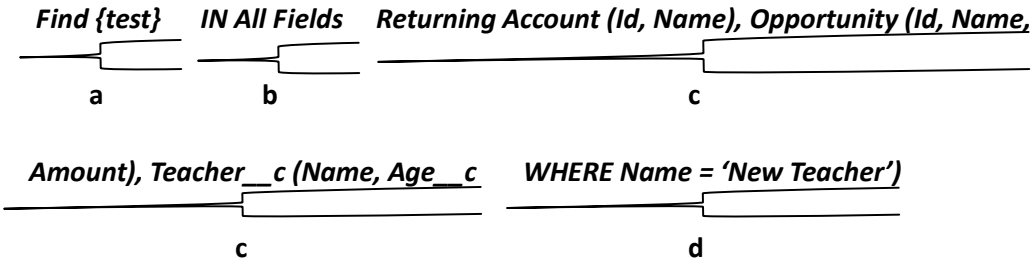
Topic	SOSL & SOQL
Created By	Siddharth Pandit (Salesforce Instructor)
Contact No.	+91 879 333 5440 / 988 12 988 13
Email	Siddharthpandit.salesforce@gmail.com

❖ SOSL (Salesforce Object Search Language):

Definition:

- Use the Salesforce Object Search Language (SOSL) to construct text searches in the search() call, in Apex statements, in Visualforce controllers and getter methods, or the Schema Explorer of the Eclipse Toolkit
- Unlike SOQL, which can query only one object at a time, SOSL enables you to search text, email and phone fields for multiple object simultaneously
- Syntax e.g. FIND {Joe Smith} IN Name Fields RETURNING lead{name, phone}

I. Syntax:



II. SOSL allows you to specify the following for source object:

a. text expression
b. scope of fields to search
c. list of objects and fields to retrieve
d. conditions for selecting rows in the source objects

III. When to Use SOSL:

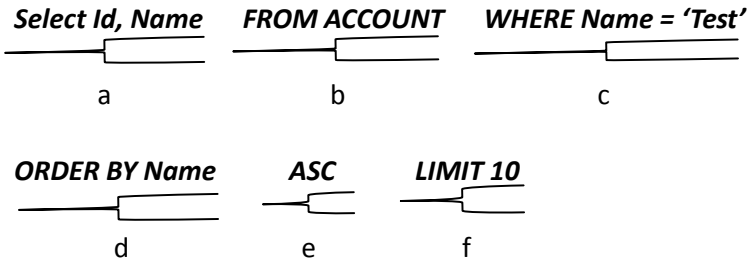
a. You don't know in which object or field the data resides and you want to find it in the most efficient way possible
b. You want to retrieve multiple objects and fields efficiently, and the objects may or may not be related to one another
c. You want to retrieve data for a particular division in an organization using the divisions feature, and you want to find it in the most efficient way possible

❖ SOQL (Salesforce Object Query Language):

Definition:

- Use SOQL to construct simple but powerful query strings in the queryString parameter in the query() call, in Apex statements, in Visualforce controllers and getter methods, or in the Schema Explorer of the Force.com IDE
- SOQL statements are similar to the select statement in SQL
- SOQL statements can be used in Apex and the Web Services API

I. Syntax:



II. SOQL allows you to specify the following for source object:

a. FieldName to be written after SELECT, multiple fields are added with comma (,)
b. ObjectName is defined after FROM
c. The query can be filtered by adding conditions using WHERE, multiple conditions can be added with OR or AND
d. Using ORDER BY the return result can be sorted. FieldName is provided to sort the result by the field
e. ASC is used to sort the result ascending order or DESC to descending order
f. Return rows can be limited by LIMIT clause.

III. SOQL statements return below data types:

a. List of sObject: Use if the SOQL statement returns multiple records
b. Single sObject: Use if the SOQL statement returns only one record
c. Integer: Use if the SOQL statement returns an Integer value
d. List of Aggregate results: Use when the SOQL statement uses a GROUP BY clause

IV. List of SOQL functions:

a. AVG()
b. COUNT()
c. COUNT_DISTINCT()

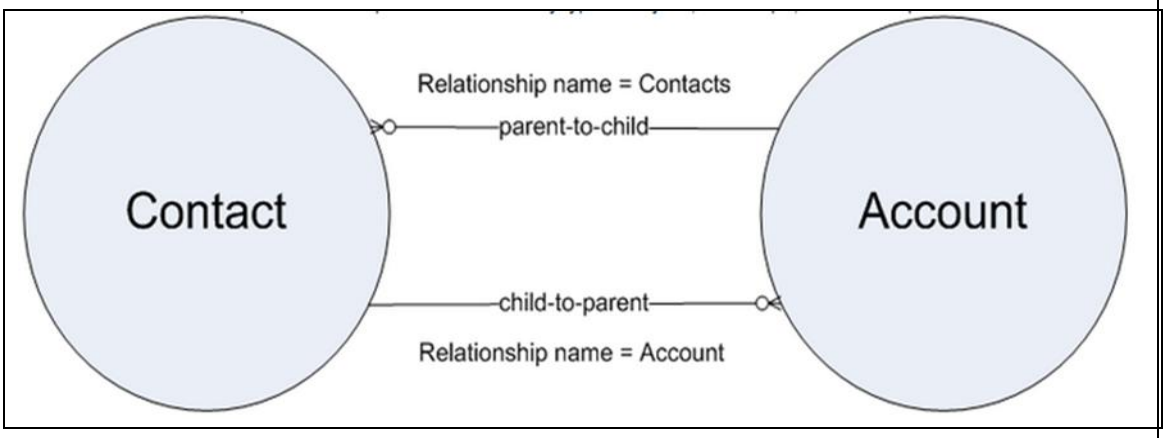
d. MIN()
e. MAX()
f. SUM()
g. CALENDAR_MONTH()
h. DAY_IN_MONTH()
i. FISCAL_YEAR()
j. WEEK_IN_YEAR()

