Salesforce GROW

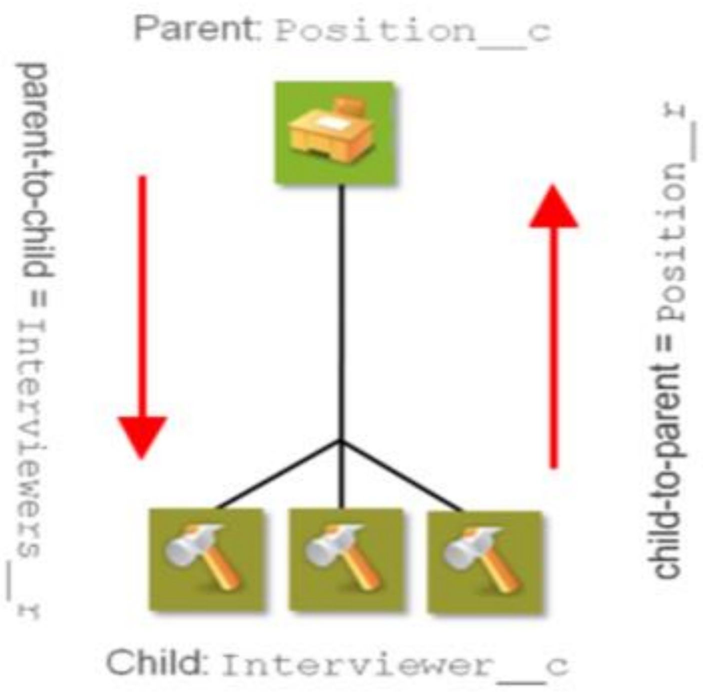
Module : Records in Database (DML, SOQL and SOSL)

Records in Database (DML, SOQL and SOSL)

sObject Relationship Naming Syntax

sObject relationships:

- Help associate sObjects with each other.
- Have different names depending on the relationship type:
  - Parent-to-child: plural version of the child object
  - Child-to-parent: singular version of the parent object
  - Custom relationships: both directions are appended with __r


Parent: Position__c
parent-to-child = Interviewers__r
child-to-parent = Position__r
Child: Interviewer__c

For more information on the API naming syntax, refer to the Force.com Web Services API Developer's Guide.

sObject Relationship: Child-to-Parent

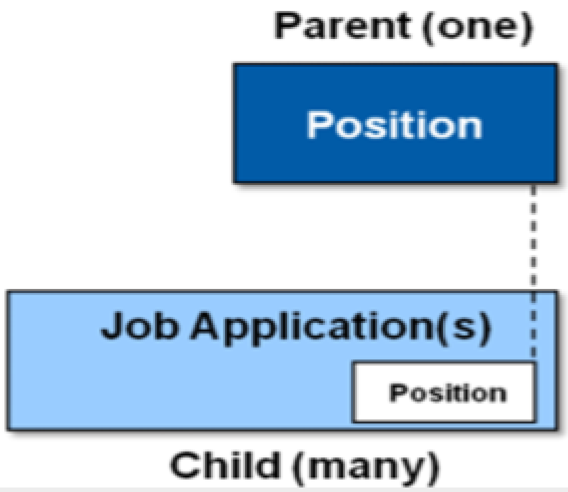
You use two member variables on the child to refer to the parent:

- The foreign key is a unique 18-digit ID of the related record.

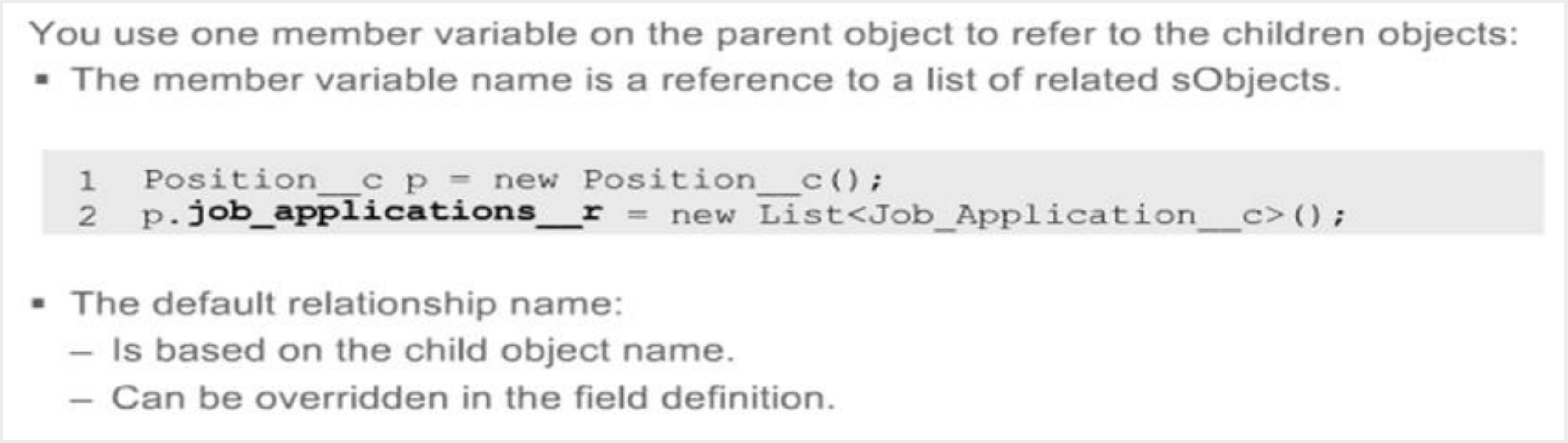
```
Job_Application__c ja1 = new Job_Application__c();
ja1.position__c = 'a05S0000000WZeYIAW';
```

- The object reference is a reference to the remote sObject object.

```
Job_Application__c ja2 = new Job_Application__c();
ja2.position__r = new Position__c();
```



sObject Relationship: Parent-to-Child

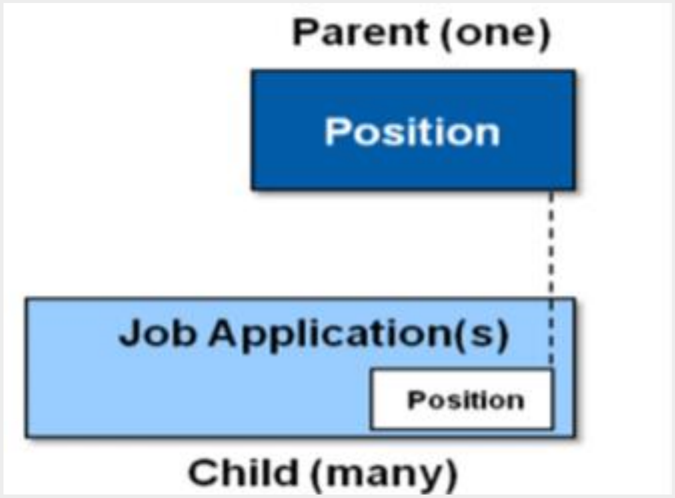


You use one member variable on the parent object to refer to the children objects:

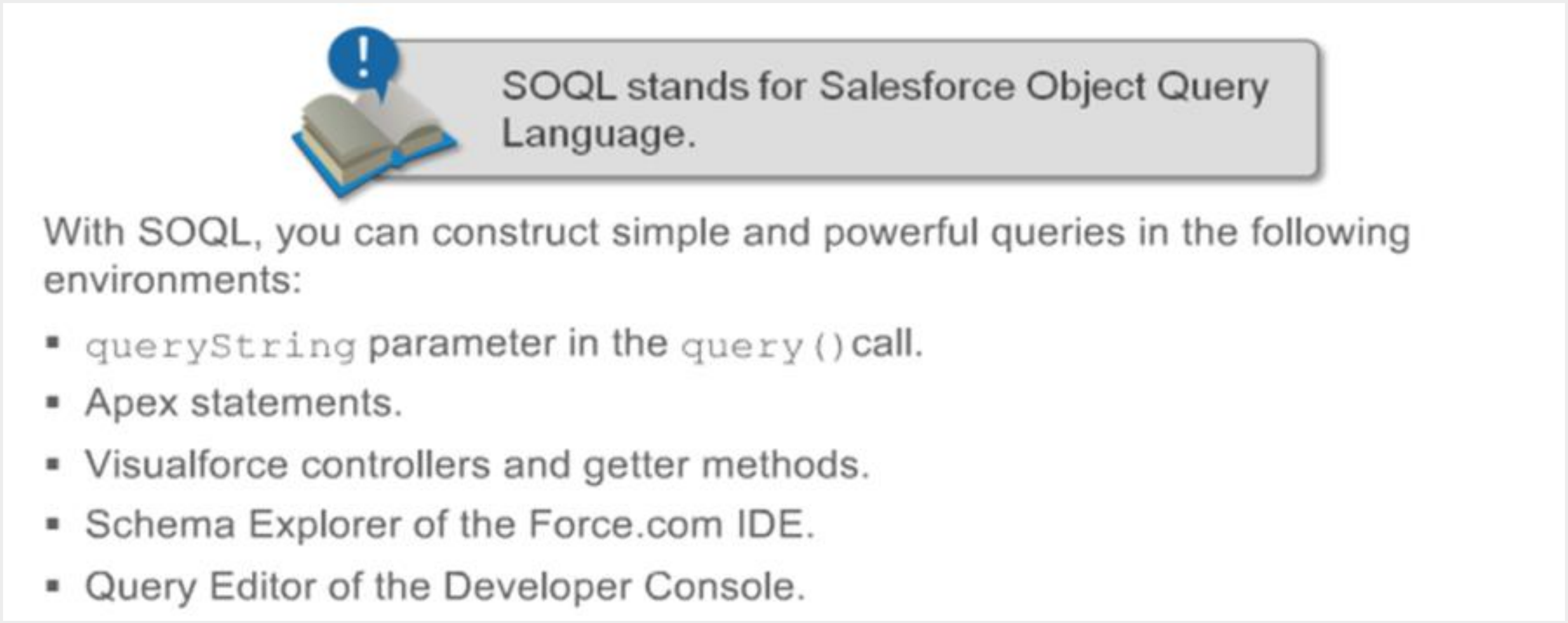
- The member variable name is a reference to a list of related sObjects.

```
Position__c p = new Position__c();
p.job_applications__r = new List<Job_Application__c>();
```

- The default relationship name:
  – Is based on the child object name.
  – Can be overridden in the field definition.



