Getting Started with Database.com

Last updated: May 17, 2014

Table of Contents

Introduction

Welcome to Database.com!

Database.com is a multitenant cloud database service that's designed to store data for mobile, social enterprise applications. You can use Database.com as the back-end database for applications that are written in any language and run on any platform or mobile device. Database.com's built-in social computing infrastructure and native support for building sophisticated REST-based APIs enable you to create employee-facing, native mobile and social apps. With Database.com, you can:

- Quickly and easily create schemas using Database.com's metadata-driven data modeling tools
- Store small to very large data sets, scaling up to millions of records
- Secure your data and share it only with certain people, whether they're within your company, your customers, or your partners
- Query your data with REST API calls, such as:

  ◊ `https://na1.salesforce.com/services/data/v26.0/query/?q=SELECT id, name, Widget_Cost__c FROM widget__c`

Because Database.com is managed by salesforce.com in the cloud, you don't need to worry about the many costs incurred by traditional client/server databases. Buying costly hardware and software, scaling, tuning, doing backups, and upgrading become distant memories. Additionally, managing user access to data is simplified because Database.com leverages Salesforce's proven identity and authentication model, as well as its sharing and security engine.

Database.com Features

- A metadata-driven data model for both structured and unstructured data
- Salesforce Object Query Language (SOQL) and Salesforce Object Search Language (SOSL) for querying your data
- Internal database services, such as the ability to create formulas, validation rules, and workflow
- Open REST and SOAP APIs for accessing and manipulating data
- The Apex programming language for extending your database with triggers, stored procedure classes, and custom Web services
- A user identity and authentication model, leveraging OAuth and SAML, with tightly integrated controls for data security, sharing, and social applications
- Chatter functionality, including Chatter feeds and social data such as users, groups, followers, and files, which you can add to your application through the Chatter REST API

See Also:

Data Model
Data Queries
Authentication and Security

DATABASE.COM QUICK START

Introduction

Use this topic to create a sample app in your development environment.

Before you begin building an integration or other client application:

- Install your development platform according to its product documentation.
- Read through all the steps in this quick start.
- Review the other Database.com documents to familiarize yourself with terms and concepts.

Step 1: Obtain an Organization

If you don't already have an account, go to `www.database.com` and follow the instructions for joining.

If you already have an organization, verify that you have the "API Enabled" permission. This permission is enabled by default, but may have been changed by an administrator. For more information, see the Database.com online help.

Step 2: Create Objects and Fields

In this step you'll create two objects, widget and model, each with a custom field. Then you'll relate the objects to each other with a one-to-many-relationship.

Create Widget Object

To create the widget object with a widget cost field:

1. Click **Create** > **Objects**.
2. Click **New Custom Object**.
3. Enter the information for the widget object:
    - Label: `Widget`
    - Plural label: `Widgets`
    - Object name: `Widget`
    - Record name: `Widget Name`
    - Data type: `Text`
4. Leave all other settings as they are and click **Save**.
5. In the Custom Fields & Relationships related list, click **New**.
6. For Data Type, select `Currency` and click **Next**.
7. Enter the custom field details.
    - Field Label: `Widget Cost`

- Length: `10`
- Decimal places: `2`
- Field Name: `Widget_Cost`

8. Leave the remaining settings as they are and click **Next**.
9. Click **Save** to accept the default field-level security settings.

Create Model Object

To create the model object with a model number field:

1. Click **Create** > **Objects**.
2. Click **New Custom Object**.
3. Enter the information for the model object:

   - Label: `Model`
   - Plural label: `Models`
   - Object name: `Model`
   - Record name: `Model Name`
   - Data type: `Text`

4. Leave all other settings as they are and click **Save**.
5. In the Custom Fields & Relationships related list, click **New**.
6. For Data Type, select `Text` and click **Next**.
7. Enter the custom field details.

   - Field Label: `Model Number`
   - Length: `10`
   - Field Name: `Model_Number`

8. Leave the remaining settings as they are and click **Next**.
9. Click **Save** to accept the default field-level security settings.

Relate the Objects

1. If you aren't already in the Model detail page, click **Create** > **Objects**, then select the Model object.
2. In the Custom Fields & Relationships related list, click **New**.
3. In the New Custom Field page, select `Master-Detail Relationship` and click **Next**.
4. In the Related To field, select the Widget object and click **Next**.
5. Accept the defaults on the remaining screens by clicking **Next** and then **Save**.

Step 3: Use a REST API Call to Query Your Objects

Let's quickly explore the REST API using Workbench. This section demonstrates some of the method calls in the REST API and their return values.

To get started with Workbench, use the hosted version. Navigate to: `workbench.developerforce.com`.

1. For **Environment** select **Production**. Leave **API Version** set to its default unless you know you want to target a different API version.
2. Accept the terms of service and click **Login with Salesforce**.

Now let's insert a record so that we have some data to query:

1. Click **Data** > **Insert**.
2. For Object Type, choose `Widget__c`.
3. Ensure that **Single Record** is selected, and click **Next**.
4. Enter the following field values:
   - **Name**: Smart Tool
   - **OwnerId**: (Leave blank.)
   - **Widget_Cost__c**: 1.99
5. Click **Confirm Insert**.

You should see a message proclaiming 1 success and 0 errors. Congratulations! You've inserted your first record. Let's use a REST API call to retrieve this new data.

1. Click **Utilities** > **REST Explorer**.
2. In the text area, enter the following: `/services/data/v24.0/query/?q=SELECT id, name, Widget_Cost__c FROM widget__c`
3. Ensure **GET** is selected, and then click **Execute**.
4. Click **Show Raw Response**.

The REST API call (to the query resource, with the query set as a parameter) returns a list of the widgets in your database (in this case in the JSON format):

```
{
    "totalSize": 1,
    "done": true,
        "records": [
        {
            "attributes": {
            "type": "Widget__c",
            "url": "/services/data/v24.0/sobjects/Widget__c/a02E0000002DYVpIAO"
        },
        "Id": "a02E0000002DYVpIAO",
        "Name": "Smart Tool",
        "Widget_Cost__c" : 1.99
        },
...
        ]
}
```

Accessing and manipulating data is simply a matter of manipulating URLs and using standard HTTP verbs like GET, POST, and DELETE. All of the URLs start with `/services/data/`, followed by a version number, followed by a path to the resource. The exact format of the URL is described in the REST API Developer's Guide, but these examples give you a feel for them.

Note how this list returns the widgets within a records element, embedded in the response. The response also contains the ID of each record. For example, in the above output, the Smart Tool widget has an ID of `a02E0000002DYVpIAO`. It also provides the REST URL needed for retrieving the contents of a particular record.

1. In the text area of the REST Explorer, enter the value of the URL attribute. In our case, it's `/services/data/v24.0/sobjects/Widget__c/a02E0000002DYVpIAO`, but your org will be different.
2. Click **Execute**, and then **Show Raw Response**.

The server will respond with the details of the Widget resource, something like this:

```
{
  "attributes": {
    "type": "Widget__c",
    "url": "/services/data/v24.0/sobjects/Widget__c/a02E0000002DYVpIAO"
  },
  "Id": "a02E0000002DYVpIAO",
  "OwnerId": "005E0000000KJS1IAO",
  "IsDeleted": false,
  "Name": "Smart Tool",
  "CreatedDate": "2012-07-16T23:34:46.000+0000",
  "CreatedById": "005E0000000KJS1IAO",
  "LastModifiedDate": "2012-07-16T23:34:46.000+0000",
  "LastModifiedById": "005E0000000KJS1IAO",
  "SystemModstamp": "2012-07-16T23:34:46.000+0000",
  "Widget_Cost__c": 1.99
}
```

This result contains a few attributes describing the record, all your custom fields, as well as a number of system fields. Note the form of the URL. In this case the resource, `Widget__c`, is there in the URL. `Widget__c` is the API name of the Widget object you created earlier. So a GET to `/services/data/v24.0/sobjects/<Object Type Name>/<ID>` returns the record of the given identifier and object. In fact, a DELETE to the same URL will delete the record, and a POST to `/services/data/v24.0/sobjects/Widget__c/` (with the correct body) will create a new record.

There are two important elements of interacting with the REST API that are masked in the above interactions:

- Every HTTP request has a header element, `Authorization`, which contains an access token. The access token was returned as part of logging in to your environment.
- Every HTTP request must be made to a base URL — the URL of the Database.com instance. This URL is also returned as part of the OAuth authentication process.

You can find these two pieces of data in Workbench:

- Navigate to **Info** > **Session Information** and expand **Connection**.

The Endpoint value starts with `https://na1.salesforce.com/services`—that's where the HTTP requests are being sent. Note that your endpoint might be different. The Session Id contains the access token.

