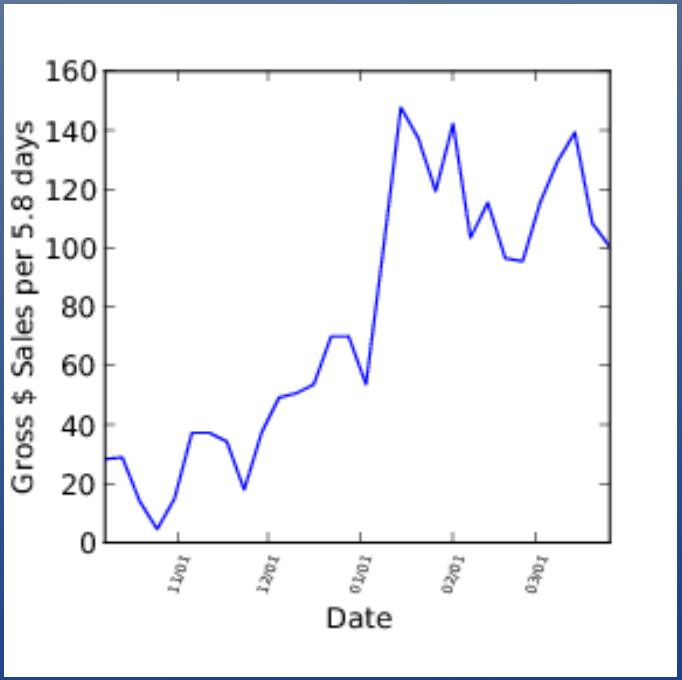
DevRev:
Developer Revenue Analysis

Karl Ostmo
Nov 1, 2010

Revenue Tracking -- First Attempt (2/2)

- Wrote Python script to split requested date range into 30-day chunks
- Parse CSV files, aggregate in histogram
- **Problem:** Records don't distinguish between apps



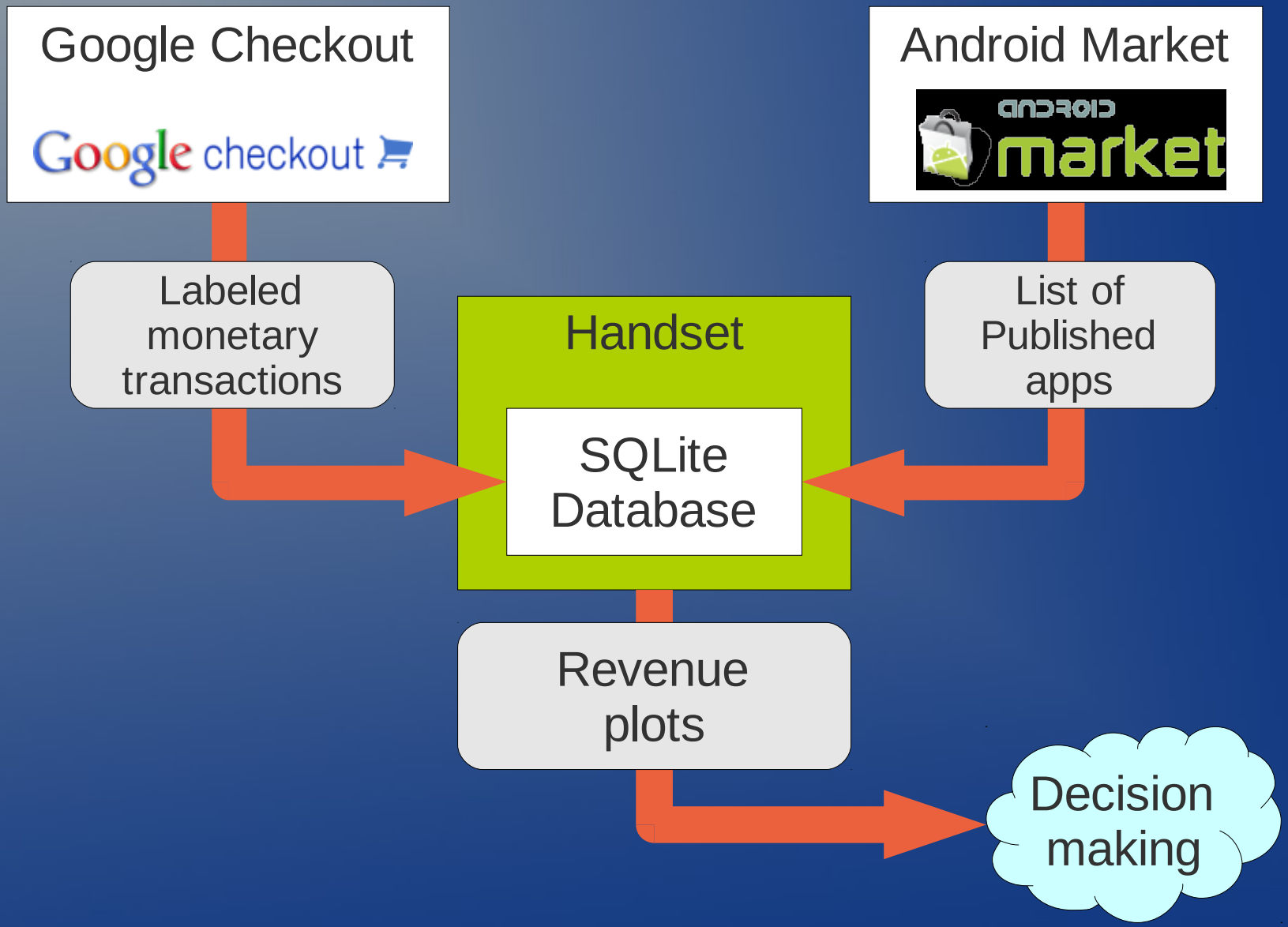
Combined sales, Oct – Mar

- Uses MatPlotLib for plotting
- Adjustable bin count
- Released online: http://www.anddev.org/general-f3/howto-graph-plot-android-market-sales-from-google-checkout-t9442.html

