

Force.com SOQL および SOSL リファレンス

バージョン 35.0, Winter '16



目次

第 1 章: Salesforce Object Query Language (SOQL)	1
SOQL 表記規則	2
引用符で囲まれた文字列のエスケープシーケンス	3
予約文字	4
SOQL SELECT の構文	4
条件式の構文 (WHERE 句)	7
fieldExpression の構文	11
USING SCOPE	25
ORDER BY	25
LIMIT	28
OFFSET	29
UPDATE TRACKING	31
UPDATE VIEWSTAT	32
WITH filteringExpression	32
GROUP BY	37
HAVING	45
TYPEOF	47
FOR VIEW	50
FOR REFERENCE	51
集計関数	51
日付関数	57
マルチ通貨組織の通貨項目のクエリ	60
SELECT 句の例	61
リレーションクエリ	63
リレーション名について	63
リレーションクエリの使用	64
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	67
クエリ結果について	69
参照関係と外部結合	71
親子リレーションの識別	72
多態的なキーとリレーションについて	75
リレーションクエリ制限について	80
履歴オブジェクトとリレーションクエリの使用	80
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	81
Partner WSDL とリレーションクエリの使用	81
クエリのバッチサイズの変更	82
オブジェクトに対する SOQL の制限	82
アーカイブデータに対する SOQL	86
シンジケーションフィールド SOQL と対応付けの構文	87

第 2 章: Salesforce Object Search Language (SOSL)	88
SOSL について	89
SOSL 表記規則	90
SOSL の構文	91
FIND {SearchQuery}	93
IN SearchGroup	98
RETURNING FieldSpec	100
WHERE conditionExpression	103
ORDER BY 句	108
UPDATE TRACKING	109
UPDATE VIEWSTAT	110
WITH DivisionFilter	110
WITH DATA CATEGORY DataCategorySpec	111
WITH NETWORK NetworkIdSpec	113
WITH SNIPPET	113
LIMIT n	117
OFFSET n	117
toLabel(fields)	118
マルチ通貨組織の通貨項目のクエリ	119
テキスト検索の例	120
SOSL の制限	121

第1章

Salesforce Object Query Language (SOQL)

トピック:

- [SOQL 表記規則](#)
- [引用符で囲まれた文字列のエスケープシーケンス](#)
- [予約文字](#)
- [SOQL SELECT の構文](#)
- [リレーションクエリ](#)
- [クエリのバッチサイズの変更](#)
- [オブジェクトに対する SOQL の制限](#)
- [シンジケーションフィード SOQL と対応付けの構文](#)

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 データ専用設計されています。

SOQL を使用すると、次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。

- `query()` コールの `queryString` パラメータ
- Apex ステートメント
- Visualforce コントローラおよび `getter` メソッド
- Force.com IDE の Schema Explorer

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、SOQL ステートメントや SOSL ステートメントをオンザフライで使用するには、大括弧で囲む必要があります。さらに、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。

 **メモ:** アーカイブデータと BigObjects では、一部の SOQL 機能のみを使用できます。詳細は、「[アーカイブデータに対する SOQL](#)」(ページ 86)を参照してください。

SOQL 表記規則

SOQL のトピックでは、カスタムの表記規則を使用します。

規則	説明
<code>SELECT Name FROM Account</code>	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。
<code>SELECT <i>fieldname</i> FROM <i>objectname</i></code>	斜体は、変数またはプレースホルダを表します。実際の値を入力してください。
<code>{ }</code>	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、 <code>UPDATE {TRACKING VIEWSTAT} [, ...]</code> 句の場合、中括弧は、 <code>UPDATE</code> の後に続く選択肢が、パイプで区切られた <code>TRACKING</code> または <code>VIEWSTAT</code> であることを示します (<code>UPDATE TRACKING</code> と <code>VIEWSTAT</code> ではない)。
<code> </code>	パイプ文字は、選択肢となる要素を区切ります。たとえば、 <code>UPDATE {TRACKING VIEWSTAT} [, ...]</code> 句の場合、 <code> </code> 文字は <code>UPDATE</code> の後に <code>TRACKING</code> または <code>VIEWSTAT</code> のどちらかを使用できることを示します。
<code>[]</code>	角括弧は、省略可能な要素を示します。たとえば、 <code>[LIMIT rows]</code> は、 <code>LIMIT</code> 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、 <code>[ORDER BY fieldOrderedList [ASC DESC] [NULLS {FIRST LAST}]]</code> 句の場合、 <code>ASC</code> 、 <code>DESC</code> 、または <code>NULLS</code> 句を使用するには <code>ORDER BY</code> 句が必要です。
<code>[...] および [, ...]</code>	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、 <code>UPDATE {TRACKING VIEWSTAT} [, ...]</code> 句の場合、 <code>[, ...]</code> は、 <code>TRACKING</code> 、 <code>VIEWSTAT</code> 、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

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。

次のSOQLキーワードは別名には使用できません。AND, ASC, DESC, EXCLUDES, FIRST, FROM, GROUP, HAVING, IN, INCLUDES, LAST, LIKE, LIMIT, NOT, NULL, NULLS, OR, SELECT, WHERE, WITH

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(%)

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)リテラル文字列として指定された場合に適切に解釈されるように、予約文字をエスケープする(前にバックスラッシュ\文字を付ける)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」をクエリする場合、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'
```

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 | REFERENCE}{[,...]} ]

[ UPDATE {TRACKING|VIEWSTAT}{[,...]} ]
```

 **メモ:** TYPEOF は、現在 SOQL 多態性機能の一部の開発者プレビューとして利用可能です。組織での TYPEOF の有効化については、Salesforce にお問い合わせください。

構文	説明
<i>fieldList subquery</i>	指定した <i>object</i> 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 <i>fieldlist</i> です。 • SELECT Id, Name, BillingCity FROM Account • SELECT count() FROM Contact • SELECT Contact.Firstname, Contact.Account.Name FROM Contact 有効な項目名を指定する必要があり、各指定項目の参照レベルの権限を持っている必要があります。<i>fieldList</i> 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は、<i>fieldList</i> にサブクエリを含めることができます。次に例を示します。 <pre>SELECT Account.Name, (SELECT Contact.LastName FROM Account.Contacts) FROM Account</pre> <i>fieldlist</i> は、COUNT() や COUNT(<i>fieldName</i>) などの集計関数とすることも、toLabel() 内でラップすることもできます。
<i>typeOfField</i>	複数のオブジェクト種別を参照できる、<i>objectType</i> の多態的なリレーション項目、または <i>objectType</i> の親の多態的な項目。たとえば、Event の What リレーション項目には Account、Campaign、Opportunity のいずれかを使用できます。<i>typeOfField</i> は、<i>fieldList</i> でも参照されるリレーション項目を参照できません。詳細は、「TYPEOF」を参照してください。

構文	説明
<i>whenExpression</i>	WHEN <i>whenObjectType</i> THEN <i>whenFieldList</i> という形式の句。TYPEOF 式内には、1つ以上の <i>whenExpression</i> 句を使用できます。詳細は、 「TYPEOF」 を参照してください。
<i>elseExpression</i>	ELSE <i>elseFieldList</i> 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。
<i>objectType</i>	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、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。
<i>filterScope</i>	API バージョン 32.0 以降で利用できます。filterScope は、クエリの結果を制限するために指定します。
<i>conditionExpression</i>	WHERE が指定されている場合は、指定したオブジェクト (<i>objectType</i>) 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が参照可能なすべての行を取得します。
<i>filteringExpression</i>	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 — 質問をクエリします。 • KnowledgeArticleVersion — 記事をクエリします。 WITH DATA CATEGORY 句についての詳細は、「WITH DATA CATEGORY filteringExpression」 を参照してください。
<i>fieldGroupByList</i>	API バージョン 18.0 以降で利用できます。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、コード内で個々のレコードを処理せずに、データを集計してクエリ結果を積み上げ集計するために、集計関数と共に使用します。 「GROUP BY」 を参照してください。
<i>fieldSubtotalGroupByList</i>	API バージョン 18.0 以降で利用できます。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。
<i>havingConditionExpression</i>	API バージョン 18.0 以降で利用できます。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。

構文	説明
<i>fieldOrderByList</i>	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、取引先責任者をクエリし、結果を姓、次に名の順に並び替えることができます。
	<pre>SELECT Id, LastName, FirstName FROM Contact ORDER BY LastName, FirstName</pre>

次の実装のヒントに注意してください。

- 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 20,000 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。これは、元の SOQL ステートメントが上限の 20,000 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。これを避けるには、SOQL ステートメントの複雑さを軽減します。
- 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()` および `convertCurrency()` 関数を含めることができます。
- Apex の動的 SOQL — Apex では、SOQL ステートメントや SOSL ステートメントをその場で使用するには、大括弧で囲む必要があります。さらに、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。

条件式の構文 (WHERE 句)

SOQL クエリの WHERE 句の条件式の構文には、1 つ以上の項目式が含まれます。論理演算子を使用して、複数の項目式を条件式に指定できます。

SOQL ステートメントの 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。

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

日付値、日付/時間値、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 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。
- *fieldExpression* が評価される順序を定義するには、括弧を使用します。たとえば次の式は、*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
```

- クライアントアプリケーションでは、演算子をネストするときに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。

- ☑ **メモ:** `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、バージョン 13.0 以降ではレコードが返されますが、13.0 より前のバージョンでは返されません。

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

ケースレコード `Id` 値は、バージョン 13.0 以降では返されますが、13.0 より前のバージョンでは返されません。

SOQL クエリの `null`

`null` キーワードを使用して `null` 値を検索できます。

SOQL クエリで `null` 値を表すには、`null` を使用します。

たとえば、次のステートメントでは、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
```

`toLabel()`

SOQL クエリの結果をユーザの言語に翻訳するには、`toLabel(fields)` を使用します。

クライアントアプリケーションは、`toLabel()` を使用してユーザの言語に翻訳されて返されるクエリの結果を使用できます。

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

`toLabel()` は、通常、複数選択、ディビジョン、または通貨コードの選択リスト項目 (関連する記述用の API コール (`describe`) によって返される選択リスト値を含むすべての項目)、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、または `RecordType` の名前で使用します。すべての組織が `toLabel()` を使用できます。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。

次に例を示します。

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

このクエリでは、リードレコードのレコードタイプ名がクエリを発行したユーザの言語に翻訳されて返されます。

 **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値または ID で絞り込みます。

`toLabel()` 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'
```

`Status` の選択リスト値が「leDraft」のリードレコードが返されます。ユーザの言語での値が比較されます。指定された選択リストをユーザの言語に翻訳できない場合、マスタ値に対して比較が実行されます。

 **メモ:** `toLabel()` メソッドは `ORDER BY` 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)。また、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()` を使用することもできません。

Boolean 項目での絞り込み

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
```

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 (つまり複数の項目を選択できる) のクエリに特定の構文を使用します。

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。
;	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。
	<code>'AAA;BBB'</code>

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'
```

多態的なリレーション項目の絞り込み

SQL 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に関連する Event に応じて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
<i>polymorphicRelationship</i>	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、Event の What リレーション項目には Account、Campaign、Opportunity のいずれかを使用できます。
<i>comparisonExpression</i>	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 Opportunity 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、「[多態的なキーとリレーションについて](#)」を参照してください。

fieldExpression の構文

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、および項目名の値との比較に使用される値で構成されます。

fieldExpression では、次の構文を使用します。

