVersion",
```

```
    "url" : "/services/data/v32.0/sobjects/KnowledgeArticleVersion/kaKD00000000001MAA"

  },
  "Id" : "kaKD00000000001MAA"
  "Title" : "San Francisco"
  "Summary" : "City and County of San Francisco"
  "snippet.text" : "<mark>San</mark> <mark>Francisco</mark>, officially the City and
County of <mark>San</mark> <mark>Francisco</mark> is the... City and County of
<mark>San</mark> <mark>Fran</mark>"
  "highlight.Title" : "<mark>San</mark> <mark>Francisco</mark>"
}, {
   "attributes" : {
     "type" : "KnowledgeArticleVersion",
    "url" : "/services/data/v32.0/sobjects/KnowledgeArticleVersion/kaBD0000000007DMAQ"

   },
   "Id" : "kaBD0000000007DMAQ",
   "Title" : "San Francisco Bay Area",
   "Summary" : "Nine county metropolitan area",
   "snippet.text" : "The <mark>SF</mark> Bay Area, commonly known as the Bay Area, is
 a populated region that"
   "highlight.Title" : "<mark>San</mark> <mark>Francisco</mark> Bay Area"
}, {
   "attributes" : {
     "type" : "KnowledgeArticleVersion",
    "url" : "/services/data/v32.0/sobjects/KnowledgeArticleVersion/ka3D0000000042OIAQ"

   },
   "Id" : "ka3D0000000042OIAQ",
   "Title" : "California",
   "Summary" : "State of California",
   "snippet.text" : "(Greater Los Angeles area and <mark>San</mark>
<mark>Francisco</mark> Bay Area, respectively), and eight of the nation's 50 most"
  } ]
```

メモ: この例では、「SF」(「San Francisco」の定義済みシノニム) と「San Fran」(「San Francisco」の語幹処理された形式) も結果内で一致した語として強調表示されます。

使用方法

`WITH SNIPPET` 句を使用する SOSL ステートメントには、`RETURNING` ***`ObjectTypeName`*** 句に `PublishStatus` 項目で絞り込みを行う `WHERE` 句を指定して使用することをお勧めします。

`RETURNING` 句では、*`ObjectTypeName`* に次のいずれかを指定します。

- サフィックス *__kav* の付いた記事タイプ名 (特定の記事タイプを検索する場合)。
- KnowledgeArticleVersion (すべての記事タイプを検索する場合)。

- ケース、ケースコメント、フィード、フィードコメント、アイデア、アイデアのコメントの種別を検索するには、Case、CaseComment、FeedItem、FeedComment、Idea、IdeaComment を使用します。次に例を示します。

```
FIND {San Francisco} IN ALL FIELDS RETURNING FeedItem, FeedComment WITH SNIPPET
(target_length=120)
```

WITH SNIPPET が指定された検索に含まれるその他のオブジェクトは、スニペットを返しません。

検索によって記事が返されないとき、またはスニペットが含まれる項目へのアクセス権がユーザーにない場合、ワイルドカードが含まれる検索語のスニペットは表示されません。WITH SNIPPET 句を追加しても、スニペットを返さない検索でスニペットは返されません。

スニペットが表示されるのは、ページに返される結果が 20 件以下の場合のみです。

ヒント: 一度に 20 件の結果のみを返すようにするには、LIMIT 句または OFFSET 句を使用します。

エスケープされた HTML タグ

HTML タグ内の一致する語がスニペットで返されると、HTML タグはエスケープされ、一致する語が結果内で強調表示されます。

例: 「*salesforce*」を検索して、テキスト「For more information, visit salesforce.com」を含む記事が返されるとします。記事内の元のハイパーリンクタグはエスケープ (符号化) され、「salesforce」がスニペットの結果内で強調表示されます。

```
For more information, visit &lt;a
href='https://salesforce.com'&gt;salesforce.com&lt;/a&gt;
```

対象スニペットの長さ

デフォルトでは、各スニペットには最大で約 300 文字が表示されます。これは、標準のブラウザーウィンドウ表示では通常 3 行のテキストに相当します。表示される文字数は、統計的に有意ではない程度の差異内で、対象の長さになります。

スニペットは、一致する語を含むテキストの 1 つ以上のフラグメントで構成されます。返されたスニペットに複数のテキストフラグメントが含まれる場合 (複数項目内に一致がある場合など)、対象の長さは、返されるフラグメントすべての最大合計長になります。

対象の長さに別の値を指定するには、省略可能な target_length パラメーターを WITH SNIPPET 句に追加します。対象の長さは、50 ～ 1,000 文字で指定できます。target_length が *0* や負の数値などの無効な数値に設定されている場合、長さはデフォルトの *300* に設定されます。

例: target_length パラメーターを 120 文字にすると、標準のモバイルインターフェースに約 3 行のテキストからなるスニペットを表示するのに便利です。

```
FIND {San Francisco} IN ALL FIELDS RETURNING KnowledgeArticleVersion(id, title WHERE
PublishStatus = 'Online' AND Language = 'en_US') WITH
      SNIPPET(target_length=120)
```

サポートされる API

`WITH SNIPPET` 句は、API バージョン 32.0 以降で使用できます。SOSL 内の `WITH SNIPPET` 句は、SOAP API、REST API、および Apex でサポートされます。

WITH SPELL_CORRECTION

`WITH SPELL_CORRECTION` 句 (省略可能) を SOSL クエリに追加できます。`true` に設定すると、スペル修正をサポートする検索のスペル修正が有効になります。`false` に設定されていると、スペル修正は有効になりません。デフォルト値は `true` です。`WITH SPELL_CORRECTION` 句は、API バージョン 40 以降で使用できます。

例: 次の SOSL ステートメントは、*「San Francisco」* という語のアカウントの検索でスペル修正を無効にします。

```
FIND {San Francisco} IN ALL FIELDS RETURNING Account WITH SPELL_CORRECTION = false
```

サポートされる API

SOSL 内の `WITH SPELL_CORRECTION` 句は、SOAP API、REST API、および Apex でサポートされます。