8-6-2017

Jonatan Gimenez – Molaro Axel

UNIVERSIDAD DE LA MATANZA

Motores de base de datos 2017

Base de datos 2 – Turno Noche

# Índice

[Índice 1](#_Toc484680353)

[Introducción 2](#_Toc484680354)

[Conceptos generales 3](#_Toc484680355)

[Presentación de los Motores de Base de Datos 4](#_Toc484680356)

[Los motores a investigar serán: 4](#_Toc484680357)

[Análisis: SQLite 5](#_Toc484680358)

[Características generales: 5](#_Toc484680359)

[Arquitectura: 5](#_Toc484680360)

[Sistemas Operativos para dispositivos móviles que la soportan: 6](#_Toc484680361)

[Modos de conexión: 6](#_Toc484680362)

[Tamaño máximo: 6](#_Toc484680363)

[Administración de consistencia de datos y concurrencia: 6](#_Toc484680364)

[Modelo de seguridad: 7](#_Toc484680365)

[Recomendaciones de cuando utilizarlo y como: 7](#_Toc484680366)

[Ventajas: 8](#_Toc484680367)

[Desventajas: 9](#_Toc484680368)

[Análisis: Microsoft Local Database for Window Phone 9](#_Toc484680369)

[Características generales: 9](#_Toc484680370)

[Arquitectura de Base de Datos para dispositivos Móviles que la identifica: 10](#_Toc484680371)

[Sistemas Operativos para dispositivos móviles que la soportan: 10](#_Toc484680372)

[Modos de Conexión: 11](#_Toc484680373)

[Administración de consistencia de datos y concurrencia: 11](#_Toc484680374)

[Modelo de Seguridad: 11](#_Toc484680375)

[Recomendaciones de cuándo y cómo utilizarla 11](#_Toc484680376)

[Ventajas: 13](#_Toc484680377)

[Desventajas: 13](#_Toc484680378)

[Análisis: InterBase 14](#_Toc484680379)

[Características generales: 14](#_Toc484680380)

[Arquitectura: 14](#_Toc484680381)

[Sistemas Operativos para dispositivos móviles que lo soportan: 14](#_Toc484680382)

[Modos de conexión: 14](#_Toc484680383)

[Tamaño máximo: 14](#_Toc484680384)

[Administración de consistencia de datos y concurrencia: 15](#_Toc484680385)

[Modelo de seguridad: 15](#_Toc484680386)

[Recomendaciones de cuando utilizarlo y como: 15](#_Toc484680387)

[Ventajas: 16](#_Toc484680388)

[Desventajas: 16](#_Toc484680389)

[Conectar un base de datos local con una remota (Punto 2) 17](#_Toc484680390)

[Sqlite – SqlServer (Android) : 17](#_Toc484680391)

[Microsoft Local Database for Window Phone – Base de datos remotas 19](#_Toc484680392)

[Glosario 20](#_Toc484680393)

[Anexo 21](#_Toc484680394)

[Bibliografía 22](#_Toc484680395)

# 

# Introducción

El motivo de este trabajo es informar acerca de distintos motores de base de datos, presentándose primero una introducción acerca de que es una base de datos, la presentación de los distintos motores de base de datos el análisis de los mismos.

En tanto al alcance del análisis este constará de:

* Características Generales.
* Arquitectura de Base de Datos para dispositivos Móviles que la identifica.
* Sistemas Operativos para dispositivos móviles que la soportan.
* Modos de Conexión.
* Servicios de Base de Datos Móviles.
* Tamaño máximo.
* Administración de consistencia de datos y concurrencia.
* Modelo de Seguridad.
* Recomendaciones de cuándo y cómo utilizarla
* Ventajas.
* Desventajas.

Finalmente se mostrarán algunas formas en las que un sistema web para dispositivo móvil puede guardar la información en la base de datos local del dispositivo, y cuando encuentra conexión sincronizarla con la base de datos principal.