```
fieldName comparisonOperator value
```

説明を次に示します。

構文	説明
<i>fieldName</i>	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。fieldList に含まれている項目である必要はありません。
<i>comparisonOperator</i>	値を比較する、大文字と小文字を区別しない演算子。
<i>value</i>	fieldName の値と比較するために使用される値。指定した項目のデータ型と一致するデータ型の値を指定する必要があります。ネイティブ値を指定する必要があります。他の項目名または計算は指定できません。引用符が必要な場合は(たとえば、日付と数値には不要)、一重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。

比較演算子

比較演算子 (=、!=、<、>、LIKE、IN など) は、SOQL クエリの `SELECT` ステートメントに含まれる `WHERE` 句の項目式に使用できます。準結合と反結合を使用してより複雑なクエリを作成することもできます。

`fieldExpression` 構文で使用される `comparisonOperator` 値を次の表に示します。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	指定した <code>fieldName</code> の値が式で指定した <code>value</code> に一致する場合、式は <code>true</code> です。等号演算子を使用する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。
!=	Not equals	指定した <code>fieldName</code> の値が指定した <code>value</code> に一致しない場合、式は <code>true</code> です。
<	Less than	指定した <code>fieldName</code> の値が指定した <code>value</code> より小さい場合、式は <code>true</code> です。
<=	Less or equal	指定した <code>fieldName</code> の値が指定した <code>value</code> 以下の場合、式は <code>true</code> です。
>	Greater than	指定した <code>fieldName</code> の値が指定した <code>value</code> より大きい場合、式は <code>true</code> です。
>=	Greater or equal	指定した <code>fieldName</code> の値が指定した <code>value</code> 以上の場合、式は <code>true</code> です。
LIKE	Like	指定した <code>fieldName</code> の値が指定した <code>value</code> のテキスト文字列の文字に一致する場合、式は <code>true</code> です。SOQL および SOSL の <code>LIKE</code> 演算子は、SQL の <code>LIKE</code> 演算子と似ています。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、ワイルドカードの使用がサポートされます。 • <code>LIKE</code> 演算子では <code>%</code> と <code>_</code> のワイルドカードがサポートされます。 • <code>%</code> ワイルドカードは、0 個以上の文字に一致します。 • <code>_</code> ワイルドカードは、1 文字のみに一致します。 • 指定した <code>value</code> のテキスト文字列は、一重引用符で囲む必要があります。 • <code>LIKE</code> 演算子は、文字列項目でのみサポートされます。 • SQL の大文字と小文字を区別する照合とは異なり、<code>LIKE</code> 演算子では大文字と小文字を区別しない照合が実行されます。 • SOQL および SOSL の <code>LIKE</code> 演算子では、特殊文字 <code>%</code> または <code>_</code> のエスケープがサポートされます。 • 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。

演算子	名前	説明
		たとえば、次のクエリは Appleton、Apple、Appl と一致しますが、Bappl とは一致しません。 <pre>SELECT AccountId, FirstName, lastname FROM Contact WHERE lastname LIKE 'appl%'</pre>
IN	IN	値が WHERE 句で指定された値のいずれかに等しい場合、式は true です。次に例を示します。 <pre>SELECT Name FROM Account WHERE BillingState IN ('California', 'New York')</pre> IN の値は括弧で囲む必要があります。文字列の値は一重引用符で囲む必要があります。 IN と NOT IN は、ID (主キー) 項目または参照 (外部キー) 項目をクエリするときに、準結合および反結合でも使用できます。
NOT IN	NOT IN	値が WHERE 句で指定された値と等しくない場合、式は true です。次に例を示します。 <pre>SELECT Name FROM Account WHERE BillingState NOT IN ('California', 'New York')</pre> NOT IN の値は括弧で囲む必要があります。文字列の値は一重引用符で囲む必要があります。 この比較演算子とは無関係ですが、論理演算子の NOT もあります。
INCLUDES EXCLUDES		複数選択リストにのみ適用されます。

IN を使用した準結合と NOT IN を使用した反結合

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クエリできます。次に例を示します。

```
SELECT Name FROM Account
WHERE BillingState IN ('California', 'New York')
```

さらに、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。

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。不要な処理が行われないようにする制限の一覧については、このセクションの最後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、ソースが 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たことができます。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
)
```

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 IN または NOT IN ステートメントを2つまで使用できます。
 - 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)項目をクエリ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クエリ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クエリできません。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'
)
```

ただし、次のような有効な形式で簡単にクエリを記述し直すことができます。

```
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。
- 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.Title LIKE 'Vacation%')
AND (Idea.LastCommentDate > YESTERDAY)
AND (Id IN (SELECT ParentId FROM Vote
            WHERE CreatedById = '005x00000000sMgYAAU'
            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
 - EventAttendee
 - 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	<i>fieldExpressionX</i> AND <i>fieldExpressionY</i>	<i>fieldExpressionX</i> と <i>fieldExpressionY</i> の両方が true の場合に true。
OR	<i>fieldExpressionX</i> OR <i>fieldExpressionY</i>	<i>fieldExpressionX</i> または <i>fieldExpressionY</i> のいずれかが true の場合に true。 OR 句で外部キーの値を使用したりレレーションクエリの動作は、API のバージョンによって異なります。OR を使用する WHERE 句でレコードの外部キーの値が null の場合、バージョン 13.0 以降で

演算子	構文	説明
		はレコードが返されますが、13.0より前のバージョンではレコードが返されません。 <pre>SELECT Id FROM Contact WHERE LastName = 'foo' or Account.Name = 'bar'</pre> 親取引先のない取引先責任者は条件を満たす姓が含まれているため、バージョン 13.0 以降では返されます。
NOT	not <i>fieldExpressionX</i>	<i>fieldExpressionX</i> 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。

日付形式と日付リテラル

SOQL クエリに日付を指定する場合、その日付は特定の日付または日付リテラルにすることができます。SOQL には、YESTERDAY、TODAY、THIS_WEEK、LAST_MONTH、NEXT_QUARTER、THIS_FISCAL_YEAR など多くの日付リテラルが定義されています。

日付リテラルは、先月、今週、来年などの相対的な時間範囲を表す、固定の式です。dateTime 項目の値は協定世界時 (UTC) で保存されます。これらの値のいずれかが Salesforce で返されるときに、組織の設定で指定したタイムゾーンに自動的に調整されます。SOQL クエリは、dateTime 項目の値を UTC 値で返します。これらの値をさまざまなタイムゾーンで処理するには、アプリケーションによる変換処理が必要になる場合があります。

日付形式

fieldExpression では、date 項目と dateTime 項目にさまざまな日付形式が使用されます。クエリで日付/時間形式を指定する場合、dateTime 項目のみで絞り込みを行うことができます。同様に、クエリで日付形式の値を指定する場合、date 項目のみで絞り込みを行うことができます。

表示形式	形式の構文	例
日付のみ	YYYY-MM-DD	1999-01-01
日付、時間、タイムゾーン のオフセット	- 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

ゾーンのオフセットは必ず UTC を基準とします。詳細は、以下を参照してください。

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

 **メモ:** 日付形式を使用する *fieldExpression* の場合、日付は一重引用符で囲みません。日付の前後に引用符は使用しないでください。次に例を示します。

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

日付リテラル

fieldExpression では、日付リテラルを使用して、値の範囲を *date* 項目または *dateTime* 項目の値と比較できます。各リテラルは、午前 0 時 (00:00:00) から始まる時間の範囲です。範囲内の値を見つけるには、`=` を使用します。範囲の一方の側の値を見つけるには、`>` または `<` を使用します。使用可能な日付リテラル、範囲、および例を次の表に示します。

日付リテラル	範囲	例
YESTERDAY	昨日の 00:00:00 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 00:00:00 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	明日の 00:00:00 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 00:00:00 から、その 7 日後までが指定されます。週の最初の日はロケールによって判断されます。	SELECT Id FROM Account WHERE CreatedDate > LAST_WEEK
THIS_WEEK	今週の最初の日の 00:00:00 から、その 7 日後までが指定されます。週の最初の日はロケールによって判断されます。	SELECT Id FROM Account WHERE CreatedDate < THIS_WEEK
NEXT_WEEK	来週の最初の日の 00:00:00 から、その 7 日後までが指定されます。週の最初の日はロケールによって判断されます。	SELECT Id FROM Opportunity WHERE CloseDate = NEXT_WEEK
LAST_MONTH	先月の最初の日の 00:00:00 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 00:00:00 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 00:00:00 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
LAST_90_DAYS	本日の 00:00:00 から、その 90 日前までが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_90_DAYS

日付リテラル	範囲	例
NEXT_90_DAYS	本日の 00:00:00 から、その 90 日後までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_90_DAYS
LAST_N_DAYS:n	数値 <i>n</i> が指定されている場合、本日の 00:00:00 から、その <i>n</i> 日前までが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_DAYS:365
NEXT_N_DAYS:n	数値 <i>n</i> が指定されている場合、本日の 00:00:00 から、その <i>n</i> 日後までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_DAYS:15
NEXT_N_WEEKS:n	数値 <i>n</i> が指定されている場合、翌週の最初の日の 00:00:00 から、その <i>n</i> 週間までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_WEEKS:4
LAST_N_WEEKS:n	数値 <i>n</i> が指定されている場合、前週の最後の日の 00:00:00 から、その <i>n</i> 週間までが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_WEEKS:52
NEXT_N_MONTHS:n	数値 <i>n</i> が指定されている場合、翌月の最初の日の 00:00:00 から、その <i>n</i> 月後までが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > NEXT_N_MONTHS:2
LAST_N_MONTHS:n	数値 <i>n</i> が指定されている場合、前月の最後の日の 00:00:00 から、その <i>n</i> 月前までが指定されます。	SELECT Id FROM Account WHERE CreatedDate = LAST_N_MONTHS:12
THIS_QUARTER	今四半期の最初の日の 00:00:00 から、今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	前四半期の 00:00:00 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	翌四半期の 00:00:00 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	翌四半期の 00:00:00 から、 <i>n</i> 期後の四半期の終わりまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate < NEXT_N_QUARTERS:2
LAST_N_QUARTERS:n	<i>n</i> 期前の四半期の最初の日の 00:00:00 から、前四半期の最終日の終わりまでが指定されます。	SELECT Id FROM Account WHERE CreatedDate > LAST_N_QUARTERS:2
THIS_YEAR	今年の 1 月 1 日 00:00:00 から、今年の 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	前年の 1 月 1 日 00:00:00 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

日付リテラル	範囲	例
NEXT_YEAR	翌年の1月1日 00:00:00 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: <i>n</i>	翌年の1月1日 00:00:00 から、 <i>n</i> 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
LAST_N_YEARS: <i>n</i>	<i>n</i> 年前の1月1日 00:00:00 から、前年の12月31日の終わりまでが指定されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_N_YEARS:5
THIS_FISCAL_QUARTER	現在の会計四半期の最初の日 00:00:00 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Account WHERE CreatedDate = THIS_FISCAL_QUARTER
LAST_FISCAL_QUARTER	前会計四半期の最初の日 00:00:00 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Account WHERE CreatedDate > LAST_FISCAL_QUARTER
NEXT_FISCAL_QUARTER	翌会計四半期の最初の日 00:00:00 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Account WHERE CreatedDate < NEXT_FISCAL_QUARTER
NEXT_N_FISCAL_QUARTERS: <i>n</i>	翌会計四半期の最初の日 00:00:00 から、 <i>n</i> 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Account WHERE CreatedDate < NEXT_N_FISCAL_QUARTERS:6
LAST_N_FISCAL_QUARTERS: <i>n</i>	<i>n</i> 期前の会計四半期の最初の日 00:00:00 から、前会計四半期の最終日の終わりまでが指定されます。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Account WHERE CreatedDate > LAST_N_FISCAL_QUARTERS:6
THIS_FISCAL_YEAR	現在の会計四半期の最初の日 00:00:00 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。会計年度は、[設	SELECT Id FROM Opportunity WHERE CloseDate = THIS_FISCAL_YEAR

日付リテラル	範囲	例
	定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	
LAST_FISCAL_YEAR	前会計年度の最初の日の 00:00:00 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_FISCAL_YEAR
NEXT_FISCAL_YEAR	翌会計年度の最初の日の 00:00:00 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。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Opportunity WHERE CloseDate < NEXT_FISCAL_YEAR
NEXT_N_FISCAL_YEARS:n	翌会計年度の最初の日の 00:00:00 から、n 年後の会計年度の最終日の終わりまでが指定されます。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	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 00:00:00 から、前会計年度の最終日の終わりまでが指定されます。会計年度は、[設定]の[組織プロファイル]>[会計年度]で表示される組織プロファイルによって定義されます。	SELECT Id FROM Opportunity WHERE CloseDate > LAST_N_FISCAL_YEARS:3

 **メモ:** Salesforce ユーザーインターフェースで[カスタム会計年度]を定義しており、定義した年以外の範囲をいずれかの FISCAL 日付リテラルで指定すると、無効な日付エラーが返されます。

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。

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、ユーザが所有するオブジェクトのみ、またはユーザ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、クエリの結果を指定した `filterScope` に制限できます。構文は次のとおりです。

```
[USING SCOPE filterScope]
```

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。

`SELECT` ステートメントで `ORDER BY` を使用して、クエリ結果の順序を制御できます。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。構文は次のとおりです。

```
[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	<code>null</code> 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、 <code>null</code> の値が最初に並び替えられます。

たとえば、次のクエリは `Account` レコードの名前をアルファベットの降順で並び替え、名前が `null` の取引先を最後に並び替えたクエリ結果を返します。

