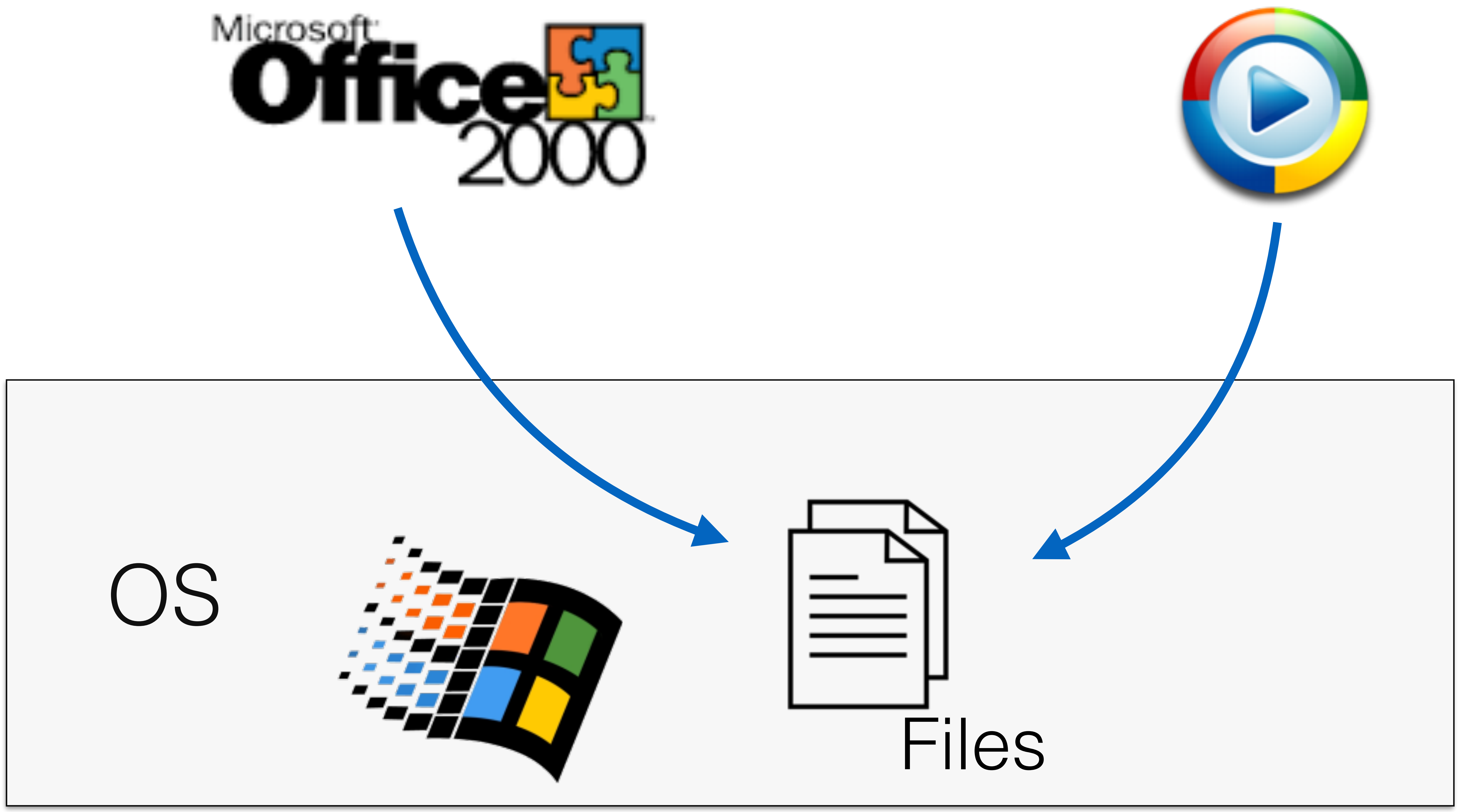
# **Earp:** Principled Storage, Sharing, and Protection for Mobile Apps