# Conceptos generales

Comenzando, una base de datos es un conjunto ordenado de información perteneciente a un mismo dominio o entidad y almacenada en formato digital. Las bases de datos constituyen el soporte principal de información de empresas, instituciones o gobiernos en todos los campos del saber de la sociedad moderna: científico, de negocios, de ocio, militar, médico, educativo, etc.  
  
Los sistemas de gestión de bases de datos son programas capaces de acceder a los datos de forma rápida y estructurada, permitiendo además diferentes tipos de operaciones, entre las que se encuentran las cuatro básicas: consulta de los datos, insertar nuevos datos, modificar datos existentes o borrar datos. Usualmente al programa que realiza estas operaciones se le denomina motor de base de datos, para diferenciarlo del archivo físico que contiene la información.

El modelo relacional, es un modelo matemático establecido en 1970 por Edgar Frank Codd, un matemático que trabajaba en los laboratorios de IBM en San José, California.  
  
Este modelo define los contenedores de información como tablas formadas por filas y columnas y que a su vez pueden formar una jerarquía de relaciones entre sí. Las operaciones de consulta devuelven a su vez los datos en forma de tablas o modifican determinadas tablas de datos.

Los datos almacenados en una base de datos pueden clasificarse en **estáticos** o **dinámicos** según su variabilidad.

En tanto a su ubicación las bases de datos pueden clasificarse en locales o remotas. Las primeras residen el mismo dispositivo o terminal donde se consulta información, esto no es el caso de la base de datos remotas, que residen fuera del dispositivo o terminal desde donde se consulta o presenta la información.

En el caso de bases de datos remotas, los dispositivos móviles que acceden a través de la red, necesitan una interfaz que permita tanto enviar como recibir datos. Esta interfaz suelen ser los llamados servicios Web.

Un servicio Web una tecnología que utiliza un conjunto de protocolos y estándares abiertos que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet.

# Presentación de los Motores de Base de Datos

## Los motores a investigar serán:

1. SQLite.
2. Microsoft Local Database.
3. **InterBase**

# Análisis: SQLite

## Características generales:

El motor de base de datos SQLite en muy popular en el mercado, ya que es **Open Source ,** consume muy pocos recursos y no necesita servicios instalados para su funcionamiento.

La biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en el terminal, que cualquiera con acceso al sistema de archivos puede leer.

## Arquitectura:

Sqlite es una librería construida en C, lo cual facilita la migración a diversas plataformas de Sistemas Operativos y de dispositivos.

La forma en que una base de datos está estructurada (cantidad de tablas, registros, índices, etc.) y el conjunto de convenciones para nombrar sus objetos se les llama **Esquema**. Por lo general el esquema inicial se guarda en un Script que nos permita recuperar las condiciones previas en cualquier momento.

Con SQLite no es diferente, por lo que debes crear un esquema predefinido para implementarlo a la hora de crear tu base de datos.

## Sistemas Operativos para dispositivos móviles que la soportan:

* [Android](https://es.wikipedia.org/wiki/Android).
* [BlackBerry](https://es.wikipedia.org/wiki/BlackBerry).
* [Windows Phone.](https://es.wikipedia.org/wiki/Windows_Phone_8)
* [iOS](https://es.wikipedia.org/wiki/Apple_iOS).

## Modos de conexión:

Normalmente cuando conectamos otro gestor de bases de datos tenemos que validar los datos del equipo, el usuario y el esquema, pero con SQLite no se requiere nada de eso, ya que podemos trabajar directamente sobre la base de datos.

## Tamaño máximo:

En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño.

## Administración de consistencia de datos y concurrencia:

Solo utiliza cinco tipos de datos: TEXT, NUMERIC, INTEGER, REAL y NONE. Aunque los campos de tipo BLOB son soportados (Binary Large Objects), se recomienda almacenar esta información en el sistema de archivo y la ruta a la misma guardarla como un texto en la base de datos.