Applications on mobile devices always contains these elements:

- OAuth dance
- Retrieval of the instance URL and access token
- Use of these in all subsequent interactions with the REST API

Step 4: Create a Remote Access Application

External applications use the OAuth protocol to verify both the Database.com user and the external application itself. To provide this functionality to your application, create a remote access application for your organization:

1. Log in to your organization. Logins are checked to ensure they are from a known IP address.
2. Click **Develop** > **Remote Access** to display the Remote Access page.
3. Click **New**.
4. Enter the information for the remote access application:
   - Application: `MyRemoteAccessApplication`
   - Callback URL: `https://no_redirect_uri`
   - Contact Email: ***your_email@domain.ext***

5. Click **Save**.

Step 5: Walk Through the Sample Code

Once you've created your remote application, you can begin building client applications that use the REST API. Use the following samples to create a basic client application. Comments embedded in the sample explain each section of code.

Java Sample Code

This section walks through a sample Java client application that uses the REST API. The purpose of this sample application is to show the required steps for logging into the login server and to demonstrate the invocation and subsequent handling of several REST API calls. This sample application performs the following main tasks:

1. Prompts the user for
   - API version
   - login URL
   - username
   - password
   - OAuth 2.0 consumer key
   - OAuth 2.0 consumer secret
2. Uses the information friom the previous step to log in to the single login server and, if the login succeeds:
3. Sends an HTTP GET request to the server URL: `https://instance.salesforce.com/services/data/v24.0/sobjects/`. This is equivalent to a calling `describeGlobal()` to retrieve a list of all available objects for the organization's data.
4. Sends an HTTP GET request to the server URL: `https://instance.salesforce.com/services/data/v24.0/sobjects/Merchandise__c/describe/`. This is equivalent to a calling `describeSObject()` to retrieve metadata (field list and object properties) for the specified object.
5. Sends an HTTP POST request to the server URL: `https://instance.salesforce.com/services/data/v24.0/sobjects/Merchandise__c/` passing a JSON object in the request body. This is equivalent to a calling `create()` to a record corresponding to the JSON object.
6. Sends an HTTP GET request to the server URL: `https://instance.salesforce.com/services/data/v24.0/query/?q=SELECT+Name+FROM+Merchandise__c`. This is equivalent to a calling `query()`, passing a simple query string ("`SELECT Name FROM Merchandise__c`"), and iterating through the returned `QueryResult`.

Java Sample Code

```
package com.example.sample.rest;

import java.awt.Desktop;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLEncoder;

import org.apache.http.Header;
import org.apache.http.HttpResponse;
```

```
import org.apache.http.StatusLine;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;

import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class RestClient extends Object {
  private static BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
  private static String OAUTH_ENDPOINT = "/services/oauth2/token";
  private static String REST_ENDPOINT = "/services/data";
  UserCredentials userCredentials;
  String restUri;
  Header oauthHeader;
  Header prettyPrintHeader = new BasicHeader("X-PrettyPrint", "1");
  Gson gson;
  OAuth2Response oauth2Response;

  public static void main(String[] args) {
    RestClient client = new RestClient();
    client.oauth2Login( client.getUserCredentials() );
    client.testRestData();
  }

  public RestClient() {
    gson = new Gson();
  }

  public HttpResponse oauth2Login(UserCredentials userCredentials) {
    HttpResponse response = null;
    this.userCredentials = userCredentials;
    String loginHostUri = "https://" +
        userCredentials.loginInstanceDomain + OAUTH_ENDPOINT;
    try {
      HttpClient httpClient = new DefaultHttpClient();
      HttpPost httpPost = new HttpPost(loginHostUri);
      StringBuffer requestBodyText =
          new StringBuffer("grant_type=password");
      requestBodyText.append("&username=");
      requestBodyText.append(userCredentials.userName);
      requestBodyText.append("&password=");
      requestBodyText.append(userCredentials.password);
      requestBodyText.append("&client_id=");
      requestBodyText.append(userCredentials.consumerKey);
      requestBodyText.append("&client_secret=");
      requestBodyText.append(userCredentials.consumerSecret);
      StringEntity requestBody =
          new StringEntity(requestBodyText.toString());
      requestBody.setContentType("application/x-www-form-urlencoded");
      httpPost.setEntity(requestBody);
      httpPost.addHeader(prettyPrintHeader);
      response = httpClient.execute(httpPost);
      if (  response.getStatusLine().getStatusCode() == 200 ) {
        InputStreamReader inputStream = new InputStreamReader(
            response.getEntity().getContent()
        );
        oauth2Response = gson.fromJson( inputStream,
            OAuth2Response.class );
        restUri = oauth2Response.instance_url + REST_ENDPOINT +
            "/v" + this.userCredentials.apiVersion +".0";
        System.out.println("\nSuccessfully logged in to instance: " +
```

```
            restUri);
        oauthHeader = new BasicHeader("Authorization", "OAuth " +
            oauth2Response.access_token);
      } else {
        System.out.println("An error has occured.");
        System.exit(-1);
      }
    } catch (UnsupportedEncodingException uee) {
      uee.printStackTrace();
    } catch (IOException ioe) {
      ioe.printStackTrace();
    } catch (NullPointerException npe) {
      npe.printStackTrace();
    }
    return response;
  }

  public void testRestData() {
    String responseBody = restGet(restUri);
    responseBody = restGet(restUri + "/sobjects/");
    responseBody = restGet(restUri +
        "/sobjects/Merchandise__c/describe/");
    responseBody = restPost(restUri +
        "/sobjects/Merchandise__c/", "{ \"Name\" : \"Wee Jet\" }\n\n");
    System.out.println(responseBody);
    JsonParser jsonParser = new JsonParser();
    JsonElement jsonElement = jsonParser.parse(responseBody);
    String id = jsonElement.getAsJsonObject().get("id").getAsString();
    responseBody = restGet(restUri +
        "/sobjects/Merchandise__c/" + id);
    System.out.println(responseBody);
    responseBody = restPost(restUri +
        "/sobjects/Merchandise__c/", "{ \"Name\" : \"Zeppelin GmbH\" }\n\n");
    System.out.println(responseBody);
    responseBody = restGet(restUri +
        "/query/?q=SELECT+Name+FROM+Merchandise__c");
    System.out.println(responseBody);
    responseBody = restPatch(restUri +
        "/sobjects/Merchandise__c/" + id, "{ \"Name\" : \"Dry Twig.\" }\n\n");
    System.out.println(responseBody);
    responseBody = restGet(restUri +
        "/sobjects/Merchandise__c/" + id);
    System.out.println(responseBody);
  }

  public String restGet(String uri) {
    String result = "";
    printBanner("GET", uri);
    try {
      HttpClient httpClient = new DefaultHttpClient();
      HttpGet httpGet = new HttpGet(uri);
      httpGet.addHeader(oauthHeader);
      httpGet.addHeader(prettyPrintHeader);
      HttpResponse response = httpClient.execute(httpGet);
      result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
      ioe.printStackTrace();
    } catch (NullPointerException npe) {
      npe.printStackTrace();
    }
    return result;
  }

  public String restPatch(String uri, String requestBody) {
    String result = "";
    printBanner("PATCH", uri);
    try {
      HttpClient httpClient = new DefaultHttpClient();
      HttpPatch httpPatch = new HttpPatch(uri);
      httpPatch.addHeader(oauthHeader);
```

```
      httpPatch.addHeader(prettyPrintHeader);
      StringEntity body = new StringEntity(requestBody);
      body.setContentType("application/json");
      httpPatch.setEntity(body);
      HttpResponse response = httpClient.execute(httpPatch);
      result = response.getEntity() != null ?
          getBody( response.getEntity().getContent() ) : "";
    } catch (IOException ioe) {
      ioe.printStackTrace();
    } catch (NullPointerException npe) {
      npe.printStackTrace();
    }
    return result;
}

 public String restPatchXml(String uri, String requestBody) {
    String result = "";
    printBanner("PATCH", uri);
    try {
      HttpClient httpClient = new DefaultHttpClient();
      HttpPatch httpPatch = new HttpPatch(uri);
      httpPatch.addHeader(oauthHeader);
      httpPatch.addHeader(prettyPrintHeader);
      httpPatch.addHeader( new BasicHeader("Accept", "application/xml") );
      StringEntity body = new StringEntity(requestBody);
      body.setContentType("application/xml");
      httpPatch.setEntity(body);
      HttpResponse response = httpClient.execute(httpPatch);
      result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
      ioe.printStackTrace();
    } catch (NullPointerException npe) {
      npe.printStackTrace();
    }
    return result;
}

 public String restPost(String uri, String requestBody) {
    String result = null;
    printBanner("POST", uri);
    try {
      HttpClient httpClient = new DefaultHttpClient();
      HttpPost httpPost = new HttpPost(uri);
      httpPost.addHeader(oauthHeader);
      httpPost.addHeader(prettyPrintHeader);
      StringEntity body = new StringEntity(requestBody);
      body.setContentType("application/json");
      httpPost.setEntity(body);
      HttpResponse response = httpClient.execute(httpPost);
      result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
      ioe.printStackTrace();
    } catch (NullPointerException npe) {
      npe.printStackTrace();
    }
    return result;
}

 /**
  * Extend the Apache HttpPost method to implement an HttpPost
  * method.
  */
 public static class HttpPatch extends HttpPost {
    public HttpPatch(String uri) {
      super(uri);
    }

    public String getMethod() {
      return "PATCH";
    }
```

```
  }

  static class OAuth2Response {
    public OAuth2Response() {
    }
    String id;
    String issued_at;
    String instance_url;
    String signature;
    String access_token;
  }

  class UserCredentials {
    String grantType;
    String userName;
    String password;
    String consumerKey;
    String consumerSecret;
    String loginInstanceDomain;
    String apiVersion;
  }

  // private methods

  private String getUserInput(String prompt) {
    String result = "";
    try {
      System.out.print(prompt);
      result = reader.readLine();
    } catch (IOException ioe) {
      ioe.printStackTrace();
    }
    return result;
  }

  private void printBanner(String method, String uri) {

System.out.println("\n------------------------------------------------------------------------\n");

    System.out.println("HTTP Method: " + method);
    System.out.println("REST URI: " + uri);

System.out.println("\n------------------------------------------------------------------------\n");

  }

  private String getBody(InputStream inputStream) {
    String result = "";
    try {
        BufferedReader in = new BufferedReader(
            new InputStreamReader(inputStream)
        );
        String inputLine;
        while ( (inputLine = in.readLine() ) != null ) {
          result += inputLine;
          result += "\n";
        }
        in.close();
    } catch (IOException ioe) {
      ioe.printStackTrace();
    }
    return result;
  }

  private UserCredentials getUserCredentials() {
    UserCredentials userCredentials = new UserCredentials();
    userCredentials.loginInstanceDomain =
        getUserInput("Login Instance Domain: ");
    userCredentials.apiVersion = getUserInput("API Version: ");
    userCredentials.userName = getUserInput("UserName: ");
```

```
    userCredentials.password = getUserInput("Password: ");
    userCredentials.consumerKey = getUserInput("Consumer Key: ");
    userCredentials.consumerSecret = g
        etUserInput("Consumer Secret: ");
    userCredentials.grantType = "password";
    return userCredentials;
  }
}
```

