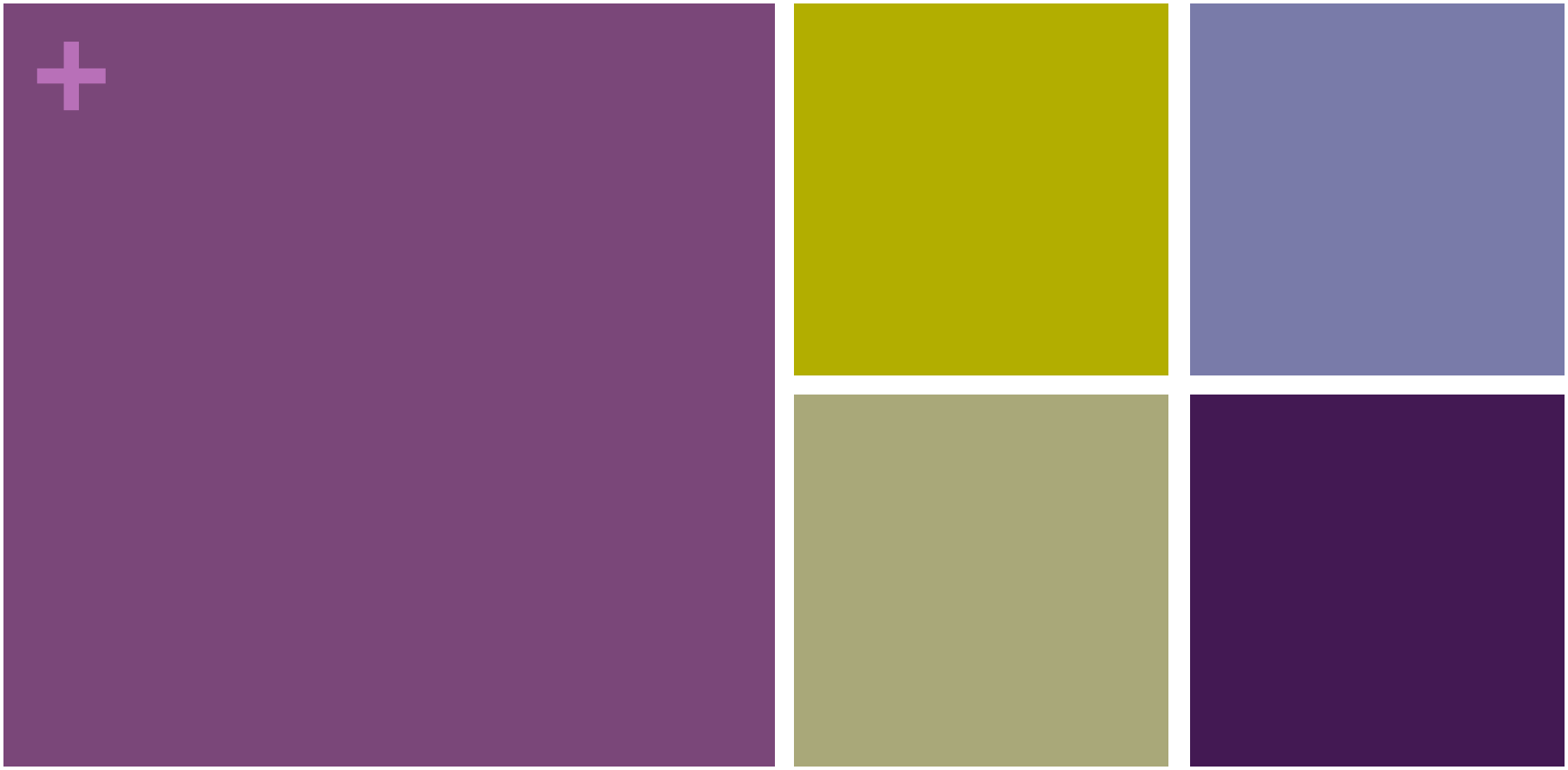


Object Relational Database Mapping

Alex Boughton

Spring 2011

Presentation Overview

- Overview of database management systems
- What is ORDM
- Comparison of ORDM with other DBMSs
- Motivation for ORDM
  - Quick Example
- How does ORDM work
  - Attributes of ORDM
  - Discussion of ORMLite framework
    - Main Components
    - Conceptual Diagrams and Comparisons
- In depth look at ORMLite and using its support for Android OS
  - More Motivation
  - Comparison using ORDM and RDBMS in an Android application
    - Trade offs specific to ORMLite and Android Applications