SQLite soporta accesos de lectura concurrentes, pero bloquea las operaciones de escritura que se realizan de manera secuencial. Esto es muy importante cuando se diseñan aplicaciones multiproceso.  
La integridad referencial o el mecanismo que vigila que los valores de los campos sean coherentes entre tablas relacionadas, no está activado por defecto.

Funciones SQLite :

1. .*help* – *.h* : Muestra una ayuda de comandos disponibles.
2. .*quit* - *.q* : Termina la sesión de la consola SQLite, retornando a la de comandos.
3. .*exit* - *.e* : Termina la sesión d ela consola SQLie . retornando a la de comando.
4. .*databases* : Lista de bases de datos en uso.
5. .*tables* : Lista las tablas de la base de datos.
6. .*tables* *‘%N’* : Lista las tablas que empiecen con N .
7. .*backup* *ARCHIVO* : Realiza un copia de seguridad de la base de datos principal a ARCHIVO.
8. .*backup* *‘BBDD’* ARCHIVO : Realiza una copia de seguridad de la base de datos ‘BBDD’ a ARCHIVO.

## Modelo de seguridad:

Para utilizar este tipo de motor de base de datos se crea un archivo para el esquema completo de una base de datos, lo que permite ahorrarse preocupaciones de seguridad, ya que los datos de las aplicaciones mobile no pueden ser accedidos por contextos externos.

No tiene sistema de autenticación de usuario.

## Recomendaciones de cuando utilizarlo y como:

Se recomienda utilizar SQLite , cuando la conexión del dispositivo al que está orientado mi aplicación , es un dispositivo con mala , lenta conexión a Internet. Támbien se recomienda utilizar este motor cuando se necesita utilizar datos con frecuencia que son irrelevantes para el componente del servidor, estos se guardan en la memoria cache de SQLite.

En términos genérales , para utilizar SQLite , primero se debe descargar la Liberia de [**https://sqlite.org/download.html**](https://sqlite.org/download.html), incluirla en el proyecto , y dependiendo del sistema operativo se deben descargar distintas dependencias y crear distintas clases para realizar las tareas de : conexión, inserción, modificación y borrado.

Para especificar un poco en el tema, se va a continuar explicando cómo implementar SQLite en una aplicación Android Mobile.

Para crear y gestionar SQLite en Android tendremos que usar **android.database.sqlite**. Aquí encontraremos las clases necesarias para crear y actualizar la base de datos, para realizar querys, sentencias precompiladas, inserts, updates y deletes, la implementación de Cursor, etc.

Veamos las principales clases:

* [**SQLiteCursor**](http://developer.android.com/reference/android/database/sqlite/SQLiteCursor.html): nos permite recuperar los datos, mediante una select o con el método query que ejecuta también una select. Podremos iterar sobre cada uno de los registros, columna a columna, como si se tratase de un resultset.
* [**SQLiteDatabase**](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html): expone métodos para gestionar los datos en una base de datos, como insertar, actualizar, eliminar, ejecutar sentencias sql, abrir y cerrar las conexiones, trabajar de forma transaccional.
* [**SQLiteOpenHelper**](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html): nos permite diseñar, crear, actualizar la base de datos.
* y gestionar la versión de la misma.
* [**SQLiteQueryBuilder**](http://developer.android.com/reference/android/database/sqlite/SQLiteQueryBuilder.html): es un helper para crear sqls.
* [**SQLiteStatement**](http://developer.android.com/reference/android/database/sqlite/SQLiteStatement.html): para trabajar con sentencias precompiladas.

Al contrario que con bases de datos como MySQL o SQL Server, en Android, no dispondremos de ninguna herramienta para construir la base de datos, sino que tendremos que hacerlo mediante comandos sql directamente en el código. Para crear la base de datos tendremos que sobreescribir la clase SQLiteOpenHelper.

