Database.com for Architects Series White Paper

Architecting Database.com apps: a design primer

WHITE PAPER

Contents

Overview

In this paper, you'll get a broad introduction to many of the key concepts and features you'll need to understand before you begin designing and architecting apps that leverage the unique power of the Database.com cloud database service.

Orgs, sandboxes, and change sets

The traditional application development lifecycle (ADLC) usually involves several on-premises databases:

- A production database to support the production application in use with end users.
- One or more development databases to support developers as they build new application features.
- One or more test databases to support the quality assurance (QA) team as they test new features.

Each database in this equation usually requires a database administrator (DBA) to install, configure, and maintain (patch/upgrade) complex database system software—software that might also require a license and ongoing support contract. DBAs are also in charge of backing up databases, monitoring system performance, tuning, and configuring the production system for disaster recovery and failover. All these requirements are significant capital expenditures and typically slow down project development and deployment.

When your app uses Database.com, all the complicated, time-consuming, and costly problems related to database provisioning, configuration, and management don't exist thanks to orgs, sandboxes, and change sets. A Database.com *organization*, or just *org* for short, can be thought of as a database. Every Database.com account automatically comes with a *production* org ready to go—but that's not where you typically do your development. When you're set to begin building your app, with a few clicks of your mouse, you can create a *sandbox* org, which is a structural (no data) or full clone (with data) of your production database that your team can use for development or testing. Once this work is complete and you want new app features to go live, you use a simple point-and-click user interface to build, push, and deploy change sets that move metadata (the description of your schema and database configuration)per from the sandbox into production. Should you need to make a quick fix in production and push these changes back to a sandbox, you can do that, too. Best of all, there's no waiting around for a DBA to fit these otherwise blocking tasks into a busy schedule.

Keep in mind that salesforce.com is responsible for managing all Database.com orgs, not you. We handle all the heavy lifting of software installation, configuration, scaling, and performance optimization as well as data back up and disaster recovery. So with Database.com, you can allocate more of your team's budget and human resources to adding value to your apps rather than to managing and solving IT headaches.

Schema objects

You've got your Database.com account, you've created a sandbox in a matter of minutes, and now you're ready to get started building a schema to support your app. Here's a quick introduction to some of the key terms and concepts related to schema objects that you'll need to understand before you begin.

Objects (aka tables) and fields

Everyone who's used a relational database knows that the primary database storage structure is a table, a data structure that has a set of fields (columns) with specific data types like text, date, and number. Apps manage table information by creating, reading, updating, and deleting records (rows).

With Database.com, the primary data structure is called an *object*, which, with a few additional features, is more or less the same concept as what we all think of as a table.

- Every object has several *standard fields* that Database.com automatically manages for each record, including an Id field that contains a primary key value, plus Owner, CreatedBy, and LastModifiedBy fields that contain the Ids of the users who own, created, and last modified the record, respectively.
- Every object also has a *Name field* that the application manages to store a natural reference key for each record. For example, it's probable that an app would refer to records in an Album object using the field Album Name, and records in an Account object using the field Account Number. Apps commonly reference natural keys like these as labels in lists and hyperlinks, so Database.com has internal optimizations for retrieving natural keys quickly and efficiently.

- For *custom fields*, Database.com supports typical scalar data types that you might expect such as date, number, and text. But there are also rich data types such as checkbox, currency, email, phone, picklist, and url, with built-in functionality that saves your crew otherwise error-prone app development chores.
- Database.com supports auto-number and *formula field* types that you can leverage to automatically populate data rather than building your own custom app logic.
- Master-detail and look up *relationship fields* make it easy to associate two objects. There's no need to declare primary and foreign keys, configure related indexes for join performance, etc.

Declarative logic

Most organizations have processes that are more efficient and reliable when you automate them. You can easily automate a significant percentage of your business processes without the burden of writing and maintaining code thanks to Database.com *workflows*. Using a simple UI, you can quickly configure conditional workflows that, when triggered by the creation or update of a record, update fields in your database or send outbound messages to other systems. In a nutshell, think of workflows as a declarative alternative to programmatic database triggers.