```
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。

- 並び替えは大文字と小文字を区別しません。
- `ORDER BY` はリレーショナル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りレレーションクエリの動作は、Force.com 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、バージョン 13.0 以降ではレコードが返され、13.0 より前のバージョンではレコードが返されません。

```
SELECT Id, CaseNumber, Account.Id, Account.Name

FROM Case

ORDER BY Account.Name
```

AccountId が空のケースレコードはバージョン 13.0 以降では返されます。

- **並び替え順は現在のユーザロケールによって決定されます。** 英語ロケールの場合、Salesforce は文字データの UTF-8 値に基づく並び替えメカニズムを使用します。アジアロケールの場合、Salesforce は ISO 14651 標準と Unicode 3.2 標準に基づく言語的並び替えメカニズムを使用します。

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。
 - `convertCurrency()` は、使用可能な場合は常にマスタ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。

- 次の制限は、Lightning Connect の OData 2.0 アダプタにのみ適用されます。
 - `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。
- 次の制限は、Lightning Connect のカスタムアダプタにのみ適用されます。
 - `NULLS FIRST` と `NULLS LAST` は無視されます。

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
```

次の要素が ORDER BY を使用して返される結果に影響します。

- 並び替えは大文字と小文字を区別しません。
- 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りレレーションクエリの動作は、Force.com 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、バージョン 13.0 以降ではレコードが返され、13.0 より前のバージョンではレコードが返されません。

```
SELECT Id, CaseNumber, Account.Id, Account.Name

FROM Case
```



```
ORDER BY Account.Name
```

AccountId が空のケースレコードはバージョン 13.0 以降では返されます。

- 並び替え順は現在のユーザロケールによって決定されます。英語ロケールの場合、Salesforce は文字データの UTF-8 値に基づく並び替えメカニズムを使用します。アジアロケールの場合、Salesforce は ISO 14651 標準と Unicode 3.2 標準に基づく言語的並び替えメカニズムを使用します。

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。
 - convertCurrency() は、使用可能な場合は常にマスタ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。

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
```

次の要素が ORDER BY を使用して返される結果に影響します。

- 並び替えは大文字と小文字を区別しません。
- 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、Force.com 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、バージョン 13.0 以降ではレコードが返され、13.0 より前のバージョンではレコードが返されません。

```
SELECT Id, CaseNumber, Account.Id, Account.Name

FROM Case

ORDER BY Account.Name
```

AccountId が空のケースレコードはバージョン 13.0 以降では返されます。

- 並び替え順は現在のユーザロケールによって決定されます。英語ロケールの場合、Salesforce は文字データの UTF-8 値に基づく並び替えメカニズムを使用します。アジアロケールの場合、Salesforce は ISO 14651 標準と Unicode 3.2 標準に基づく言語的並び替えメカニズムを使用します。

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。
 - convertCurrency() は、使用可能な場合は常にマスタ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。

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]
```

以下に例を示します。

```
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カウントできます。

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への開始行オフセットを指定するには、`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
```

例として、SOQL 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。

```
SELECT Name

FROM Merchandise__c

WHERE Price__c > 5.0

ORDER BY Name

LIMIT 100

OFFSET 10
```

前述の例の結果セットは、完全な結果セットの行 11 ～ 50 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` 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。次のクエリは、サブクエリでの `OFFSET` の有効な使用を示します。

```

SELECT Name, Id

(

    SELECT Name FROM Opportunity 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。
- 同様に、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。

```

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側のカーソルは作成されません。`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。一貫したサーバ側カーソルでレコードの複数のページをクエリする必要がある場合は、SOAP API の `queryMore()` を使用します。

- 最大オフセットサイズとサーバ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。
- `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。

UPDATE TRACKING

SOQL クエリで `UPDATETRACKING` 句 (省略可能) を使用すると、Salesforce ナレッジ記事の検索で使用するキーワードを追跡できます。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するキーワードを追跡できます。

 **例:** 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。

```
SELECT Title FROM FAQ__kav

WHERE Keyword='Apex' and

Language = 'en_US' and

KnowledgeArticleVersion = 'ka230000000PCiy'

UPDATE TRACKING
```

UPDATE VIEWSTAT

Salesforce ナレッジ記事が表示された回数を判別するには、SOQL クエリで `UPDATE VIEWSTAT` 句 (省略可能) を使用します。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カウントを取得できます。

`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。開発者は、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カウントを増加できます。

```
SELECT Title FROM FAQ__kav

WHERE PublishStatus='online' and

Language = 'en_US' and

KnowledgeArticleVersion = 'ka230000000PCiy'

UPDATE VIEWSTAT
```

WITH *filteringExpression*

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、ユーザのプロファイルフィールドで追跡されている変更をクエリ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 filteringExpression](#)」を参照してください。
- ユーザプロファイルフィールドで追跡されるレコードの変更をクエリして取得する場合。『[Salesforce および Force.com のオブジェクトリファレンス](#)』の「[UserProfileFeed](#)」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。構文は次のセクションで説明します。

- `SELECT Title FROM KnowledgeArticleVersion WHERE PublishStatus='online' WITH DATA CATEGORY Geography__c ABOVE usa__c`
- `SELECT Id FROM UserProfileFeed WITH UserId='005D00000001AamR' ORDER BY CreatedDate DESC, Id DESC LIMIT 20`

WITH DATA CATEGORY *filteringExpression*

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が参照可能なレコードを検索できます。

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、指定したデータカテゴリに関連付けられていてユーザ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、`ObjectName` が次と一致する FROM `ObjectName` 句も含める必要があります。

- `KnowledgeArticleVersion` (すべての記事タイプをクエリする場合)
- 記事タイプの API Name (特定の記事タイプをクエリする場合)
- `Question` (質問をクエリする場合)

`ObjectName` 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クエリする場合)
 - `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 に基づいて記事をクエリする場合)

記事タイプまたは質問についての詳細は、Salesforce ヘルプの「記事タイプの管理」および「質問の検索と表示」を参照してください。

🔒 注意: WITH DATA CATEGORY 句はバインド変数をサポートしていません。

filteringExpression

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

```
dataCategorySelection [AND [dataCategorySelection2] [...]]
```

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。構文は次のセクションで説明します。

- **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、絞り込みセクタ、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
<i>dataCategoryGroupName</i>	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
<i>filteringSelector</i>	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クタ。有効なセクタのリストは、「絞り込みセクタ」を参照してください。

構文	説明
<code>dataCategoryName</code>	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。カテゴリ表示設定についての詳細は、Salesforceヘルプの「カテゴリグループ表示設定について」を参照してください。 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: <code>WITH DATA CATEGORY Geography__c AT (usa__c,france__c,uk__c)</code> 複数のデータカテゴリの指定に括弧の代わりに <code>AND</code> 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。 <code>WITH DATA CATEGORY Geography__c AT usa__c AND france__c</code>

絞り込みセレクト

SOQL クエリの `WITH 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。

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

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

```

セレクト	説明
<code>AT</code>	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 <code>asia__c</code> を選択します。 <pre>WITH DATA CATEGORY Geography__c AT asia__c</pre>

セレクトタ	説明
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。 <pre>WITH DATA CATEGORY Geography__c ABOVE usa__c</pre>
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。 <pre>WITH DATA CATEGORY Geography__c BELOW northAmerica__c</pre>
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。 <pre>WITH DATA CATEGORY Geography__c ABOVE_OR_BELOW europe__c</pre>



メモ: データカテゴリグループ、データカテゴリ、親カテゴリ、およびサブカテゴリについての詳細は、Salesforce ヘルプの「データカテゴリとは?」を参照してください。

WITH DATA CATEGORY 句の例

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

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	<pre>SELECT Title FROM Question WHERE LastReplyDate < 2005-10-08T01:02:03Z WITH DATA CATEGORY Product__c AT mobile_phones__c</pre>
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	<pre>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</pre>

検索タイプ	例
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__c 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	<pre>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</pre>

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。

API バージョン 18.0 以降では、コードで個々のレコードを処理せずに、データを集計してクエリ結果をロールアップするために、GROUP BY を SUM() や MAX() などの[集計関数](#)と共に使用できます。構文は次のとおりです。

```
[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 値のカウントを増分します。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
```

SOQL でサポートされる集計関数の一覧は、「[集計関数](#)」を参照してください。

オブジェクトのすべての個別値(null を含む)をクエリするには、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
```

`COUNT_DISTINCT()` 関数は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。

GROUP BY を使用するときの考慮事項

GROUP BY 句を使用する SOQL クエリを作成する場合、注意が必要な考慮事項がいくつかあります。

- 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。[DescribeObjectResult](#)に関連付けられたField オブジェクトには、GROUP BY 句に項目を含めることができるかどうかを定義する `groupable` 項目があります。
- クエリで LIMIT 句と集計関数を使用する場合は、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](#)」(ページ 67)を参照してください。

GROUP BY および queryMore()

GROUP BY 句を含まないクエリの場合、クエリ結果のオブジェクトには、デフォルトで最大 500 行のデータが含まれます。クエリ結果が 500 行を超える場合は、クライアントアプリケーションで `queryMore()` コールとサーバ側のカーソルを使用して、追加の行を 500 行のチャンクで取得できます。

ただし、クエリに GROUP BY 句が含まれている場合は、`queryMore()` を使用できません。QueryOptions ヘッダーのデフォルトサイズを最大 2,000 行に増やすことができます。クエリ結果が 2,000 行を超える場合、より小さなチャンクでデータをクエリするように検索条件を変更する必要があります。要求されるバッチサイズが、実際のバッチサイズになるとは限りません。パフォーマンスを最大化するために変更が行われます。詳細は「[クエリのバッチサイズの変更](#)」を参照してください。

GROUP BY と小計

クエリで小計の計算を行い、コードのロジックを維持しないで済むようにするには、「[GROUP BY ROLLUP](#)」を参照してください。グループ化された項目のすべての可能な組み合わせを対象に小計を計算するには(たとえば、クロス集計レポートを作成するため)、代わりに「[GROUP BY CUBE](#)」を参照してください。

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 `expri` 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)` の明示的な別名は `min` です。`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 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。

API バージョン 18.0 以降では、`GROUP BY ROLLUP` を `SUM()` や `COUNT(fieldName)` などの[集計関数](#)と共に使用できます。構文は次のとおりです。

```
[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
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)
```

次の表に、クエリ結果を示します。第1レベルの小計行と総計行に注目してください。各行の説明をコメント列に示します。

状況	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
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

状況	LeadSource	cnt	コメント
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

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。

小計を識別する `GROUPING(fieldName)` の使用

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。

`GROUP BY ROLLUP` または `GROUP BY CUBE` 句によって小計が追加され、`GROUPING(fieldName)` 関数で行が項目の小計であるかどうかを識別されます。

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。

`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。

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

さらに詳しく理解する最も簡単な方法は、クエリとその結果を確認することです。

```
SELECT LeadSource, Rating,

      GROUPING(LeadSource) grpLS, GROUPING(Rating) grpRating,

      COUNT(Name) cnt

FROM Lead

GROUP BY ROLLUP (LeadSource, Rating)
```

このクエリは、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。

次の表に、クエリ結果を示します。各行の説明をコメント列に示します。

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
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 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。

API バージョン 18.0 以降では、GROUP BY CUBE を SUM() や COUNT(fieldName) などの **集計関数** と共に使用できます。

構文は次のとおりです。

```
[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。各行の説明をコメント列に示します。

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

Type	BillingCountry	grpType	grpCty	accts	コメント
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)
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
合計	3	1	8	12

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。

API バージョン 18.0 以降では、GROUP BY 句と一緒に HAVING 句を使用して、SUM() などの集計関数によって返される結果を絞り込みます。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。構文は次のとおりです。

```
[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
```

SOQL でサポートされる集計関数の一覧は、「[集計関数](#)」を参照してください。

HAVING を使用するときの考慮事項

`HAVING` 句を使用する SOQL クエリを作成する場合、いくつか注意が必要な考慮事項があります。

- `HAVING` 句は、集計された値で絞り込めます。また、`GROUP BY` 句に含まれる任意の項目で絞り込むこともできます。その他の項目値で絞り込む場合、その検索条件を `WHERE` 句に追加します。たとえば、次のクエリは有効です。

```
SELECT LeadSource, COUNT(Name)

FROM Lead

GROUP BY LeadSource

HAVING COUNT(Name) > 100 and LeadSource > 'Phone'
```

次のクエリは、`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%'
```

- 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 句に準結合または反結合を含めることはできません。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。

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クエリできます。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。

-  **メモ:** TYPEOF は、現在 SOQL 多態性機能の一部の開発者プレビューとして利用可能です。組織での TYPEOF の有効化については、Salesforce にお問い合わせください。

TYPEOF は、API バージョン 26.0 以降で使用できます。

```
SELECT [fieldList,]

    [TYPEOF typeOfField

        {WHEN whenObjectType THEN whenFieldList} [...]