SOQL



SOQL stands for Salesforce Object Query Language.

With SOQL, you can construct simple and powerful queries in the following environments:

- `queryString` parameter in the `query()` call.
- Apex statements.
- Visualforce controllers and getter methods.
- Schema Explorer of the Force.com IDE.
- Query Editor of the Developer Console.

SOQL Return Types

List of sObjects

When more than 1 record is returned from your query

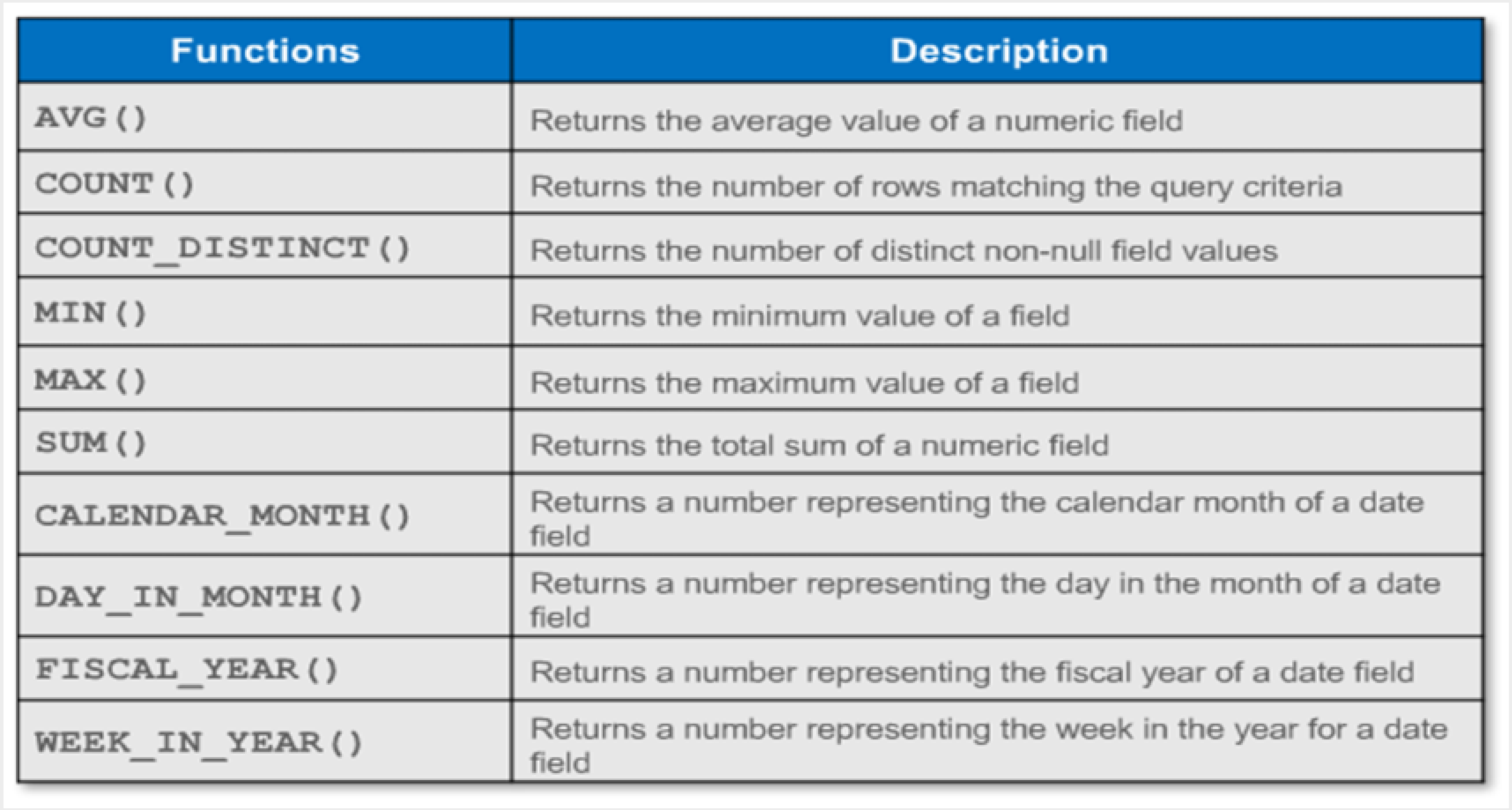
Integer

When a query returns an integer such as Count() of records in a query

Single sObject

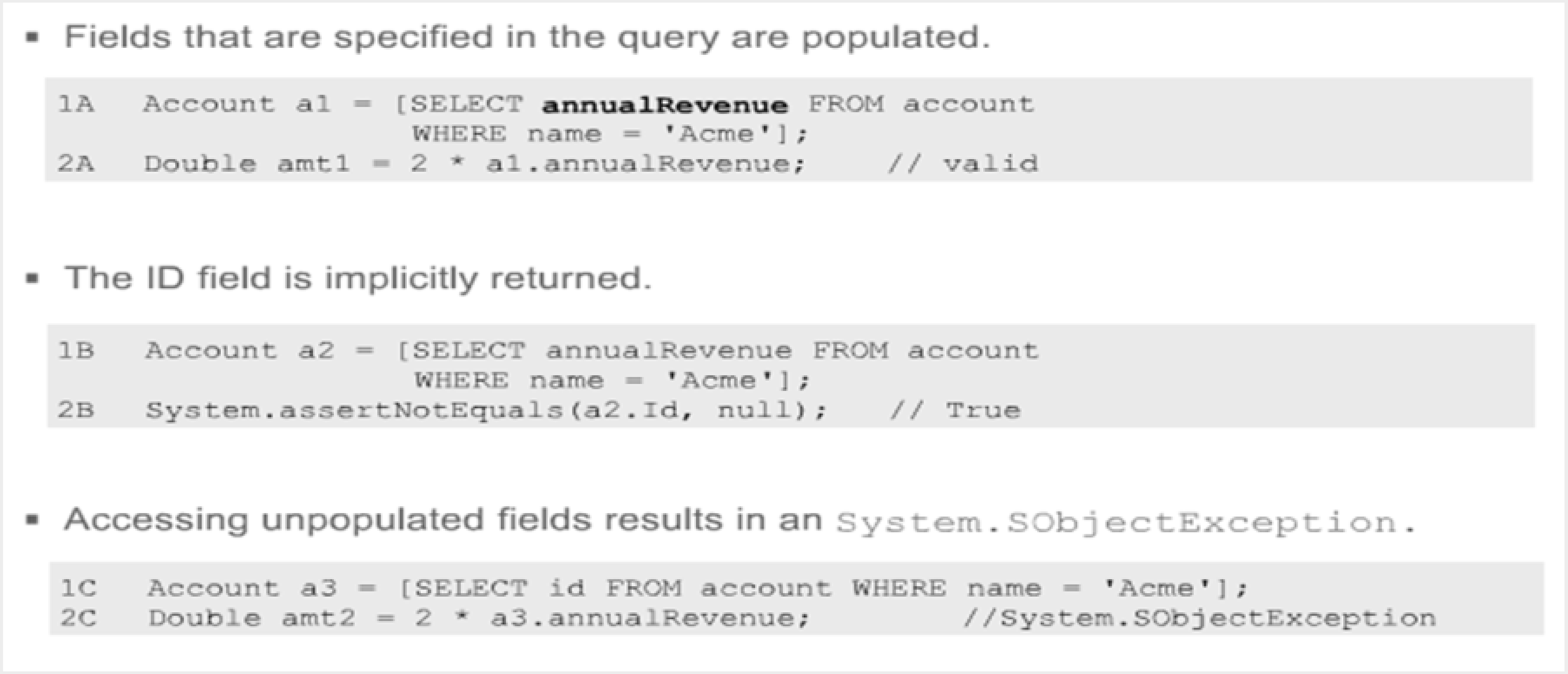
When a query returns 1 row

SOQL Functions



Functions	Description
AVG()	Returns the average value of a numeric field
COUNT()	Returns the number of rows matching the query criteria
COUNT_DISTINCT()	Returns the number of distinct non-null field values
MIN()	Returns the minimum value of a field
MAX()	Returns the maximum value of a field
SUM()	Returns the total sum of a numeric field
CALENDAR_MONTH()	Returns a number representing the calendar month of a date field
DAY_IN_MONTH()	Returns a number representing the day in the month of a date field
FISCAL_YEAR()	Returns a number representing the fiscal year of a date field
WEEK_IN_YEAR()	Returns a number representing the week in the year for a date field

SOQL Returned Data



- Fields that are specified in the query are populated.

```
1A    Account a1 = [SELECT annualRevenue FROM account
                     WHERE name = 'Acme'];
2A    Double amt1 = 2 * a1.annualRevenue;     // valid
```