Overview of Database Technologies

- Relational Database Management Systems introduced in 1970[3]
  - mySQL, SQLite3
- Object Oriented Database Management Systems were released next

Overview of Database Technologies (cont.)

- Object Relational Mapping Database Management APIs were developed out of motivation to combine the good things from the previous two.
  - Introduced in 1990's[1]
  - Combined rich data type support and code minimization offered by OODBMS with the speed and portability of RDBMS

* There are other types of database management systems (i.e. Document-oriented, NoSQL) but they are not mentioned here because they are not in the scope of object oriented development.

What is Object Relational Database Mapping?

- Programming technique typically utilized through an API
  - Can be implemented by the application
- A few popular APIs[1]
  - Hibernate – widely used Java implementation
  - ORMlite – Java implementation, gaining popularity with its new Android support (more on this API later)
  - Django – Python framework that has ORDM support built in
  - Core Data – Cocoa framework that ships with Mac OSX and iPhoneOS

Object Relational Database Mapping Advantages

- Allows developers to convert data from rich data types used in object oriented programming languages to lower level relational database types
  - Allows the use of self defined ADTs in database storage
  - Helps developers maintain continuity in databases that are shared between applications
    - For example instead of storing "address" as a VARCHAR, you can store it as an ADT Address with the specified format of that data type. Have all applications use the same Address class and continuity is achieved.

Object Relational Mapping Limitations

- With the ability to have complex ADTs, efficiency may be lower for large scale applications
- Database size can also increase depending on the amount of methods stored with the objects and the number of entries stored
- Many larger applications with frequent database read/write activity will not gain from ORDM.
- ORDM can be very useful for mobile phone use(example later) and small applications

What is a Relational Database?

- Relational Database Management System (RDBMS)
  - Simple data with queries
  - Example of data types in SQLite3 (Supported on Android OS and iPhone OS)[2]

Data Type	Description
NULL	Null value
INTEGER	signed integer
REAL	floating point value, stored as an 8-byte IEEE floating point
BLOB	blob of data, stored exactly as it was input
TEXT	text string, stored using the database encoding

Relational Database Data Type Limitations and Advantages[3]

- Lack of support for abstract data types (ADTs) causes extra code in applications for going in between raw data types and ADTs
- The simplistic data types can be faster for large databases and applications where speed is important
- Uses Structured Query Language (SQL) which makes databases portable across different systems

What are Object Oriented Databases?[1]

- Essentially allows objects to be stored persistently with support for storage and retrieval
- Allows for complex data storage but does not use SQL for query support
- ODL – Object Defining Language
- OID – Object ID
- OQL – Object Query Language; similar to SQL but has features for inheritance, polymorphism, and object identity

Object Oriented Database Management Systems Limitations and Advantages[3]

- Decreases the amount of code in applications because they don't need to generate SQL statements
- Poor performance and scalability compared to other data storage methods
- Difficulty catching on because with lack of standards developers are cautious about using these systems
- Introduced the idea for better data type support in database systems

Motivation for ORDM Technology[4]

- Wanted the efficiency from having the underlying RDBMS implementation, along with the portability gained by using the SQL
- Wanted ADTs, inheritance, and polymorphism as in OODBMS implementation
- Result was a software layer between RDBMS and applications to provide both.
- ORDM Technology arose in the 1990's and gained popularity with the rise of social media and web applications[3]