        [ELSE elseFieldList]

    END] [...]

FROM objectType
```

複数の多態的なリレーション項目をクエリする必要がある場合、1つの 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
<i>fieldList</i>	指定した <i>objectType</i> 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。これは SELECT ステートメントで使用される項目の標準リストで、多態的なリレーションのオブジェクト種別に関係なく使用されます。多態的なリレーションで参照されるオブジェクトの項目のみが必要な場合、SELECT ステートメントからこのリストを除外できます。この

構文	説明
	項目リストは、同じクエリで使用する <i>typeOfField</i> 項目でも参照されるリレーション項目は参照できません。
<i>typeOfField</i>	複数のオブジェクト種別を参照できる、 <i>objectType</i> の多態的なリレーション項目、または <i>objectType</i> の親の多態的な項目。たとえば、Event の What リレーション項目には Account、Campaign、Opportunity のいずれかを使用できます。 <i>typeOfField</i> は、 <i>fieldList</i> でも参照されるリレーション項目を参照できません。
<i>whenObjectType</i>	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、 <i>typeOfField</i> 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を確認されます
<i>whenFieldList</i>	指定された <i>whenObjectType</i> 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。
<i>elseFieldList</i>	WHEN 句のすべてのオブジェクト種別が <i>typeOfField</i> 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。
<i>objectType</i>	クエリするオブジェクトの種別を指定します。これは、SELECT ステートメントで必須の標準オブジェクト種別です。

TYPEOF の使用について、次の点に注意してください。

- 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 () や [集計クエリ](#) 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、次の準結合句は有効です。

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

- GROUP BY、GROUP BY ROLLUP、GROUP BY CUBE、および HAVING は、TYPEOF を使用する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 `TYPEOF` のその他の例は、「[多態的なキーとリレーションについて](#)」を参照してください。

FOR VIEW

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。SOQL クエリで `FOR VIEW` 句を使用し、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。

Salesforce では、レコード参照に関する情報がインターフェースに保存され、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。`FOR VIEW` 句を使用すると、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場合に、Salesforce に通知して、最近参照したデータを常に最新のものにすることができます。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` 句と `FOR REFERENCE` 句を組み合わせて使用することを検討してください。

この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。

メモ:

- 取得されたレコードをログインユーザが確実に参照する場合にのみ、この句を使用してください。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。

次に、取引先責任者を1件取得して現在のユーザに表示し、`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する SOQL クエリの例を示します。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。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、この句を FOR VIEW 句と一緒に使用することを検討してください。

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。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 REFERENCE 句を FOR VIEW 句と一緒に使用することを検討してください。

この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。

メモ:

- クエリの影響を受けるレコードが確実に参照される場合にのみ、この句を使用してください。参照されないと、取得されたレコードの最近の参照に関する情報が、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。

次に、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 項目を更新する SOQL クエリの例を示します。

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

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。

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。 <pre>SELECT CampaignId, AVG (Amount) FROM Opportunity GROUP BY CampaignId</pre> API バージョン 18.0 以降で利用できます。
COUNT () および COUNT (<i>fieldName</i>)	クエリ条件に一致する行数を返します。COUNT () を使用した例: <pre>SELECT COUNT () FROM Account WHERE Name LIKE 'a%'</pre> COUNT (<i>fieldName</i>) を使用した例: <pre>SELECT COUNT (Id) FROM Account WHERE Name LIKE 'a%'</pre>  メモ: SOQL の COUNT (Id) は、SQL の COUNT (*) に相当します。 COUNT (<i>fieldName</i>) 構文は、API バージョン 18.0 以降で使用できます。GROUP BY 句を使用している場合は、COUNT () の代わりに COUNT (<i>fieldName</i>) を使用します。詳細は、「COUNT () および COUNT (<i>fieldName</i>)」を参照してください。
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。 <pre>SELECT COUNT_DISTINCT (Company) FROM Lead</pre>  メモ: SOQL の COUNT_DISTINCT (<i>fieldName</i>) は、SQL の COUNT (DISTINCT <i>fieldName</i>) に相当します。オブジェクトのすべての個別値(null を含む) をクエリするには、「GROUP BY」を参照してください。 API バージョン 18.0 以降で利用できます。

集計関数	説明
MIN()	項目の最小値を返します。次に例を示します。 <pre>SELECT MIN(CreatedDate), FirstName, LastName FROM Contact GROUP BY FirstName, LastName</pre> 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。 API バージョン 18.0 以降で利用できます。
MAX()	項目の最大値を返します。次に例を示します。 <pre>SELECT Name, MAX(BudgetedCost) FROM Campaign GROUP BY Name</pre> API バージョン 18.0 以降で利用できます。
SUM()	数値項目の合計を返します。次に例を示します。 <pre>SELECT SUM(Amount) FROM Opportunity WHERE IsClosed = false AND Probability > 60</pre> API バージョン 18.0 以降で利用できます。

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
```

COUNT() および COUNT(fieldName)

COUNT() 句 (省略可能) を SOQL クエリの SELECT ステートメントに使用すると、クエリが返す行数を取得できます。

COUNT() の構文には次の 2 つのバージョンがあります。

- COUNT()
- COUNT(fieldName)

GROUP BY 句を使用している場合は、COUNT() の代わりに COUNT(fieldName) を使用します。

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、クエリ結果の `size` 項目が行数を返します。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 () 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。代わりに COUNT (*fieldName*) を使用します。
- API バージョン 19.0 以降では、COUNT () と GROUP BY 句は一緒に使用できません。代わりに COUNT (*fieldName*) を使用します。

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。この構文は COUNT () より新しく、API バージョン 18.0 以降で使用できます。

次に例を示します。

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

COUNT (*Id*) は COUNT () と同じ数を返します。したがって前のクエリと次のクエリは同等になります。

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



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

COUNT(**fieldName**) の場合、records 項目の AggregateResult 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%'
```

このクエリの場合、この数は AggregateResult オブジェクトの `expr0` 項目で返されます。詳細は、「[GROUP BY での別名の使用](#)」を参照してください。

COUNT() の代わりに COUNT(**fieldName**) を使用することには、利点があります。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
```

COUNT() とは異なり、API バージョン 18.0 以降では GROUP BY 句と COUNT(**fieldName**) を一緒に使用できません。これにより、レコードを分析して、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
```

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

データ型	AVG ()	COUNT ()	COUNT_DISTINCT ()	MIN ()	MAX ()	SUM ()
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。[DescribeObjectResult](#) に関連付けられた Field オブジェクトには、GROUP BY 句に項目を含めることができるかどうかを定義する groupable 項目があります。

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)
```

日付関数は、API バージョン 18.0 以降で使用できます。

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、「[日付関数のタイムゾーンの変換](#)」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
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

日付関数	説明	例
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。  メモ: 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。Salesforce ヘルプの「会計年度について」を参照してください。	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 Salesforce オンラインヘルプの「会計年度の設定」を参照してください。
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。  メモ: 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。Salesforce ヘルプの「会計年度について」を参照してください。	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。  メモ: 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。Salesforce ヘルプの「会計年度について」を参照してください。	2009
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)
```

日付関数のタイムゾーンの変換

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
```

マルチ通貨組織の通貨項目のクエリ

通貨項目をユーザ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)
```

以下に例を示します。

```
SELECT Id, convertCurrency(AnnualRevenue)

FROM Account
```

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、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。

`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 USD5000 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。上記の例を使用すると、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。そのため、`convertCurrency()` は `ORDER BY` と共には使用できません。

クエリに `GROUP BY` 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。`convertCurrency()` 関数をコールして集計関数の結果をユーザの通貨に換算することはできません。

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
```

SELECT 句の例

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

検索タイプ	例
リレーションクエリ: 子-親	<pre>SELECT Contact.FirstName, Contact.Account.Name FROM Contact</pre> <pre>SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'</pre>
リレーションクエリ: 親-子	<pre>SELECT Name, (SELECT LastName FROM Contacts) FROM Account</pre> <pre>SELECT Account.Name, (SELECT Contact.LastName FROM Account.Contacts) FROM Account</pre>
WHERE を使用するリレーションクエリ	<pre>SELECT Name, (SELECT LastName FROM Contacts WHERE CreatedBy.Alias = 'x') FROM Account WHERE Industry = 'media'</pre>
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	<pre>SELECT Id, FirstName__c, Mother_of_Child__r.FirstName__c FROM Daughter__c WHERE Mother_of_Child__r.LastName__c LIKE 'C%'</pre>
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	<pre>SELECT Name, (SELECT Name FROM Line_Items__r) FROM Merchandise__c WHERE Name LIKE 'Acme%'</pre>
多態的なキーを使用するリレーションクエリ	<pre>SELECT Id, Owner.Name FROM Task WHERE Owner.FirstName like 'B%'</pre> <pre>SELECT Id, Who.FirstName, Who.LastName FROM Task WHERE Owner.FirstName LIKE 'B%'</pre> <pre>SELECT Id, What.Name FROM Event</pre>
TYPEOF を使用する多態的なリレーションクエリ	<pre>SELECT TYPEOF What WHEN Account THEN Phone, NumberOfEmployees WHEN Opportunity THEN Amount, CloseDate ELSE Name, Email END FROM Event</pre>  メモ: TYPEOF は、現在 SOQL 多態性機能の一部の開発者プレビューとして利用可能です。組織での TYPEOF の有効化については、Salesforce にお問い合わせください。
集計を使用するリレーションクエリ	<pre>SELECT Name, (SELECT CreatedBy.Name FROM Notes) FROM Account</pre> <pre>SELECT Amount, Id, Name, (SELECT Quantity, ListPrice, PricebookEntry.UnitPrice, PricebookEntry.Name FROM OpportunityLineItems) FROM Opportunity</pre>
単純なクエリ: 各ユーザの UserId と LoginTime	<pre>SELECT UserId, LoginTime from LoginHistory</pre>
特定時間範囲のユーザあたりのログイン回数を使用するリレーションクエリ	<pre>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</pre>

 **メモ:** Apex では、SOQL ステートメントや SOSL ステートメントをオンザフライで使用するには、大括弧で囲む必要があります。さらに、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。

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辿ります。

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 におけるリレーションクエリは、有効なリレーションを辿る必要があります。

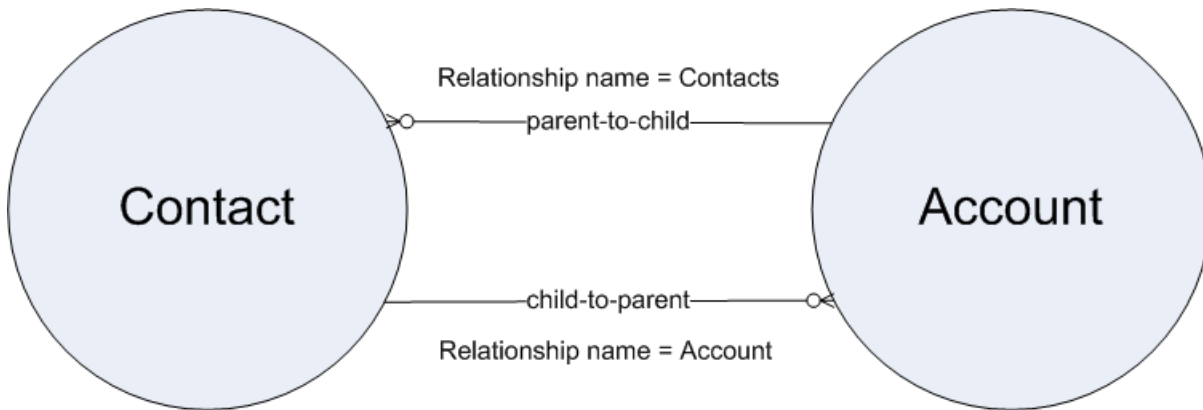
たとえば、リレーションクエリを使用すると、ある種別のオブジェクトを別の種別のオブジェクトに適用される条件に基づいて返すことができます。たとえば、「Bob Jones によって作成されたすべての取引先と、その取引先に関連付けられた取引先責任者を返す」ことができます。オブジェクトを接続している親子または子親リレーションがなければなりません。「Bob Jones によって作成された取引先とユーザをすべて返す」のような、任意のクエリを記述することはできません。

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](#)
- [リレーションクエリの使用](#)
- [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](#)
- [クエリ結果について](#)
- [参照関係と外部結合](#)
- [親子リレーションの識別](#)
- [多態的なキーとリレーションについて](#)
- [リレーションクエリ制限について](#)
- [履歴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](#)
- [Partner WSDL とリレーションクエリの使用](#)