**Yuanzhong Xu**, Tyler Hunt, Youngjin Kwon, Martin Georgiev, Vitaly Shmatikov[†], Emmett Witchel

UT Austin, [†]Cornell Tech

Santa Clara, CA, 3/18/2016

# Desktop era

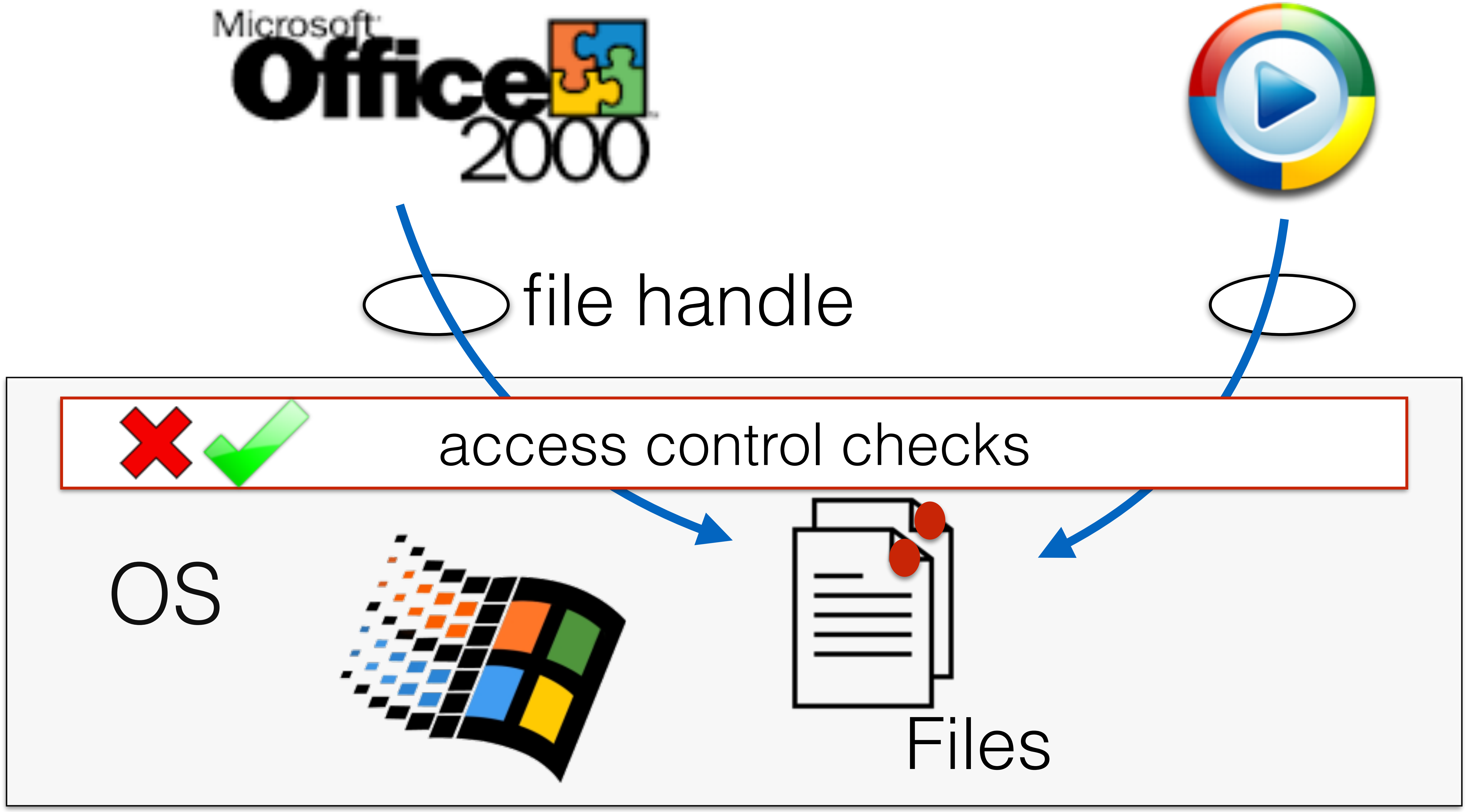


OS

Files

- Applications  mostly work individually
- They rely on the OS to store and exchange data, in the form of **files**