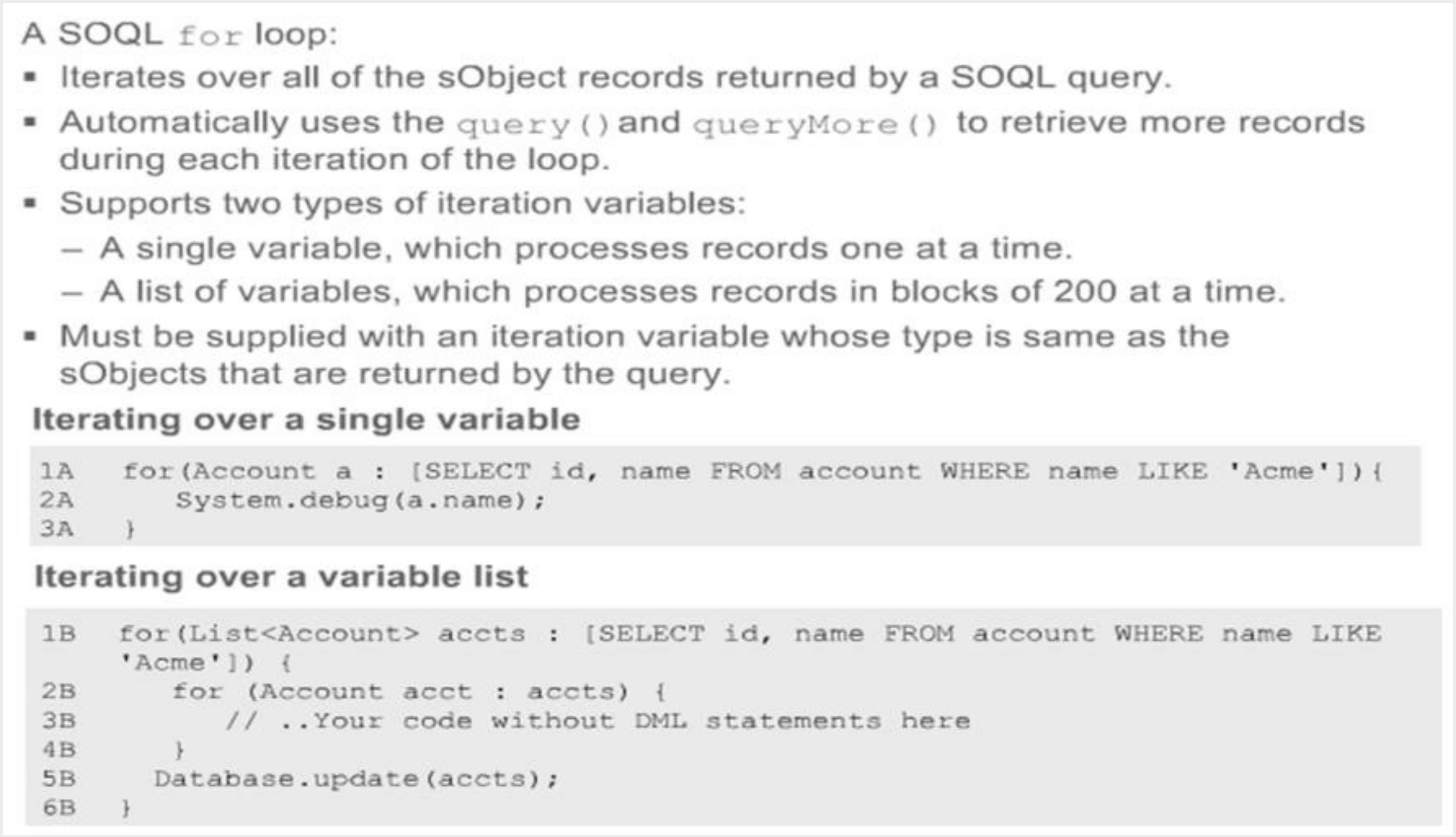
- The ID field is implicitly returned.

```
1B    Account a2 = [SELECT annualRevenue FROM account
                     WHERE name = 'Acme'];
2B    System.assertNotEquals(a2.Id, null);    // True
```

- Accessing unpopulated fields results in an `System.SObjectException.`

```
1C    Account a3 = [SELECT id FROM account WHERE name = 'Acme'];
2C    Double amt2 = 2 * a3.annualRevenue;         //System.SObjectException
```

The SOQL *for* Loop Statement



A SOQL `for` loop:

- Iterates over all of the sObject records returned by a SOQL query.
- Automatically uses the `query()` and `queryMore()` to retrieve more records during each iteration of the loop.
- Supports two types of iteration variables:
  - A single variable, which processes records one at a time.
  - A list of variables, which processes records in blocks of 200 at a time.
- Must be supplied with an iteration variable whose type is same as the sObjects that are returned by the query.

Iterating over a single variable

```
for(Account a : [SELECT id, name FROM account WHERE name LIKE 'Acme']){
    System.debug(a.name);
}
```

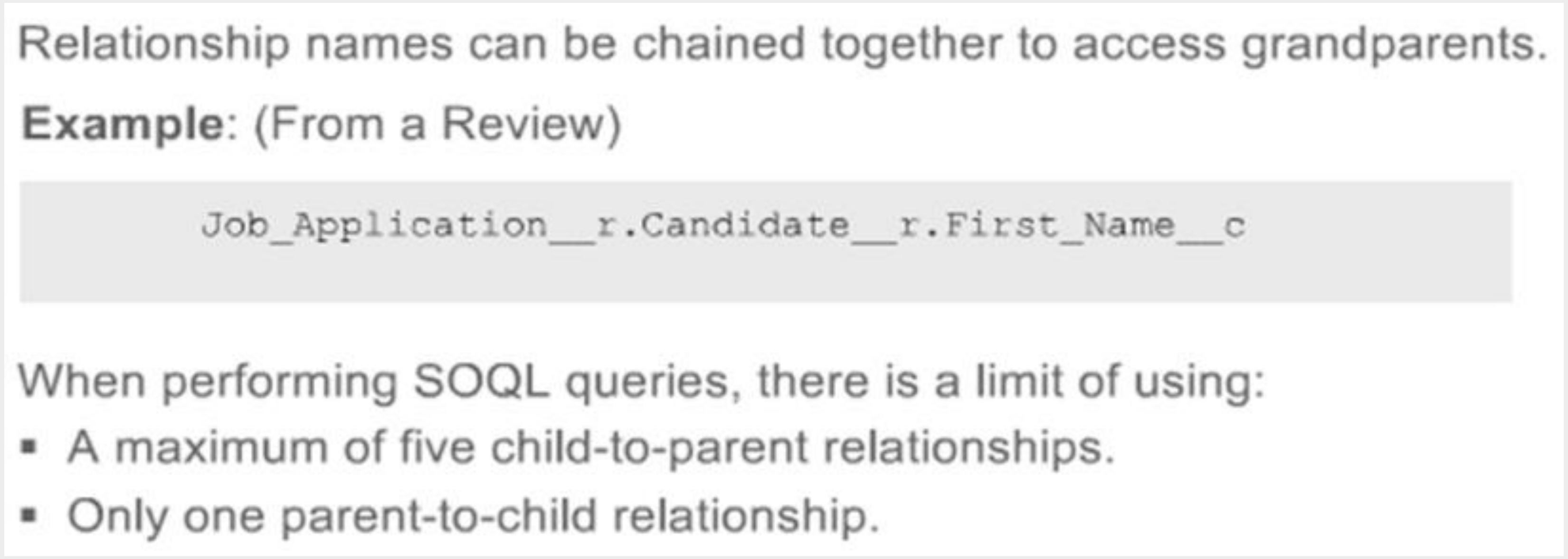
Iterating over a variable list

```
for(List<Account> accts : [SELECT id, name FROM account WHERE name LIKE
'Acme']) {
    for (Account acct : accts) {
        // ..Your code without DML statements here
    }
  Database.update(accts);
}
```

SOQL Join Terminology

SQL Join	SOQL Join
Left Outer Join	All Parents
Right Outer Join	All Children
Inner Join	All Parents with Children or All Children that have parents
Left anti-join	All Parents without Children
Right anti-join	All Children without Parents

Multilevel-Relationships



Relationship names can be chained together to access grandparents.

Example: (From a Review)

```
Job_Application__r.Candidate__r.First_Name__c
```

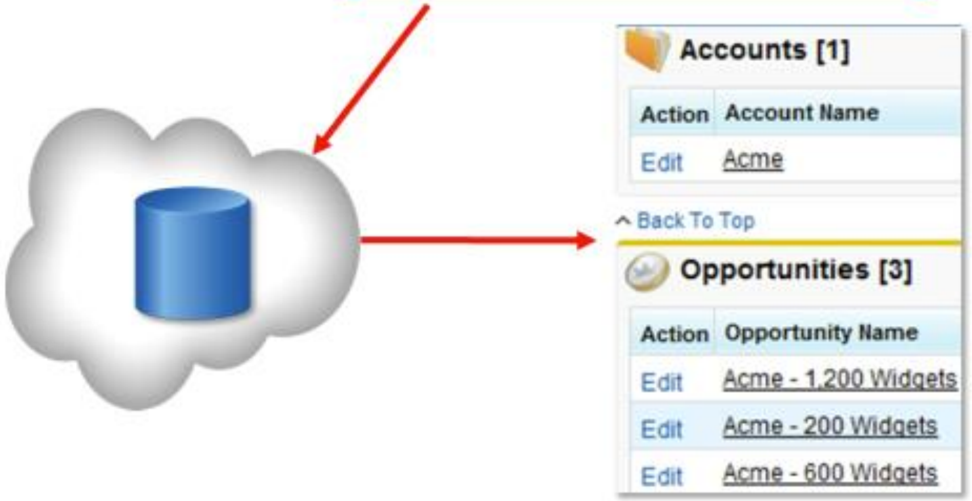
When performing SOQL queries, there is a limit of using:

- A maximum of five child-to-parent relationships.
- Only one parent-to-child relationship.