Code Example for RDBMS[4]

```
SELECT InitCap(C.Surname) || ', ' || InitCap(C.FirstName), A.city
FROM Customers C JOIN Addresses A ON A.Cust_Id=C.Id -- the join
WHERE A.city="New York"
```

- Query to select names of customers whose address is in New York
- The query requires information from "Customers" and "Addresses" because the database does not know the relationship between customer id's and the unique identifier in the table "Addresses"

Code Example for ORDBMS (cont.)

```
SELECT Formal( C.Name )
FROM Customers C
WHERE C.address.city="New York" -- the linkage is 'understood' by the ORDB
```

- The same query to select the names of customers who live in New York
- The query becomes more simple for the developer because the database knows that the customers are linked to the address by their id's

Attributes Common to ORDM APIs

- Database
  - Underlying RDBMS implementation
  - ORMLite supports MySQL, SQLite, Postgres, Oracle and others
- Query Representation
  - Offer a simplified way for developers to issue queries
  - Many APIs also offer support for executing raw SQL statements
- Data Object
  - Some API's have interfaces that the application defined objects must implement
  - Usually require an empty constructor
  - Must use some mark up language(@ notation in eclipse) or some other way to denote which data fields will be stored

Attributes Common to ORDM APIs (cont.)

- Iterator
  - Sometimes referred to as a cursor
  - Typically the API will implement this class as a subclass of the RDBMS's Iterator class, giving the developer a better encapsulation for
- Objects for logging support
- Other Database specific classes that vary with the RDBMS implementation

A Closer Look at a Specific ORDM API

Why ORMLite?

- ORMLite is commonly used for Java applications and supports many different types of underlying relational database implementations
- Mobile application development is becoming more and more important for software solutions and ORMLite has released a new and more reliable Android supported version of their API
  - Until looking into ORMLite, support for more advanced database uses in Android apps was sparse

A Closer Look at a Specific ORDM API (cont.)

- Learning to use ORMLite with mobile application development will help make better mobile applications that can be more easily maintained and created
- ORMLite in Android shows how an object oriented language based application can benefit from using ORDM technology
  - With out the technology there is a bottleneck at the data transfer point – all of your lovely class structures and OO designs need to be flattened and simplified back to a construct that (in the case of SQLite) may only have 5 data types!!
  - There is an elegant solution to the data storage problem that complements the designing and analysis that is involved in using OO, and ORMLite is a "lite" way to start learning about it

ORMLite: Main Components

ORMLiteSqliteOpenHelper[5]

- Abstract class that extends android.database.sqlite.SQLiteOpenHelper
- The implementation should include:
  - Wrappers for the getDao(Class<T> data) method should be implemented for each type of Dao that is needed (ie number of data classes in the system)

```
public Dao<MyData, Integer> getMyDataDao() throws SQLException {
                    if (MyDataDao == null) {
                            MyDataDao = getDao(MyData.class);
                    }
                    return MyDataDao;
            }
```

ORMLite: Main Components

Activity Support

- ORMLite extends all the variations of Android Activity (ie ListActivity, TabActivity)
- Inherits regular methods from activities in Android, but then adds a few of its own to assist with data management