Procedural logic and Apex

If declarative workflows don't meet your specific needs, then it's time to turn to a programmatic approach using *Apex*, Database.com's procedural database language. Apex, very similar to Java, is akin to the procedural languages most database systems have to build *stored procedures*, *database triggers*, and *unit tests* that implement reliable, database-centralized, complex business logic. Here's a quick look at some examples of how you can use Apex with Database.com.

A *stored procedure*, implemented as a method within an Apex class, is a named, database-side code block that centralizes some common logic that one or more apps need to execute. For example, you can use a stored procedure to implement a complex business operation that modifies related records in more than one object, all within the context of an ACID transaction. Using Apex classes, you can build a custom API for your apps by exposing class methods as *Apex Web services* (RESTful or SOAP).

An Apex *database trigger* is essentially a stored procedure you associate with an object that automatically fires (executes) before or after an app inserts or modifies a record in the object. For example, the following simple trigger prevents the deletion of a master Album record when there are related child records in the Track object. Triggers let you program Database.com objects so they automatically react to state changes in data.

```
trigger DeleteRestrictAlbums on Album__c (before delete) {

   // With each of the albums targeted by the trigger that have tracks,
   // add an error to prevent them from being deleted.
   for (Album__c targetAlbum : [SELECT Id
                                FROM Album__c
                                WHERE Id IN (SELECT Album__c FROM Track__c) AND
                                      Id IN :Trigger.old]){
       Trigger.oldMap.get(targetAlbum.id).addError('Cannot delete album with tracks');
   }

}
```

To handle complex long-running processes, consider building and asynchronously executing batch Apex. For example, batch Apex is perfect for when you want to build an archiving solution that runs on a nightly basis, looking for records past a certain date and adding them to an archive. Or you can use batch Apex to execute a data cleansing operation that operates in the background to scan an object and reassign specific records based on custom criteria. Batch Apex is easy to use: just code the class with your logic, schedule a method to execute or execute it directly, then monitor its progress as it runs in the background.

An Apex callout lets you tightly integrate your Apex with an external service by making a call to an external Web service or sending an HTTP request from an Apex script and then receiving the response. Apex callouts provide integration capabilities with Web services that use SOAP and WSDL, or HTTP services (RESTful services).

Data access and APIs

Once you have your schema and centralized business logic in place, you can start building your app to manage data. But how do you code an app to work with Database.com? Earlier, you learned how you could create your own API with Apex Web services. Now you'll explore several standard Database.com APIs that make it open and flexible enough to meet the needs of any application development environment.

CRUD APIs

The ubiquitous adoption of the World Wide Web has triggered a tremendous shift to apps that rely on fundamental Web-related technologies such as HTTP and JSON. Along with this shift, Web apps are increasingly using lightweight RESTful APIs for interaction with the persistence layer.

To create, read, update, and delete (CRUD) data records in your org, apps can rely on Database.com's *REST API*. Each resource in the REST API is a named URI that apps can use with an HTTP method (HEAD, GET, POST, PATCH, or DELETE).

For example, suppose you'd like to retrieve detailed information about a specific record in an object named Track. To do so, all your app has to do is submit a simple request similar to the following:

```
https://na1.services/data/v22.0/sobjects/Track__c/a01T000000LimtjIAB
```

The output from this request, in JSON format (XML is also an option), is as follows:

```
HTTP/1.1 200 OK
Server:
Last-Modified: Thu, 04 Aug 20xx 21:36:11 GMT
Content-Encoding: gzip
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 19 Aug 20xx 14:45:57 GMT

{
  "attributes" : {xx
    "type" : "Track__c",
    "url" : "/services/data/v22.0/sobjects/Track__c/a01T000000LimtjIAB"
  },
  "Id" : "a01T000000LimtjIAB",
  "IsDeleted" : false,
  "Name" : "01 - Help!",
  "CreatedDate" : "2011-08-04T21:36:11.000+0000",
  "CreatedById" : "00530000004puQVAAY",
  "LastModifiedDate" : "2011-08-04T21:36:11.000+0000",
  "LastModifiedById" : "00530000004puQVAAY",
  "SystemModstamp" : "2011-08-04T21:36:11.000+0000",
  "Album__c" : "a00T0000004xb2QIAQ",
  "Price__c" : 0.99
}
```

If your apps prefer SOAP to access data, Database.com also has a *SOAP-based Web services API*. After downloading a WSDL that describes your org, apps can use the API to CRUD records in your org. To make using the API even easier, there are several language-specific toolkits that provide clients for the API in a particular language, which lets your developers use familiar technology and patterns to quickly access services available in Database.com. Toolkits are available for many popular languages, including Java, .NET, and PHP.