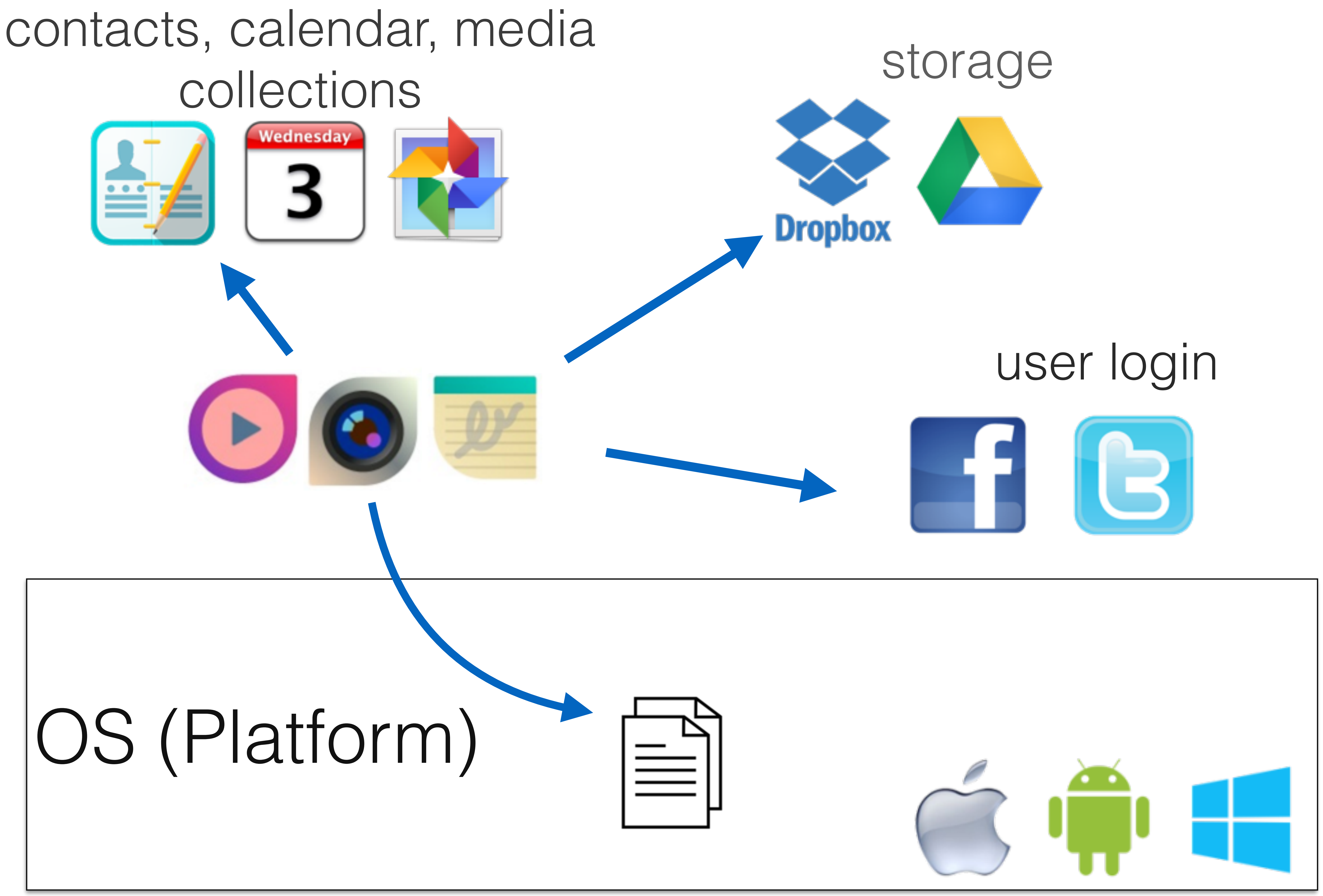
# Data protection in desktop era



OS protects data:

- File ownership and permissions
- App processes hold file handles (file descriptors)