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辿れ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、辿られます。

```
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、取引先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 句内のみを辿ることができます。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いくらか異なります。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クエリできます。

SOQL を使用して次のリレーションのクエリを行うことが可能です。

- 子-親リレーションのクエリを行ってください (多対一の場合が多くなります)。SELECT、FROM、または WHERE 句内で、ドット (.) 演算子を使用して、これらのリレーションを直接指定してください。

以下に例を示します。

```
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。

- 親子のクエリを行ってください (それはほとんど常に一対多です)。サブクエリ内の FROM 句の初期のメンバーが外部クエリ FROM 句の初期のメンバーの場合、サブクエリ (括弧で囲まれた) を使用して、これらのリレーションを指定してください。サブクエリでは、オブジェクトの複数形の名前が各オブジェクトのリレーション名であるため、複数形の名前を指定する必要があります。

次に例を示します。

```
SELECT Name,

(

    SELECT LastName

    FROM Contacts

)

FROM Account
```

クエリはすべての取引先に関して名前を返します。そして各取引先に関しては、取引先責任者の姓を返します。

- 集計クエリ内の外部キーとしての親子リレーションを辿ってください。

次に例を示します。

```
SELECT Name,

(

    SELECT CreatedBy.Name

    FROM Notes

)

FROM Account
```

このクエリは組織内の取引先を返します。そして、(結果セットが空でない場合) メモを作成したユーザ名と一緒に、取引先に関して、取引先の名前、それらの取引先のメモを返します(メモがない場合、空の結果セットの可能性あります)。

- 同様の例で、集計クエリ内の親子リレーションを辿ってください:

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

同じクエリを使用して、(項目のデータを示す)商品群を指定することによって、Product2 の値を得ることができます。

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

- クエリ(サブクエリを含む)には WHERE 句を含めることができます。WHERE 句は、現在のクエリの FROM 句のオブジェクトに適用されます。これらの句は親子リレーション経由で、現在の範囲(クエリのルート要素から到達可能)のどのようなオブジェクト上でも絞り込みを行うことが可能です。

次に例を示します。

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

このクエリは、業種がメディアであるすべての取引先の名前を返し、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を返します。

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の名前を使用して、クエリがリレーションを辿る場合、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。

Daughter Help for this Page ?

New Relationship

Step 3. Enter the label and name for the lookup field
Step 3 of 6

[Previous](#)
[Next](#)
[Cancel](#)

Field Label Mother of Child i

Field Name Mother_of_Child i

Description Relationship Mother of Child

Help Text i

Child Relationship Name Daughters i

Required ☒ Always require a value in this field in order to save a record

What to do if the lookup record is deleted?

☐ Clear the value of this field. You can't choose this option if you make this field required.

☐ Delete this record also. This allows users to delete large numbers of records without regard for sharing or visibility constraints.

☒ Don't allow deletion of the lookup record that's part of a lookup relationship.

[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
Field Name	Mother_of_Child
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。

サブクエリの結果でも、通常のクエリ結果と同様に、子の数が多い場合、すべてのレコードを取得するには `queryMore()` の使用が必要になることがあります。たとえば、取引先にサブクエリを含むクエリを発行する場合、クライアントアプリケーションは、次のようにサブクエリからの結果も処理する必要があります。

1. `Account` オブジェクトにクエリを実行します。
2. `queryMore()` で取引先 `QueryResult` を反復処理します。
3. 各取引先オブジェクトについては、取引先責任者オブジェクトの `QueryResult` を取得します。
4. 各取引先責任者の `QueryResult` で `queryMore()` を使用して子取引先責任者を反復処理します。

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

API バージョン 13.0 以降では、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。

動作の変化は、リレーションクエリの次のタイプにあてはまります。

- ORDER BY 句では、レコードの外部キーが null の場合、レコードはバージョン 13.0 以降では返されますが、13.0 より前のバージョンでは返されません。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、バージョン 13.0 以降では返されます。

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。バージョン 13.0 以降では、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。それより前のバージョンでは、そのようなレコードは、クエリによって返されません。

- OR を使用する WHERE 句でレコードの外部キーの値が null の場合、バージョン 13.0 以降ではレコードが返されますが、13.0 より前のバージョンではレコードが返されません。たとえば、組織に、LastName 項目の値が foo に一致し、AccountId 項目の値が null の取引先責任者オブジェクトが存在し、かつ、LastName 項目の値が foo とは異なり、bar という名前の親取引先オブジェクトを持つ別の取引先責任者オブジェクトが存在するとします。この場合に以下のクエリを実行すると、bar と等しい LastName 項目の値を持った取引先責任者オブジェクトのみが返されます。

```
SELECT Id FROM Contact WHERE LastName = 'foo' or Account.Name = 'bar'
```

親取引先のない取引先責任者は条件を満たす姓が含まれているため、バージョン 13.0 以降では返されます。

- 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、バージョン 13.0 以降ではレコードが返されますが、13.0 より前のバージョンでは返されません。以下に例を示します。

```
SELECT Id

FROM Case

WHERE Contact.LastName = null
```

ケースレコード Id 値は、バージョン 13.0 以降では返されますが、13.0 より前のバージョンでは返されません。

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『SalesforceObjectReference』の「DataModel」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"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

        <element name="AccountId" nillable="true" minOccurs="0"
            type="tns:ID"/>

        ...

    </sequence>

</extension>

</complexContent>

</complexType>

```

 **メモ:** すべてのリレーシヨンがAPIで公開されるわけではありません。リレーシヨ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、リレーシヨンのサフィックス `__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>

```

多態的なキーとリレーションについて

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オブジェクトのいずれかにすることができます。

たとえば、Event の What リレーション項目には Account、Campaign、Opportunity のいずれかを使用できます。多態的なリレーションでクエリを実行またはレコードを更新するときには、リレーションに設定されている実際のオブジェクト種別を確認し、それに応じた操作を行う必要があります。多態的なリレーションには、次の方法でアクセスできます。

- リレーションの多態的なキーを使用する。
- クエリで TYPEOF 句を使用する。
- 多態的な項目で Type 修飾子を使用する。

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。それぞれの方法について次に説明します。

-  **メモ:** TYPEOF は、現在 SOQL 多態性機能の一部の開発者プレビューとして利用可能です。組織での TYPEOF の有効化については、Salesforce お問い合わせください。

多態的なキーの使用

多態的なキーでは、その ID で複数の種別のオブジェクトを親として参照できます。たとえば、取引先責任者もリードも ToDo の親となることができます。つまり、ToDo の whoId 項目には、取引先責任者かリードのいずれかの ID が含まれる可能性があります。オブジェクトが複数のオブジェクト種別を親に持つことができる場合、多態的なキーは 1 つのオブジェクト種別のかわりに Name オブジェクトを参照します。

describeSObjects() コールを実行すると、Name オブジェクトが返され、このオブジェクトの項目 Type には、クエリしたオブジェクトの親に設定可能なオブジェクト種別のリストが含まれます。DescribeSObjectResult 内の namePointing 項目は、リレーションが Name オブジェクトを参照することを示します。これはリレーションが多態的であるために必要です。たとえば、ToDo レコードの whoId 項目の値は、取引先責任者である場合と、リードである場合が考えられます。

-  **メモ:** 組織で SOQL 多態性機能が有効になっていると、多態的なリレーション項目は Name ではなく、sObject を参照します。

親のオブジェクト種別が不明なリレーションをトラバースするには、次の項目を使用してクエリを作成できます。

- owner: この項目は、親のオブジェクト種別に関係なく、子オブジェクトを所有している親のオブジェクトを表します。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、カレンダーまたはユーザ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、子に関連付けられた親のオブジェクト種別を表します。

```
SELECT Id, Who.FirstName, Who.LastName

FROM Task

WHERE Owner.FirstName LIKE 'B%'
```

このクエリ例は、所有者がカレンダーまたはユーザのいずれかで、「who」で表される親が取引先責任者またはリードのいずれかである 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、取引先責任者に関連付けられたすべての ToDo をクエリすることもできます。

```
SELECT Id, Who.Id, Who.Type
```



```
FROM Task

WHERE Who.Type='Contact'
```

- what: この項目は、子に関連付けられた親のオブジェクト種別を表します。この場合、オブジェクトは人物以外(つまり、取引先責任者、リード、ユーザ以外)を表します。

```
SELECT Id, What.Name

FROM Event
```

このクエリ例は、取引先かソリューション、またはその他のオブジェクト種別のいずれかを親とする行動に対して機能します。

`describeSObjects()` を使用してオブジェクトの親と子に関する情報を取得することもできます。詳細は、`describeSObjects()` を参照してください。特に `namePointing` が `true` に設定されている場合は、項目が `Name` を参照することを示します。

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 26.0 以降で使用できます。

多態的なリレーションの各オブジェクト種別に対してクエリ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 `NumberOfEmployees` 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。

- `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 () や **集計クエリ** 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、次の準結合句は有効です。

```
SELECT

  TYPEOF What

    WHEN Account THEN Phone

    ELSE Name

  END

FROM Event

WHERE CreatedById IN

(

  SELECT CreatedById

  FROM Case

)
```

- `GROUP BY`、`GROUP BY ROLLUP`、`GROUP BY CUBE`、および `HAVING` は、`TYPEOF` を使用するクエリでは使用できません。

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 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。

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、35 個以下です。カスタムオブジェクトには最大 25 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。
- 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。
- 各指定リレーション内で、親-子リレーションの 1 つのレベルだけが 1 つのクエリ内で指定可能です。たとえば、`FROM` 句が `Account` を指定している場合、`SELECT` 句では `Contact` かそのレベルの他のオブジェクトのみを指定できます。`Contact` の子オブジェクトを指定することはできません。
- 情報を得るために、メモと添付ファイルをクエリ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、有効ではありません。

```
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
```

- 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。
 - 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、Lightning Connect の OData 2.0 アダプタ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辿るために、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 foo__history)

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。

- 一对多リレーションの `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 に記載されている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

`query()` または `queryMore()` コールで返されるバッチサイズ (クエリ結果オブジェクトで返される行数) をデフォルトの 500 行から変更できます。

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。C# クライアントアプリケーションでこの設定を変更するには、`query()` コールを呼び出す前に、`QueryOptions` コールで SOAP ヘッダー部分にバッチサイズを指定します。最大バッチサイズは 2,000 レコードです。ただし、この設定はあくまでも目安です。要求されるバッチサイズが、実際のバッチサイズになるとは限りません。パフォーマンスを最大化するために変更が行われます。