Para crear la base de datos tendremos que sobreescribir la clase SQLiteOpenHelper, los métodos onCreate(SQLiteDatabase) y onUpgrade(SQLiteDatabase), pudiendo también sobreescribir onOpen(SQLiteDatabase), onDowngrade(), etc. En onCreate vamos a insertar nuestros comandos de creación de las tablas, en onUpdate, introduciremos el código para los casos en que se produzca un cambio en la versión de la base de datos y por lo tanto haya que modificar alguna tabla. Finalmente, onOpen, lo utilizaremos para abrir una base de datos que ya está creada.

## Ventajas:

* Es Open Source.
* Motor de base de datos liviano.
* Se puede usar el modo ventana de comandos o embebido en aplicaciones de código.
* Es portable.
* [SQLite](http://www.sqlite.org/) es usada por muchos frameworks como base de datos por defecto durante el desarrollo de nuevos proyectos. La ventaja de esto es que así se puede trabajar desde el primero momento con bases de datos relacionales, sin tener que preocuparnos por configurar adecuadamente ningún sistema gestor tradicional.
* No requiere el soporte de un servidor, ya que no ejecuta un proceso para administrar la información, si no que implementa un conjunto de librerías encargadas de la gestión.
* Libera al programador de todo tipo de configuraciones de puertos, tamaños, ubicaciones, etc.

## Desventajas:

* Se limita a algunos tipos de datos.
* Las consultas SQL, aunque siguen prácticamente todo el estándar ANSI SQL-92, no soportan RIGHT ni FULL OUTER JOINs que permiten unir varias tablas en el resultado de una consulta compleja.
* Por último, las columnas una vez definidas, no pueden ser modificadas o borradas con la sentencia ALTER TABLE.

# Análisis: Microsoft Local Database for Window Phone

## Características generales:

Microsoft Corporation ha realizado su propia implementación de un motor de base de datos relacional para los dispositivos móviles que ejecuten el sistema operativo Windows Phone.

Utiliza un mecanismo denominado LINQ to SQL, para todas las operaciones con la base de datos que es almacenada en el almacenamiento aislado de la aplicación (isolated storage). Entre las operaciones se incluyen tanto las de consulta y modificación de datos así como las de definición o creación del propio esquema de la base de datos.  
  
LINQ o Language Integrated Query (Lenguaje Integrado de Consulta), es un sublenguaje de manipulación de datos que está incluido en toda la familia de lenguajes de programación de la plataforma Microsoft .NET, tales como Visual Basic o C#.

Aunque LINQ soporta inicialmente consultas en colecciones en memoria, bases de datos relacionales y datos XML, es una arquitectura extensible que permite a desarrolladores de orígenes de datos adicionales el uso del LINQ, implementando los operadores de consulta estándar como métodos extensores para sus orígenes de datos, o mediante la implementación de la interfaz IQueryable que permite convertir un árbol de expresión en tiempo de ejecución para transformarlo en algún lenguaje de consultas.  
Los operadores de consulta estándar son usados para objetos también y permiten consultar objetos en la memoria con la misma sintaxis LINQ.  
Cuando la aplicación en Windows Phone se ejecuta y se produce un acceso a los datos relacionales, LINQ to SQL traduce las consultas de lenguaje integrado a sentencias del dialecto Trasact-SQL que utiliza Microsoft en sus bases de datos y las envía al motor de base de datos local para que sean ejecutadas.  
Cuando el motor devuelve los resultados, LINQ to SQL los traduce en objetos que puedan ser manipulados desde el propio lenguaje de programación que estemos utilizando en Windows Phone.

## Arquitectura de Base de Datos para dispositivos Móviles que la identifica:

Para almacenar y recuperar datos en una base de datos local, una aplicación de Windows Phone utiliza LINQ a SQL. LINQ a SQL proporciona un enfoque orientado a objetos para trabajar con datos y comprende un modelo de objeto y un tiempo de ejecución.