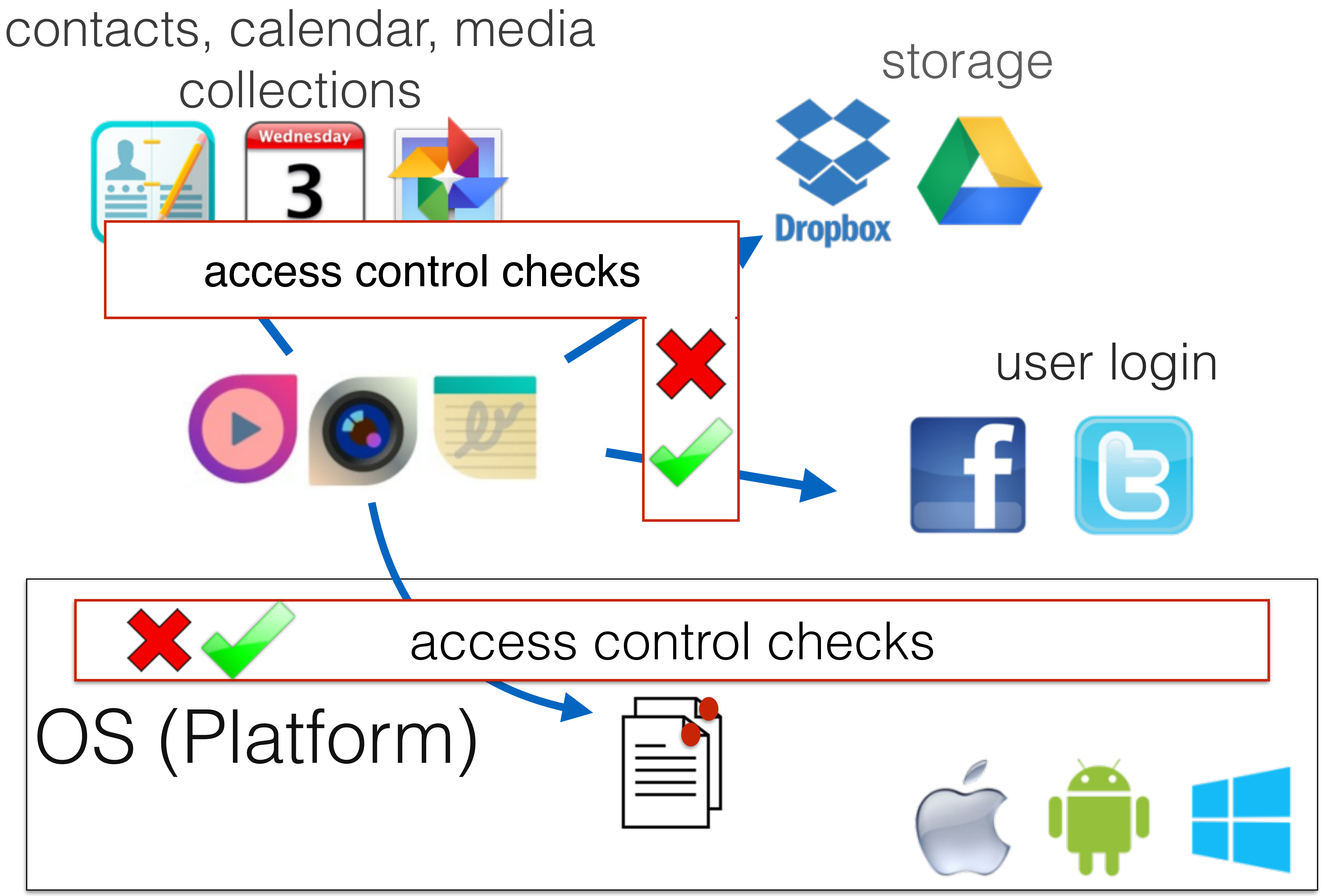
# Mobile era



- Apps interact with each other as much as with the platform — an app “ecosystem”