 **メモ:** SOQL ステートメントがロングテキストタイプの 2 つ以上のカスタム項目を選んだ場合、バッチサイズは 200 未満になります。これは、大きな SOAP メッセージを防止するためです。

次のサンプル Java (WSC) コードは、バッチサイズを 250 レコードに設定する方法を示しています。

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

次のサンプル C# (.NET) コードは、バッチサイズを 250 レコードに設定する方法を示しています。

```
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、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。
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。 <pre>SELECT fieldList [...] FROM objectType [USING SCOPE filterScope] [WHERE conditionExpression] [ORDER BY field {ASC DESC} [NULLS {FIRST LAST}}]</pre> • <i>fieldList</i> には、リレーションを使用しない項目のみを含めることができます。 • FROM には、オブジェクトを1つだけ含めることができます。 • WHERE では、EQUAL または IN のみ使用できます。 • ORDER BY では、項目を1つだけリストできます。 • 比較演算子には IN および = を使用できます。 • 論理演算子には AND を使用できます。
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。 • 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 (<i>fieldName</i>) 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

オブジェクト	説明
	- LIKE 演算子 - toLabel () 関数 - TYPEOF 句 - WITH 句 次の制限は、Lightning Connect の OData 2.0 アダプタ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、Lightning Connect のカスタムアダプタにのみ適用されます。 • 外部オブジェクトのロケーションベースの 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、クエリに失敗します。 - 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
NewsFeed	• ログインしたユーザに「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。
KnowledgeArticleVersion	• クエリで 1 つ以上の主キー ID を指定する場合を除き、必ず PublishStatus の値は 1 つだけ指定します。セキュリティをサポートするために、PublishStatus の値が Draft の記事は「記事の管理」権限を持つユーザにのみ表示されます。

オブジェクト	説明
	- アーカイブ済み記事のバージョンは、<code>articletype_kav</code> 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クエリするには、記事の <code>Id</code> を指定し、<code>sLatestVersion='0'</code> を設定します。 - 必ず <code>Language</code> の値は1つだけ指定します。ただし、SOQL では、<code>Id</code> または <code>KnowledgeArticleId</code> に対する条件がある場合、複数の <code>Language</code> を指定できます。
RecentlyViewed	RecentlyViewed オブジェクトは、ログインユーザ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 <code>FOR VIEW</code> または <code>FOR REFERENCE</code> 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。
TopicAssignment	ログインしたユーザに「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。 <code>LIMIT</code> 句に 1,100 件以下のレコードを指定する。 「<code>=</code>」を指定した <code>WHERE</code> 句を使用する場合に、<code>Id</code> または <code>Entity</code> 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。 <code>ORDER BY</code> 句を含めることができます。SELECT <code>HasAccess</code> の場合は <code>ORDER BY HasAccess</code>、SELECT <code>MaxAccessLevel</code> の場合は <code>ORDER BY MaxAccessLevel</code> 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、<code>LIMIT</code> 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 <code>ORDER BY</code> は使用できません。SOQL クエリでは、<code>ORDER BY</code> 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 <code>WITH UserId = {userId}</code> を含める必要があります。
Vote	<code>ParentId = [単一の ID]</code> <code>Parent.Type = [単一型]</code> <code>Id = [単一の ID]</code> <code>Id IN = [ID のリスト]</code>

アーカイブデータに対する SOQL

SOQL を使用してアーカイブ項目をクエリできます。

使用可能な SOQL コマンドのサブセットを使用してアーカイブデータを取得し、きめ細かい処理を行うことができます。WHERE 句を使用して FieldHistoryType、ParentId、および CreatedDate 項目の比較式を(必ずこの順序で)指定し、クエリを絞り込むことができます。つまり、ParentId または CreatedDate を使用して絞り込みを行う場合は、前の項目を使用した絞り込みも行う必要があります。クエリの最終的な比較式では、比較演算子 =、<、>、<=、>= のいずれかを使用できます。他のすべての比較式で利用できる演算子は = のみです。!= 演算子は使用できません。

LIMIT 句を使用して返される結果の数を制限できます。LIMIT 句を使用しない場合、最大 2,000 件の結果が返されます。queryMore() を使用すると、さらに結果のバッチを取得できます。

```
SELECT fieldList

FROM FieldHistoryArchive

[WHERE FieldHistoryType expression [AND ParentId expression[AND CreatedDate expression]]
]

[LIMIT rows]
```

例: 使用可能なクエリ

絞り込みなし

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue FROM FieldHistoryArchive
```

FieldHistoryType で絞り込み

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account'
```

FieldHistoryType および ParentId で絞り込み

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account' AND ParentId='906F00000008unAIAQ'
```

FieldHistoryType、ParentId、および CreatedDate で絞り込み

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account' AND ParentId='906F00000008unAIAQ' AND CreatedDate
> LAST_MONTH
```

次の表では、アーカイブ項目のクエリに使用できる SOQL 機能について説明します。

 **メモ:** アーカイブデータの SOQL クエリから返される数値項目にはすべて標準記数法が使用されています。これは、標準オブジェクトのエンティティ履歴内の数値項目で使用されている科学的記数法とは異なります。

表 1: アーカイブ項目に使用可能な SOQL 機能

機能	詳細
DATE LITERALS	<i>yesterday</i> 、 <i>last_week</i> など
LIMIT	
WHERE	FieldHistoryType、ParentId、および CreatedDate でのみ絞り込み

シンジケーションフィールド SOQL と対応付けの構文

シンジケーションフィールドサービスでは、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ルドは、公開サイトに対して定義できます。

クエリフィールド定義での 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ルドに関する Salesforce オンラインヘルプを参照してください。

第 2 章

Salesforce Object Search Language (SOSL)

トピック:

- [SOSL について](#)
- [SOSL 表記規則](#)
- [SOSL の構文](#)
- [FIND {SearchQuery}](#)
- [IN SearchGroup](#)
- [RETURNING FieldSpec](#)
- [WHERE conditionExpression](#)
- [ORDER BY 句](#)
- [UPDATE TRACKING](#)
- [UPDATE VIEWSTAT](#)
- [WITH DivisionFilter](#)
- [WITH DATA CATEGORY DataCategorySpec](#)
- [WITH NETWORK NetworkIdSpec](#)
- [WITH SNIPPET](#)
- [LIMIT n](#)
- [OFFSET n](#)
- [toLabel\(fields\)](#)
- [マルチ通貨組織の通貨項目のクエリ](#)
- [テキスト検索の例](#)
- [SOSL の制限](#)

Salesforce Object Search Language (SOSL) を使用してテキスト検索クエリを作成できます。

SOSL クエリは、次の環境でのテキスト検索に使用できます。

- `search()` コール
- Apex ステートメント
- Visualforce コントローラと getter メソッド
- Eclipse Toolkit の Schema Explorer

一度に 1 つのオブジェクトのみをクエリできる SOQL とは異なり、SOSL では、同時に複数のオブジェクトのテキスト、メール、および電話項目を検索できます。

SOSL について

Salesforce Object Search Language (SOSL) 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。SOSL では、カスタムオブジェクトも含め、アクセス権のあるすべてのオブジェクトを1つのクエリで検索できます。

SOSL では、ソースオブジェクトに次を指定できます。

- テキスト式
- 検索する項目範囲
- 取得するオブジェクトと項目のリスト
- ソースオブジェクトの行の選択条件

`search()` コールの `search` パラメータで全体の SOSL 式を渡します。

 **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 と SOQL の比較

Salesforce Object Query Language (SOQL) と同様、SOSL では組織の Salesforce データから特定の情報を検索できます。一度に1つのオブジェクトのみをクエリできる SOQL とは異なり、SOSL では、1つのクエリでアクセス可能なすべてのオブジェクト(カスタムオブジェクトを含む)を検索できます。APIは指定された範囲内で検索を実行し、アプリケーションのログインに使用されたユーザ権限に基づいて使用可能な情報のみを返します。

- 1つのオブジェクトのレコードを選択するには、SOQL の `query()` コールを使用します。
- 1つ以上のオブジェクトのレコードを検索するには、SOSL の `search()` コールを使用します。`search()` コールはオブジェクトのほとんどのテキスト項目を検索します。検索される項目については、「[検索範囲](#)」を参照してください。

効率的なテキスト検索の設計

検索内容が一般的すぎる場合、検索に時間がかかり多数の結果が返されます。より効率的に検索するためには、次を使用します。

- [IN 句](#) (検索する列の種類を制限)
- [RETURNING 句](#) (検索するオブジェクトを制限)
- [LIMIT 句](#) (検索結果を制限)
- [OFFSET 句](#) (検索結果をページング処理)

検索範囲

`search()` コールは、アクセス権のあるほとんどのオブジェクト(カスタムオブジェクトを含む)とテキスト項目を検索します。次のオブジェクトと項目は検索しません。

- 検索不可と定義されている (`searchable` が `false` の) 選択リストなどの要素。特定のオブジェクトが検索可能かどうかを確認するには、アプリケーションでオブジェクトに対する `describeSObjects()` コールを実行し、`DescribeSObjectResult` の `searchable` プロパティを確認します。
- 数値、日付、またはチェックボックス項目。これらの情報を検索するには、代わりに `query()` コールを使用します。
- `textarea` 項目 (`ALL FIELDS` 検索グループを使用しない場合)。
- `Account`、`Contact`、または `Opportunity` などの特定のオブジェクトに関連付けられた `Attachment` レコード。

次の実装のヒントに注意してください。

- ユーザの検索結果ページの固定表示オブジェクトを含め、ログインユーザのデフォルトの検索範囲内にいるオブジェクトの順序付きリストを取得するには、`describeSearchScopeOrder()` を使用します。
- `search()` コールは、記事のシノニム照合、ストップワード、語幹処理、およびスペル修正を提供します。多くの場合、`search()` コールはシノニム照合やストップワードなどの特殊な検索機能は提供しません。
- Apex では、SOQL ステートメントや SOSL ステートメントをオンザフライで使用するには、大括弧で囲む必要があります。さらに、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。

SOSL 表記規則

SOSL のトピックでは、特定の表記規則に従います。

次の表記規則を使用します。

規則	説明
<code>FIND Name IN Account</code>	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。
<code>FIND <i>fieldname</i> IN <i>objectname</i></code>	斜体は、変数またはプレースホルダを表します。実際の値を入力してください。
<code> </code>	パイプ文字は、選択肢となる要素を区切ります。たとえば、 <code>UPDATE TRACKING VIEWSTAT[,...]</code> 句の場合、 <code> </code> 文字は <code>UPDATE</code> の後に <code>TRACKING</code> または <code>VIEWSTAT</code> のどちらかを使用できることを示します。
<code>[LIMIT n]</code>	角括弧は、省略可能な要素を示します。たとえば、 <code>[LIMIT n]</code> は、 <code>LIMIT</code> 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、 <code>[ORDER BY <i>fieldname</i> [ASC DESC] [NULLS {FIRST LAST}]]</code> 句の場合、 <code>ASC</code> 、 <code>DESC</code> 、または <code>NULLS</code> 句を使用するには <code>ORDER BY</code> 句が必要です。

規則	説明
[...] 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

SOSL の構文

SOSL クエリは `FIND` 句 (必須) で開始します。その後に省略可能な句を使用して、オブジェクト種別、項目、データカテゴリなどに基づい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、結果が返される順序、返される行数などを決定できます。

SOSL クエリ構文は、必須の `FIND` 句とそれに続く 1 つ以上の省略可能な句で構成されます。次のような順序で指定します。

```

FIND {SearchQuery}

[ IN SearchGroup [ convertCurrency(Amount) ] ]

[ RETURNING FieldSpec [ toLabel(fields) ] ]

[ WITH DivisionFilter ]

[ WITH DATA CATEGORY DataCategorySpec ]

[ WITH SNIPPET[(target_length=n)] ]

[ WITH NETWORK NetworkIdSpec ]

[ LIMIT n ]

[ OFFSET n ]