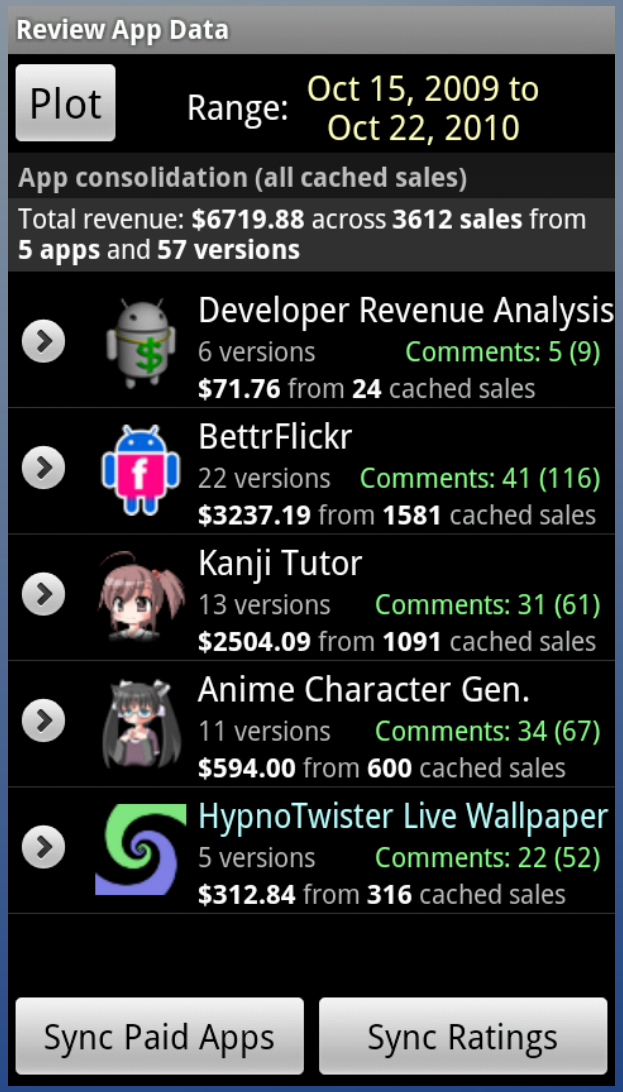
Revenue Tracking -- A Better Solution

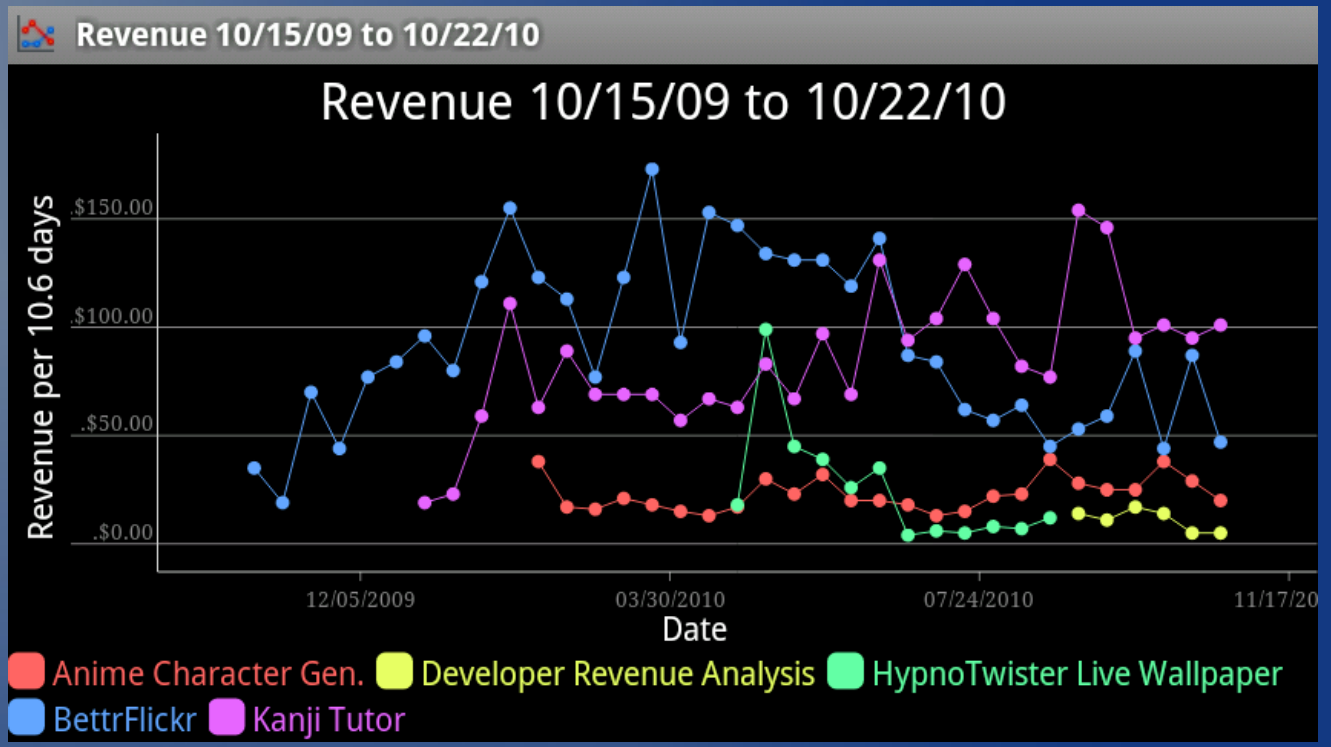
- Use Google Checkout's "Notifications" API to obtain more complete data
  - XML web service (well-documented)
- Combine with Android Market API
  - Unpublished API, but reverse engineered (independent open source project)
  - Gets a list of a publisher's current apps (with icons)
- Productize as a new Android app!

System Overview



Preview of Final Product





- Revenue summaries by App
- Time plots

Challenges

- Obtain data
- Design schema
- Platform bug
- Improve speed

Challenges

- **Obtain data**
- Design schema
- Platform bug
- Improve speed

Google Checkout "Notifications" API
What data is available?

- A "Full" record (a.k.a. <new-order-notification>)
  - Created for every new order placed
  - Large (~4KB)
    - Order number
    - Product ID
    - Many extras: Billing address, product description, etc.
- A "Charged" record (a.k.a. <charge-amount-notification>)
  - Created for every purchase unreturned after 24 hours
  - Minimal data (< 0.5KB per record)
    - Order number
    - Charge amount
    - Timestamp

Google Checkout "Notifications" API
Constraints

- A "full" record has every field that a "charged" record does, but has no way to discriminate between *returns* and *completed orders*.
  - Makes both types of query necessary
- Because of returns, there will be fewer "charged" records than "full" records
- Can **query** for records by **date range** or **order number**