El modelo de objetos LINQ a SQL está compuesta principalmente por el System.Data.Linq.DataContext objeto, que actúa como un proxy para la base de datos local. El tiempo de ejecución de LINQ to SQL es responsable de salvar el mundo de los objetos (el objeto DataContext) con el mundo de los datos (la base de datos local). Esta relación se resume en la siguiente imagen.



## Sistemas Operativos para dispositivos móviles que la soportan:

* Windows Phone.

## Modos de Conexión:

En una implementación de aplicaciones estándar, la base de datos local se crea en la carpeta local la primera vez que se ejecuta la aplicación.Después de eso, se añade datos de la aplicación a la base de datos como se utiliza la aplicación. Para incluir un conjunto de pre-poblada de datos de referencia, junto con su aplicación, añadir un archivo de base de datos local para su aplicación.

## Administración de consistencia de datos y concurrencia:

El contexto de datos es un proxy, un objeto que representa la base de datos. Un contexto de datos contiene Tabla objetos, cada uno de los cuales representa una tabla en la base de datos. Cada objeto de tabla se compone de entidades que corresponden a filas de datos en una base de datos. Cada entidad es un simple objeto CLR con atributos. Los atributos de cada entidad determinan la estructura de la tabla de base de datos y definen la correspondencia entre el modelo de objeto de los datos y el esquema de la base de datos.

LINQ a SQL proporciona capacidades de mapeo objeto-relacional que permiten su aplicación logrado utilizar Language Integrated Query (LINQ) para comunicarse con una base de datos relacional (que sólo “habla” de Transact-SQL). LINQ a SQL asigna el modelo de objetos, que manifiesta con código administrado de .NET Framework, a una base de datos relacional. Cuando su aplicación se ejecuta, LINQ a SQL traduce consultas en lenguaje-integrado en Transact-SQL y luego envía las consultas a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ a SQL traduce los resultados de vuelta a los objetos que se puede trabajar con en su propio lenguaje de programación.

## Modelo de Seguridad:

La base de datos local proporciona protección mediante contraseña y el cifrado para ayudar a asegurar su base de datos. Cuando se utiliza una contraseña con la base de datos, la base de datos está cifrada. Con el fin de cifrar la base de datos, proporcionar una contraseña en la cadena de conexión de base de datos (el constructor contexto de datos) antes de crear la base de datos. Cada vez que acceda a la base de datos, tendrá que proporcionar la contraseña. No se puede cifrar la base de datos después de que ha sido creado. La base de datos se cifra mediante AES-128 y la contraseña se hash usando SHA-256.

## 

## Recomendaciones de cuándo y cómo utilizarla

Se recomienda utilizar cuando se necesita una base de datos compatible con apps Windows Phone.

**Como:**

Inserción de datos

Insertar datos en la base de datos es un proceso de dos pasos. Primero se debe agregar un objeto al contexto de datos, a continuación, llamar al contexto de datos SubmitChanges método para persistir los datos como una fila en la base de datos.

actualización de datos

Hay tres pasos para la actualización de datos en la base de datos local. En primer lugar, consultar la base de datos para el objeto que se va a actualizar. A continuación, modifique el objeto como se desee. Por último, llame al SubmitChanges método para guardar los cambios en la base de datos local.

Eliminación de datos

Eliminación de datos en la base de datos también se compone de tres pasos. En primer lugar, consultar la base de datos para los objetos que se van a eliminar. Luego, dependiendo de si usted tiene uno o más objetos a eliminar, llamar a la DeleteOnSubmit o DeleteAllOnSubmit método, respectivamente, para poner esos objetos en un estado pendiente eliminar. Por último, llame al SubmitChanges método para guardar los cambios en la base de datos local.