SOSL

- SOSL stands for Salesforce Object Search Language.
- SOSL queries:
  - Enable text searches across multiple sObjects.
  - Return a list of lists of sObjects.
  - Search all searchable fields (excluding dates, ids, and numbers), by default.
  - Return only IDs, by default.

```
List<List<sObject>> acmeObjects = [ FIND  'Acme'
                                    RETURNING Account(Name),
                                              Opportunity(Name) ];
```



Accounts [1]

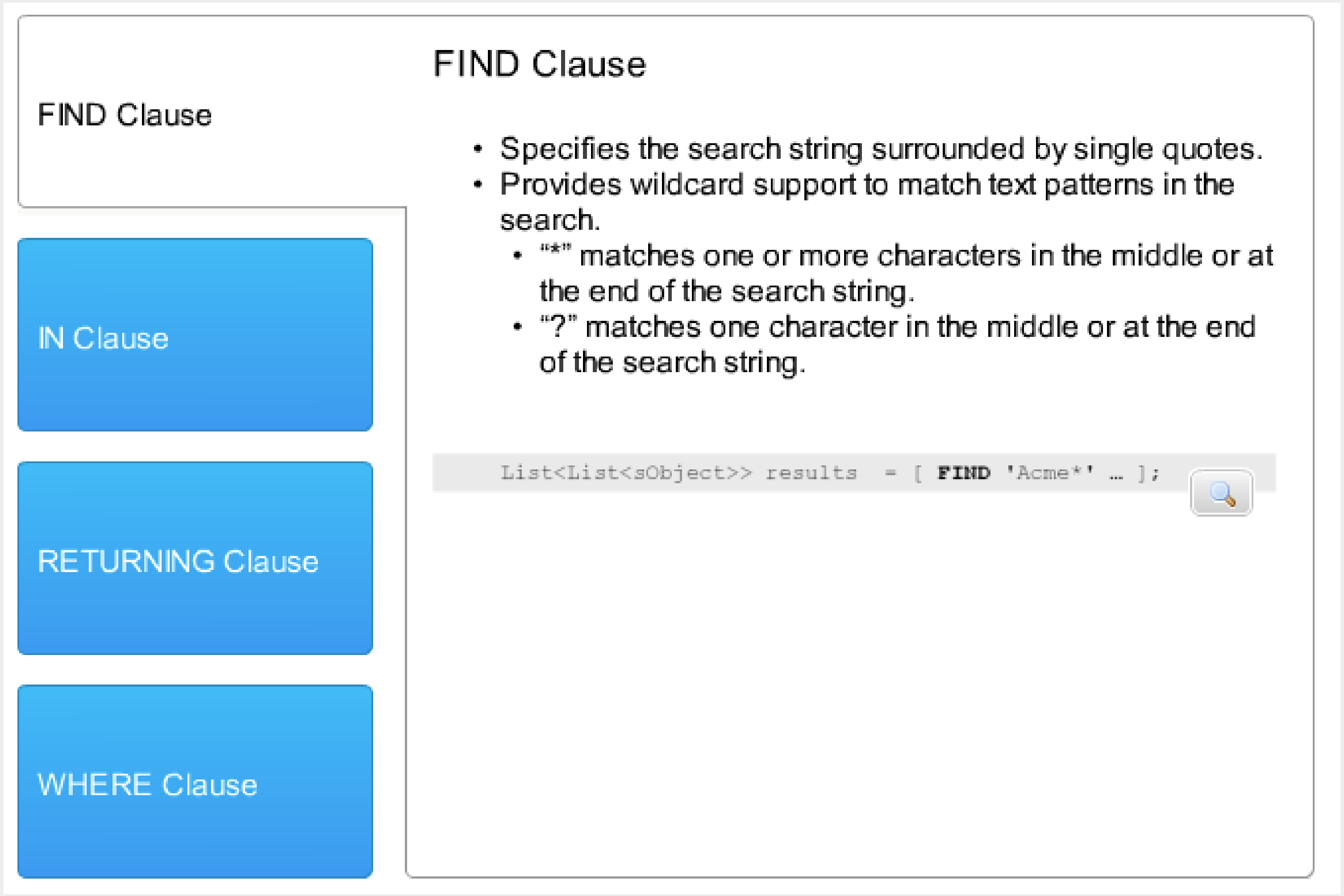
Action	Account Name
Edit	Acme

^ Back To Top

Opportunities [3]

Action	Opportunity Name
Edit	Acme - 1,200 Widgets
Edit	Acme - 200 Widgets
Edit	Acme - 600 Widgets

SOSL Syntax



FIND Clause

IN Clause

RETURNING Clause

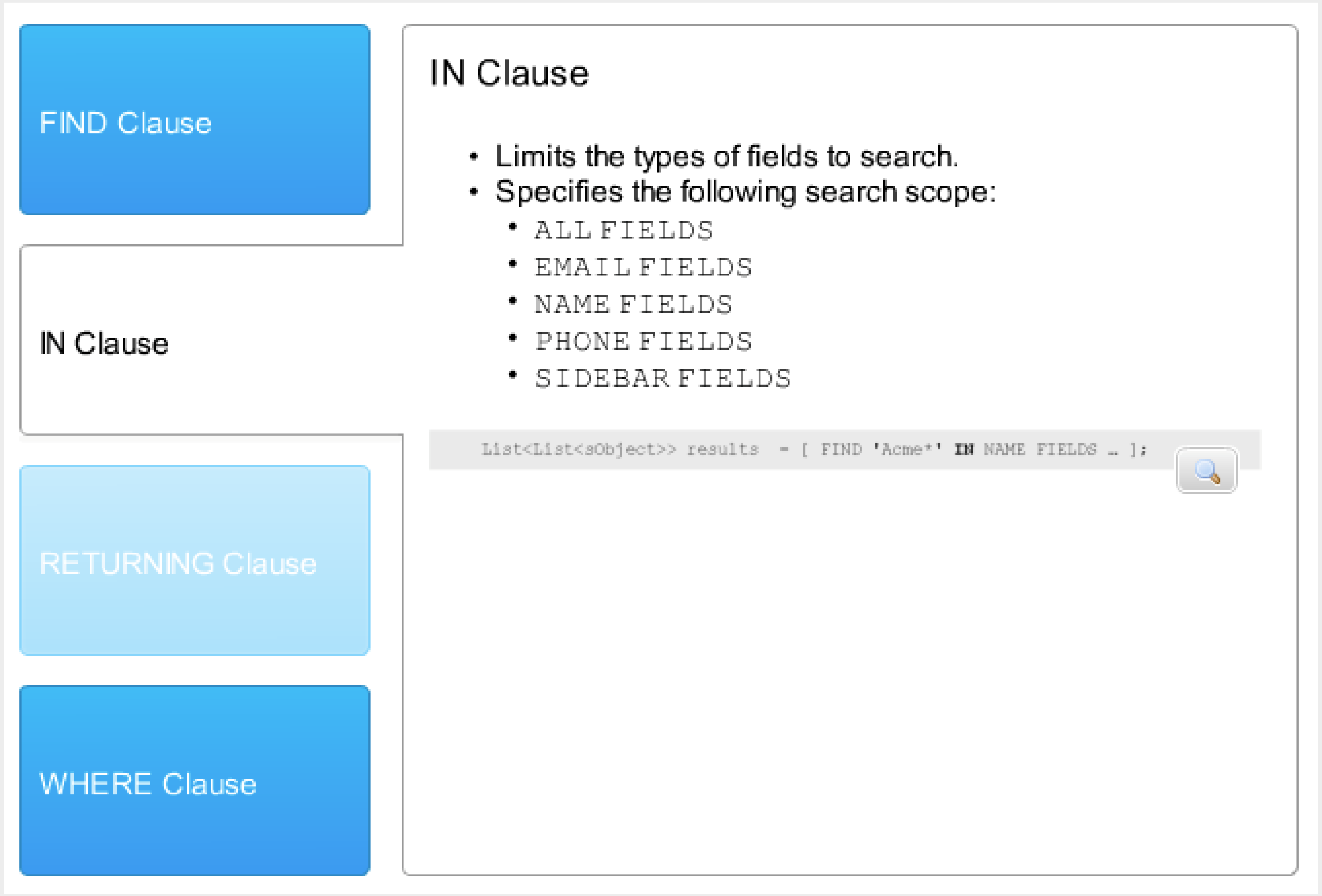
WHERE Clause

FIND Clause

- Specifies the search string surrounded by single quotes.
- Provides wildcard support to match text patterns in the search.
  - "*" matches one or more characters in the middle or at the end of the search string.
  - "?" matches one character in the middle or at the end of the search string.

```
List<List<sObject>> results  = [ FIND 'Acme*' … ];
```