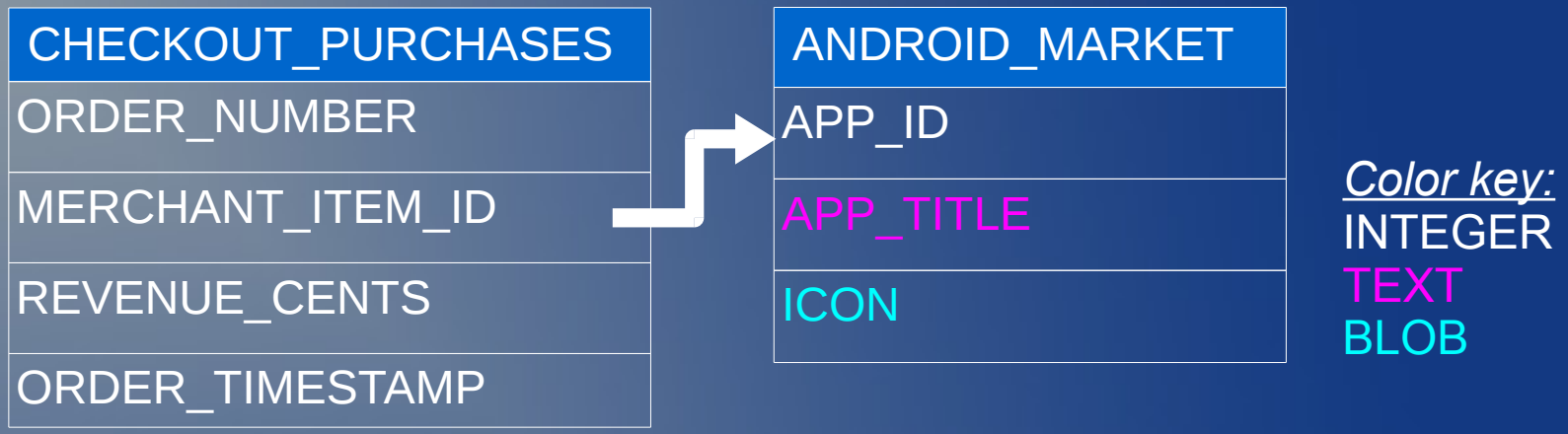
Google Checkout "Notifications" API
Strategy

- First query "charged" records by date
  - Obtains order numbers for all "kept" purchases
  - API gives 50 per page, with "next page" token
- Next query "full" records by order number
  - May specify 16 order numbers at a time
- Implement a SAX parser, insert records into DB

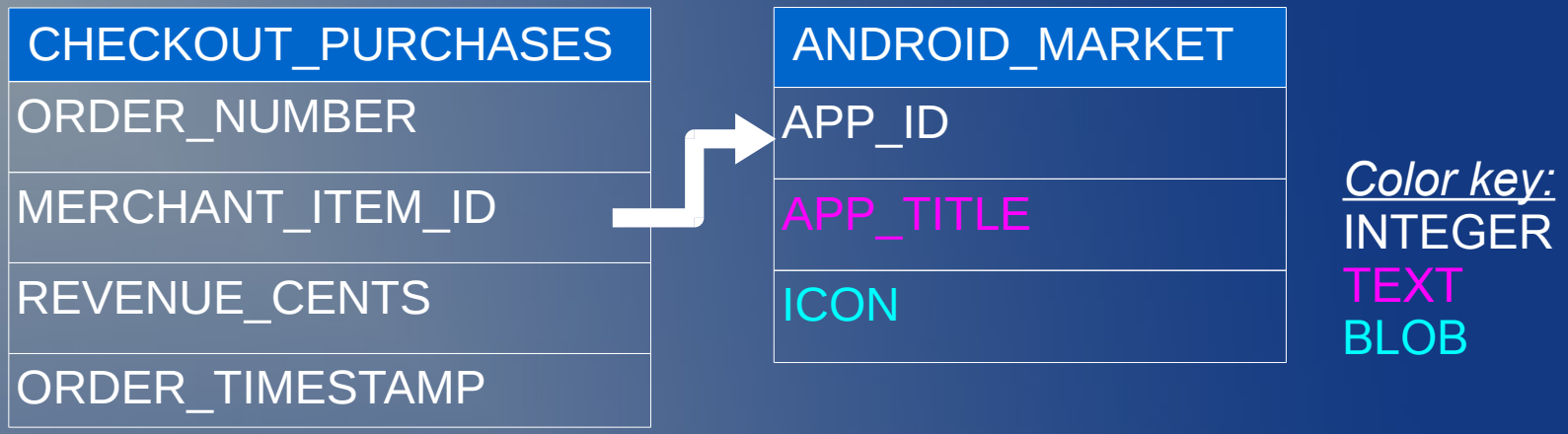
Challenges

- Obtain data
- **Design schema**
- Platform bug
- Improve speed

Naïve Database Schema

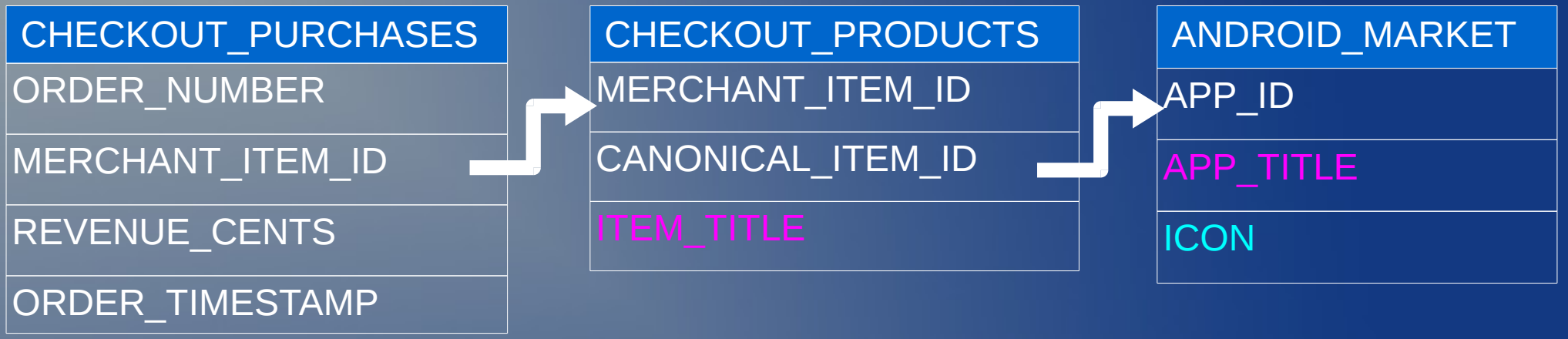


Naïve Database Schema



- **Problem:** New app versions (updates from the developer) are treated as distinct products in Google Checkout (they get a new **`MERCHANT_ITEM_ID`**)
  - The most recent app version has a **`MERCHANT_ITEM_ID`** that matches the **`APP_ID`**
  - All other app versions have unique **`MERCHANT_ITEM_ID`**s
  - Many-to-one mapping between **`MERCHANT_ITEM_ID`** and **`APP_ID`**