Cambiar el esquema de base

El Microsoft.Phone.Data.Linqespacio de nombres proporciona la DatabaseSchemaUpdater clase para ayudar con los cambios de esquema de base de datos.

El DatabaseSchemaUpdater clase puede realizar cambios a la base de datos de aditivos, tales como la adición de tablas, columnas, índices o asociaciones. Para cambios más complejos, tendrá que crear una nueva base de datos y copiar los datos en el nuevo esquema, según sea el caso.

## Ventajas:

* Reside en la carpeta local de su aplicación. Una base de datos local sólo se puede acceder mediante la aplicación de teléfono de Windows correspondiente. Debido a que el archivo de base de datos reside en la carpeta local, no hay otras aplicaciones pueden acceder a esos datos
* Son livianas

## Desventajas:

* Una base de datos local sólo se puede acceder con LINQ a SQL; Transact-SQL no es compatible.

# **Análisis: InterBase**

## Características generales:

InterBase es un gestor de base de datos relacionales que se caracteriza por su bajo consumo de recursos (una instalación mínima puede ocupar alrededor de 400KB, mientras que una instalación máxima alrededor de 40MB), su adaptación a funcionar integrado en un servidor (embebido) o de manera regular y ofrece opciones de encriptación **AES** en 256bit. Es **OpenSource**.

## Arquitectura:

InterBase al ser multiplataforma puede usarse en distintos sistemas operativos (Windows, Linux, Solaris). Soporta diferentes paradigmas o lenguajes de programación, ademas de tener excelente integración con código en C, C++, .NET, Delphi, PHP y Ruby.

## Sistemas Operativos para dispositivos móviles que lo soportan:

* [Android](https://es.wikipedia.org/wiki/Android)(2.3.3 mínimo).
* iOS(5.1 mínimo).

## Modos de conexión:

Se requiere una librería de acceso a datos como FireDac. Esta se puede configurar para acceder a la base de datos de diversas maneras (TLS/SSL, remoto vía TCP/IP, de forma local, a un archivo del disco).

Se puede elegir entre autenticación del sistema operativo o la propia de InterBase.

## Tamaño máximo:

No existe un limite impuesto, ya que es dependiente del tamaño de **paginación** del dispositivo.

Ejemplo:

1KB de paginación: Máximo de 2TB.

2KB de paginación: Máximo de 4TB.

4KB de paginación: Máximo de 8TB.

8KB de paginación: Máximo de 16TB.

16KB de paginación: Máximo de 32TB.

## Administración de consistencia de datos y concurrencia:

InterBase utiliza un control de concurrencia mediante versiones múltiples (Multiversion concurrency control o MVCC). Esto significa que mientras un usuario se encuentre trabajando con un objeto de la base de datos, existirán copias del mismo para el acceso de otros usuarios. Mientras el usuario inicial está realizando una transacción, los demás usuarios verán la copia “original”. Cuando el primer usuario finalice su transacción, si el siguiente usuario quiere realizar alguna modificación sera advertido de que hubo una modificación en el registro en el que está trabajando.

InterBase soporta los datos: BLOB, BOOLEAN, CHAR, DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, NUMERIC, SMALLINT, TIME, TIMESTAMP y VARCHAR.

## Modelo de seguridad:

InterBase guarda nombres y contraseñas autorizadas en una base de datos especial de seguridad que reside en el directorio de instalación, la cual usa claves de encriptación.

## Recomendaciones de cuando utilizarlo y como:

Es muy útil cuando se necesita acceder a datos y no se tiene conexión a Internet.

**Ejemplo de uso en Android e iOS**

Cuando trabajamos con bases de datos en Android e iOS, necesitamos establecer la ruta donde se encuentra el fichero que contiene esa base de datos a través del Deployment Manager (Administrador de despliegues). El archivo RECIPES.IB está almacenado en la misma carpeta que el proyecto.