V. SOQL Keywords:

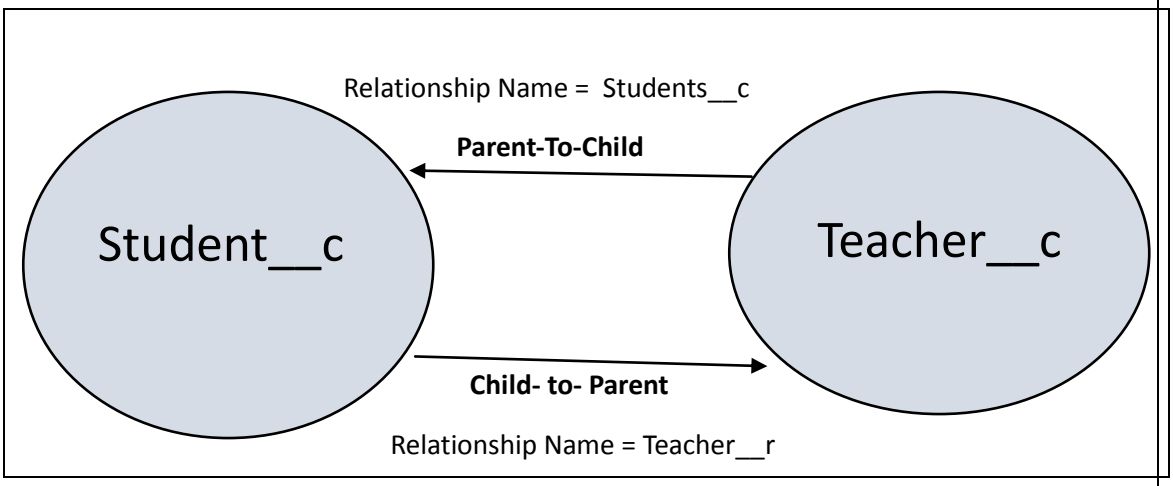
a. IN [used for bulk queries; uses a Set of a List of sObject(s) as an argument]
b. LIKE
c. AND/OR
d. NOT
e. ORDER BY
f. GROUP BY
g. LIMIT [returns the TOP record when used with ORDER BY]
h. FOR UPDATE [locks the record from being updated by another request]
i. ALL ROWS [returns active and deleted rows]

VI. Understanding Relationship Names:

Parent-to-Child and Child-To-Parent relationships exit between many types of objects.
Below is the example of Account and Contact standard objects where Account is parent object of Contact



Below is the example of Teacher and Student Custom objects where Teacher is parent object of Student



VII. Relationship Queries:

a. Child-To-Parent:

Select Id, Name , AccountId, Account.Name, Account.AnnualRevenue FROM Contact

Above query will fetch id, Name, AccountId fields from Contact (Child) object as well as Name, Annual Revenue from Account (Parent) object

b. Child-To-Parent:

Select Id, Name, (Select id, Name FROM Contacts) FROM Account

Above query will fetch Id, Name from Account (Parent) object as well as Id, Name from Contact (Child) object

VIII. Semi Joins with IN:

a. You can query values in a field where another field on the same object has a specified set of values, using IN. For example:

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

b. In addition, you can create more complex semi-joins using IN to query ID fields. For example, the following query returns account IDs if an associated opportunity is lost:

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

Notice that the subquery returns a single field of the same type as the field to which it is compared.

IX. Anti Joins with NOT IN:

a. The following example returns account IDs for all accounts that do not have any open opportunities:

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

X. Multiple Semi-Joins or Anti-Joins:

a. You can combine semi-join or anti-join clauses in a query.

For example, the following query returns account IDs that have open opportunities if the last name of the contact associated with the account is like the last name "Apple":

```
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)
```

XI. When to use SOQL:

a. You know in which objects or fields the data resides
b. You want to retrieve data from a single object or from multiple objects that are related to one another
c. You want to count the number of records that meet specified criteria
d. You want to sort results as part of the query
e. You want to retrieve data from number, date, or check-box fields

XII. Consideration while using SOQL & SOSL

a. Both SOSL search queries and SOQL WHERE filters can specify text to look for
b. When a given search can use either language, SOSL is generally faster than SOQL if the search expression uses leading wildcards or a CONTAINS term
c. In some cases, when multiple WHERE filters are being used in SOQL, indexes cannot be used even though the fields in the WHERE clause may be indexed. In this situation, decompose the single query into multiple queries each with one WHERE filter and then combine the results
d. Executing a query with a null in a WHERE filter makes it impossible to use indexing. Such queries must scan the entire database to find appropriate records. Design the data model not to rely on nulls as valid field values
e. If dynamic values are being used for the WHERE field and null values can be passed in, don't let the query run to determine there are no records; instead check for the nulls and avoid the query if necessary

XIII. Basic Limit:

a. No more than two IN or NOT IN statements per WHERE clause.

b. You cannot use the NOT operator as a conjunction with semi-joins and anti-joins. Using them converts a semi-join to an anti-join, and vice versa. Instead of using the NOT operator, write the query in the appropriate semi-join or anti-join form.
c. In the main WHERE clause, the left operand of any semi-join or anti-join query must query a single primary ID field. However, the selected field in a subquery can be a foreign key
For example:

```
SELECT Id
 FROM Idea
 WHERE (Id IN (SELECT ParentId FROM Vote WHERE CreatedDate
> LAST_WEEK))
```