SOSL Syntax



FIND Clause

IN Clause

RETURNING Clause

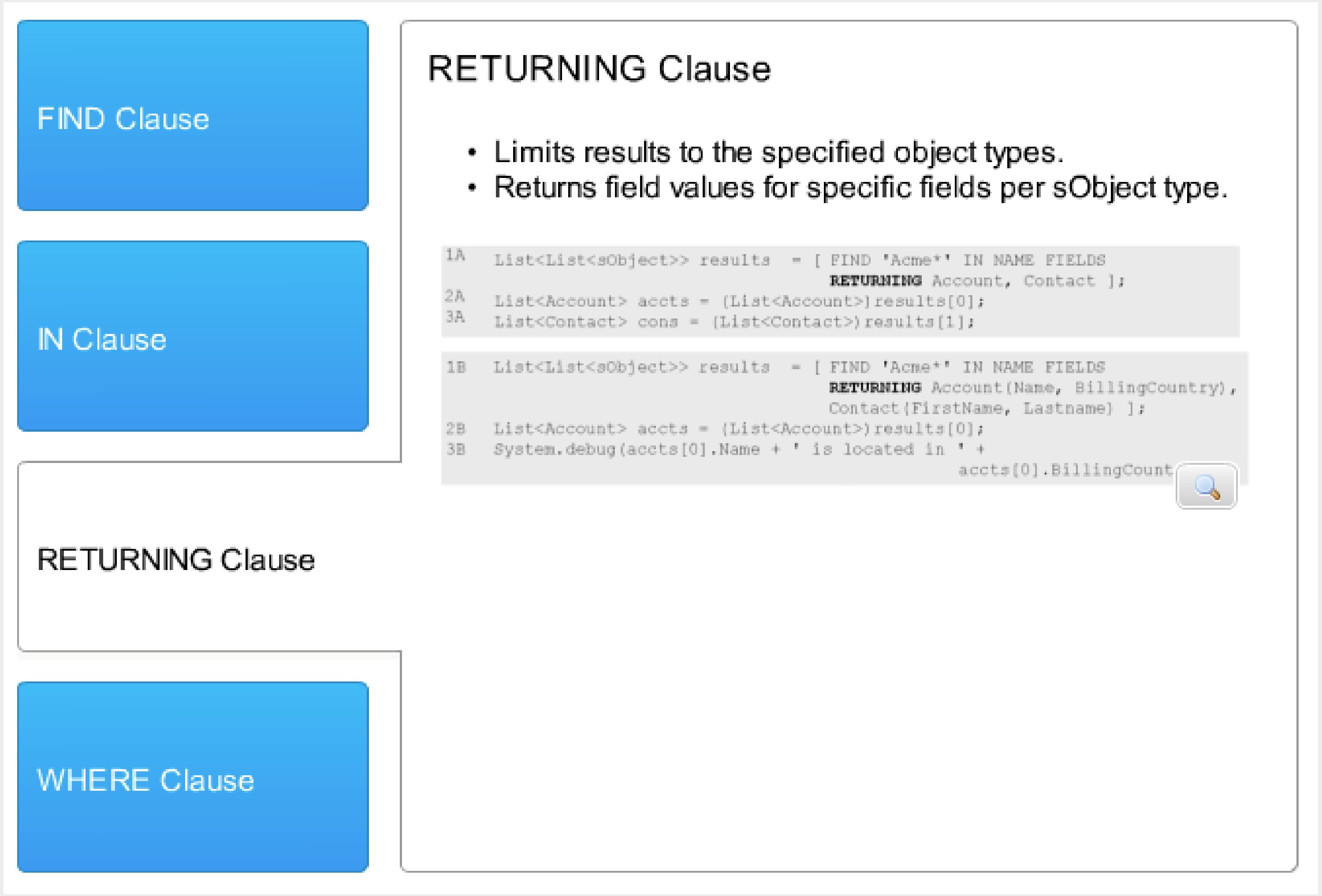
WHERE Clause

IN Clause

- Limits the types of fields to search.
- Specifies the following search scope:
  - ALL FIELDS
  - EMAIL FIELDS
  - NAME FIELDS
  - PHONE FIELDS
  - SIDEBAR FIELDS

```
List<List<sObject>> results = [ FIND 'Acme*' IN NAME FIELDS … ];
```

SOSL Syntax



FIND Clause

IN Clause

RETURNING Clause

WHERE Clause

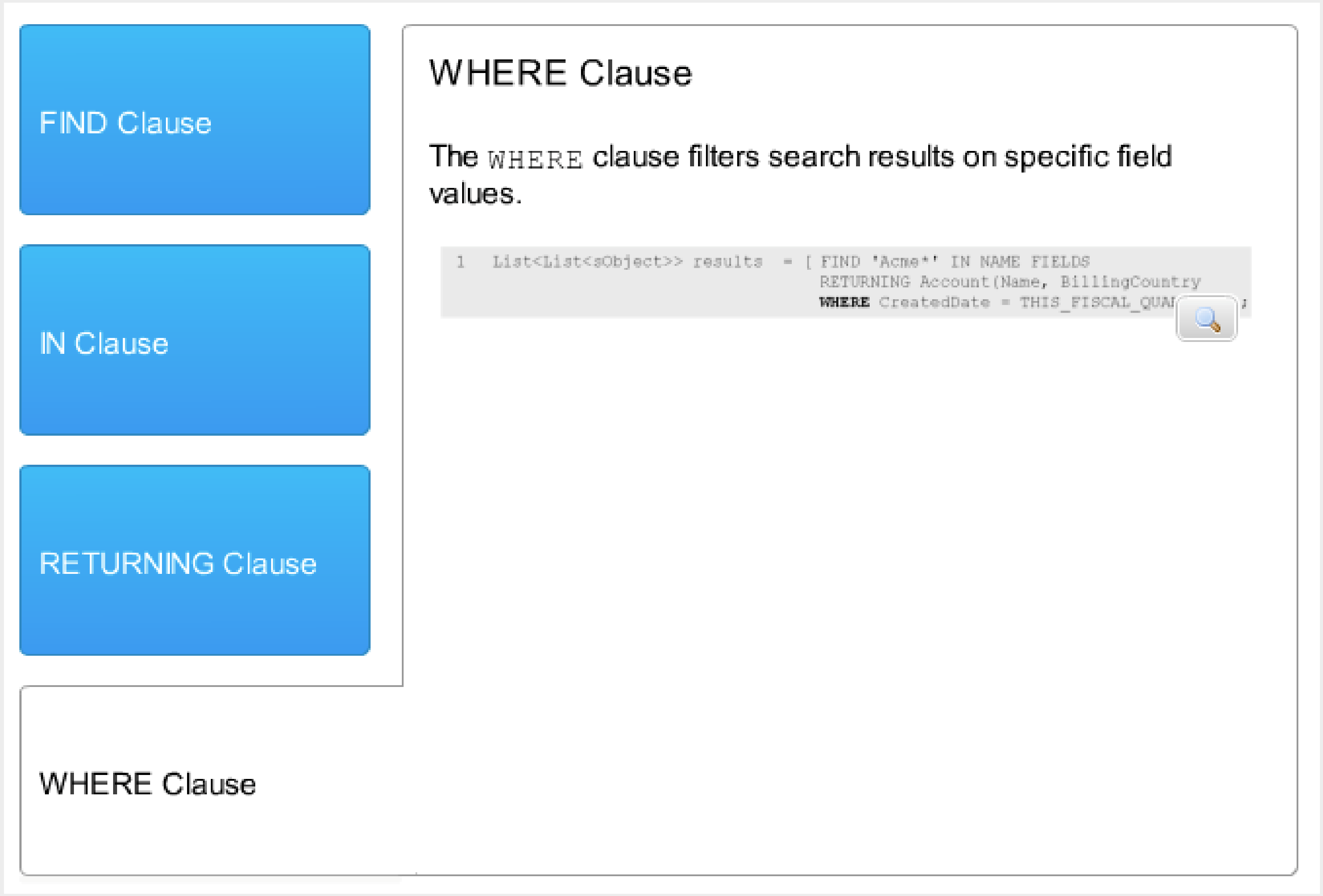
RETURNING Clause

- Limits results to the specified object types.
- Returns field values for specific fields per sObject type.

```
List<List<sObject>> results  = [ FIND 'Acme*' IN NAME FIELDS
                                 RETURNING Account, Contact ];
List<Account> accts = (List<Account>)results[0];
List<Contact> cons = (List<Contact>)results[1];
```

```
List<List<sObject>> results  = [ FIND 'Acme*' IN NAME FIELDS
                                 RETURNING Account(Name, BillingCountry),
                                 Contact(FirstName, Lastname) ];
List<Account> accts = (List<Account>)results[0];
System.debug(accts[0].Name + ' is located in ' +
                                           accts[0].BillingCount
```

SOSL Syntax



FIND Clause

IN Clause

RETURNING Clause

WHERE Clause

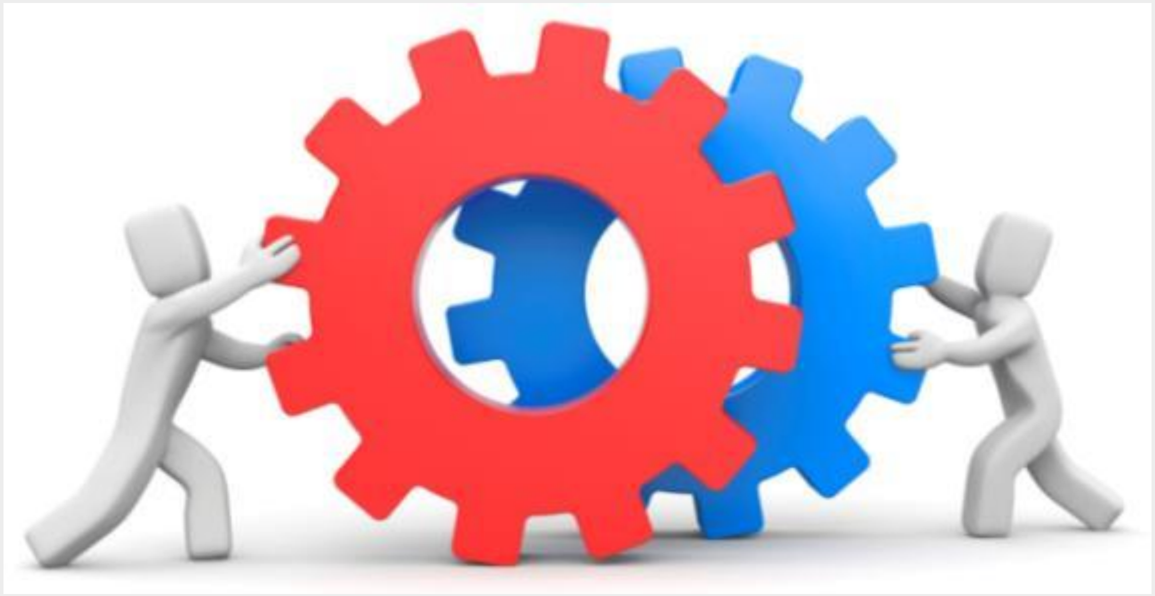
WHERE Clause

The `WHERE` clause filters search results on specific field values.

```
List<List<sObject>> results  = [ FIND 'Acme*' IN NAME FIELDS
                                 RETURNING Account(Name, BillingCountry
                                 WHERE CreatedDate = THIS_FISCAL_QUA[illegible];
```