Metadata APIs

Database.com's Metadata API is a SOAP-based API you can use to retrieve, deploy, create, update, or delete schema information in an org, such as objects and field definitions. This API is for managing schema objects and for building tools that can manage metadata, not data itself.

Bulk API

For bulk processing sizeable amounts of data, Database.com's REST *bulk* API lets you query, insert, update, upsert, or delete a large number of records asynchronously. You first send several batches to Database.com, which processes the batches in the background. As Database.com is processing batches, you can track progress by checking the status of the job. All operations use HTTP GET or POST methods to send and receive XML or CSV data.

Streaming API

Database.com makes it easy for apps to expose a near-real-time stream of data in a secure and scalable way with its *streaming API*. The API supports a publish/subscribe model in which you create one or more

named topics, each associated with an SOQL query. Apps can subscribe to one or more topics, using the Bayeux protocol. As relevant data changes, the platform reevaluates the query and, when a change occurs that alters the results of the query, the platform publishes a notification to subscribed apps.

Social API

See "Social Apps" later in this document for information about Salesforce Chatter and Database.com's related social API.

Query languages

Apps can use the *Salesforce Object Query Language (SOQL)* to construct simple but powerful database queries. Similar to the SELECT command in the Structured Query Language (SQL), SOQL lets you specify the source object, a list of fields to retrieve, and conditions for selecting rows in the source object. For example, the following SOQL query returns the value of the Id and Name field for all Account records with a Name equal to the string 'Sandy':

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

Database.com also includes a full-text, multilingual search engine that automatically indexes all text-related fields. Apps can leverage this pre-integrated search engine using the *Salesforce Object Search Language (SOSL)* to perform text searches. Unlike SOQL, which can query only one object at a time, SOSL lets you search text, email, and phone fields for multiple objects simultaneously. For example, the following SOSL statement searches for records in the Lead and Contact objects that contain the string 'Joe Smith' in the name field and returns the name and phone number field from each record found:

```
FIND {"Joe Smith"} IN Name Fields RETURNING lead(name, phone), contact(name, phone)
```

Security

Once you've got your Database.com schema along with an app's user interface and business logic in place to manage data, how do you lock things down so that only the right users see the right data at the right time? The good news is that you don't have to implement any complex code in your app because of Database.com's ingrained security model. In this section, you'll get a quick preview of powerful Database.com features you can use to quickly and easily build secure apps.

Users

Database.com has three user license types: *administrators*, *standard users*, and *light users.*

- Every organization has one or more *administrators*, super-users who can do most anything in the organization, including managing data, other users and their security, and configurations of the organization.
- A *standard user* is subject to all Database.com security controls, including authentication, profiles (system and object-level data access controls), and record sharing/row-level data access controls (more about these various controls follows).
- A *light user* is subject to Database.com authentication and profiles, but not to record-level data sharing controls. *Light users* are intended for situations where you want to code, test, and maintain your own security controls into your application.

Authentication

OAuth (Open Authorization) is an open protocol to allow secure API user authorization in a simple and standardized way for desktop, Web, and mobile apps. Database.com implements the OAuth 2.0 standard, so users can authorize apps to access org resources (via the REST and SOAP Web service APIs) on their behalf without revealing their passwords or other credentials to those apps. Apps authenticate REST calls using standard *OAuth* 2.0 access tokens, and can receive data in both XML and JSON formats.