OVERVIEW

Database.com User Licenses

A user license entitles a user to particular functionality within Database.com. The following user licenses are delivered with Database.com.

User License	Description	Default Number of Available Licenses
Database.com Admin	Designed for users who need to administer Database.com, or make changes to Database.com schemas or other metadata using the point-and-click tools in the Database.com Console.	Database.com Edition: 3
Database.com User	Designed for users who need API access to data stored in Database.com.	Database.com Edition: 3 Contact salesforce.com to obtain Database.com User Licenses
Database.com Light User	Designed for users who need only API access to data, need to belong to Chatter groups (but no other groups), and don't need to belong to roles or queues. Access to data is determined by organization-wide sharing defaults.	Database.com Edition: 0 Contact salesforce.com to obtain Database.com Light User Licenses

To view your organization's number of active user licenses, click **Company Profile** > **Company Information** in the Database.com Console.

To increase your number of available licenses, contact salesforce.com.

You can also purchase additional licenses using the Database.com console. Click **Checkout Summary**, click **Proceed to Checkout**, and follow the instructions on the page.

Data Model

Database.com's database model includes the data storage features that you'd expect from a typical relational database, augmented with powerful metadata-driven features you can use to quickly and easily create applications.

Database.com Terminology

To reflect the added functionality provided by metadata-driven features, Database.com uses different terminology compared to a relational database.

Relational Database Term	Equivalent Term in Database.com
Database	Organization
Table	Object
Column	Field
Row	Record

In a relational database, tables contain columns (to define the data types) and rows (to store the data). You relate tables to other tables by using primary keys and foreign keys, which map the rows of one table to the rows of another table.

In Database.com, an *organization* is the equivalent of a database, but with built-in user identity, security, and social features. *Objects* contain *fields* and *records*. You relate objects to other objects by using *relationship fields*, such as *lookup relationships* and *master-detail relationships*, instead of primary and foreign keys.

Database.com Field Types

Database.com's field types allow you to store data and easily configure how the data is used by applications accessing your database. The following table describes some of Database.com's field types.

Field Type	Description
Checkbox	Represents a value that can be true or false.
Currency	Represents currency values.
Date	Represents a date value. Database.com includes built-in functions for manipulating date values.
Date/Time	Represents a date/time combination. Database.com includes built-in functions for manipulating date/time values.
Email	Represents an email address. Database.com validates values for this field to ensure proper format.
Formula	Allows for the automatic calculation of values based on other values or fields. The field is updated whenever any of the source fields are changed.
Lookup Relationship	Creates a relationship between two records so you can associate them with each other. For example, say you have a recruiting application. You can associate a hiring manager user with an open position by adding a User object lookup relationship field to a Position object.
Master-Detail Relationship	Creates a relationship between records where the master record controls certain behaviors of the detail record, such as record deletion and security. For example, say you have a blog and it contains blog posts. If a blog is the master record and the posts in that blog are the detail records, the blog posts are automatically deleted

Field Type	Description
	when the blog is deleted. Likewise, a user that has permission to view the blog can view all associated blog posts.
Number	Represents a real number, with optional decimal points.
Phone	Represents phone numbers.
Picklist	Represents a list of options from which one option can be selected at a time.
Roll-up Summary	A count of related records or the calculated sum, minimum, or maximum value of a numerical attribute on related records.
Text	Any combination of letters, numbers, or symbols.

See Also:

"Field Types" in the Force.com SOAP API Developer's Guide

Identity Field

All Database.com objects include an ID field that contains a 15-character unique identifier for each record in the object. This field is analogous to a primary key in relational databases.

See Also:

"ID Field Type" in the Force.com SOAP API Developer's Guide

Data Queries

In Database.com, you can query your data by using the following:

Database.com Object Query Language (SOQL)

Use SOQL to construct simple but powerful query strings. In a manner similar to SQL, SOQL is an object query language that uses relationships, instead of joins, to support intuitive navigation of data. Use SOQL in the following contexts:

- In the `queryString` parameter in the `query()` call to select records for a single object
- In Apex statements
- In the Schema Explorer of the Force.com IDE

Database.com Object Search Language (SOSL)

Use SOSL to construct text searches in the following contexts:

- In the `search()` call to find records for one or more objects
- In Apex statements
- In the Schema Explorer of the Force.com IDE

SOQL Example

Similar to the SELECT command in SQL, SOQL allows you to specify the source object, a list of fields to retrieve, and conditions for selecting rows in the source object.

Note: SOQL does not support all advanced features of the SQL SELECT command. For example, you cannot use SOQL to perform arbitrary join operations, use wildcards in field lists, or use calculation expressions.

SOQL uses the SELECT statement combined with filtering statements to return sets of data, which may optionally be ordered:

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

For example, the following SOQL query returns the value of the `Id` and `Name` field for all Merchandise records if the value of `Name` is `Wee Jet`:

```
SELECT Id, Name
FROM Merchandise__c
WHERE Name = 'Wee Jet'
```

Note that you can embed SOQL queries directly into your Apex code by surrounding the query in brackets. For example:

```
Merchandise__c m = [SELECT Id, Name
                    FROM Merchandise__c
                    WHERE Name = 'Wee Jet']
```

See Also:

"Salesforce Object Query Language (SOQL)" in the Database.com SOQL and SOSL Reference Guide
"An Introduction to the Force.com IDE"

SOSL Examples

The following SOSL examples search for text in Database.com.

Look for `joe` anywhere in the system. Return the IDs of the records where `joe` is found.

```
FIND {joe}
```

Look for the name `Joe Smith` anywhere in the system, in a case-insensitive search. Return the IDs of the records where `Joe Smith` is found.

```
FIND {Joe Smith}
```

Delimiting “and” and “or” as literals when used alone:

```
FIND {"and" or "or"}
FIND {"joe and mary"}
FIND {in}
```

```
FIND {returning}
FIND {find}
```

See Also:

"About SOSL" in the Database.com SOQL and SOSL Reference Guide
"An Introduction to the Force.com IDE"

SQL and Database.com

If you need to use SQL with Database.com, you can employ drivers for ODBC and JDBC that are provided by third-party vendors such as Progress Software.

Authentication and Security

Traditional databases assume that you implement your application security within the application tier. While you can use this approach with Database.com, you wouldn't be taking advantage of Database.com's sophisticated, declarative user and security model. This model has been proven not only with Database.com applications, but also with all the applications developers have created using the Force.com platform. Because this model abstracts security functionality from application code, it increases flexibility and improves time to market.