Improved Database Schema



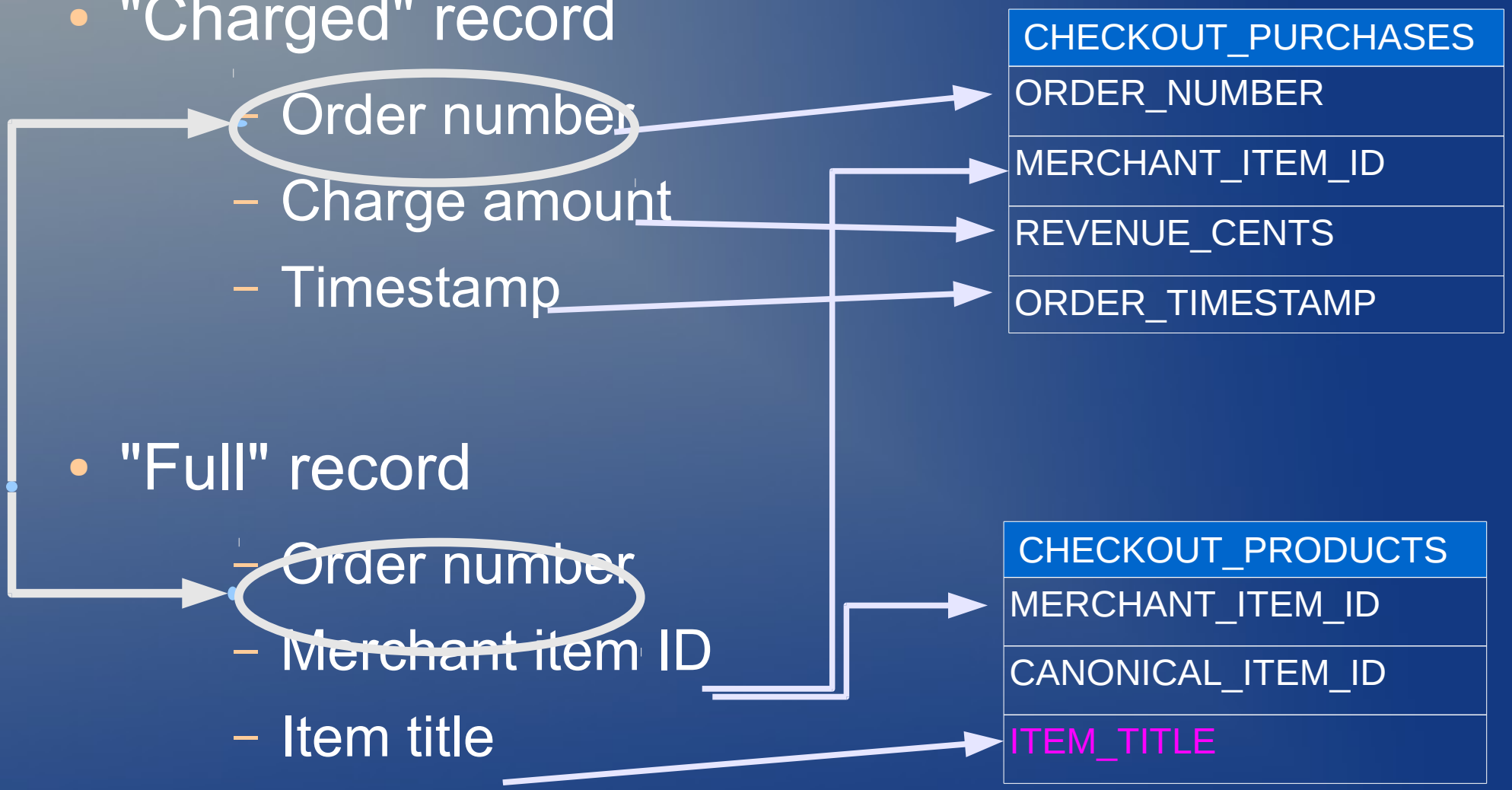
- **CHECKOUT_PURCHASES** is **JOIN**ed with **CHECKOUT_PRODUCTS** in a **VIEW**
- **ITEM_TITLE** is compared to **APP_TITLE** and used to automatically set **CANONICAL_ITEM_ID** to the appropriate **APP_ID**

Populating Database (1/2)

- "Charged" record
  - Order number
  - Charge amount
  - Timestamp
- "Full" record
  - Order number
  - Merchant item ID
  - Item title

Populating Database (2/2)

- "Charged" record
  - Order number
  - Charge amount
  - Timestamp
- "Full" record
  - Order number
  - Merchant item ID
  - Item title



Challenges

- Obtain data
- Design schema
- **Platform bug**
- Improve speed

Platform Bug

- Complete data loss in SQLite
  - Sporratic but reproducible by inserting thousands of records
    - Was I using the database incorrectly?
  - Tracked to a bug in the version of SQLite used by the Android platform
    - Browsed source code (http://source.android.com/)
    - SQLite mailing list archive
  - Remedied in Android 2.2
    - Postponed release and set minimum platform version requirement