ORMLite: Main Components

DAOs

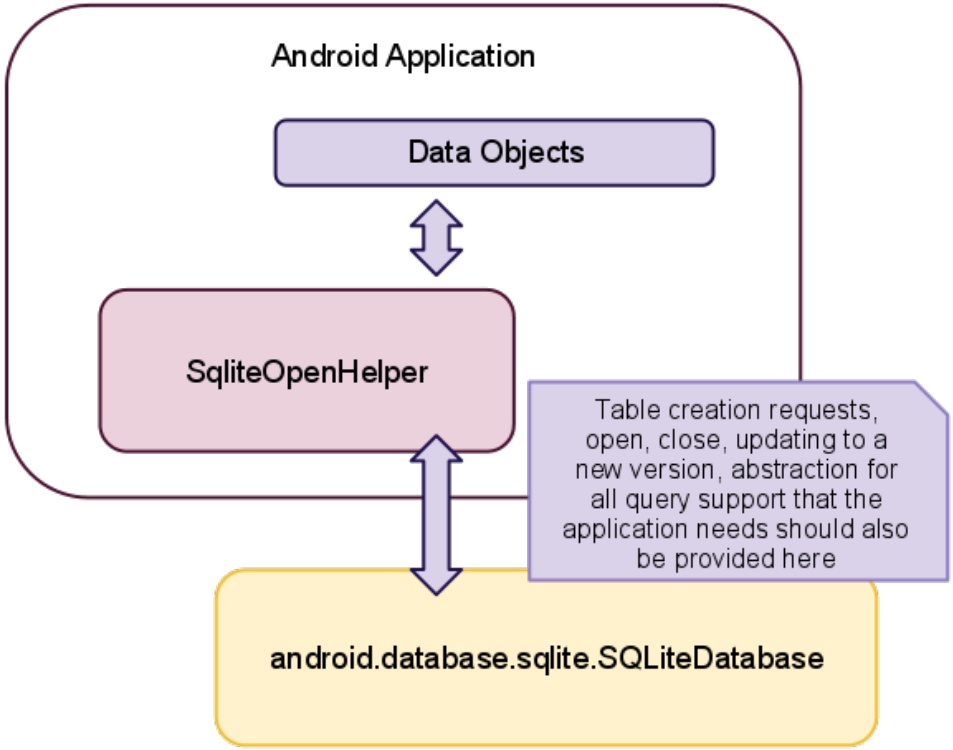
- Data Access Objects
- Act as a bridge between the database implementation and the data class
- Dao<T,ID> is an interface and it is implemented by BaseDaoImpl<T,ID>
- ORMLite uses the factory design pattern to create new Dao's within the implementation of OrmLiteSqliteOpenHelper