The Database.com user and security model includes:

- Identity and user management
- Data security access and sharing controls
- Automatic authentication
- User profiles and permission sets
- A social data model and social APIs

Layered Security and Sharing Design

Specifying the data set that each user or group of users can access is one of the key decisions that affects data security. When deciding on the data that you want to expose to your users, it's necessary to strike a balance between limiting access to data (to reduce the risk of stolen or misused data) and providing your users with the ability to access data that's critical to their success.

To help you meet your data security, Database.com provides a flexible, layered sharing design that makes it easy to expose different data sets to different sets of users. You can:

- Use permission sets and profiles to specify the objects that users can access
- Employ field-level security to specify the fields that a user can access
- Manage organization-wide sharing settings, define a role hierarchy, and create sharing rules to specify the individual records that a user can view and edit

Use the following security and sharing settings to control users' access to data.

Object-Level Security (Permission Sets and Profiles)

Object-level security provides the bluntest way to control data. Using object permissions, you can prevent a user from viewing, creating, editing, or deleting any instance of a particular type of object. Object permissions let you hide entire objects from particular users, so they don't even know that type of data exists.

You specify object permissions in *permission sets* and *profiles*. Permission sets and profiles are collections of access settings and permissions that determine what a user can do in the application, similar to a group in a Windows network, where all members of the group have the same folder permissions and access to the same software.

Field-Level Security (Permission Sets and Profiles)

Field-level security controls whether a user can see, edit, and delete the value for a particular field on an object. It lets you protect sensitive fields without having to hide the whole object from users. Field-level security is also controlled in permission sets and profiles.

Record-Level Security (Sharing)

After setting object- and field-level access permissions, you may want to configure access settings for the actual records themselves. Record-level security lets you give users access to some object records, but not others.

To specify record-level security, set your organization-wide sharing settings, define a hierarchy, and create sharing rules.

- **Organization-wide sharing settings**—The first step in record-level security is to determine the organization-wide sharing settings for each object. Organization-wide sharing settings specify the default level of access users have to each others' records. The settings can be Private, Public Read Only, or Public Read/Write. You use organization-wide sharing settings to lock down your data to the most restrictive level, and then use the other record-level security and sharing tools to selectively open up access to other users. For example, let's say users have read and edit permissions on an object, and the organization-wide sharing setting is Read-Only. By default, those users can view all object records, but can't edit any unless they own the record or are granted additional permissions.
- **Role hierarchy**—Once you've specified organization-wide sharing settings, the first way you can give wider access to records is with a role hierarchy. Similar to an organization chart, a role hierarchy represents a level of data access that a user or group of users needs. In a role hierarchy, users higher in the hierarchy always have access to the data visible to users below them in the hierarchy. This ensures that managers always have access to the same data as their employees, regardless of the organization-wide default settings. Role hierarchies don't have to match your organization chart exactly. Instead, each role in the hierarchy should represent a level of data access that a user or group of users needs.
- **Sharing rules**—Sharing rules let you make automatic exceptions to organization-wide sharing settings for particular sets of users, to give them access to records they don't own or can't normally see. Sharing rules, like role hierarchies, are only used to give users additional access to records—they can't be stricter than your organization-wide default settings. Sharing rules work best when defined for a particular set of users that you determine or predict in advance, rather than a set of users that frequently changes. A set of users can be a public group, a role, or a queue.
- **Apex managed sharing**—If sharing rules don't provide the control you need, you can use Apex managed sharing to programmatically share objects. When you use Apex managed sharing to share an object, only users with the “Modify All Data” permission can add or change the sharing on the object's record, and the sharing access is maintained across record owner changes.

See Also:

"Object Permissions" in the Database.com Online Help
"Managing Field-Level Security" in the Database.com Online Help
"Setting Your Organization-Wide Sharing Defaults" in the Database.com Online Help
"Managing Roles" in the Database.com Online Help
"Creating Custom Object Sharing Rules" in the Database.com Online Help
Database.com Apex Code Developer's Guide

OAuth

You can use the OAuth protocol to authenticate applications that access data in Database.com. OAuth is an open protocol that allows you to provide your users access to their data while protecting their account credentials. Enable OAuth by creating a remote access application, as described in Step 4: Create a Remote Access Application on page 5.

See Also:

"Using OAuth to Authorize External Applications"

Database.com Limits

Object Limits

Feature	Limit
Objects	2,000
Objects: Maximum Number of Master Detail Relationships	Each relationship is included in the maximum number of custom fields allowed. When data is substituted for the tokens in the URL, the link may exceed 3,000 bytes. Your browser may enforce additional limits for the maximum URL length.
Objects: Maximum Number of Deleting Combined Objects and Child Records	100,000
Sharing Rules	300 sharing rules per object, including up to 50 criteria-based rules

Field Limits

Feature	Limit
Fields	• Relationship Fields: 25 per object • Roll-up Summary Fields: 10 per object • Rich Text Area and Long Text Area Fields: 25. Additionally, each object can contain a total of 1.6 million characters across long text area and rich text area fields. The default character limit for long text area and rich text area fields is 32,000 characters. A long text area or rich text area field needs to contain at least 256 characters. • All Other Fields: 800 per object
Custom Settings: Maximum Number of Fields Per Custom Setting	300
Field History Tracking: Maximum Number of Fields Tracked per Object	20
Formulas: Maximum Number of Characters	3,900 characters
Formulas: Maximum Formula Size (in Bytes) When Saved	4,000 bytes

Feature	Limit
Formulas: Maximum Formula Size (in Bytes) When Compiled	5,000 bytes
Formulas: Number of Unique Relationships Per Object	10
Active Validation Rules	500 per object

Administration Limits

Feature	Limit
Certificates: Maximum Number of Certificates	50
Custom Settings: Cached Data Limit	The lesser of 10 MB or 1 MB multiplied by the number of Database.com Admin user licenses in your organization.
Permission Sets	1,000
Recycle Bin: Maximum Number of Records	25 times your storage capacity in MBs
Tags	A user is limited to a maximum of: • 500 unique personal tags • 5,000 instances of personal tags applied to records Across all users, your organization can have a maximum of: • 1,000 unique public tags • 50,000 instances of public tags applied to records • 5,000,000 instances of personal and public tags applied to records
Users: Maximum Number of Users Created	Unlimited

Workflow Limits

Feature	Limit
Active Workflow Rules	50 per object
Total Workflow Rules Allowed (Limits apply to any combination of active and inactive rules.)	300 per object 1,000 per organization
Total Actions Allowed Per Workflow Rule	200
Workflow Rules	Each workflow rule can have: • 10 time triggers • 40 immediate actions • 40 time-dependent actions per time trigger Note that for both immediate and time-dependent actions, there can be no more than: • 10 field updates • 10 outbound messages
Workflow Time Triggers Per Hour	1,000

Concurrent Web Requests Limits

The limit for concurrent Web requests is 25.

API Query Cursor Limits

A user can have up to 10 query cursors open at a time. If 10 `QueryLocator` cursors are open when a client application, logged in as the same user, attempts to open a new one, then the oldest of the 10 cursors is released. If the client application attempts to open the released query cursor, an error results.

Bulk API Limits

Bulk API Limit	Limit Description
Batch limit	You can submit up to 5,000 batches per rolling 24 hour period. You can't create new batches associated with a job that is more than 24 hours old.
Batch lifespan	You can submit up to 5,000 batches per rolling 24 hour period. You can't create new batches associated with a job that is more than 24 hours old.
Batch size	You can submit up to 5,000 batches per rolling 24 hour period. You can't create new batches associated with a job that is more than 24 hours old.
Batch processing time	There is a five-minute limit for processing 100 records. Also, if it takes longer than 10 minutes to process a batch, the Bulk API places the remainder of the batch back in the queue for later processing. If the Bulk API continues to exceed the 10-minute limit on subsequent attempts, the batch is placed back in the queue and reprocessed up to 10 times before the batch is permanently marked as failed.
Job open time	The maximum time that a job can remain open is 24 hours. The Bulk API doesn't support clients that, for example, post one batch every hour for many hours.

Concurrent API Request Limits

The following table lists the limits for various types of organizations for concurrent requests (calls) with a duration of 20 seconds or longer.

Organization Type	Limit
Database.com Production organization	25
Database.com Test database organization	25

Total API Request Limits

The following table lists the limits for the total API requests (calls) per 24-hour period for an organization.

	API Calls	Minimum	Maximum
Debugging Header on API testing calls for Apex specified. Valid in API version 20 and later.	N/A	1,000	1,000

	API Calls	Minimum	Maximum
Total API Requests	5,000	5,000	Unlimited. However, at any high limit, it is likely that other limiting factors such as system load may prevent you from using your entire allocation of calls in a 24–hour period.

Limits are enforced against the aggregate of all API calls made by the organization in a 24 hour period; limits are not on a per-user basis. When an organization exceeds a limit, all users in the organization may be temporarily blocked from making additional calls. Calls will be blocked until usage for the preceding 24 hours drops below the limit.