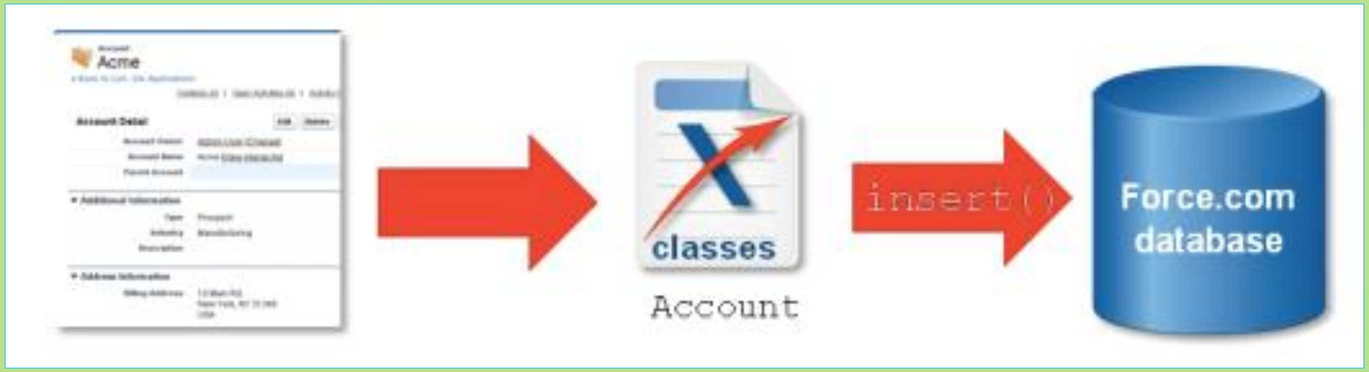
SOQL or SOSL Syntax

SOQL is best when:
The sObject is predetermined
Result Data needs to be joined
You want to query beyond text, email, or phone fields
SOSL is best when
Searches need to occur across multiple sObject types



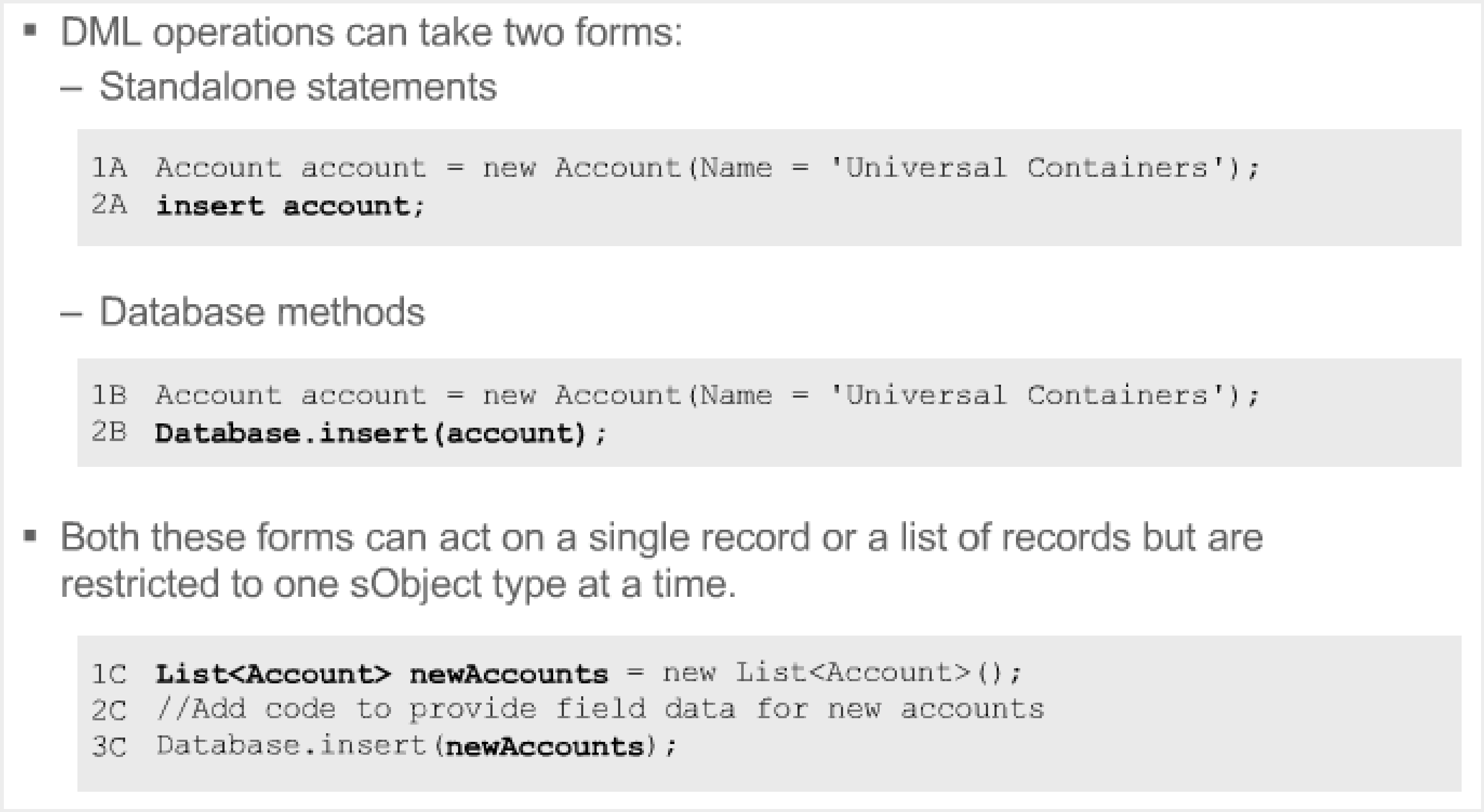
DML

- DML stands for Data Manipulation Language.



- The DML statements such as insert `insert`, `update`, `upsert`, `delete`, `merge`, and `undelete` can be used to perform database operations.
- The DML statements can take a single sObject or an sObject list.

DML Standalone Statements vs Database Methods



- DML operations can take two forms:
  - Standalone statements

```
Account account = new Account(Name = 'Universal Containers');
insert account;
```

  - Database methods

```
Account account = new Account(Name = 'Universal Containers');
Database.insert(account);
```

- Both these forms can act on a single record or a list of records but are restricted to one sObject type at a time.

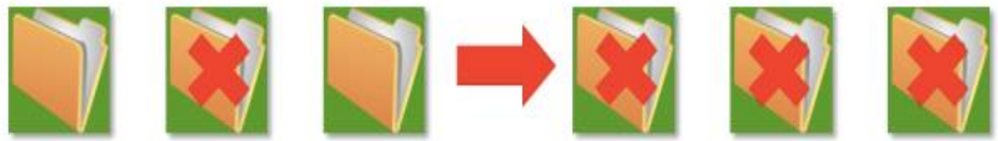
```
List<Account> newAccounts = new List<Account>();
//Add code to provide field data for new accounts
Database.insert(newAccounts);
```

DML Standalone Statements vs Database Methods

DML Standalone Statement

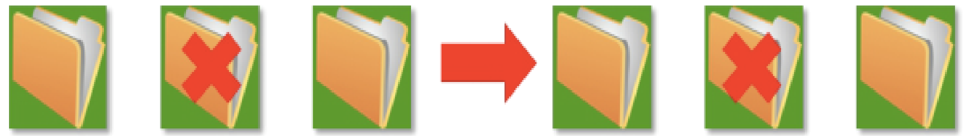
Example: `insert accounts;`

If an error is encountered with any one record, then all of the records fail.