ORMLite: Main Components

DAOs

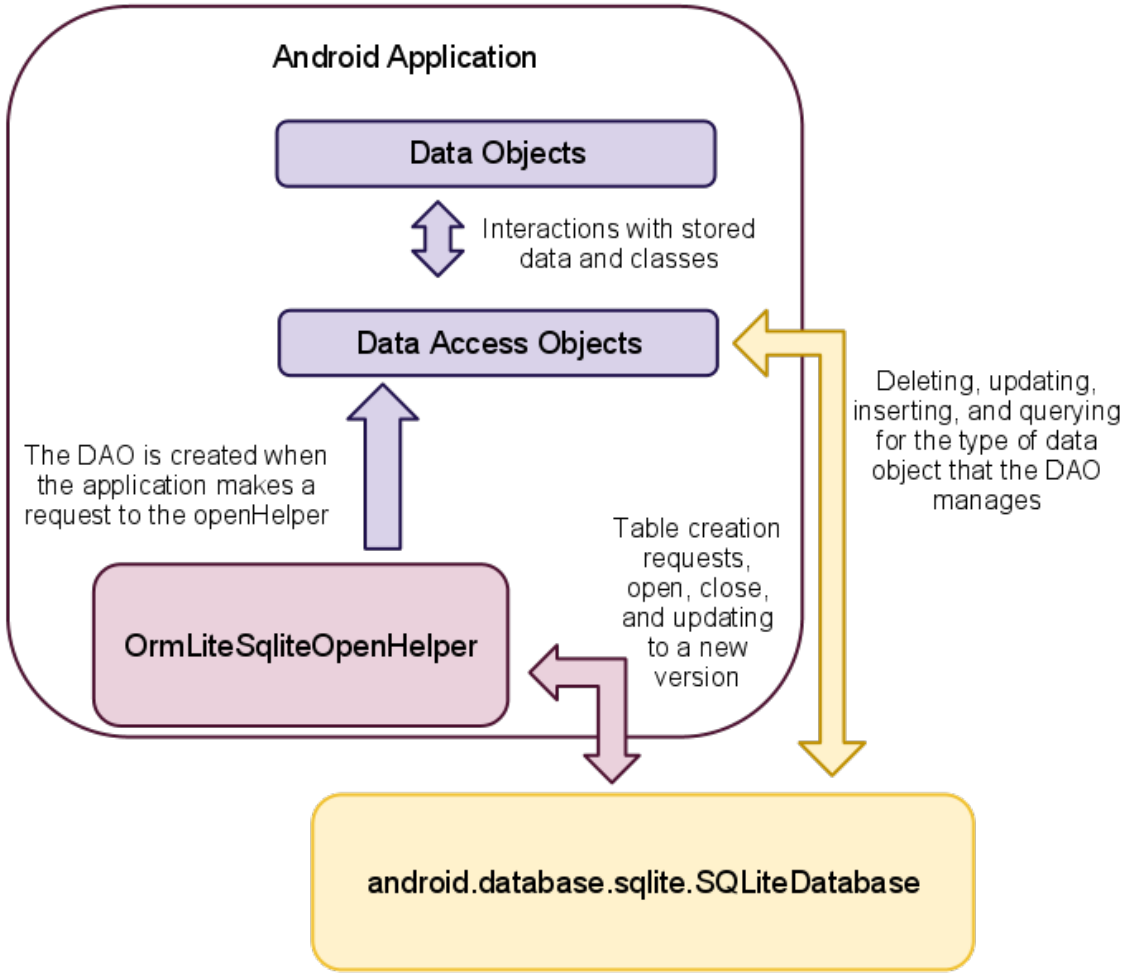
- Because it implements Iterator, you can use the iterator design pattern to loop over all the entries in a table
- Used for accessing and storing info on the db with some of these methods:
  - create(T data)
  - delete(T data)
  - queryForAll()
  - Refresh(T data)