Profiles

A *profile* is a collection of functional permissions and settings that control what users can do. You create and customize profiles to meet various types of application user requirements, and then assign each user to the appropriate profile. Profiles control the following types of access:

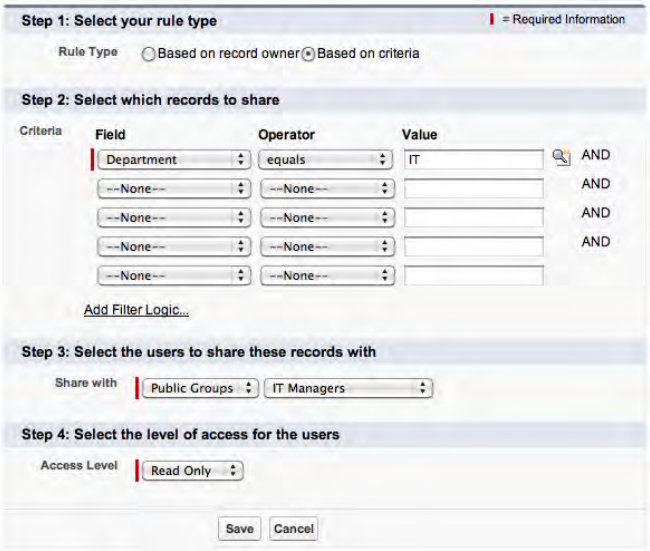
- System-level access, including time- and IP-based login restrictions as well as permissions that apply to different functions within an organization, such as the ability to manage users
- Object-level access, including CRUD permissions for each object in the database
- Field-level access, including the ability to read or edit fields
- Access to invoke Apex classes and custom logic

Record sharing

Row-level security is easy to implement with Database.com because inherent in the system's design is the notion of record-based access controls and record ownership. Database.com determines a user's access to a specific record considering the user's CRUD access (from the user's profile), the object's organization-wide default record sharing model (public or private), and the user's sharing access (which defaults to full when the user owns the record).

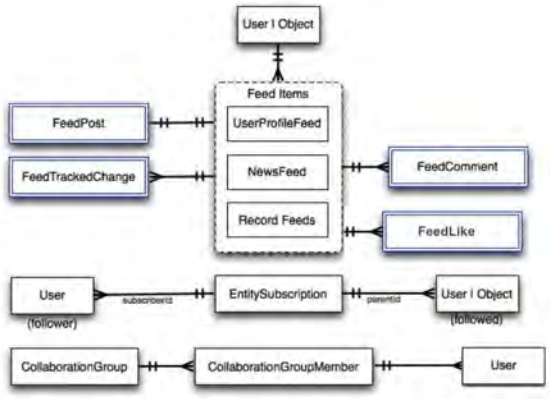
To facilitate the sharing of private records with non-owners (standard users only), you can use one or more features, including roles, groups, declarative owner-based and criteria-based sharing rules, and programmatic sharing rules.

- Users with the same *role* all live at a level of an organizational hierarchy in terms of access privilege requirements. Role hierarchies make it easy to automatically provide record access from lower to higher levels in a hierarchy (for example, a manager can access records that people in his/her group own).
- A *group* is a collection of users who all require access to a common set of records. Create a group, add users or other groups to the group, and then share records with the group so everyone can access them.
- You can easily declare *sharing rules* that determine with whom to share records based on record ownership or criteria based on field values in records. As the example figure here shows, let's say you use a custom object for job applications, with a custom picklist field named "Department." You can create a criteria-based sharing rule that shares all job applications in which the Department field is set to "IT" with all IT managers in your organization.
- For unique situations when you need to share specific records with a user or a group, developers can build *Apex-managed sharing rules* that programmatically share records in custom objects.



Social apps

Database.com includes *Salesforce Chatter*, which lets your developers embed social networking functionality into an application with minimal effort. Chatter-enabled apps let end users collaborate privately and securely by "following" each other, data records, and groups, and by sharing files and status updates. See what someone is up to. Follow the feed for a data record and get notifications when someone updates the record. These are just a few examples of the power of Chatter.



Internally, Database.com includes several core system objects that power Chatter. To associate them with your app objects, your apps leverage Database.com's *Chatter REST-based API* to access feeds and social data such as users, groups, followers, and files. Rather than building and maintaining all this complex logic yourself in your applications, simply use Chatter when you want to integrate social functions into any of your applications.

