

Data Integrity

Database vs Application

Background

- Need to Enforce Data Integrity
 - Typically Use Foreign Keys w/ Cascades
- Made Unit Testing Difficult
 - Fixture Loading w/ Dependencies
 - Massive Slowdown w/ Frequent Testing

Enforcing Integrity

- Database Layer
 - Use FK w/ Cascades or Triggers
- Application Layer
 - Use an ORM with Support
- Never-Ending Argument

Low-Level Concerns

- Speed
 - Database Tends to Be Bottleneck
 - Database May be Faster Than App
 - More Requests to Enforce From App
 - Deletes are Probably Rare
 - Updates are Less Important w/
Surrogates

Low-Level Concerns

- Atomicity
 - Transactions
 - DBMS w/ FK Support Probably Supports Transactions

High-Level Concerns

- Application Development
 - ORMs Available for Use
 - ActiveRecord (Rails)

High-Level Concerns

- Multiple Paths Into Data
 - Website
 - DBA Directly Managing
 - Desktop Management Application
 - DRY
 - FKs In Database
 - Provide Service Layer Between Database and Application(s)
 - Consolidate All Shared Logic

High-Level Concerns

- Distribution of Application Logic
 - Foreign Keys are Business Decisions
 - If I Delete This User Should I Lose All Their Photos?
 - What If People Have Linked To Those Photos? (etc.)
 - Other Business Decisions are in App Logic
 - Keep it All in One Place

Conclusion

- No Cut and Dry Rule
- Depends On Application Needs
- Save Yourself a Headache w/ Rails and Keep Logic in the Application

Link

- <http://www.agiledata.org/essays/referentialIntegrity.html>