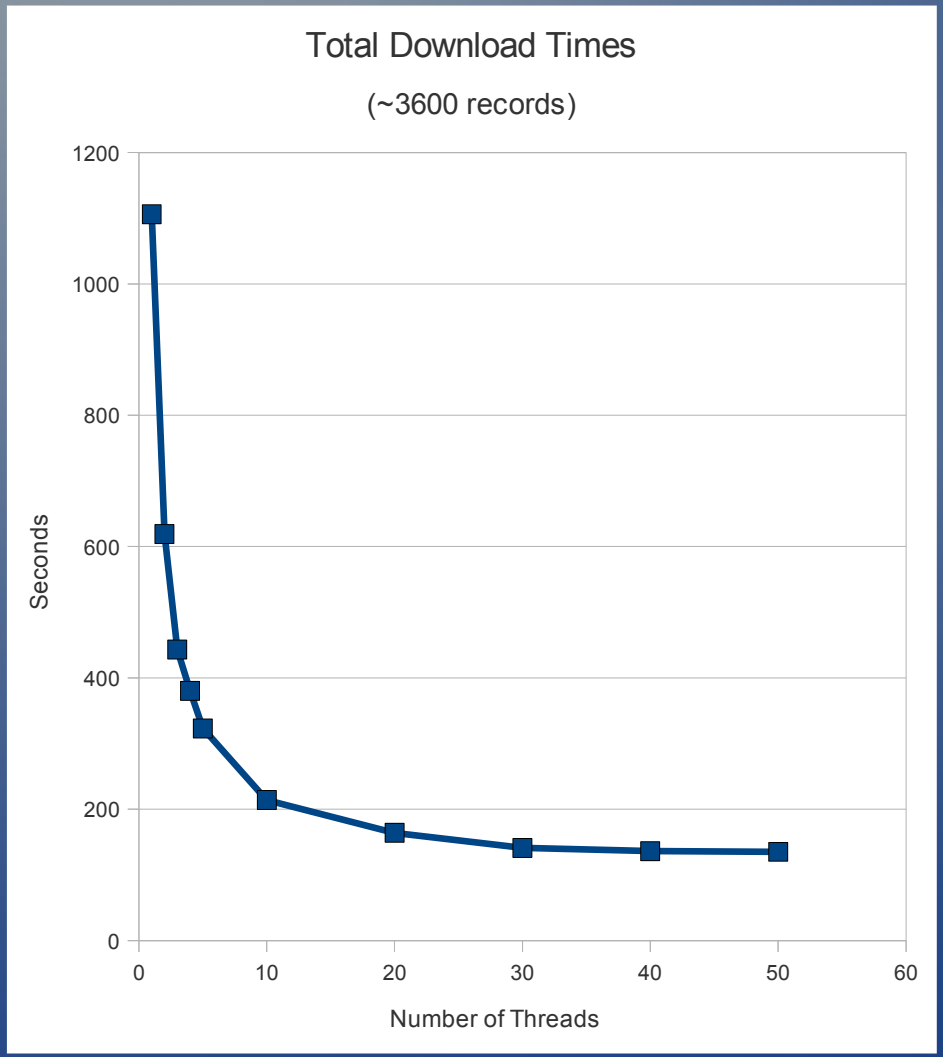
Challenges

- Obtain data
- Design schema
- Platform bug
- **Improve speed**

Reducing Download Times (1/2)

- Problem: Unsatisfactory download times
  - Almost 20 minutes for me (around 3600 records)
  - What about other (more successful) sellers?
    - My target audience: 100K sales or fewer
    - At least, runs as a background process (start and forget, notified when complete)
- Solution: Multithreaded downloads
  - Server-bound, not bandwidth-bound
  - Required refactoring design from sequential to parallel

Reducing Download Times (2/2)

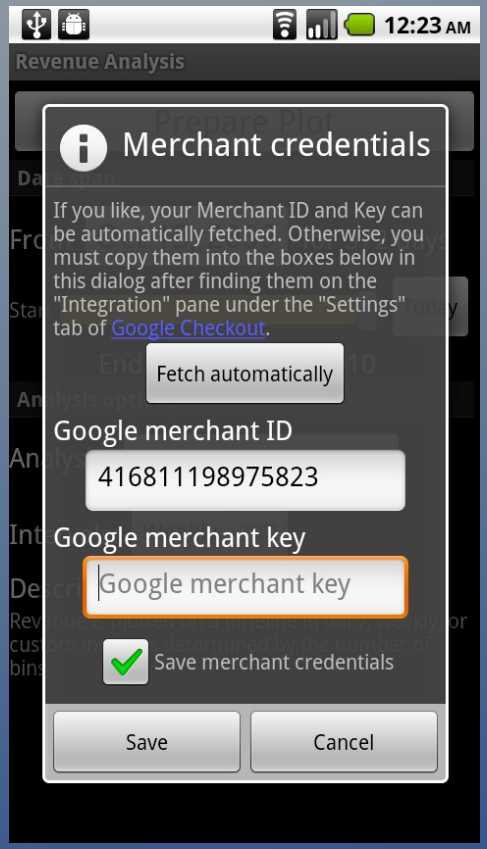


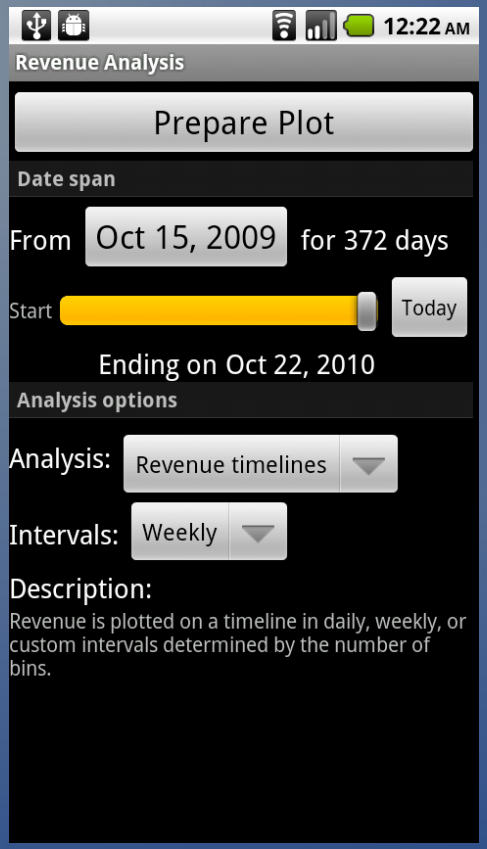
- Download takes 18 minutes without multithreading
- Adding a second thread nearly doubles speed
- No significant improvement beyond 30 threads (2:15)

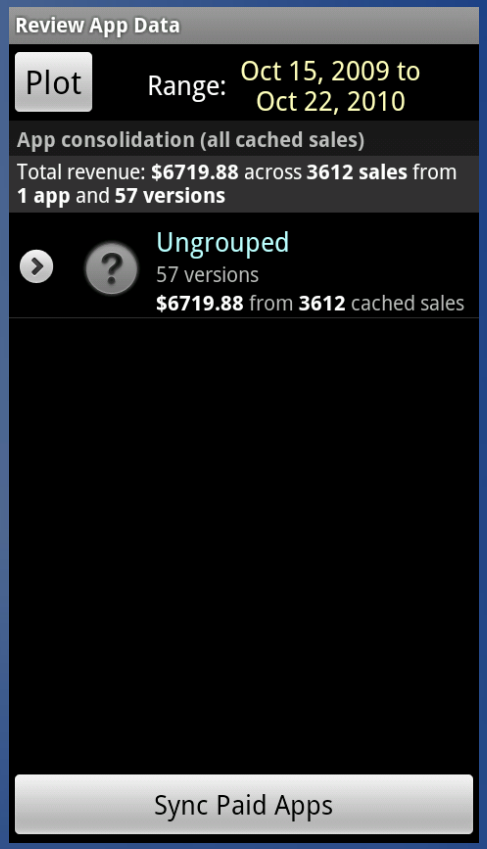
User Interaction Sequence (1/2)