In Database.com, administrators can view how many API requests have been issued in the last 24 hours on the Company Information page at **Company Profile** > **Company Information**.

Any action that sends a call to the API counts toward usage limits, except the following:

- Queries from a syndicated feed on a public site
- Outbound messages
- Apex callouts

Apex Governor Limits

Because Apex runs in a multitenant environment, the Apex runtime engine strictly enforces a number of limits to ensure that runaway Apex doesn't monopolize shared resources.

Per-Transaction Apex Limits

These limits count for each Apex transaction. For Batch Apex, these limits are reset for each execution of a batch of records in the `execute` method.

This table lists limits for synchronous Apex and asynchronous Apex (Batch Apex and future methods) when they're different. Otherwise, this table lists only one limit that applies to both synchronous and asynchronous Apex.

Description	Synchronous Limit	Asynchronous Limit
Total number of SOQL queries issued[1]	100	200
Total number of records retrieved by SOQL queries	50,000	
Total number of records retrieved by `Database.getQueryLocator`	10,000	
Total number of SOSL queries issued	20	
Total number of records retrieved by a single SOSL query	2,000	
Total number of DML statements issued[2]	150	
Total number of records processed as a result of DML statements or `database.emptyRecycleBin`	10,000	
Total stack depth for any Apex invocation that recursively fires triggers due to `insert`, `update`, or `delete` statements[3]	16	
Total number of callouts (HTTP requests or Web services calls) in a transaction	10	
Maximum timeout for all callouts (HTTP requests or Web services calls) in a transaction	120 seconds	

Description	Synchronous Limit	Asynchronous Limit
Total number of methods with the `future` annotation allowed per Apex invocation	10	
Total number of describes allowed[4]	100	
Total heap size[5]	6 MB	12 MB
Maximum CPU time on the Database.com servers[6]	10,000 milliseconds	60,000 milliseconds
Maximum execution time for each Apex transaction	10 minutes	
Maximum number of unique namespaces referenced[7]	10	

[1] In a SOQL query with parent-child relationship sub-queries, each parent-child relationship counts as an additional query. These types of queries have a limit of three times the number for top-level queries. The row counts from these relationship queries contribute to the row counts of the overall code execution. In addition to static SOQL statements, calls to the following methods count against the number of SOQL statements issued in a request.

- `Database.countQuery`
- `Database.getQueryLocator`
- `Database.query`

[2] Calls to the following methods count against the number of DML queries issued in a request.

- `Approval.process`
- `Database.convertLead`
- `Database.emptyRecycleBin`
- `Database.rollback`
- `Database.setSavePoint`
- `delete` and `Database.delete`
- `insert` and `Database.insert`
- `merge` and `Database.merge`
- `undelete` and `Database.undelete`
- `update` and `Database.update`
- `upsert` and `Database.upsert`
- `System.runAs`

[3] Recursive Apex that does not fire any triggers with `insert`, `update`, or `delete` statements exists in a single invocation, with a single stack. Conversely, recursive Apex that fires a trigger spawns the trigger in a new Apex invocation, separate from the invocation of the code that caused it to fire. Because spawning a new invocation of Apex is a more expensive operation than a recursive call in a single invocation, there are tighter restrictions on the stack depth of these types of recursive calls.

[4] Describes include the following methods and objects.

- ChildRelationship objects
- RecordTypeInfo objects
- PicklistEntry objects
- `fields` calls

[5] Email services heap size is 36 MB.

[6] CPU time is calculated for all executions on the Database.com application servers occurring in one Apex transaction—for the executing Apex code, and any processes that are called from this code, such as package code and workflows. CPU time is private for a transaction and is isolated from other transactions. Operations that don't consume application server CPU time

aren't counted toward CPU time. For example, the portion of execution time spent in the database for DML, SOQL, and SOSL isn't counted, nor is waiting time for Apex callouts.

[7] In a single transaction, you can only reference 10 unique namespaces. For example, suppose you have an object that executes a class in a managed package when the object is updated. Then that class updates a second object, which in turn executes a different class in a different package. Even though the second package wasn't accessed directly by the first, because it occurs in the same transaction, it's included in the number of namespaces being accessed in a single transaction.

Note:

- Limits apply individually to each `testMethod`.
- Use the Limits methods to determine the code execution limits for your code while it is running. For example, you can use the `getDMLStatements` method to determine the number of DML statements that have already been called by your program, or the `getLimitDMLStatements` method to determine the total number of DML statements available to your code.

Force.com Platform Apex Limits

The limits in this table aren't specific to an Apex transaction and are enforced by the Force.com platform.

Description	Limit
The maximum number of asynchronous Apex method executions (Batch Apex, future methods, and scheduled Apex) per a 24-hour period[1]	250,000 or the number of user licenses in your organization multiplied by 200, whichever is greater
Number of synchronous concurrent requests for long-running requests that last longer than 5 seconds for each organization. [2]	10
Maximum simultaneous requests to URLs with the same host for a callout request[3]	20
Maximum number of Apex classes scheduled concurrently	100
Maximum number of Batch Apex jobs queued or active	5
Maximum number of Batch Apex job `start` method concurrent executions[4]	1
Total number of test classes that can be queued per a 24-hour period[5]	The greater of 500 or 10 multiplied by the number of test classes in the organization
Maximum number of query cursors open concurrently per user[6]	50
Maximum number of query cursors open concurrently per user for the Batch Apex `start` method	15
Maximum number of query cursors open concurrently per user for the Batch Apex `execute` and `finish` methods	5

[1] For Batch Apex, method executions include executions of the `start`, `execute`, and `finish` methods. This is an organization-wide limit and is shared with all asynchronous Apex: Batch Apex, scheduled Apex, and future methods.

[2] If additional requests are made while the 10 long-running requests are still running, they're denied.

[3] The host is defined by the unique subdomain for the URL, for example, `www.mysite.com` and `extra.mysite.com` are two different hosts. This limit is calculated across all organizations that access the same host. If this limit is exceeded, a `CalloutException` will be thrown.

[4] Batch jobs that haven't started yet remain in the queue until they're started. Note that this limit doesn't cause any batch job to fail and `execute` methods of batch Apex jobs still run in parallel if more than one job is running.

[5] This limit applies to tests running asynchronously. This includes tests started through the Database.com user interface including the Developer Console or by inserting `ApexTestQueueItem` objects using SOAP API.

[6] For example, if 50 cursors are open and a client application still logged in as the same user attempts to open a new one, the oldest of the 50 cursors is released. Cursor limits for different Database.com features are tracked separately. For example, you can have 50 Apex query cursors, 15 cursors for the Batch Apex `start` method, and 5 cursors for the Batch Apex `execute` and `finish` methods each.

Static Apex Limits

Description	Limit
Default timeout of callouts (HTTP requests or Web services calls) in a transaction	10 seconds
Maximum size of callout request or response (HTTP request or Web services call)[1]	3 MB
Maximum SOQL query run time before the transaction can be canceled by Database.com	120 seconds
Maximum number of class and trigger code units in a deployment of Apex	5,000
For loop list batch size	200
Maximum number of records returned for a Batch Apex query in `Database.QueryLocator`	50 million

[1] The HTTP request and response sizes count towards the total heap size.

Size-Specific Apex Limits

Description	Limit
Maximum number of characters for a class	1 million
Maximum number of characters for a trigger	1 million
Maximum amount of code used by all Apex code in an organization[1]	3 MB
Method size limit [2]	65,535 bytecode instructions in compiled form

[1] This limit does not apply to certified managed packages installed from AppExchange (that is, an app that has been marked AppExchange Certified). The code in those types of packages belong to a namespace unique from the code in your organization. For more information on AppExchange Certified packages, see the Force.com AppExchange online help. This limit also does not apply to any code included in a class defined with the `@isTest` annotation.

[2] Large methods that exceed the allowed limit cause an exception to be thrown during the execution of your code.

Miscellaneous Apex Limits

SOQL Query Performance

For best performance, SOQL queries must be selective, particularly for queries inside of triggers. To avoid long execution times, non-selective SOQL queries may be terminated by the system. Developers will receive an error message when a non-selective query in a trigger executes against an object that contains more than 100,000 records. To avoid this error, ensure that the query is selective. See More Efficient SOQL Queries.

Event Reports

The maximum number of records that an event report returns for a user who is not a system administrator is 20,000; for system administrators, 100,000.

DATABASE.COM FEATURES

Internal Database Services

Database.com provides the following internal database services.

Apex Stored Procedure Classes

You can write Apex classes to perform data operations using Data Manipulation Language (DML), Database.com Object Query Language (SOQL), and Database.com Object Search Language (SOSL) to insert, update, delete, and query records.

Apex Web Services

Apex Web Services allow you to easily extend your database with new API methods.