- “Hub” apps provide services to other apps

# Data protection in mobile platforms

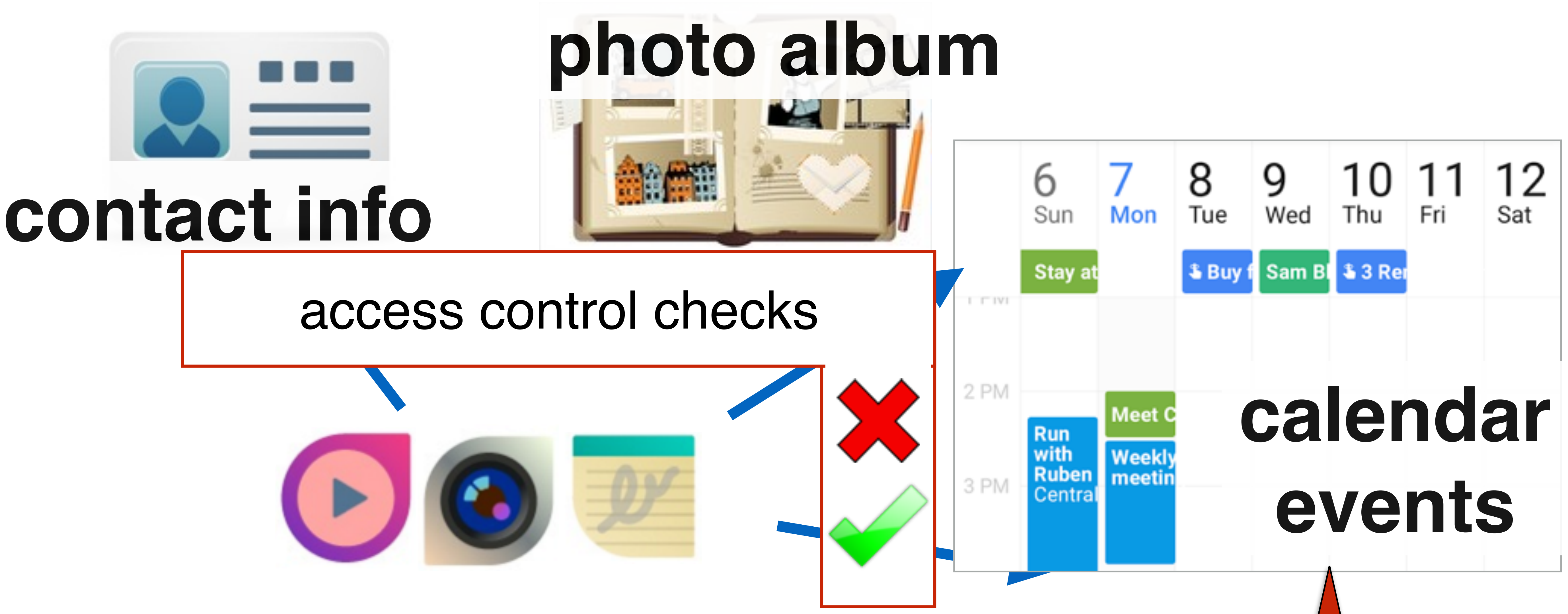


Check what apps have access to what data

- Apps check interactions

- Platform checks file access

# No principled solution for app-level checks

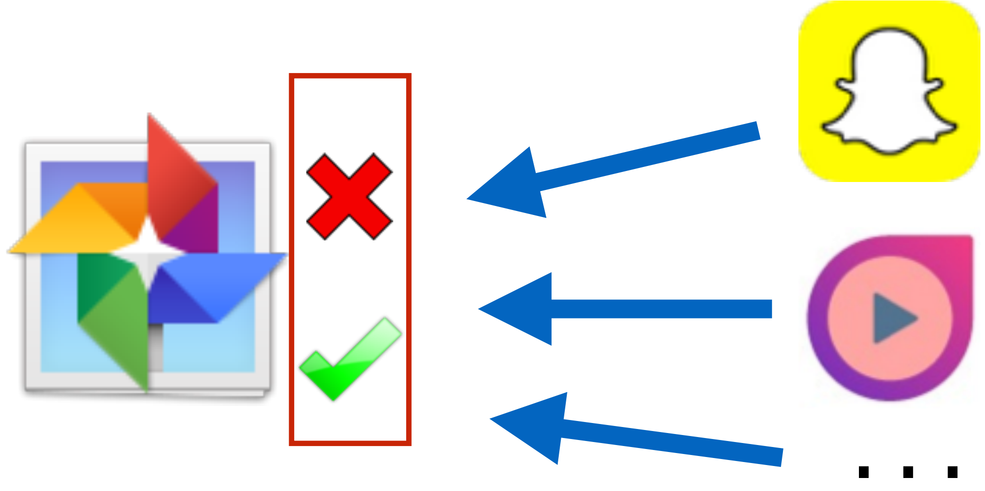