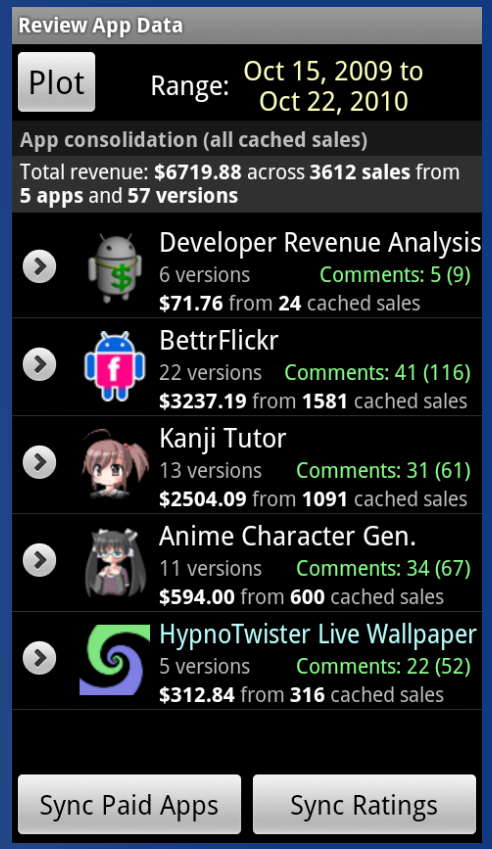
1. Obtain login credentials
2. Download sales within a certain date range
    - Cache sales in database (persistent storage)
    - Mark a date range as cached
        - Allow cache date ranges to be merged
3. Automatically/manually consolidate Google Checkout "products" as Android Apps
4. View/Export sales plots

User Interaction Sequence (2/2)







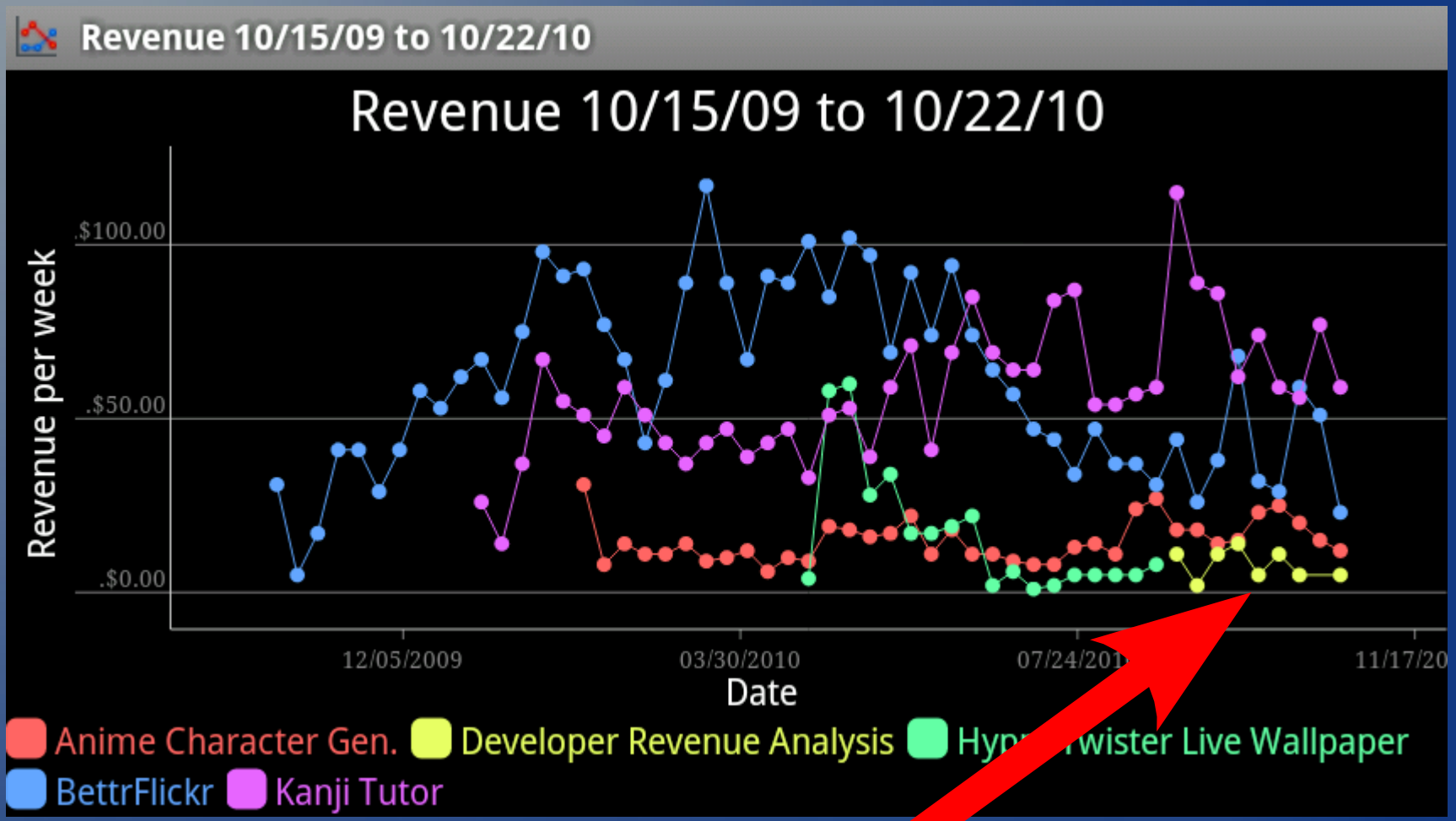


(1) Authenticate

(2) Specify date range (and analysis parameters)