[ UPDATE [TRACKING], [VIEWSTAT] ]

```

各項目は次のとおりです。

構文	説明
<code>FIND {SearchQuery}</code>	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区切る必要があります。 <code>SearchQuery</code> 文字列が 10,000 字を超えると、結果行は返されません。<code>SearchQuery</code> 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 <code>SearchQuery</code> を含むステートメント内の <code>AND</code> 演算子は、デフォルトの <code>OR</code> 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。
<code>IN SearchGroup</code>	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。 • <code>ALL FIELDS</code> • <code>NAME FIELDS</code> • <code>EMAIL FIELDS</code> • <code>PHONE FIELDS</code> • <code>SIDEBAR FIELDS</code> 指定されていない場合、デフォルトは <code>ALL FIELDS</code> です。<code>RETURNING FieldSpec</code> 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。<code>RETURNING</code> 句にこれらのオブジェクトのいずれか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。
<code>convertCurrency (Amount)</code>	省略可能。組織でマルチ通貨が有効になっている場合、通貨項目をユーザの通貨に換算します。
<code>RETURNING FieldSpec</code>	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。
<code>toLabel (fields)</code>	省略可能。クエリの結果がユーザの言語に翻訳されて返されます。
<code>WITH DivisionFilter</code>	省略可能。組織でディビジョンを使用する場合、<code>Division</code> 項目の値に基づいてすべての検索結果を絞り込みます。

構文	説明
<code>WITH DATA CATEGORY <i>DataCategorySpec</i></code>	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。
<code>WITH SNIPPET [(target_length=n)]</code>	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、標準のブラウザウィンドウでは通常 3 行のテキストに相当します。対象の長さに別の値 (100 ~ 500 文字) を指定するには、target_length パラメータ (省略可能) を追加します。
<code>WITH NETWORK <i>NetworkIdSpec</i></code>	省略可能。検索結果をコミュニティ ID で絞り込みます。
<code>LIMIT n</code>	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。以前のバージョンでは最大 200 行までが返されます。
<code>[UPDATE [TRACKING VIEWSTAT][,...]]</code>	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。 • <code>UPDATE TRACKING</code> 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。 • <code>UPDATE VIEWSTAT</code> は、記事の参照統計を更新します。 • <code>UPDATE TRACKING, VIEWSTAT</code> は両方を行います。

 **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の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 20,000 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 は `MALFORMED_SEARCH` 例外コードを返します。結果の行は返されません。

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)

検索エンジンは、すべての検索可能なテキストをより小さい要素 (トークン) に分解し、検索インデックスに保存します。検索を実行すると、結果には、検索語のトークンでの一致が含まれます。

スペースで区切られた複数の単語の検索は、別個のトークンに分割されます。たとえば、検索「acme inc」はトークン「acme」と「inc」に分割されます。

形態的トークン化により、中国語、日本語、韓国語、タイ語 (CJKT) などの単語間にスペースのない東アジア言語の検索で、正確な検索結果が得られるようになりました。

 **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	京都
東京都	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	都
東京	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。標準ルックアップ検索を実行する場合以外は、検索語の先頭にアスタリスクを使用しないでください。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 (\ 文字をその前に付けます)。
?	疑問符は、検索語の途中または末尾 (先頭ではない) 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。

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。

- ワイルドカードは先行する文字の種類を表します。たとえば、「aa*a」は「aaaa」と「aabceda」に一致しますが、「aa2a」や「aa.//a」には一致せず、「p?n」は「pin」と「pan」には一致しますが、「p1n」や「p/n」には一致しません。同様に、「1?3」は「123」と「143」には一致しますが、「1a3」や「1b3」には一致しません。
- ワイルドカード (*) は、中国語、日本語、韓国語、タイ語 (CJKT) での検索で、完全に一致する語句の検索を除いて、1 つの文字の最後に追加されます。
- 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
" "	完全に一致する語句を検索するには、検索語を引用符で囲みます。これは、句読点を含むテキストの検索で特に役立ちます。たとえば、 <code>"acme.com"</code> は完全に一致するテ

演算子	説明
	キスト <code>acme.com</code> を含む項目を検索します。"<code>monday meeting</code>" は完全に一致する語句 <code>monday meeting</code> を含む項目を検索します。 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	すべての検索キーワードに一致するデータを検索します。たとえば、<code>john AND smith</code> は <code>john</code> と <code>smith</code> の両方の単語を含む項目を検索します。多くの場合、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、<code>john OR smith</code> は <code>john</code> か <code>smith</code>、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、<code>john AND NOT smith</code> は単語 <code>john</code> を含み、単語 <code>smith</code> を含まない項目を検索します。
()	論理演算子と組み合わせた検索語を括弧で囲み、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。 <code>("Bob" and "Jones") OR ("Sally" and "Smith")</code>: Bob Jones または Sally Smith を検索します。 <code>("Bob") and ("Jones" OR "Thomas") and Sally Smith</code>: 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。

検索順序

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. () (丸括弧)
2. AND および AND NOT (右から左に計算されます)
3. OR

次の例は、検索文字列がどう計算されるかを示します。

検索文字列	同等の検索	検出されるデータに含まれる単語
東京都 AND 中央区 AND NOT ミーティング	東京都 AND (中央区 AND NOT ミーティング)	東京都と中央区はあるが、ミーティングはない

検索文字列	同等の検索	検出されるデータに含まれる単語
東京都 AND NOT 中央区 AND ミーティング	東京都 AND NOT (中央区 AND ミーティング)	東京都はあるが、中央区とミーティングは両方ともない
東京都 AND 中央区 OR ミーティング	(東京都 AND 中央区) OR ミーティング	東京都と中央区があるデータと、ミーティングがあるデータ
東京都 AND (中央区 OR ミーティング)	東京都 AND (中央区 OR ミーティング)	東京都と中央区があるデータと、東京都とミーティングがあるデータ

予約文字

次の文字が予約されています。

```
? & | ! { } [ ] ( ) ^ ~ * : \ " ' + -
```

テキスト検索で指定された場合に適切に解釈されるように、予約文字をエスケープする(前にバックスラッシュ \ 文字を付ける)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	<code>FIND {MyProspect OR "John Smith"}</code>
AND/ORを使用した複雑なクエリ	<code>FIND {MyProspect AND "John Smith" OR MyCompany}</code> <code>FIND {MyProspect AND ("John Smith" OR MyCompany)}</code>
AND NOT を使用した複雑なクエリ	<code>FIND {MyProspect AND NOT MyCompany}</code>
ワイルドカード検索	<code>FIND {My*}</code>
エスケープシーケンス	<code>FIND {Why not\??}</code>
無効または不完全な語句 (成功しません)	<code>FIND {"John Smith}</code>

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
```

- Force.com API の場合、`FIND` 句の値は中括弧で区切られます。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、『*Force.com Apex コード開発者ガイド*』を参照してください。

IN SearchGroup

SOSL クエリで検索するテキスト項目の種別を指定するには、`IN SearchGroup` 句 (省略可能) を使用します。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。

次のいずれかの値を指定できます (数値項目は検索不可)。指定されていない場合の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を検索します。

-  **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のいずれか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

範囲	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、これがデフォルト設定です。

範囲	説明
EMAIL FIELDS	メール項目のみを検索します。
NAME FIELDS	名前項目のみを検索します。 ほとんどの標準項目にある標準の名前項目に加え、次の標準オブジェクトでは IN NAME FIELDS を使用し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 - 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 プロパティが true に設定されています (詳細は、DescribeSObjectResult の fields パラメータの 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、すべての項目を検索する必要がある場合以外は検索範囲を指定することをお勧めします。たとえば、メールアドレスのみを検索する場合、検索を効率的にするため IN EMAIL FIELDS を指定します。

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

検索タイプ	例
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 (成功しません)	FIND {MyProspect} IN Accounts

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。

指定されていない場合の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およびカスタムオブジェクト (カスタムタブがない場合も含む) の ID を返します。返される最大数は、LIMIT *n* 句で指定された最大数または 2,000 (API バージョン 28.0 以降) のうち小さい方になります。結果では、オブジェクトはその句で指定された順序で表示されます。高度な検索とグローバル検索で検索可能な項目についての詳細は、Salesforce ヘルプの「オブジェクトで検索可能な項目」を参照してください。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 conditionExpression] [ORDER BY Clause] [LIMIT n] [OFFSET n])]

[, ObjectTypeName [(FieldList [WHERE conditionExpression] [ORDER BY Clause] [LIMIT n]
[OFFSET n])]]
```

RETURNING には次の要素を含めることができます。

名前	説明
<i>ObjectTypeName</i>	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

名前	説明
	できます。複数の <i>ObjectTypeName</i> 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 <i>ObjectTypeName</i> を繰り返すことはできません。RETURNING 句で指定されていないオブジェクトは、search() コールによって返されません。
<i>FieldList</i>	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。
WHERE <i>conditionExpression</i>	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 <i>FieldList</i> を含める必要があります。たとえば、次の構文は正しくありません。 <pre>RETURNING Account (WHERE name like 'test')</pre> 次の構文は有効です。 <pre>RETURNING Account (Name, Industry WHERE Name like 'test')</pre> 詳細は、「conditionExpression」を参照してください。
ORDER BY <i>Clause</i>	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、1つ以上の指定された項目を含む <i>FieldList</i> を指定する必要があります。たとえば、次の構文は正しくありません。 <pre>RETURNING Account (ORDER BY id)</pre> 次の構文は有効です。 <pre>RETURNING Account (Name, Industry ORDER BY Name)</pre>
LIMIT <i>n</i>	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、1つ以上の指定された項目を含む <i>FieldList</i> を含める必要があります。たとえば、次の構文は正しくありません。 <pre>RETURNING Account (LIMIT 10)</pre> 次の構文は有効です。 <pre>RETURNING Account (Name, Industry LIMIT 10)</pre>

名前	説明
<code>OFFSET n</code>	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クエリするときのみ使用できます。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、1つ以上の指定された項目を含む <i>FieldList</i> を含める必要があります。たとえば、次の構文は正しくありません。 <pre>RETURNING Account (OFFSET 25)</pre> 次の構文は有効です。 <pre>RETURNING Account (Name, Industry OFFSET 25)</pre>

 **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	<code>FIND {MyProspect}</code>
1つの sObject、項目なし	<code>FIND {MyProspect} RETURNING Contact</code>
複数の sObject オブジェクト、項目なし	<code>FIND {MyProspect} RETURNING Contact, Lead</code>
1つの sObject、1つ以上の項目	<code>FIND {MyProspect} RETURNING Account (Name)</code> <code>FIND {MyProspect} RETURNING Contact (FirstName, LastName)</code>
カスタム sObject	<code>FIND {MyProspect} RETURNING CustomObject_c</code> <code>FIND {MyProspect} RETURNING CustomObject_c (CustomField_c)</code>
複数の sObject オブジェクト、1つ以上の項目、制限	<code>FIND {MyProspect} RETURNING Contact (FirstName, LastName LIMIT 10), Account (Name, Industry)</code>
複数の sObject オブジェクト、不定数の項目	<code>FIND {MyProspect} RETURNING Contact (FirstName, LastName), Account, Lead (FirstName)</code>
検索不可の sObject オブジェクト	<code>FIND {MyProspect} RETURNING RecordType</code> <code>FIND {MyProspect} RETURNING Pricebook</code>
無効な sObject オブジェクト	<code>FIND {MyProspect} RETURNING FooBar</code>
無効な sObject 項目	<code>FIND {MyProspect} RETURNING Contact (fooBar)</code>

検索タイプ	例
1つのオブジェクトの制限	<code>FIND {MyProspect} RETURNING Contact (FirstName, LastName LIMIT 10)</code>
複数のオブジェクトの制限とクエリ制限	<code>FIND {MyProspect} RETURNING Contact (FirstName, LastName LIMIT 20), Account (Name, Industry LIMIT 10), Opportunity LIMIT 50</code>
1つのオブジェクトのオフセット	<code>FIND {MyProspect} RETURNING Contact (FirstName, LastName OFFSET 10)</code>

 **メモ:** Apex では、SOQL ステートメントや SOSL ステートメントをオンザフライで使用するには、大括弧で囲む必要があります。さらに、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。

WHERE *conditionExpression*

デフォルトでは、オブジェクトに対する SOSL クエリでは、ユーザ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。たとえば次の式は、*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
```

- クライアントアプリケーションでは、演算子をネストするときに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。

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。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、大文字と小文字を区別しない演算子。
value	fieldName の値と比較するために使用される値。指定した項目のデータ型と一致するデータ型の値を指定する必要があります。ネイティブ値を指定する必要があります。他の項目名または計算は指定できません。引用符が必要な場合は(たとえば、日付と数値には不要)、一重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。

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	指定した fieldName の値が式で指定した value に一致する場合、式は true です。等号演算子を使用する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。
!=	Not equals	指定した fieldName の値が指定した value に一致しない場合、式は true です。
<	Less than	指定した fieldName の値が指定した value より小さい場合、式は true です。
<=	Less or equal	指定した fieldName の値が指定した value 以下の場合、式は true です。
>	Greater than	指定した fieldName の値が指定した value より大きい場合、式は true です。
>=	Greater or equal	指定した fieldName の値が指定した value 以上の場合、式は true です。

演算子	名前	説明
LIKE	Like	指定した <i>fieldName</i> の値が指定した <i>value</i> のテキスト文字列の文字に一致する場合、式は <code>true</code> です。SOQL および SOSL の <code>LIKE</code> 演算子は、SQL の <code>LIKE</code> 演算子と似ています。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、ワイルドカードの使用がサポートされます。 • <code>LIKE</code> 演算子では <code>%</code> と <code>_</code> のワイルドカードがサポートされます。 • <code>%</code> ワイルドカードは、0 個以上の文字に一致します。 • <code>_</code> ワイルドカードは、1 文字のみに一致します。 • 指定した <i>value</i> のテキスト文字列は、一重引用符で囲む必要があります。 • <code>LIKE</code> 演算子は、文字列項目でのみサポートされます。 • SQL の大文字と小文字を区別する照合とは異なり、<code>LIKE</code> 演算子では大文字と小文字を区別しない照合が実行されます。 • SOQL および SOSL の <code>LIKE</code> 演算子では、特殊文字 <code>%</code> または <code>_</code> のエスケープがサポートされます。 • 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。 たとえば、次のクエリは <code>Appleton</code>、<code>Apple</code>、<code>Appl</code> と一致しますが、<code>Bappl</code> とは一致しません。 <pre>SELECT AccountId, FirstName, lastname FROM Contact WHERE lastname LIKE 'appl%'</pre>
IN	IN	値が <code>WHERE</code> 句で指定された値のいずれかに等しい場合、式は <code>true</code> です。次に例を示します。 <pre>SELECT Name FROM Account WHERE BillingState IN ('California', 'New York')</pre> <code>IN</code> の値は括弧で囲む必要があります。文字列の値は一重引用符で囲む必要があります。 <code>IN</code> と <code>NOT IN</code> は、ID (主キー) 項目または参照 (外部キー) 項目をクエリするときに、準結合および反結合でも使用できます。
NOT IN	NOT IN	値が <code>WHERE</code> 句で指定された値と等しくない場合、式は <code>true</code> です。次に例を示します。 <pre>SELECT Name FROM Account WHERE BillingState NOT IN ('California', 'New York')</pre>