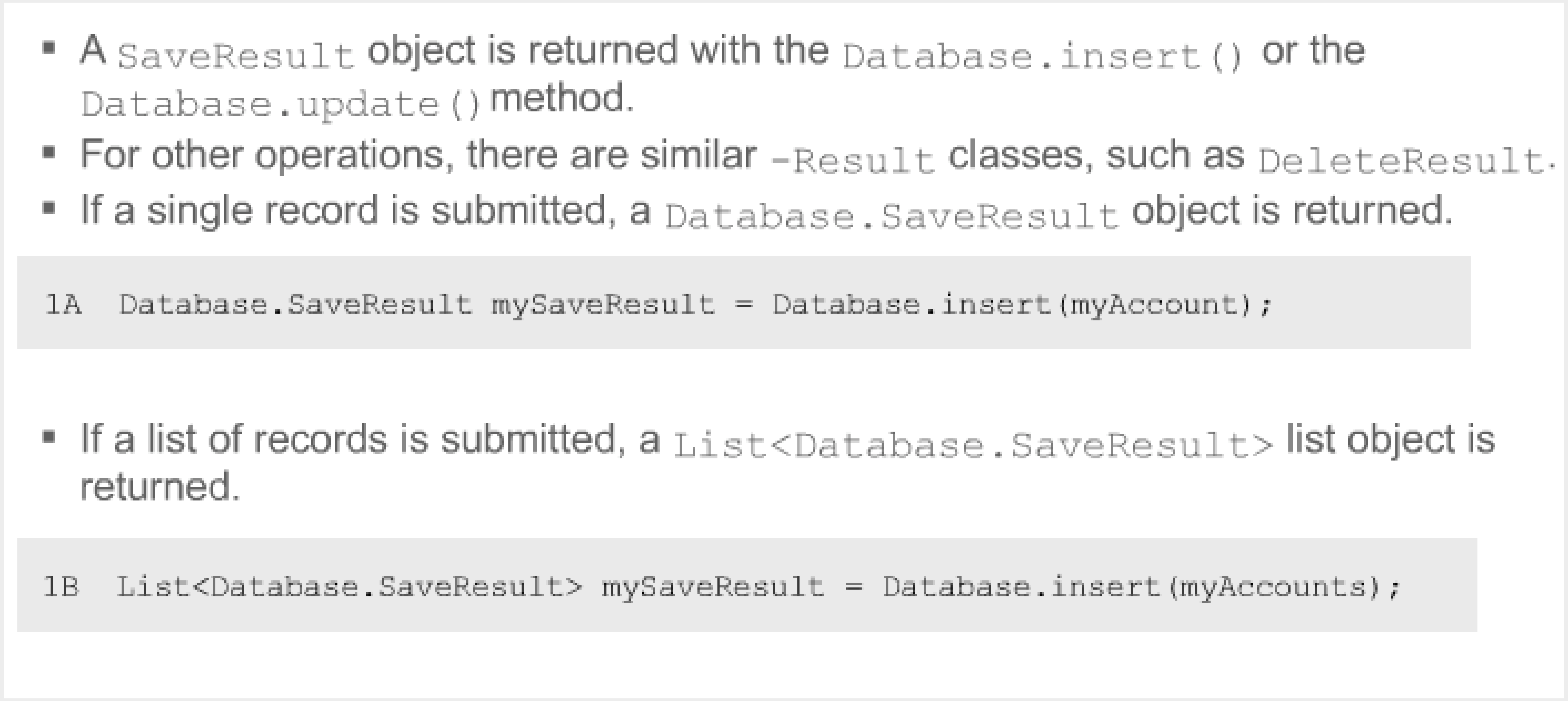
DML Standalone Statement

- Are static methods available in the `Database` class.
- Include an optional Boolean `allOrNone` parameter that defaults to `true`.
  - Example: `Database.insert(accounts, false)`



- Return a `SaveResult` object.

SaveResult Object



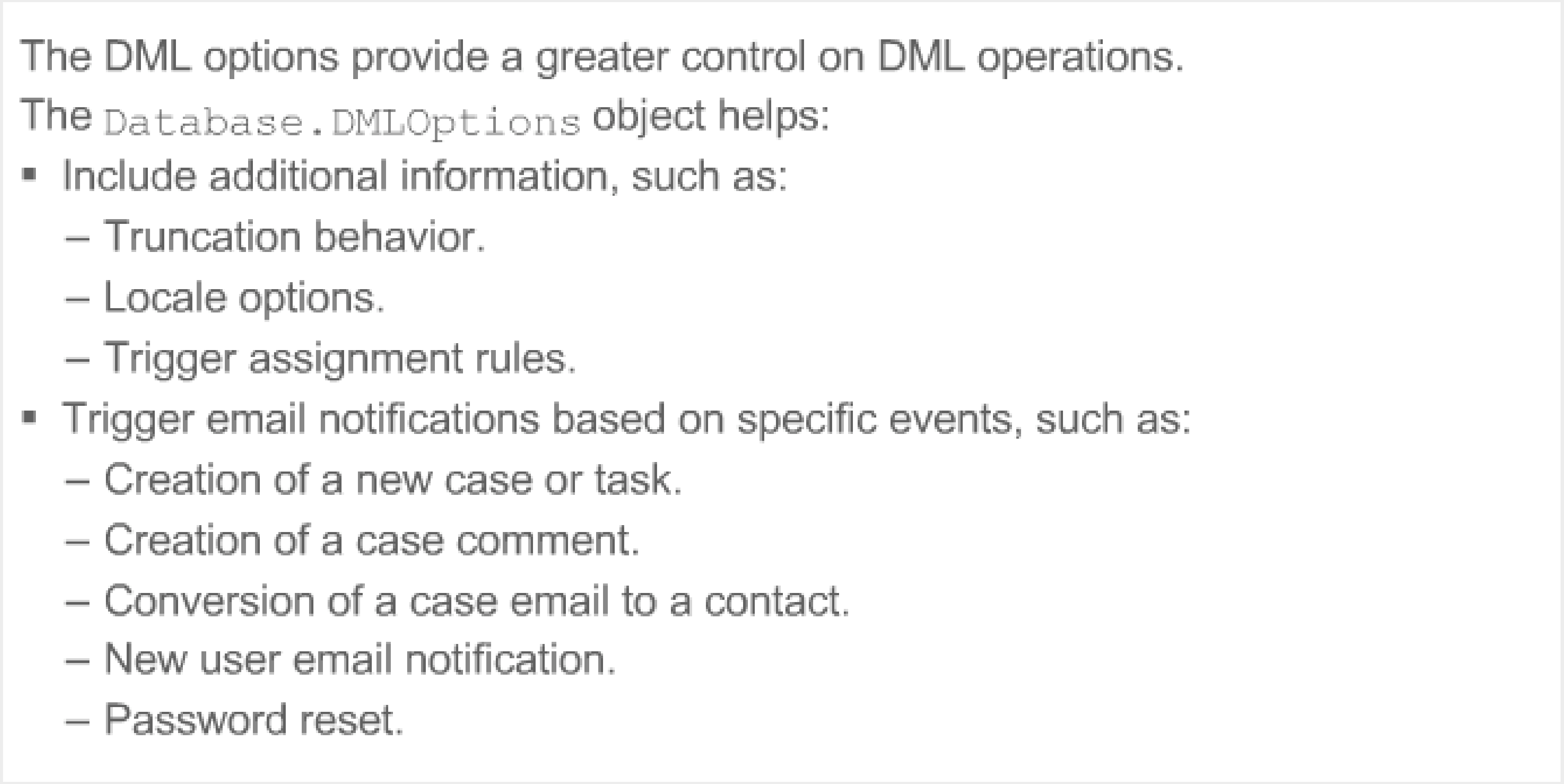
- A `SaveResult` object is returned with the `Database.insert()` or the `Database.update()` method.
- For other operations, there are similar `-Result` classes, such as `DeleteResult`.
- If a single record is submitted, a `Database.SaveResult` object is returned.

```
1A   Database.SaveResult mySaveResult = Database.insert(myAccount);
```

- If a list of records is submitted, a `List<Database.SaveResult>` list object is returned.

```
1B   List<Database.SaveResult> mySaveResult = Database.insert(myAccounts);
```

Database.DMLOptions have Greater Control than DML Ops



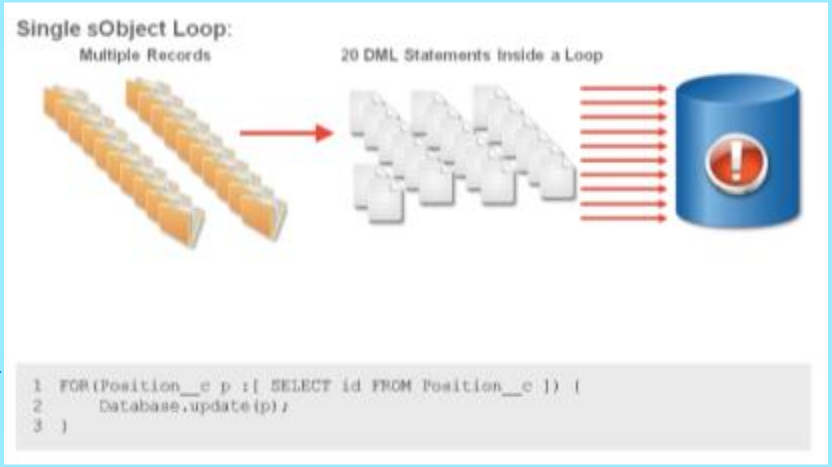
The DML options provide a greater control on DML operations.

The `Database.DMLOptions` object helps:

- Include additional information, such as:
  - – Truncation behavior.
  - – Locale options.
  - – Trigger assignment rules.
- Trigger email notifications based on specific events, such as:
  - – Creation of a new case or task.
  - – Creation of a case comment.
  - – Conversion of a case email to a contact.
  - – New user email notification.
  - – Password reset.