Formulas

The Formula field type behaves much like a spreadsheet formula—the field's value is a calculation that's based on other values or fields. Formula fields can calculate and manipulate strings, dates, numbers, and regular expressions.

Outbound Messages

An outbound message is a workflow action that sends the information you specify to an endpoint you designate, such as an external service, in the form of a SOAP message.

Security

Database.com's flexible security model allows you to control who has access to the objects, records, and fields in your database.

Track Field History

Field history automatically tracks edits to records. Examples include tracking the user who edited the value of a field, the date and time when the value was changed, and the value of the field before and after the edit was made.

Triggers

Written in Apex, triggers are pieces of code that execute before or after a record is inserted, updated, deleted, or restored.

Validation Rules

Validation rules improve the quality of your data by preventing incorrect data from being saved.

Validation rules consist of an error condition and corresponding error message. For example, a validation rule can be used to ensure that the value of a field always falls within a particular range. If the number entered is not within the range, an error message is returned.

Workflow Field Updates

Workflow field updates allow you to automatically update a field value to one that you specify when a workflow rule is triggered.

Workflow

Triggers

See Also:

"What is Apex" in the Database.com Apex Code Developer's Guide
"Exposing Apex Methods as Web Services" in the Database.com Apex Code Developer's Guide
"About Formulas" in the Database.com Online Help
"Managing Outbound Messages" in the Database.com Online Help
"Securing Data Access" in the Database.com Online Help
"Tracking Field History" in the Database.com Online Help
Triggers
"About Validation Rules" in the Database.com Online Help
"Defining Field Updates" in the Database.com Online Help

Workflow

Database.com's workflow features enable you add standardized internal procedures, which are similar to triggers, to your application using point-and-click tools.

You design a *workflow rule* and associate it with a *workflow action*. A workflow action is triggered when its associated workflow rule executes.

Workflow actions include field updates and outbound messages. A field update automatically specifies a value for a field. An outbound message sends the information you specify to an endpoint you designate, such as an external service, in the form of a SOAP message. Outbound messages are a great way to notify external applications when specific events occur, or specific conditions have been met within Database.com.

Each workflow rule consists of:

- Criteria that determine when Database.com executes the workflow rule. Any change that causes a record to match this criteria can trigger the workflow rule—even changes to hidden fields.
- Workflow actions.
- Time-dependent actions that Database.com queues when the workflow rule executes.

You can use the Developer Console to debug workflow rules. The Developer Console lets you view debug log details and information about workflow rules and actions, such as the name of the user who triggered the workflow rule and the name and ID of the record being evaluated.

See Also:

"Creating Workflow Rules" in the Database.com Online Help
"Using the Developer Console" in the Database.com Apex Code Developer's Guide

Triggers

Every trigger runs with a set of context variables that provide access to the records that caused the trigger to fire. Triggers run in bulk, that is, they process several records at once.

The following trigger is associated with the Device object and executes before a new Device record is inserted.

```
trigger myDeviceTrigger on Device__c (before insert)
{
```

```
    Device__c[] devices = Trigger.new;
    MyClass.process(devices);
}
```

In all triggers, the first line of code defines the trigger. It assigns the trigger to a name, specifies the object on which it operates, and defines the events that cause it to fire. In this example, the trigger runs before new device records are inserted into the database.

The third line in the example creates a list of device records named `devices` and assigns it the contents of a trigger context variable called `Trigger.new`. Trigger context variables such as `Trigger.new` are implicitly defined in all triggers, and provide access to the records that caused the trigger to fire. In this example, `Trigger.new` contains all of the new devices that are about to be inserted.

The fourth line in the example calls the static method `process` in the `MyClass` class. It passes in the array of new devices.

See Also:

"Triggers and Order of Execution" in the Database.com Apex Code Developer's Guide

Database.com APIs

Apex
Data Access
Chatter REST API

Apex

Apex is the proprietary object-oriented programming language for executing flow and transaction control statements. Using syntax that looks like Java and acts like database stored procedures, Apex allows you to add business logic to most system events, including record updates. These include:

- Creating triggers to add logic for code to execute before or after a record is inserted, updated, deleted, or restored.
- Creating batch Apex jobs to build complex, long-running processes on Database.com. For example, you could build an archiving solution that looks for records past a certain date and adds them to an archive. Or you could build a data cleansing operation that goes through certain records and reassigns them if necessary, based on custom criteria.
- Creating Apex stored procedure classes to perform data operations using DML, SOQL, and SOSL for inserting, updating, deleting, and querying records.
- Scheduling Apex classes to run on a regular basis and batching Apex jobs.
- Employing Apex Web Services to easily extend your database with new API methods.
- Invoking external Web or HTTP services by using callouts.

Apex Development Process

Apex code must be developed on a test database organization. Apex runs on test database and production organizations but can't be written or modified in production organizations. You must create unit tests for all new Apex classes and

triggers that you write. Unit tests must have at least 75% code coverage. After you write and test your Apex code in test database, you can deploy it to a production organization.

See Also:

"What is Apex" in the Database.com Apex Code Developer's Guide
"What is the Apex Development Process" in the Database.com Apex Code Developer's Guide
"Batch Apex" in the Database.com Apex Code Developer's Guide
"Apex Scheduler" in the Database.com Apex Code Developer's Guide
"Exposing Apex Methods as Web Services" in the Database.com Apex Code Developer's Guide
"Invoking Callouts Using Apex" in the Database.com Apex Code Developer's Guide

Data Access

Database.com provides the following tools to help you query and work with your data.

Database.com REST API and SOAP API

Use the REST API and SOAP API to interact with Database.com by creating, retrieving, updating, and deleting records, maintaining passwords, performing searches, and much more. You can use the APIs with any language that supports Web services.

REST API provides a powerful, convenient, and simple Web services interface. Advantages of working with the REST API include ease of integration and development. The REST API is an excellent choice of technology to use when working with mobile applications and Web 2.0 projects.

SOAP API is optimized for real-time client applications that update small numbers of records at a time.

Force.com Bulk API

The Bulk API is based on REST principles, and is optimized for loading or deleting large sets of data. Use to insert, update, upsert, delete, or restore a large number of records asynchronously by submitting a number of batches that are processed in the background by Database.com. The Bulk API is designed to simplify the processing of a few thousand to millions of records.

Apex Data Manipulation Language (DML)

Use DML statements to insert, delete, and update data from within your Apex code.

Apex Web Services

You can expose your Apex methods as Web service operations that can be called by external Web client applications. This is a powerful tool for building efficient communication between your data service and application tier. By aggregating business logic onto Database.com, you can:

- Prevent unnecessary communication between your data service and the client
- Simplify client development and maintenance by providing a coarse-grained application-level API

- Build more robust applications, since all of the logic implemented in Apex is executed within a transaction on Database.com

See Also:

Database.com REST API Developer's Guide
Database.com Bulk API Developer's Guide
"Apex Data Manipulation Language (DML) Operations" in the Database.com Apex Code Developer's Guide
"Exposing Apex Methods as Web Services" in the Database.com Apex Code Developer's Guide

Chatter REST API

Chatter is an application that helps people share business information securely and in real time. Users employ Chatter to share information, learn about their colleagues, connect with others, and keep up with the latest record and document updates.

You can add Chatter functionality to your apps using the REST-based Chatter REST API, which is optimized to work with Web 2.0 resources. Chatter REST API makes it easy to add social functionality to applications that use Database.com. With the Chatter REST API, you can:

- Build a mobile client that displays a Chatter feed.
- Integrate a third-party Web application with Chatter so it can notify groups of users about events.
- Display a Chatter feed on an external system, such as an intranet site, after users are authenticated to your application.
- Make feeds actionable and integrated with third-party sites. For example, an app that posts a Chatter item to Twitter whenever the post includes #tweet hashtag.

See Also:

Database.com Chatter Rest API Developer's Guide

Salesforce Features in Database.com

Some Salesforce features are visible in the Database.com user interface, but contain functionality that isn't useful for managing and accessing data. We recommend you ignore the following features:

- Custom Object Import Wizard
- Desktop Integration
- Personal Groups
- Fiscal Years
- Visualforce pages in the System Log Console

USEFUL UTILITY APPLICATIONS

Workbench

Workbench is an online tool that enables developers to interact with data in their organizations via the APIs, providing a simple user interface to:

- Describe, query, manipulate, and migrate both data and metadata
- Test and troubleshoot the APIs
- Debug API traffic logs
- Test backward compatibility with previous API versions

For more information about Workbench and links to the open source community, where you can find support, see https://workbench.developerforce.com/about.php.

Workbench is a free resource provided by salesforce.com to support its users and partners, but is not considered part of our Services for purposes of the salesforce.com Master Subscription Agreement.

Data Loader

If you want to use a user interface or command line, instead of the API, to import data into your objects, you can use Data Loader.