演算子	名前	説明
		NOT IN の値は括弧で囲む必要があり、文字列の値は一重引用符で囲む必要があります。 この比較演算子とは無関係ですが、論理演算子の NOT もあります。
INCLUDES EXCLUDES		複数選択リストにのみ適用されます。

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	<i>fieldExpressionX</i> AND <i>fieldExpressionY</i>	<i>fieldExpressionX</i> と <i>fieldExpressionY</i> の両方が true の場合に true。
OR	<i>fieldExpressionX</i> OR <i>fieldExpressionY</i>	<i>fieldExpressionX</i> または <i>fieldExpressionY</i> のいずれかが true の場合に true。 OR 句で外部キーの値を使用したりレクエリの動作は、API のバージョンによって異なります。OR を使用する WHERE 句でレコードの外部キーの値が null の場合、バージョン 13.0 以降ではレコードが返されますが、13.0 より前のバージョンではレコードが返されません。 <pre>SELECT Id FROM Contact WHERE LastName = 'foo' or Account.Name = 'bar'</pre> 親取引先のない取引先責任者は条件を満たす姓が含まれているため、バージョン 13.0 以降では返されます。
NOT	not <i>fieldExpressionX</i>	<i>fieldExpressionX</i> 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

シーケンス	意味
\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 (%)

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'))
```

```
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'))
```

例

```

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', 'ka0D000000002HClAY'))

```

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)

```

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。

構文	説明
<code>NULLS [first last]</code>	<code>null</code> のレコードを結果の先頭または最後に配置します。デフォルトでは、 <code>null</code> 値は昇順の最後、降順の先頭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)
```

UPDATE TRACKING

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す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
```

UPDATE VIEWSTAT

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カウントを増加できます。

```
FIND {Title}

RETURNING FAQ__kav (Title WHERE PublishStatus="Online" and

language="en_US" and

KnowledgeArticleVersion = 'ka230000000PCiy')

UPDATE VIEWSTAT
```

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 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
<i>groupName</i>	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。
<i>Operator</i>	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クエリします。 - 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クエリします。 - 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クエリします。 - 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クエリします。
<i>category</i>	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。

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「記事タイプの管理」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。	<pre>FIND {tourism} RETURNING KnowledgeArticleVersion (Id, Title WHERE PublishStatus='online') WITH DATA CATEGORY Location__c AT America__c</pre>
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。	<pre>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</pre>
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。	<pre>FIND {tourism} RETURNING FAQ__kav (Id, Title WHERE PublishStatus='archived') WITH DATA CATEGORY Geography__c AT Iceland__c</pre>
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。	<pre>FIND {tourism} RETURNING KnowledgeArticleVersion (Id, Title WHERE PublishStatus='draft') WITH DATA CATEGORY Geography__c BELOW Europe__c</pre>

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 NETWORK *NetworkIdSpec*

コミュニティユーザ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。検索結果をコミュニティで絞り込む場合、各コミュニティはコミュニティ ID (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つ以上のコミュニティによる絞り込みをサポートします。
- WITH NETWORK = 'NetworkId' は、1つのコミュニティによる絞り込みのみサポートします。

ユーザ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、すべてのコミュニティ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。

- 同じコミュニティでは、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、特定のコミュニティのユーザと組織全体の取引先は、一緒に検索できません。

グループの検索結果をコミュニティ別に絞り込むには、代わりに WHERE 句を使用します。

WITH NETWORK *NetworkIdSpec* 句の例

複数のコミュニティの「テスト」という文字列を含むユーザ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')
```

1つのコミュニティ(ネットワークID)の「テスト」という文字列を含むユーザ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'
```

WITH SNIPPET

WITH SNIPPET 句(省略可能)を記事検索の SOSL クエリに追加すると、検索結果に含まれる記事について追加のコンテキストをユーザに提供できます。検索語が記事の概要項目に含まれていない場合、スニペットにより、ユーザは検索結果内で探しているコンテンツを容易に特定できるようになります。

検索の強調表示とスニペットは、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

検索の強調表示とスニペットは、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)
```

検索語、定義済みシノニム、および検索語を語幹処理した形式がスニペットの結果のコンテキスト内で タグにより強調表示されます。

 例:

```
[ {
  "attributes" : {
    "type" : "KnowledgeArticleVersion",
    "url" : "/services/data/v32.0/subjects/KnowledgeArticleVersion/kaKD0000000001MAA"
  },
  "Id" : "kaKD0000000001MAA"
  "Title" : "San Francisco"
```

```

    "Summary" : "City and County of San Francisco"

    "snippet.text" : "<em>San</em> <em>Francisco</em>, officially the City and County
of <em>San</em> <em>Francisco</em> is the... City and County of <em>San</em>
<em>Fran</em>"

    "snippet.whole.Title" : "<em>San</em> <em>Francisco</em>"
}, {

    "attributes" : {

        "type" : "KnowledgeArticleVersion",

        "url" : "/services/data/v32.0/subjects/KnowledgeArticleVersion/kaBD0000000007DMAQ"

    },

    "Id" : "kaBD0000000007DMAQ",

    "Title" : "San Francisco Bay Area",

    "Summary" : "Nine county metropolitan area",

    "snippet.text" : "The <em>SF</em> Bay Area, commonly known as the Bay Area, is a
populated region that"

    "snippet.whole.Title" : "<em>San</em> <em>Francisco</em> Bay Area"
}, {

    "attributes" : {

        "type" : "KnowledgeArticleVersion",

        "url" : "/services/data/v32.0/subjects/KnowledgeArticleVersion/ka3D00000000042OIAQ"

    },

    "Id" : "ka3D00000000042OIAQ",

    "Title" : "California",

    "Summary" : "State of California",

    "snippet.text" : "(Greater Los Angeles area and <em>San</em> <em>Francisco</em> 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検索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。

スニペットは記事についてのみ表示されます。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)。

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。

スニペットは、ワイルドカードが含まれる検索語については表示されません。

スニペットを返さない検索 (検索で記事を返さない場合、検索語にワイルドカードが含まれる場合、ユーザにスニペットが含まれる項目へのアクセス権がない場合など) に WITH SNIPPET 句を追加しても、結果にスニペットは表示されません。

スニペットは、検索結果で返される最初の 20 件の記事について表示されます。返される記事が 20 を超えると、スニペットは生成されません。

 **ヒント:** 一度に 20 件の記事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 タグ内で一致した検索語がスニペットで返されると、HTML タグはエスケープされ、一致した検索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場合、記事内の元のハイパーリンクタグはエスケープされ、「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='http://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、標準のブラウザウィンドウ表示では通常 3 行のテキストに相当します。

スニペットは、統計的に有意ではない差異範囲内で対象の長さまでの検索語が含まれるテキストの、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。

 **例:** `target_length` パラメータは、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。

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。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
```

オブジェクトの制限はクエリに含まれている順序で評価され、最大クエリ制限は各オブジェクトの処理後に調整されます。たとえば、7 件の取引先のみが上記のクエリ文字列に一致する場合、最大で 93 件の取引先責任者レコードが返されます。同様に、次のクエリが 15 件の取引先と 30 件の取引先責任者を返す場合、Opportunity オブジェクトの制限は 75 に設定されていますが、55 件の商談のみが返されます。

```
FIND {test} RETURNING Account(id LIMIT 20), Contact, Opportunity(id LIMIT 75) 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クエ

りするときのみ使用できます。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 MALFORMED_SEARCH: 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。

toLabel(fields)

SOSL クエリの結果をユーザの言語に翻訳するには、toLabel(fields) を使用します。

クライアントアプリケーションは、toLabel() を使用してユーザの言語に翻訳されて返されるクエリの結果を使用できます。

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

次に例を示します。

```
FIND {Joe} RETURNING Lead(company, toLabel(Recordtype.Name))
```

このクエリでは、リードレコードのレコードタイプ名がクエリを発行したユーザの言語に翻訳されて返されます。

- ☑ **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値または ID で絞り込みます。

`toLabel()` 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みます。次に例を示します。

```
FIND {test} RETURNING Lead(company, toLabel(Status) WHERE toLabel(Status) = 'le Draft' )
```

`Status` の選択リスト値が「leDraft」のリードレコードが返されます。ユーザの言語での値が比較されます。指定された選択リストをユーザの言語に翻訳できない場合、マスタ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)。

マルチ通貨組織の通貨項目のクエリ

通貨項目をユーザの通貨に変換するには、SOQL 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、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。

`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 USD 5001 の商談は返されますが JPY 7000 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。上記の例を使用すると、JPY 5001、EUR 5001、USD 5001 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。

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、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。商談の名前が Joe Smith または Joe Smythe の場合、その商談は返されません。

```
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 사용되는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、SOQL ステートメントや SOSL ステートメントをオンザフライで使用するには、大括弧で囲む必要があります。さらに、前にコロンの(:)がある場合は、Apex スクリプト変数と式を使用できます。

SOSL の制限

SOSL では、特定の制限が検索結果のオブジェクトまたは状況に適用されます。その結果、ユーザが探しているレコードが検索されるレコードの最大数の枠内に含まれないことがあるため、ユーザがいつも有効な一致結果すべてを取得できるとは限りません。次の概要と説明図は、検索エンジンが SOSL 検索をどのように処理するかを示します。

組織の Salesforce データは、最も関連性の高い結果がユーザに返されるように、インデックスが作成され、取得されます。

ただし、ユーザが探しているレコードが検索されるレコードの最大数に含まれない場合、ユーザはアクセス可能なすべての一致結果を見つけることができません。

検索エンジンは、次の順序で SOSL 検索を処理します (リストの番号が下記画像の番号に対応します)。

1. 検索エンジンは、最大数のレコード内で検索語との一致を探します。
2. 次に、SOSL は特定の制限をオブジェクトまたは状況に適用します。

通常、SOSL の制限は [SOQL の制限と同じ](#) です。API バージョン 28.0 以降では、返される結果の全体の制限が 2,000 件になりました。以前のバージョンでは最大 200 件の結果が返されます。

この結果セットは、検索を実行するユーザに応じて、また時間の経過と共に変化する可能性があります。

- ユーザが最近アクセスしたレコードは、検索結果に含まれる可能性が高くなります。
 - インデックスは、ユーザがレコードの削除や作成を行うと1日の間に变化する可能性があります。
3. 管理者(「すべてのデータの参照」権限を持つユーザ)には、RETURNING 句の指定に従って、返された結果セットのすべてが表示されます。
 4. 個々のユーザには、参照権限を持つレコードのみが表示されます。



-  **例:** たとえば、上記画像の各雨滴がレコード、各色がオブジェクト、じょうごの外に落ちる1つの青い雨滴が取引先レコード「Acme, Inc.」を表すとします。この例では、返される一致レコードの最大数は21です(じょうご内に落ちる雨滴の数で表される)。ユーザが「Acme, Inc.」を検索すると、検索エンジンは、一致レコード全体の制限である21に達するまで一致を検索しますが、その中には最初の5件の取引先レコードしか含まれません。「Acme, Inc.」は検索対象のレコードのセットに含まれないため、検索を行うユーザにそのレコードへのアクセス権があっても、検索結果に含まれません。

外部オブジェクトに関する SOSL の制限

- 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
```

次の制限は、Lightning Connect の OData 2.0 アダプタにのみ適用されます。

- Lightning Connect の OData 2.0 アダプタでは、FIND 句で論理演算子をサポートしていません。外部システムでは、検索クエリ文字列全体が大文字と小文字が区別される 1 つの句として受信され、英数字文字以外はすべて削除されます。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。
- また、デフォルトでは、検索クエリ文字列は \$filter システムクエリオプションの substringof 値として使用されます。次の例では、検索クエリ文字列は Acme です。

```
http://services.example.org/my.svc/Shippers?
$select=CompanyName,Phone,ShipperID&$inlinecount=allpages&
$filter=substringof('Acme',CompanyName)+eq+true+
or+substringof('Acme',Phone)+eq+true&$top=26
```

OData プロデューサのフリーテキスト検索カスタムクエリオプションを実装することをお勧めします。外部データソースの [Salesforce 検索のカスタムクエリオプション] 項目でそのクエリ文字列パラメータの名前を指定します。次の例では、カスタムクエリパラメータの名前は doSearch で、検索クエリ文字列は Acme です。

```
http://services.example.org/my.svc/Shippers?
$select=CompanyName,Phone,ShipperID&
$inlinecount=allpages&doSearch=Acme&$top=26
```

次の制限は、Lightning Connect のカスタムアダプタ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。