A continuación se selecciona esa ruta para dar el data module (módulo de lectura/escritura de datos) para la base de datos. Luego se utiliza una función para que funcione en todas las plataformas y establezca en tiempo de ejecución la respectiva carpeta.

*uses System.IOUtils;*

*procedure TDataModule1.DataModuleCreate(Sender: TObject);*

*begin*

*SQLConnection1.Params.Values['Database'] := IncludeTrailingPathDelimiter*

*(TPath.GetDocumentsPath) + ‘RECIPES.IB’;*

*end;*

## Ventajas:

* Es Open Source.
* Sistema de versiones múltiples para administrar concurrencia.
* Cercanía al estardar SQL.
* Facilidad y uso de configuración.
* Poco uso de recursos.
* Mínima necesidad de administración.

## Desventajas:

* Dificultad para particionar la base de datos en distintos archivos, lo que perjudica su rendimiento físico.
* El sistema de versiones múltiples puede afectar el rendimiento. Una función COUNT debe recorrer todos los registros, aunque exista algún índice sobre la tabla requerida.

# Conectar un base de datos local con una remota (Punto 2)

Llamamos sincronización local ,a el proceso mediante el cual los datos que existan en la base de datos remota se guarden en la base de datos mobile (En este caso).El caso contrario es denominado sincronización remota.

## **Sqlite – SqlServer (Android) :**

Una de las formas de sincronización es mediante la implementación del patrón [SyncAdapter](https://developer.android.com/intl/es/training/sync-adapters/index.html) ;

El patrón está basado en un componente llamado [SyncAdapter](https://developer.android.com/intl/es/training/sync-adapters/index.html" \t "_blank), cuya funcionalidad es sincronizar datos en segundo plano entre una aplicación Android y un servidor remoto.

Se caracteriza por realizar las acciones asincrónicamente, es decir, en periodos de tiempo sin inicio o fin determinados. Por ello puede que la transferencia de datos no suceda cuando se espera, sin embargo,se asegura la integridad de la información.

Esta arquitectura se enfoca en evitar realizar la sincronización en el UI Thread. Lo que libera a las actividades de operaciones en la base de datos, peticiones Http, etc.

Si observas el diagrama anterior, se muestra un [ContentProvider para implementar el modelo de datos](https://developer.android.com/intl/es/training/sync-adapters/creating-stub-provider.html). Esto se debe a la seguridad que brinda a la estructura de la base de datos Sqlite y la capacidad de compartir dicha información entre aplicaciones.

Además, el content provider permite la actualización de la interfaz de usuario en tiempo real gracias a los Loaders.

Implementar Un SyncAdapter

Este proceso es controlado por un elemento del sistema llamado SyncManager, el cual se encarga de añadir a una cola de gestión las sincronizaciones de todas las aplicaciones.

Incluso comprueba que la conexión a la red esté disponible antes de iniciar una sincronización. También se encarga de reiniciar una sincronización fallida por si nuestro Sync Adapter ha fallado.

Para [acceder a este framework](https://developer.android.com/intl/es/training/sync-adapters/creating-sync-adapter.html) simplemente debemos usar cuatro piezas de código:

* Un Content provider que proporcione flexibilidad y seguridad a los datos locales. Este elemento es obligatorio. Si tu aplicación no usa un content provider, entonces crea uno auxiliar para satisfacer la característica.
* Un Sync Adapter que extienda de la clase AbstractThreadedSyncAdapter que maneje la sincronización de la aplicación. Donde se sobrescribe el método onPerfomSync() para indicar las acciones de actualización, peticiones http, parsing, etc.
* Un bound service que esté registrado para escuchar el filtro android.content.SyncAdapter y que retorne en su método onBind() el Sync Adapter. Este servicio se comunicará con el sistema para controlar la sincronización.
* Una definición XML que le diga al Sync Manager de qué forma se manipularán los datos a sincronizar.

¿Cómo se inicia un Sync Adapter?

Existen varias formas que pueden ejecutar la sincronización  de este elemento.