Data Loader is a client application that you can use for bulk importing or exporting of data using the Web Service APIs. Use it to insert, update, delete, or export Database.com records. When importing data, Data Loader reads, extracts, and loads data from comma separated values (CSV) files or from a database connection. When exporting data, Data Loader outputs CSV files.

You can use Data Loader interactively through its user interface, or set up automated batch processes launched from the command line. When you use the interface, you work interactively to specify the configuration parameters, CSV files used for import and export, and the field mappings that map the field names in your import file with the field names in Database.com. When you set up batch processes through the command line, you specify the configuration, data sources, mappings, and actions in files used for automated processing.

The Data Loader offers the following key features:

- An easy-to-use wizard interface for interactive use
- An alternate command line interface for automated batch operations
- Support for large files with up to 5 million records
- Drag-and-drop field mapping
- Detailed success and error log files in CSV format
- A built-in CSV file viewer
- Support for Microsoft® Windows® 7 and Windows® XP

See Also:

Data Loader User Guide

Frequently Asked Questions About Database.com

List of Frequently Asked Questions about Database.com

Use Cases and Architecture

- What are the top use cases for Database.com?
- What architectures are typically used with Database.com

Support

- What programming languages does Database.com support?
- What platforms does Database.com support?
- With what mobile devices can users view Database.com data?
- Can I write triggers for Database.com?
- Can I write custom Web services for Database.com?

Infrastructure, Performance, and Limits

- How scalable is Database.com?
- Is there a limit to how much data can be stored in Database.com?
- Is there a limit to how many users Database.com can support?
- Will I see a degradation in performance as my application's data and number of users increases?
- What happens when the system goes down?
- What are the usage limits for an organization?

Security

- Does Database.com use my data for internal uses?
- How can I be assured my data will be kept private?
- How can I be sure my application's data is secure?
- How can I be sure my data won't be lost?
- How do I allow or restrict access to particular objects?
- How do I allow or restrict access to particular fields?
- How do I allow or restrict users' access to records they don't own?
- How do I allow only certain users to share data?
- If I use organization-wide default settings to restrict access to all records of an object, can I give some users access to certain records?
- Do I need to use Database.com's user model?

Data

- How can I import data into Database.com?
- Can I import amounts in different currencies?
- How can I import data that exists in multiple languages?
- How can I migrate data from an existing database into Database.com?

Concepts and Terminology

Salesforce and Force.com

Use Cases and Architecture

What are the top use cases for Database.com?

Building cloud and mobile applications

The world is changing. Collaboration and mobility are becoming more and more critical to today's workforce. Database.com's APIs and data feeds enable end users to connect and share data more efficiently and provide an excellent data platform for native mobile applications, including security-critical enterprise mobile applications.

Enabling secure sharing of critical business data across organizational boundaries

Business collaboration is dependent on the ability to make specific data available to certain people at the correct time, using a defined set of permissions. Database.com's data access and security capabilities allow you to declaratively model user roles, hierarchies and business rules that drive data access decisions, down to a granular level. This isolates your data access rules from the rest of your business logic, which help to make your applications more adaptable, scalable, and easier to maintain.

What architectures are typically used with Database.com?

Native mobile apps and devices

Mobile applications introduce new challenges for application development. Database.com's Chatter feeds are an ideal way to quickly surface relevant data to the end user using a small form factor device. Custom API capabilities provide the most effective way of simplifying native mobile application development by pushing most of the heavy lifting to the server side. For example, the Chatter REST API provides full and easy access to Chatter functionality.

Ruby on Heroku and other cloud platforms

Ruby applications deployed on Heroku can easily access data in Database.com through the comprehensive suite of REST and SOAP-based APIs.

Support

What programming languages does Database.com support?

Database.com supports applications written in any language that supports Web services, for example: Java, .NET, Ruby, Objective C, and PHP.

What platforms does Database.com support?

Database.com supports any platform that supports Web services. For example, mobile platforms such as iPhone®, iPad®, and Android®, as well as cloud platform such as Google App Engine®, Microsoft Azure®, Amazon Web Services®, and Facebook®.

With what mobile devices can users view Database.com data?

Users can view Database.com data that is exposed in apps on all mobile devices.

Can I write triggers for Database.com?

Yes. You can write triggers using Apex, which is Database.com's proprietary trigger and stored procedure language.

See Also:

"Triggers and Order of Execution" in the Database.com Apex Code Developer's Guide

Can I write custom Web services for Database.com?

Yes. You can write custom Web services using Apex, which is salesforce.com's trigger and stored procedure language.

Infrastructure, Performance, and Limits

How scalable is Database.com?

Database.com has the capacity to scale to the largest of applications. The architecture behind Database.com was designed to handle millions of users and large amounts of data. We use scalable application and database servers, and can scale as rapidly as your application requires.

Is there a limit to how much data can be stored in Database.com?

No. By default, the Database.com Edition includes 100,000 records. Contact salesforce.com to increase your number of available records.

Is there a limit to how many users Database.com can support?

No. By default, the Database.com Edition provides two Database.com Admin licenses and five Database.com User licenses.

To view your organization's number of active user licenses, click **Company Profile** > **Company Information** in the Database.com Console.

To increase your number of available licenses, contact salesforce.com.

Will I see a degradation in performance as my application's data and number of users increases?

The architects of Database.com are very conscious of performance and have designed Database.com to always stay ahead of customer demand. Database.com's architecture allows for easy additions of Web, application, and database servers to accommodate more data and users.

What happens when the system goes down?

Salesforce.com builds redundancy into all systems to minimize system failures that could be perceived as customer outages. All components are proactively monitored and managed so faults are detected before system outages. While there may occasionally be system outages due to issues beyond our control, we employ numerous escalation procedures to notify the proper personnel in the event of a system outage, and remedy issues as quickly as possible.

See Also:

http://trust.database.com/trust/security/

What are the usage limits for an organization?

The default usage limits are:

- 100,000 records
- Three enterprise users

- 50,000 transactions per month

To purchase additional capacity, contact salesforce.com.

Security

Does Database.com use my data for internal uses?

No. As outlined in the Privacy Statement, salesforce.com does not review, share, distribute, print, or reference your data except as provided in the salesforce.com Terms of Use, or as may be required by law. For exact information, refer to the Privacy Statement, as well as the Terms of Use agreement. You can view both items by clicking their links below the copyright at the bottom of any page.

How can I be assured my data will be kept private?

We are committed to keeping your data private and secure. For a greater understanding of the legal obligations salesforce.com adheres to regarding data privacy, refer to the Privacy Statement, as well as the Terms of Use agreement. You can view both items by clicking the relevant link below the copyright at the bottom of any page.

How can I be sure my application's data is secure?

URLs for applications using Database.com as a database are preceded with `https://` instead of `http://`, indicating that a secure connection is used. Furthermore, whenever a user's password is changed or reset, or when a user logs in to the application from a computer or device that they haven't used to log in before, they might need to activate the computer or device to successfully log in. Activating a computer or device allows Database.com to verify user identity and prevent unauthorized access.

Additionally, we use a multi-layered approach to protect key information, constantly monitoring and improving our application, systems, and processes to meet the growing demands and challenges of security.

See Also:

"Setting Login Restrictions" in the Database.com online help

How can I be sure my data won't be lost?

We back up your data with a variety of methods to ensure that your organization does not experience any data loss. Every transaction is stored to RAID disks in real-time with archive mode enabled, allowing the database to recover all transactions prior to any system failure. Every night all data is backed up to a separate backup server and high speed automatic tape library. The backup tapes are cloned as an additional precautionary measure, and the cloned tapes are transported to an off-site, fireproof vault twice a month. In addition, the facility that stores our servers is architecturally designed to withstand catastrophic events and earthquakes up to 8.0 on the Richter scale.

How do I allow or restrict access to particular objects?

Use object-level security to control the data that users can see, create, edit, and delete. Users without access to the object won't know that the object or its data exists.

See Also:

Layered Security and Sharing Design

How do I allow or restrict access to particular fields?

Use field-level security to control whether a user can see, edit, and delete the value for a particular field on an object. This allows you to protect sensitive fields without having to hide the whole object from certain users.

See Also:

"Field-Level Security Overview" in the Database.com online help

How do I allow or restrict users' access to records they don't own?

Define the default sharing model for your organization by setting organization-wide defaults, which specify the default level of access to records. For objects, organization-wide defaults can be set to Private, Public Read Only, or Public Read/Write. To access sharing and organization-wide default settings, click **Security Controls** > **Sharing Settings**.

In environments where the sharing model for an object has been set to Private or Public Read Only, you can grant users additional access to records by setting up a role hierarchy and defining sharing rules. Role hierarchies and sharing rules can only be used to grant additional access; they cannot be used to restrict access to records beyond what was originally specified with the sharing model through organization-wide defaults.

To access your organization's role settings, click **Manage Users** > **Roles**.

How do I allow only certain users to share data?