What does it look like with out ORMLite?

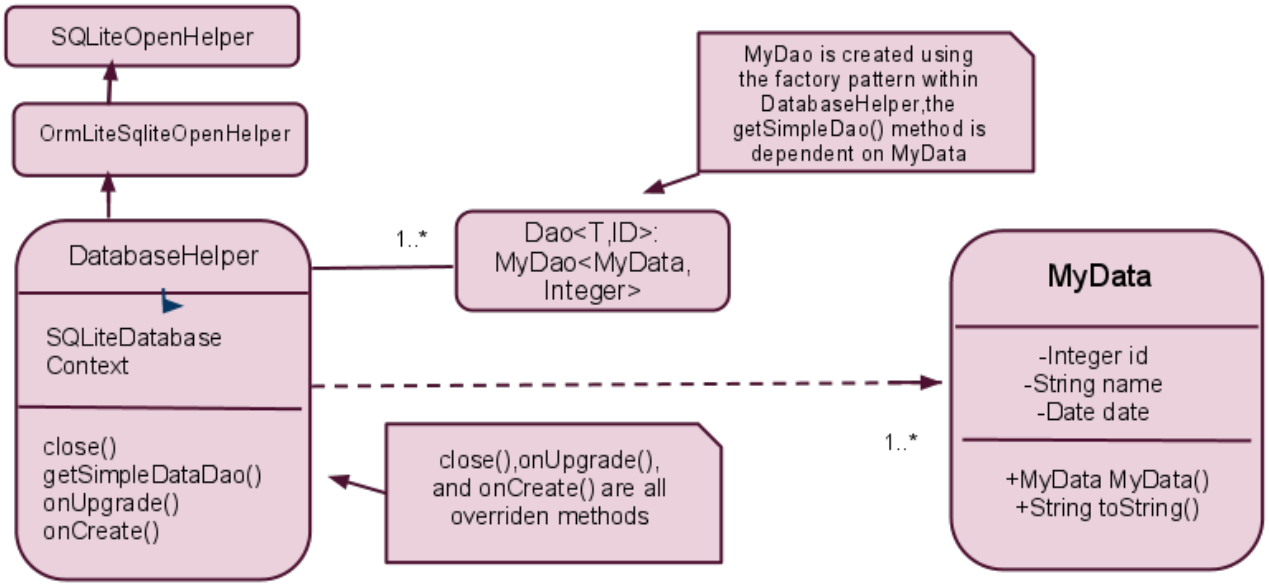


ORMLite: Main Components

Overview



ORMLite: UML Diagram



ORMLite Mechanics: DatabaseField[5]

```
// id is generated by the database and set on the object automagically
@DatabaseField(generatedId = true)
int id;
@DatabaseField(index = true)
String string;
@DatabaseField
long millis;
@DatabaseField
Date date;
@DatabaseField
boolean even;

SimpleData() {
    // needed by ormlite
}
```

Real Life Motivation

- If an Android project is using a database for anything more then the need of persistent storage (Android Cache files are not guaranteed to be persistent), then ORDM is likely to be a good choice for implementing the database
- ORMLite with Android support is a very new API (weeks old)
- ICUPDb an example of a database intensive application that was built before ORMLite was released

Example: Creating a Database in Android without ORDM[7]

```
/**
 * Table creation statements
 */
private static final String TABLE_CREATE_BUSINESS =
    "create table business (_id integer primary key autoincrement, "
    + "name text not null, longitude text not null, latitude text not null,"
    + " category text not null, dto_string string not null, id text not null);";

private static final String TABLE_CREATE_UPDATES =
    "create table updates (_id integer primary key autoincrement, "
    + "business_update_time text not null, events_update_time text not null,"
    + " featured_update_time text not null);";


private static final String TABLE_CREATE_EVENTS =
    "create table events (_id integer primary key autoincrement, "
    + "business_name text not null, name text not null,"
    + "start_date bigint not null, end_date bigint not null,"
    + "longitude text not null, latitude text not null,"
    + " category text not null, dto_string text not null, id text not null);";

private static final String TABLE_CREATE_FEATURED =
    "create table featured (_id integer primary key autoincrement, access_date text not null,"
    + "dto_string text not null);";

private static final String TABLE_CREATE_FAVBUSINESS =
    "create table favbusiness (_id integer primary key autoincrement, dto_string text not null,"
    + "id text not null);";

private static final String TABLE_CREATE_FAVEVENTS =
    "create table favevents (_id integer primary key autoincrement,"
    + "dto_string text not null, id text not null);";
/**
 * Database creation sql statement
 */
private static final String DATABASE_CREATE =
    TABLE_CREATE_EVENTS +";"+ TABLE_CREATE_BUSINESS + ";" + TABLE_CREATE_FEATURED + ";" + TABLE_CREATE_FAVBUSINESS + ";"
    + TABLE_CREATE_FAVEVENTS + ";"+ TABLE_CREATE_UPDATES + ";" ;
```

Example: Creating a Database in Android without ORDM (cont.)[7]

```
private static final String DATABASE_NAME = "data";
private static final String TABLE_EVENTS = "events";
private static final String TABLE_FEATURED = "featured";
private static final String TABLE_BUSINESS = "business";
private static final String TABLE_UPDATES = "updates";
private static final String TABLE_FAVBUSINESS = "favBusiness";
private static final String TABLE_FAVEVENTS = "favEvents";
private static final int DATABASE_VERSION = 95;
```

Example: Creating a Database in Android with ORDM[5]

```
// name of the database file
private static final String DATABASE_NAME = "helloAndroid.db";

private static final int DATABASE_VERSION = 6;
```

Example: Query a Database in Android without ORDM

```
/**
 * Gets the JSON String from the database for the EventDTO based on the ID
 * @param OrganizationDTO
 * @return JSONString
 * @throws SQLException
 */
public String fetchFavEventJSON(EventDTO e) throws SQLException {

    Cursor mCursor =
        mDb.query(true, TABLE_EVENTS, new String[] {KEY_JSON_DTO},  KEY_ID +"=" + String.valueOf(e.getId()) , null,
                null, null, null, null);
    if (mCursor != null && mCursor.moveToFirst()) {
        String JSON = mCursor.getString(mCursor.getColumnIndex(KEY_JSON_DTO));
        mCursor.close();
        mCursor.deactivate();
        return JSON;
    }
    mCursor.close();
    mCursor.deactivate();
    return null;
}
```