Mobile apps

Database.com is perfect for building mobile apps, either native or HTML5. In the final section of this paper, you'll revisit several aforementioned features of Database.com and explore some new ones that are of particular interest when designing mobile apps.

REST APIs and OAuth

RESTful APIs and OAuth 2.0 implementations, such as those in Database.com, are emerging as the preferred approach to database access and user authentication/authorization for mobile apps due to their adherence to open standards and lightweight nature.

Apex Web services

Optimized mobile apps require server-side features to help minimize the number of round-trips needed to perform work. With this requirement in mind, you build custom API calls for complex transactions using Apex Web services that can help curtail unnecessary network communication.

Data feeds

The preferred user experience for a limited form factor device such as a smart phone or tablet is not forcing users to navigate complex user interfaces or type in search terms to find data of interest—it's letting users subscribe to data feeds of their choice and then pushing that data to the mobile app to consume the data in a meaningful way. To support this development pattern easily, mobile apps can leverage Database.com's integrated Chatter user, record, and new feeds to publish information of interest to users, as indicated by their subscriptions.

Security

Building secure, lightweight, and scalable mobile apps is easy with Database.com thanks to its built-in security model because there's no complex security authorization code to write and maintain.

- **Step 1:** Configure users, profiles, and sharing rules in your org.
- **Step 2:** Build your app, focusing on its user interface and business logic, with simple API calls to manage data.
- **Step 3:** Configure remote access for OAuth.

When the app is live and a user logs in, Database.com automatically limits data access based on that particular user's permissions.

Mobile SDK

The Salesforce Mobile SDK provides essential libraries for quickly building native mobile apps that seamlessly integrate into the Salesforce cloud architecture. Virtually all enterprise apps require secure user authentication. By providing an out-of-the-box implementation of the OAuth 2.0 protocol, developers don't need to worry about the complexity of securely storing tokens or fetching refresh tokens when a session expires—everything is taken care behind the scenes. The SDK also provides REST app wrappers, making it easy to retrieve, store, and manipulate data in your native programming language. Over time, the SDK will provide more libraries to deliver best-in-class mobile applications that take advantage of the latest mobile device capabilities.

HTML5 is quickly emerging as the dominant technology for developing cross-platform mobile applications. Although developers can create sophisticated apps with HTML5 and JavaScript, some limitations remain, including session management, access to the camera and address book, and the inability to distribute apps inside public App Stores. The Salesforce Mobile SDK will make it possible to combine the ease of Web app development with the power of a native platform by wrapping a Web app inside a thin native container, resulting in a hybrid application.

Summary

Database.com is a proven cloud database service you can rely on to provide a management-free, scalable, and secure persistence layer for any app. To begin the app development process, you self-provision production orgs and sandbox orgs (databases) and move changes among orgs using change sets in a matter of minutes with a few mouse clicks. Then, you can rapidly construct app schemas inside orgs using Database.com objects (tables) and server-side business logic, including declarative workflows and programmatic Apex stored procedures and database triggers. Once you're ready to shift focus to app-side functionality, use the development language and platform of your choice, thanks to Database.com's embracement of open, standards-based APIs for managing your schema's data. Developing fast, secure apps that support social and mobile design requirements couldn't be easier because of Database.com's built-in features such as a full-text search engine for fast app navigation and record search, embedded system and data access controls for security, and the Chatter data model for social collaboration and data feeds for mobile apps. And don't forget, it's all backed by salesforce.com, the leader in cloud computing technology.

For More Information
Contact your account executive to learn how we can help you accelerate your CRM success.

Corporate Headquarters
The Landmark @ One Market
Suite 300
San Francisco, CA, 94105
United States
1-800-NO-SOFTWARE
www.salesforce.com

Global Offices

Latin America	+1-415-536-4606
Japan	+81-3-5785-8201
Asia/Pacific	+65-6302-5700
EMEA	+4121-6953700

WP_DBDCArchitectPrimer_2011-08-25