If your organization has a Private or Public Read Only sharing model, you can allow certain users to share information. You can create public groups and then set up sharing rules to specify that users in certain roles or groups will always share their data with users in another role or public group.

To access sharing settings, click **Security Controls** > **Sharing Settings**.

See Also:

"Sharing Rules Overview" in the Database.com online help

If I use organization-wide default settings to restrict access to all records of an object, can I give some users access to certain records?

Yes. Record-level security allows you to grant users access to some object records, but not others.

See Also:

"Securing Data Access" in the Database.com online help
http://trust.salesforce.com/trust/security

Do I need to use Database.com's user model?

No. However, there are many benefits to using Database.com's user model.

When you use Database.com's user model, you can manage the identity, authentication, and data security needs for your application while reducing the development resources necessary to handle these tasks.

The Database.com user and security model includes:

- Identity and user management
- Data security access and sharing controls
- Automatic authentication
- User profiles and permission sets
- A social data model and social APIs

Data

How can I import data into Database.com?

Insert your data into objects using Data Loader.

See Also:

"Inserting, Updating, or Deleting Data Using the Data Loader" in the Database.com online help

Can I import amounts in different currencies?

Yes. If your database is enabled with the ability to use multiple currencies, you can import amounts in different currencies using Data Loader or the APIs. Contact salesforce.com to enable multi-currency support for your database.

See Also:

"Inserting, Updating, or Deleting Data Using the Data Loader" in the Database.com online help

How can I import data that exists in multiple languages?

Use Data Loader.

See Also:

"Data Loader Overview" in the Database.com online help

How can I migrate data from an existing database into Database.com?

You can use Data Loader or the Web Services APIs such as SOAP API, REST API, or Bulk API. For more complex projects involving the migration of on-premise database schemas and data to Database.com, you can use a third-party migration tool, such as Informatica Cloud.

What happens to records that are deleted?

Records that are deleted are placed into the Recycle Bin.

The Recycle Bin lets you view and restore recently deleted records for 30 days before they are permanently deleted. Your organization can have up to 5,000 records per license in the Recycle Bin at any one time. For example, if your organization has five user licenses, 25,000 records can be stored in the Recycle Bin. If your organization reaches its Recycle Bin limit, Database.com automatically removes the oldest records, as long as they have been in the Recycle Bin for at least two hours.

See Also:

"undelete()" in the Force.com SOAP API Developer's Guide

How do I permanently delete records from the Recycle Bin?

Determine the IDs of the records you want to permanently delete, construct an array of the IDs, then pass the array into the emptyRecycleBin() SOAP API call.

See Also:

"emptyRecycleBin()" in the Force.com SOAP API Developer's Guide

How can I retrieve deleted data?

You can restore records that were deleted using Data Loader if they weren't hard deleted. To retrieve deleted data, use the `queryAll()` call to identify deleted records, then use the `undelete()` call to restore the deleted data. Data Loader doesn't provide an undelete function.

Concepts and Terminology

What's an organization?

An organization is the equivalent of a database, as defined in standard relational database terminology. However, unlike a traditional database, an organization contains built-in user identity, security, and social features.

What's an object?

An object is the equivalent of a database table, as defined in standard relational database terminology. In a database, each entity is represented by a *table*. A database table is simply a list of information, presented with rows and columns, about the category of person, thing, or concept you want to track.

Note, however, that an object is much more than a table because the full functionality of Database.com is behind it. Each object automatically has built-in features like a security and sharing model, Web service API access, workflow processes, and much more.

What's a field?

A field is the equivalent of a column, as defined in standard relational database terminology.

What are the differences between SOQL and SQL?

SOQL (Database.com Object Query Language) is an object query language designed for use in a multi-tenant environment. It was created to be as similar as possible to SQL, but still run efficiently and safely. Because all customers share the same resources, very expensive queries such as `SELECT * FROM *` aren't allowed.

Other differences include:

- SOQL queries can only return data sets. You can't modify the data retrieved; every SOQL command is a SELECT command.
- SOQL provides the following clauses: FROM, WHERE, WITH, GROUP BY, HAVING, ORDER BY, and LIMIT. There are also a few functions for dealing with dates and currency conversion.
- You cannot use SOQL to join unrelated sets of data. Some semi-join and anti-join queries are available, and limited traversal of child-to-parent and parent-to-child relationships are supported.

See Also:

"Salesforce Object Query Language (SOQL)" in the SOAP API Developer's Guide

Salesforce and Force.com

What are the differences between Force.com and Database.com?

Force.com is a complete application development platform that provides tools for managing the database, logic, and user interfaces of your cloud apps. Database.com provides database services only, and doesn't include the other Force.com user interface customization tools. You can use Force.com and Database.com together or separately—the tools provided by Database.com are also included in Force.com. When you use Database.com without Force.com, you can build user interfaces with the development platform of your choice.

How do Force.com and Database.com compare in available features?

If you're a Database.com user and have the Developer, Enterprise, Unlimited, or Performance Edition, you're already using Database.com when you're performing tasks such as creating custom objects, managing security, or importing data with the Force.com platform and API.

The following table compares the features included with Force.com and the standalone version of Database.com.

Area	Feature	In Force.com?	In Standalone Version of Database.com?
User Interface	System Overview page	No	Yes
Security	User management	Yes	Yes
	Ability to control access to data and functions through authentication, permission sets, profiles, roles, and sharing	Yes	Yes
Database	Standard objects	Yes	No
	Custom objects	Yes	Yes
	Schema editing	Yes	Yes
	Formulas	Yes	Yes
	Validation rules	Yes	Yes
Workflow and approvals	Outbound messages	Yes	Yes
	Field updates	Yes	Yes
	Visual Workflow	Yes	No
	Approvals	Yes	No
Coding capabilities	Triggers using Apex	Yes	Yes
	Apex classes	Yes	Yes
	Force.com SOAP API, REST API, and Bulk API	Yes	Yes
	Chatter REST API	Yes	Yes
	Force.com IDE	Yes	Yes
User interface	Page layouts	Yes	No
	Visualforce	Yes	No
	Custom views	Yes	No
	Sites	Yes	No
Data import capabilities	Data Import Wizard	Yes	No
	Data Loader	Yes	Yes
	APIs	Yes	Yes
Reporting and analytics	Reports	Yes	No
	Dashboards	Yes	No
Email messaging	Ability to send and receive emails	Yes	No
Chatter	Chatter user interface	Yes	No

Area	Feature	In Force.com?	In Standalone Version of Database.com?
	Chatter REST API	Yes	Yes
Deployment	Sandbox	Yes	No
	Test database	No	Yes
Packaging	Packages	Yes	No

If I'm already using Force.com, why would I want to use Database.com?

Database.com provides the underlying data persistence layer within Force.com. If you're using a Force.com organization, you're already using Database.com and can build applications in other languages, platforms, and devices that access your data through the APIs. If you have projects you want to deploy to a separate organization for administrative or data isolation purposes, then using a separate Database.com organization is an option.

What are the differences between the API in Database.com and the API in Force.com?

Some API operations and objects that are available in Force.com aren't available in Database.com; these correspond to the following features:

- Standard objects such as Account and Lead
- Visualforce pages and controllers
- Sites
- Approval processing
- Email services
- Packages

What are the differences between Apex in Database.com and Apex in Force.com?

Some Apex classes, methods, and interfaces that are available in Force.com aren't available in Database.com; these correspond to the following features:

- Standard objects such as Account and Lead
- Visualforce pages and controllers
- Sites
- Approval processing
- Email services
- Packages

What are the main Salesforce features that aren't included in Database.com?

End user features:

- Accounts
- Activities
- Analytics
- Answers
- Calendar
- Campaigns
- Case Management
- Contacts
- Contracts
- Content
- Customer Portals
- Dashboards
- Discussions
- Email
- Forecasting
- Ideas
- Knowledge
- Leads
- Offline
- Opportunities
- Quotes
- Search
- Solutions
- Tags
- Territories

Administrator and developer features:

- Approvals
- Assignment Rules
- Auto-Response Rules
- Custom Views
- Data Import Wizards
- Layouts
- Packaging
- Partner Portals
- Record Types
- Sites
- Visualforce

What Salesforce objects aren't supported in Database.com?

- Account
- Asset
- Campaign
- Case
- Contact
- Contract
- Document
- Event
- Idea
- Lead
- Opportunity
- Order
- Pricebook
- Product
- Question
- Quote
- Service Entitlement
- Solution
- Task

How can I access Salesforce from my Database.com organization?

You can access Database.com from your Database.com organization by using the API or Workbench.

Workbench is a free resource provided by salesforce.com to support its users and partners, but is not considered part of our Services for purposes of the salesforce.com Master Subscription Agreement.

See Also:

Data Access
Workbench

Additional Resources

Besides this guide, you can find additional Database.com documentation at docs.database.com.

Index

M

N

O

P

Q

R

S

T

U

V

W