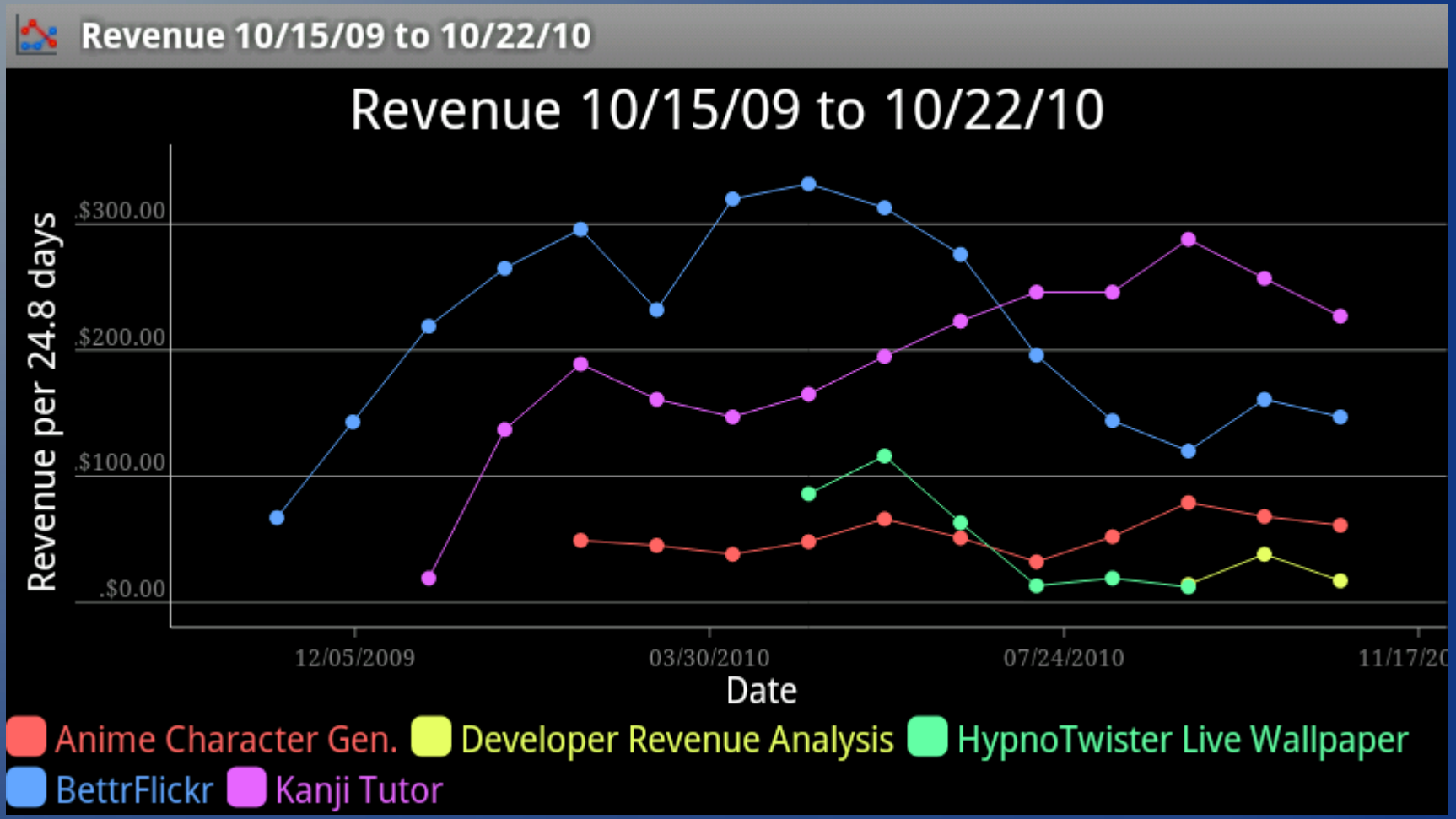
(3) Consolidate

Finished product

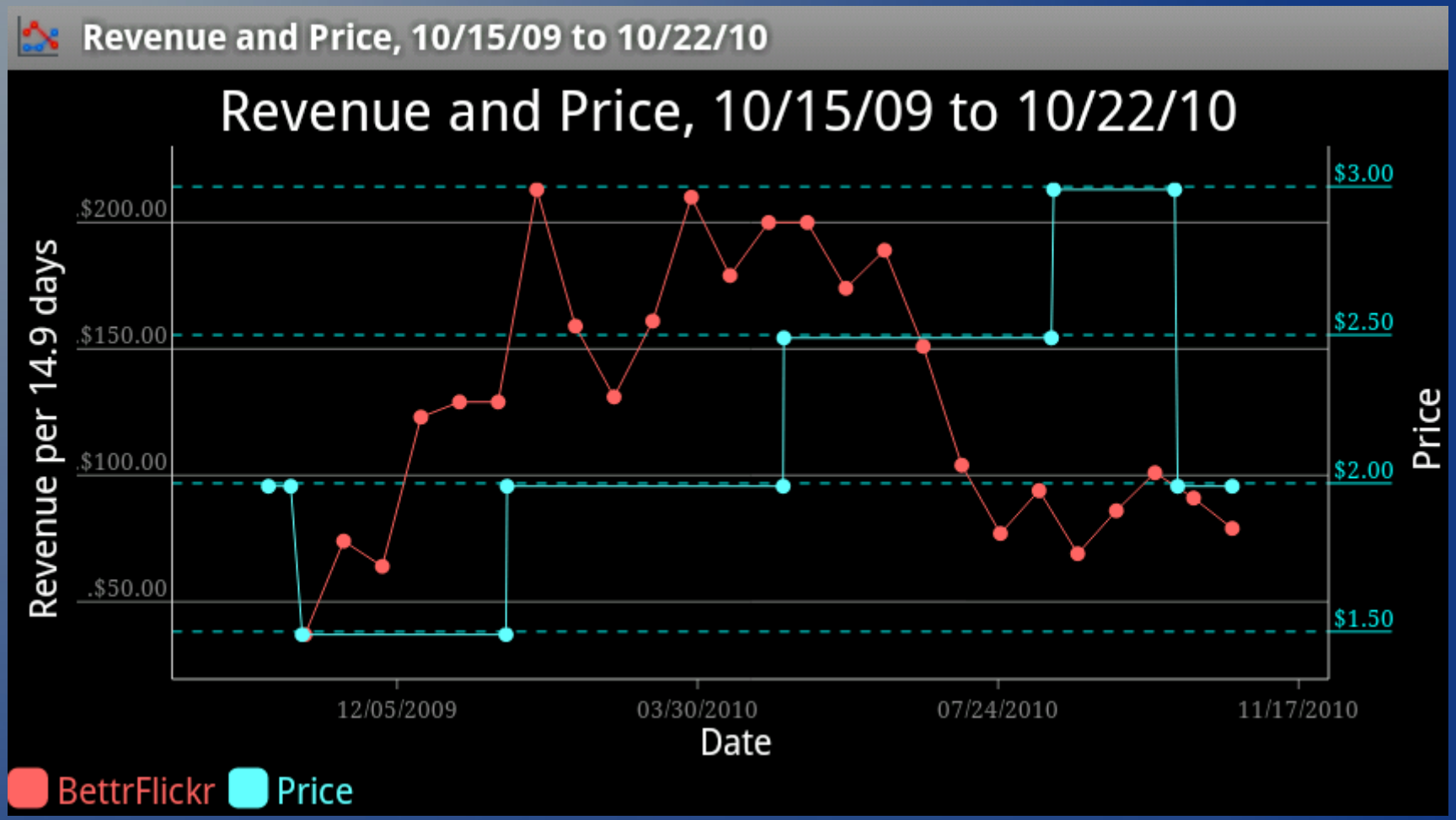


- App is self-hosting

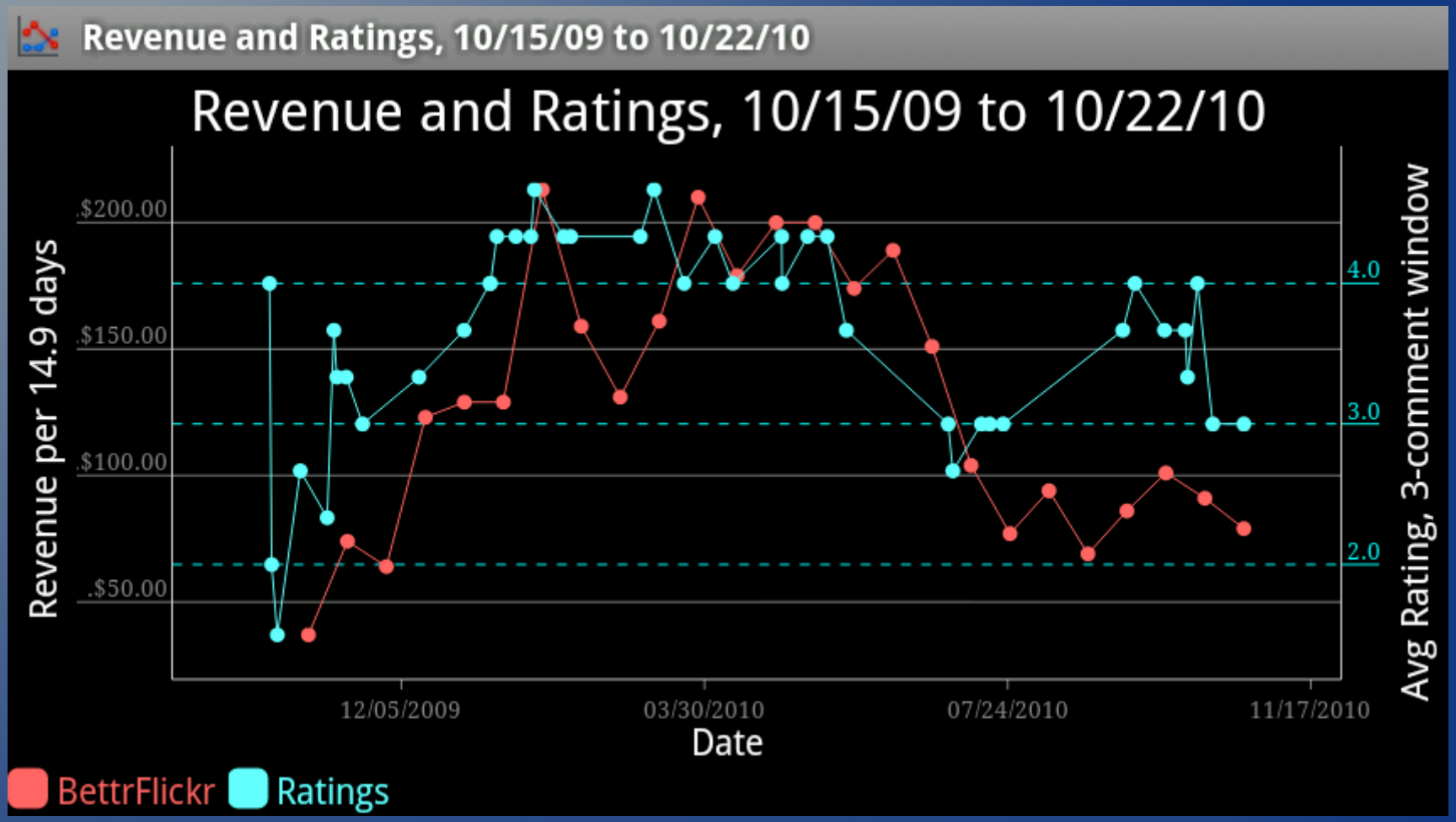
Smoothing with Larger Bin Widths



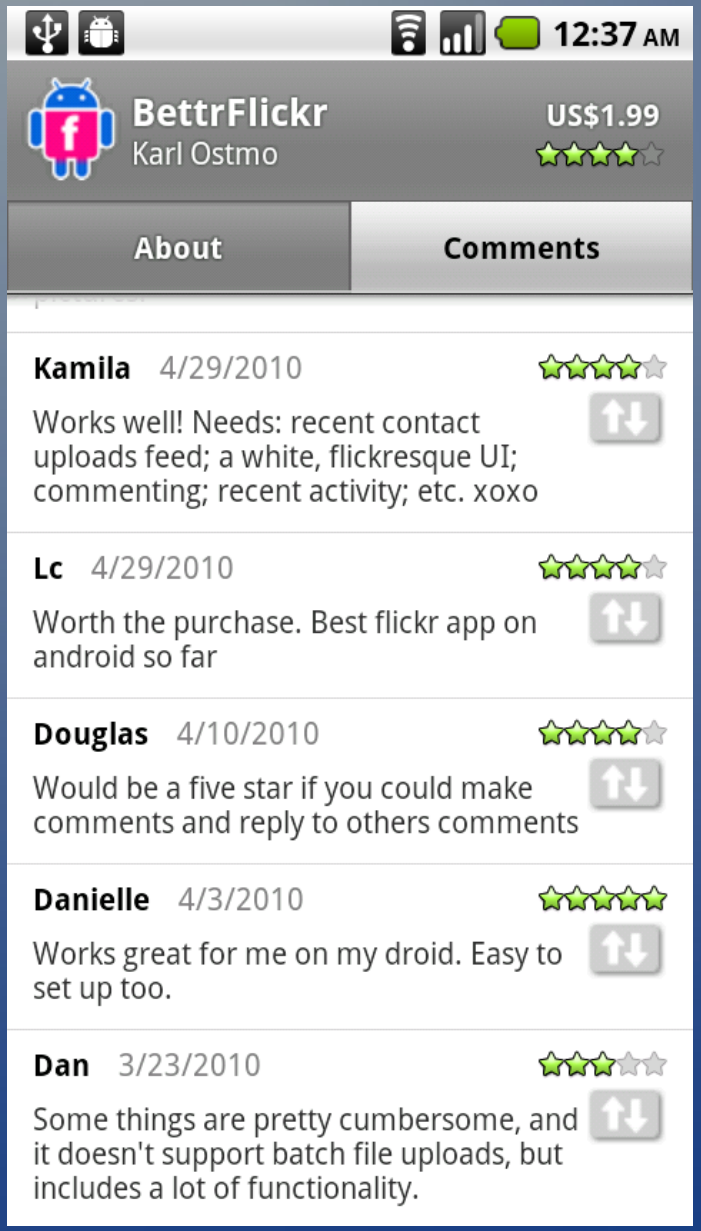
Cause of a Sales Trend? (1/2)



Cause of a Sales Trend? (2/2)



Conclusion



- User comments have an important effect on sales
- More data points needed to determine effect of price
  - Preferably on an app with stable ratings
- The two APIs were integrated successfully to produce revenue plots