

Nombre: Aguirre Yoder

Fecha : 28-06-2012

Arquitectura de base de datos para móviles Androide

Referencia: <http://androideity.com/2011/10/12/manejo-de-bases-de-datos-en-android-i/>

**SQLDatabase y Cursor**

La clase SQLiteDatabase provee los métodos insert(), update() y delete() y execSQL() que nos ayuda a ejecutar sentencias SQL directamente. El objeto [*ContentValues*](http://developer.android.com/reference/android/content/ContentValues.html) permite definir claves y valores en las sentencias Insert y Update. La clave (key) corresponde a la columna y el valor es el valor para la columna.

Las consultas se pueden crear a través del método rawQuery() que acepta como parámetro una sentencia en SQL o el método query() que proporciona una interfaz para especificar los datos dinámicos o un objeto de tipo [*SQLiteQueryBuilder*](http://developer.android.com/reference/android/database/sqlite/SQLiteQueryBuilder.html). SQLiteBuilder es similar a la interfaz de un proveedor de contenidos por lo que suele utilizarse con Content Providers. Hay que saber también que toda consulta que realicemos nos retornará un objeto de tipo Cursor.

**Un acercamiento a SQLite**

**SQLite**, como su nombre indica, utiliza el lenguaje SQL para las consultas (SELECT), manipulación de datos (INSERT, DELETE, etc.), y de definición de datos (CREATE TABLE, etc). **SQLite** presenta unas pequeñas variaciones donde se desvía del **estándar SQL-92**, que aplica para la mayoría de bases de datos SQL.

Entre ellas, el uso de FOREIGN KEY, transacciones anidadas, RIGHT OTHER JOIN, FULL OTHER JOIN y algunos usos de ALTER TABLE no son válidos en SQLite.

Para utilizar SQLite no se requiere de ninguna configuración o administrador de base de datos, por lo que únicamente deberemos preocuparnos por el manejo de SQL y todo lo demás se gestionará a través del entorno de desarrollo Android.

Debes tomar en cuenta desde el principio que el uso de bases de datos implica tareas de I/O, que siempre tienen un impacto en el rendimiento de las aplicaciones, por lo que en ocasiones todo esto puede tornarse lento.

SQLite soporta los tipos de datos TEXT (similar a los String en Java), INTEGER (similar a Integer en Java) y REAL (similar a Double en Java). Si hacemos uso de cualquier otro tipo de dato, de manera automática serán convertidos para que sean compatibles con estos tres tipos de datos.

**SQLiteOpenHelper**

Para crear y trabajar con bases de datos en Android, es necesario hacer uso de la clase [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html). En esta clase es necesario sobreescribir los métodos onCreate() para crear la base de datos, y onUpgrade() para actualizar la base de datos en caso de que existan cambios en el esquema de la misma. Ambos métodos reciben como parámetro un objeto [*SQLiteDatabase*](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html).

SQLiteOpenHelper ofrece los métodos getReadableDatabase() y getWriteableDatabase() para trabajar con un objeto SQLiteDatabase y poder tener acceso de lectura y escritura sobre una base de datos.

Para definir la llave primaria de una base de datos es indispensable hacer uso del identificador \_id ya que muchas de las funciones con las que trabajaremos toman en cuenta este estándar.

**SQLite** es un manejador open source de bases de datos que combina una interfaz muy limpia de SQL y que nos permite trabajar con poca memoria y con una velocidad bastante decente, características que son necesarias cuando hablamos de entornos móviles.

SQLite soporta las características estándar de las bases de datos relacionales como la sintaxis que se basa en SQL, transacciones y la elaboración de consultas. Debido a esto, cualquier desarrollador que haya trabajado con bases de datos sin importar el entorno, no encontrará una dificultad especial en trabajar con bases de datos locales en Android. Por otro lado, hay que dejar en claro que la API nativa de **SQLite** no es compatible con JDBC contrario a lo que podría pensarse tomando en cuenta el uso de Java como lenguaje nativo en aplicaciones Android.