**Por cambios en el servidor** : En este escenario, el Sync Adapter se inicia debido a que se produce una petición desde el servidor hacia el dispositivo Android, cuando los datos en él cambian .

**Por cambios en el contenido local:** Cuando los datos del Content Provider son modificados en la aplicación local, el Sync Adapter puede iniciarse automáticamente para subir los datos nuevos al servidor y asegurar una actualización.

**Al enviar mensaje de red**: Android comprueba la disponibilidad de la red enviando un mensaje de prueba con frecuencia. Podemos indicar a nuestro Sync Adapter que se inicie cada vez que este mensaje es liberado.

**Programando intervalos de tiempo**: En este caso podemos programar el Sync Adapter para que se ejecute cada cierto tiempo continuamente o si lo deseas, en una hora determinada del día**.**

**Manualmente:** El sincronizador se inicia por petición del usuario en la interfaz.

## Microsoft Local Database for Window Phone – Base de datos remotas

Cuando se logre conectar la base de datos Microsoft Local Database for Window Phone , a la misma red del servidor que se comunica con la base de datos remota, se pueden sincronizar mediante es uso de Web Services WCF .

# Glosario

**Datos estáticos:** de solo lectura.

**Datos dinámicos:** estos datos se pueden borrar, modificar y agregar. (Datos de lectura y escritura).

**Bases de datos locales:** Su acceso es muy rápido y por lo general contiene información que no es compartida con otros usuarios.

**Base de datos remotas:** La velocidad de acceso es más lenta, con fuerte dependencia de la conectividad y ancho de banda de la red a través de la cual se realizan las consultas, pero con la ventaja de poseer un motor muy potente y grandes volúmenes de información que pueden compartirse entre miles de usuarios.

**Open Source :** (Código abierto) es el término con el que se conoce al software distribuido y desarrollado libremente.

**android.database.sqlite :** se trata de un paquete genérico, no es específico de ninguna base de datos. Aquí tenemos interfaces tan importantes como Cursor (nos permitirá recuperar datos) y clases de ayuda como DatabaseUtils.

# Anexo

**SQLite :** Es la base de datos seleccionada por Research in Motion, Apple y Google para servir de base de datos local en sus aplicaciones desplegadas en los sistemas operativos para dispositivos móviles Blackberry OS, Blackberry PlayBook OS, iOS y Android.

**Link de descarga de SQLite :** [**https://sqlite.org/download.html**](https://sqlite.org/download.html) **.**

**Link to SQL LINQ to SQL** en Windows Phone no soporta directamente la ejecución de Transact-SQL, incluyendo las declaraciones de datos Modeling Language (DML) lenguaje de definición de datos (DDL) o. Además, las aplicaciones de Windows Phone no pueden utilizar LINQ a SQL para acceder directamente a los objetos de ADO.NET.

**DatabaseSchemaUpdater :** La base de datos no cambia para reflejar las actualizaciones de la DatabaseSchemaUpdater objeto hasta que la ejecución se llama al método. Cuando se llama a este método, todos los cambios se envían a la base de datos local como una sola transacción, incluyendo actualizaciones de versión. El uso de una sola transacción ayuda a la base de datos a mantener la integridad, como por ejemplo en los casos en que el usuario sale de la aplicación durante una actualización.

# Bibliografía

<http://jmaw.blogspot.com.ar/2012/07/bases-de-datos-en-dispositivos-moviles_23.html>

<https://prezi.com/q8a2xvgjzqri/arquitectura-y-estructura-de-sqlite/>

<https://medium.com/jmtorres/almacenar-datos-en-sqlite-2e6785462cc4>

http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

<https://msdn.microsoft.com/en-us/library/windows/apps/hh202860(v=vs.105).aspx>

<http://www.hermosaprogramacion.com/2015/07/como-sincronizar-sqlite-con-mysql-en-android/>

http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/