Example: Query a Database in Android with ORDM[5]

```
// get our dao
Dao<SimpleData, Integer> simpleDao = getHelper().getSimpleDataDao();
// query for all of the data objects in the database
List<SimpleData> list = simpleDao.queryForAll();
```

Example: Insert into a Database in Android without ORDM

```
/**
 * Create a new event using the title and body provided. If the event is
 * successfully created return the new rowId for that note, otherwise return
 * a -1 to indicate failure.
 * @return rowId or -1 if failed
 */
public long createEvent(EventDTO e, String JSONString) throws SQLException {
    Long rowID = rowExists(TABLE_EVENTS, e);
    if(rowID == -1){
        ContentValues initialValues = new ContentValues();
        if(e.getCategories()!=null){
            //initialValues.put(KEY_CATEGORY, e.getCategories());
        }

        initialValues.put(KEY_CATEGORY, "\"" + e.getCategories().toString().replace('[',' ').replace(']',' ')+"\"");
        initialValues.put(KEY_BUSINESS_NAME, e.getOrganizationName());
        initialValues.put(KEY_NAME, e.getName());
        initialValues.put(KEY_LON, String.valueOf(e.getLon()));
        initialValues.put(KEY_LAT, String.valueOf(e.getLat()));
        initialValues.put(KEY_START_DATE, e.getStartDatetime().getTime());
        initialValues.put(KEY_END_DATE, e.getStopDatetime().getTime());
        //Note: need to escape all \n and \r otherwise JSONMapping FAILS
        initialValues.put(KEY_JSON_DTO, JSONString.replace("\n", "\\n").replace("\r", "\\r"));
        initialValues.put(KEY_ID, String.valueOf(e.getId()));
        //Log.w(TAG, "CREATED NEW ROW FOR TABLE EVENTS");
        rowID = mDb.insert(TABLE_EVENTS, null, initialValues);
        //Log.i(TAG, String.valueOf(rowID));
    }
    return rowID;
```

Example: Insert into a Database in Android with ORDM

```
// create a new simple object
long millis = System.currentTimeMillis();
SimpleData simple = new SimpleData(millis);
// store it in the database
int ret = simpleDao.create(simple);
```

ORDM and Android Specific Trade Offs

- Your Activity classes will need to be either from ORMLite or extend them, this can create some complex inheritance structures
- The code will be easier to maintain and less tedious to work with at any point in development
- For application development in particular bringing in another library may not be an option if size is a big constraint of the application

References

- [1] *Wikipedia* retrieved March 28, 2011 from
  - http://en.wikipedia.org/wiki/Object-relational_mapping
  - http://en.wikipedia.org/wiki/ORDBMS
  - http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
- [2] *SQLite* retrieved March 28, 2011 from
  - http://www.sqlite.org/datatype3.html
- [3] Ramakanth Devarakonda, Object-Relational Database Systems – The Road Ahead, Crossroads 2001
- [4] Praveen Seshadre, Enhanced Abstract Data Types in Object-Relational Databases, Springer- Verlag New York Inc. 1998
- [5] *ORMLite* Retrieved March 30, 2011 from
  - http://ormlite.com/javadoc/ormlite-android/
  - http://ormlite.com/javadoc/ormlite-core/
  - http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_7.html#SEC54
  - http://ormlite.com/android/examples/
- [6] Ming Wang, Implementation of Object-Relational DBMSs in a Relational Database Course, SIGCSE 2001
- [7] ICUP 2011 Senior Project, Team Members: Alex Boughton, Kim Nguyen, Jon Tsui, Matthew Ripley, and Zainab Jarourdi