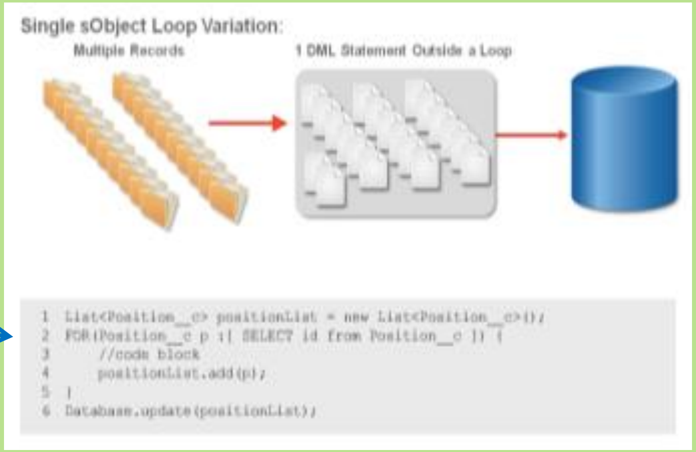
DML Loops

Will Result in multiple DML Statements



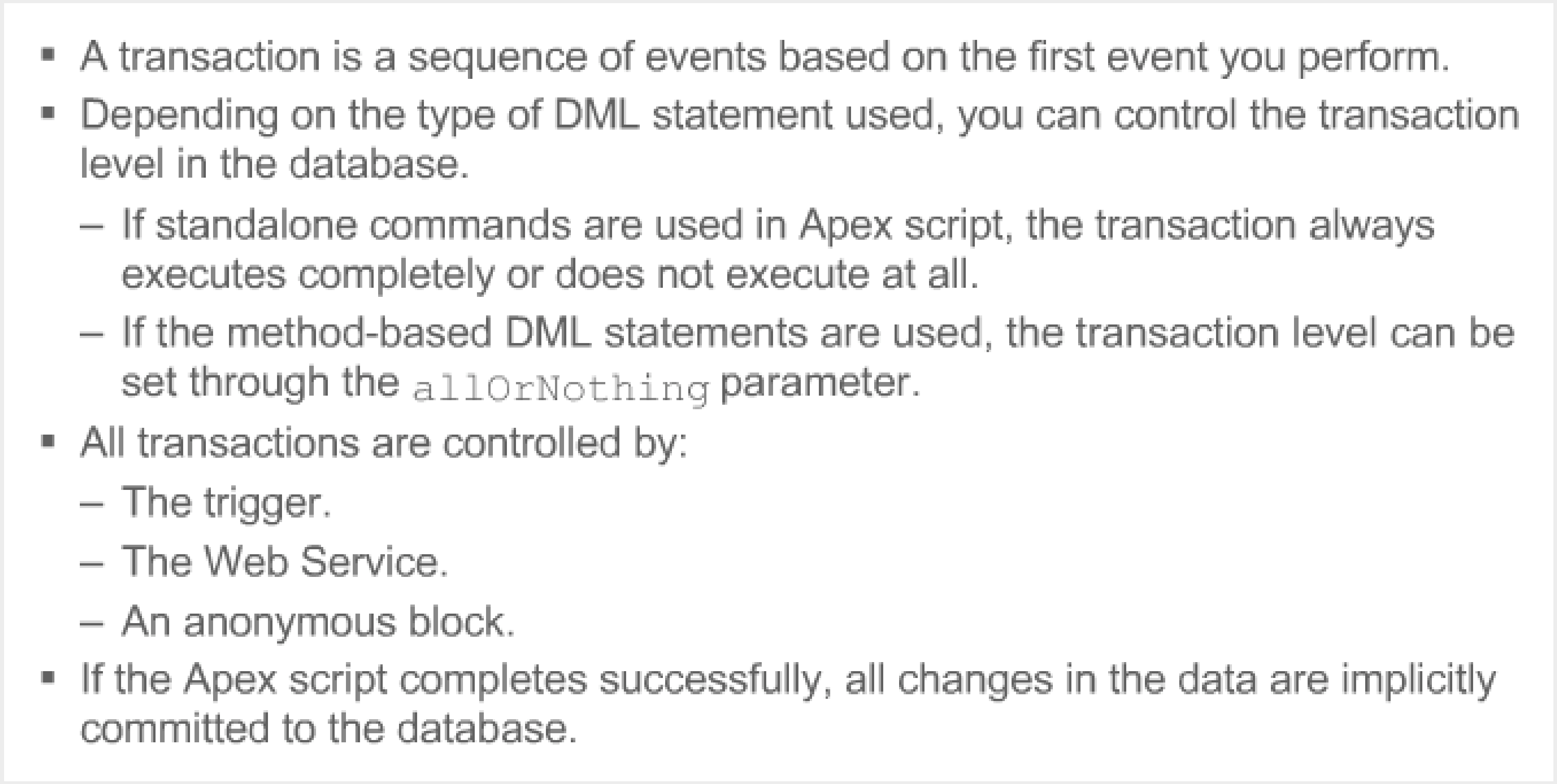
```
FOR(Position__c p :[ SELECT id FROM Position__c ]) {
    Database.update(p);
}
```

Will Result in a single DML Statement


Single sObject Loop Variation:
Multiple Records
1 DML Statement Outside a Loop

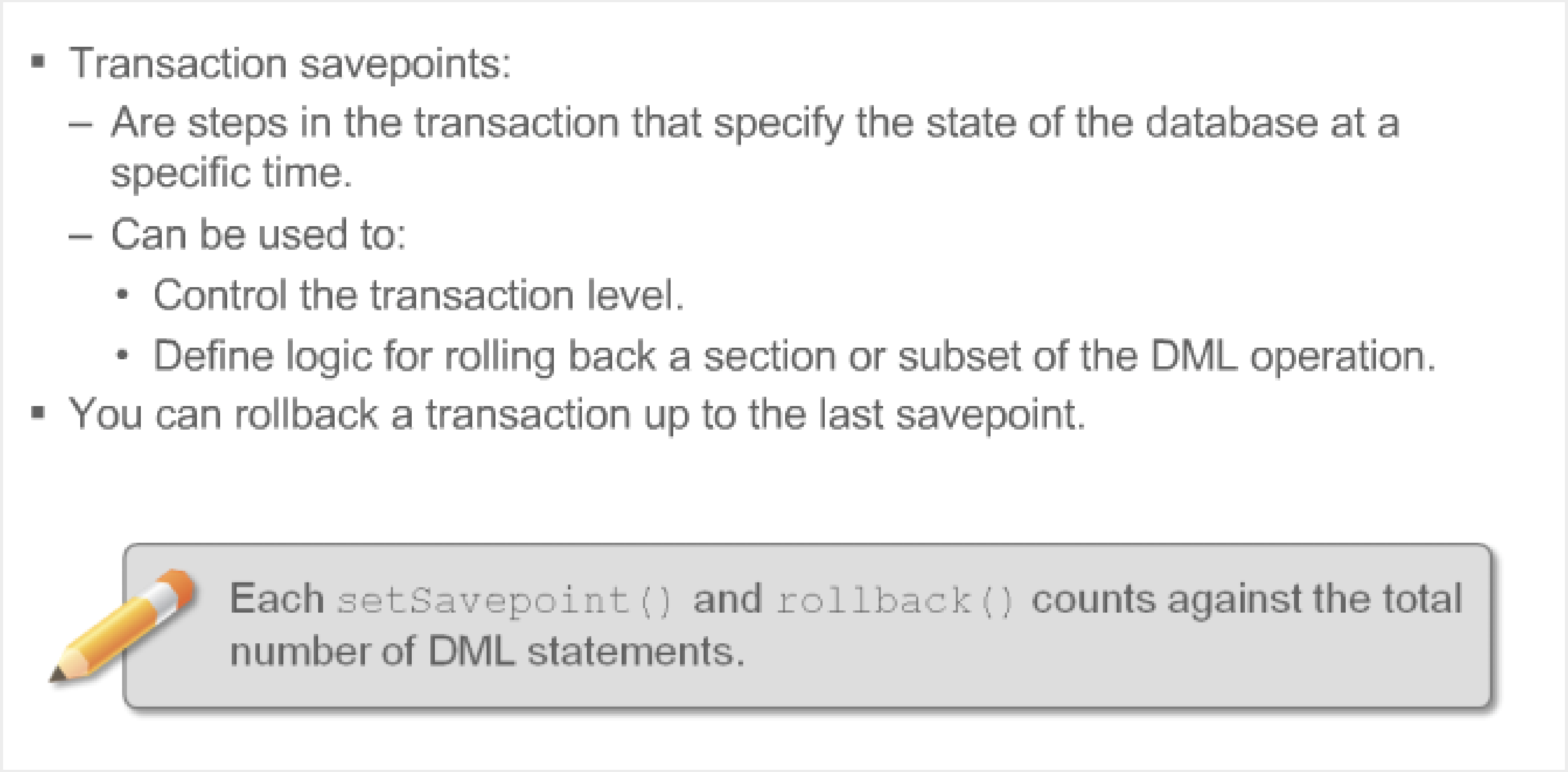
```
List<Position__c> positionList = new List<Position__c>();
FOR(Position__c p :[ SELECT id from Position__c ]) {
    //code block
    positionList.add(p);
}
Database.update(positionList);
```

Database Transactions



- A transaction is a sequence of events based on the first event you perform.
- Depending on the type of DML statement used, you can control the transaction level in the database.
  - If standalone commands are used in Apex script, the transaction always executes completely or does not execute at all.
  - If the method-based DML statements are used, the transaction level can be set through the `allOrNothing` parameter.
- All transactions are controlled by:
  - The trigger.
  - The Web Service.
  - An anonymous block.
- If the Apex script completes successfully, all changes in the data are implicitly committed to the database.

Transaction Savepoints



- Transaction savepoints:
  - Are steps in the transaction that specify the state of the database at a specific time.
  - Can be used to:
    - Control the transaction level.
    - Define logic for rolling back a section or subset of the DML operation.
- You can rollback a transaction up to the last savepoint.

Each `setSavepoint()` and `rollback()` counts against the total number of DML statements.

Hands-On Exercise

Apex: Record in a Database Exercise Guide
Exercise 4-1: Exploring Query Relationships Using Force.com IDE Tools
Exercise 4-2: Creating an Automated Chatter Subscription (Part 1)

https://lms.cfs-api.com/v1/content/bfe56b4e-035b-4c16-a78a-bd0cc2768a0d/presentation_content/external_files/recordsinthedatabaseexerciseguide.pdf