**Different high-level semantics: not just files!**

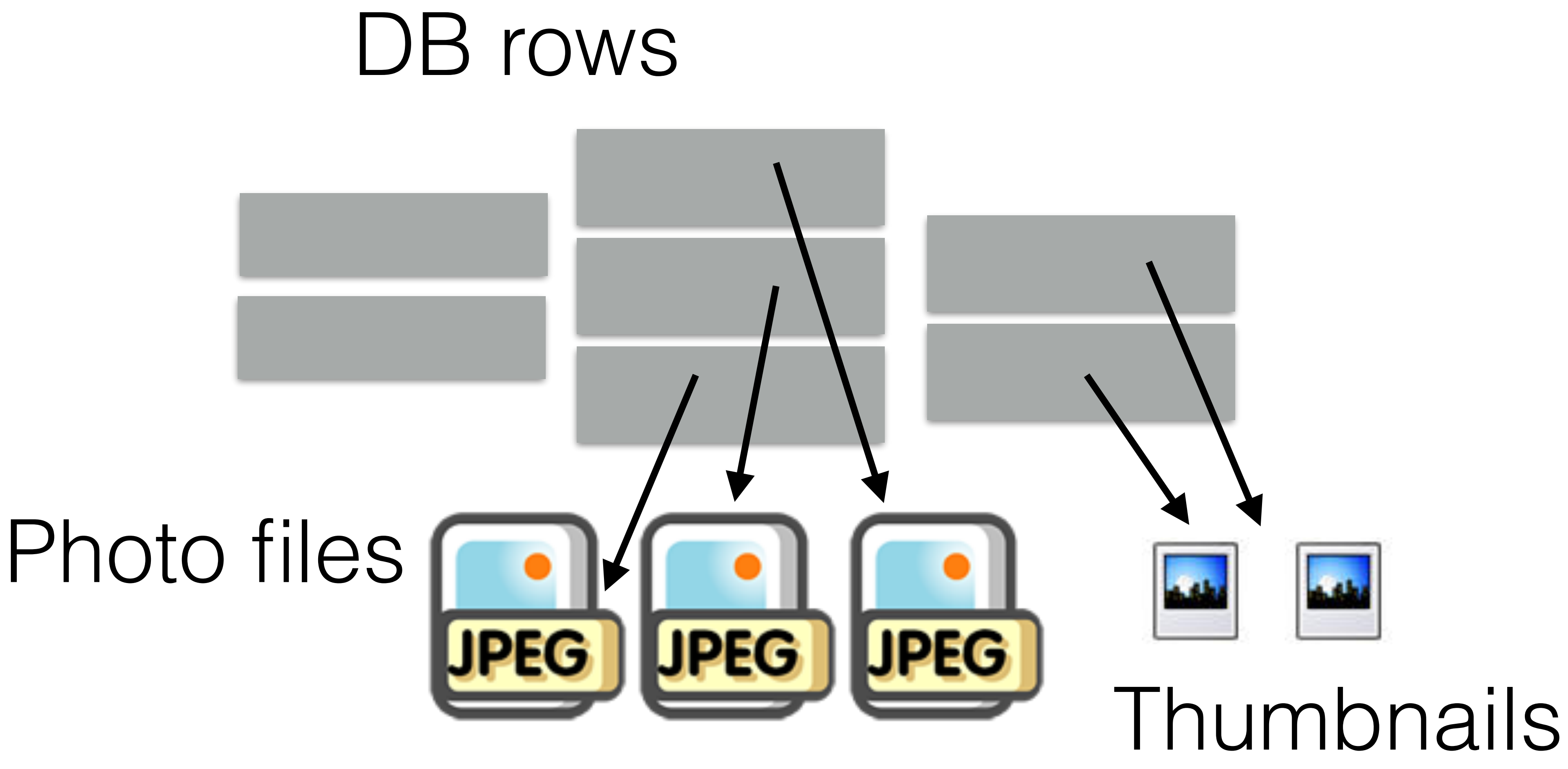
- Different **data models** — how data structures represent semantics
- Different protection requirements
- Developers have to write **ad hoc** checks

# How would a developer write ad hoc checks?

Example: implement a photo manager



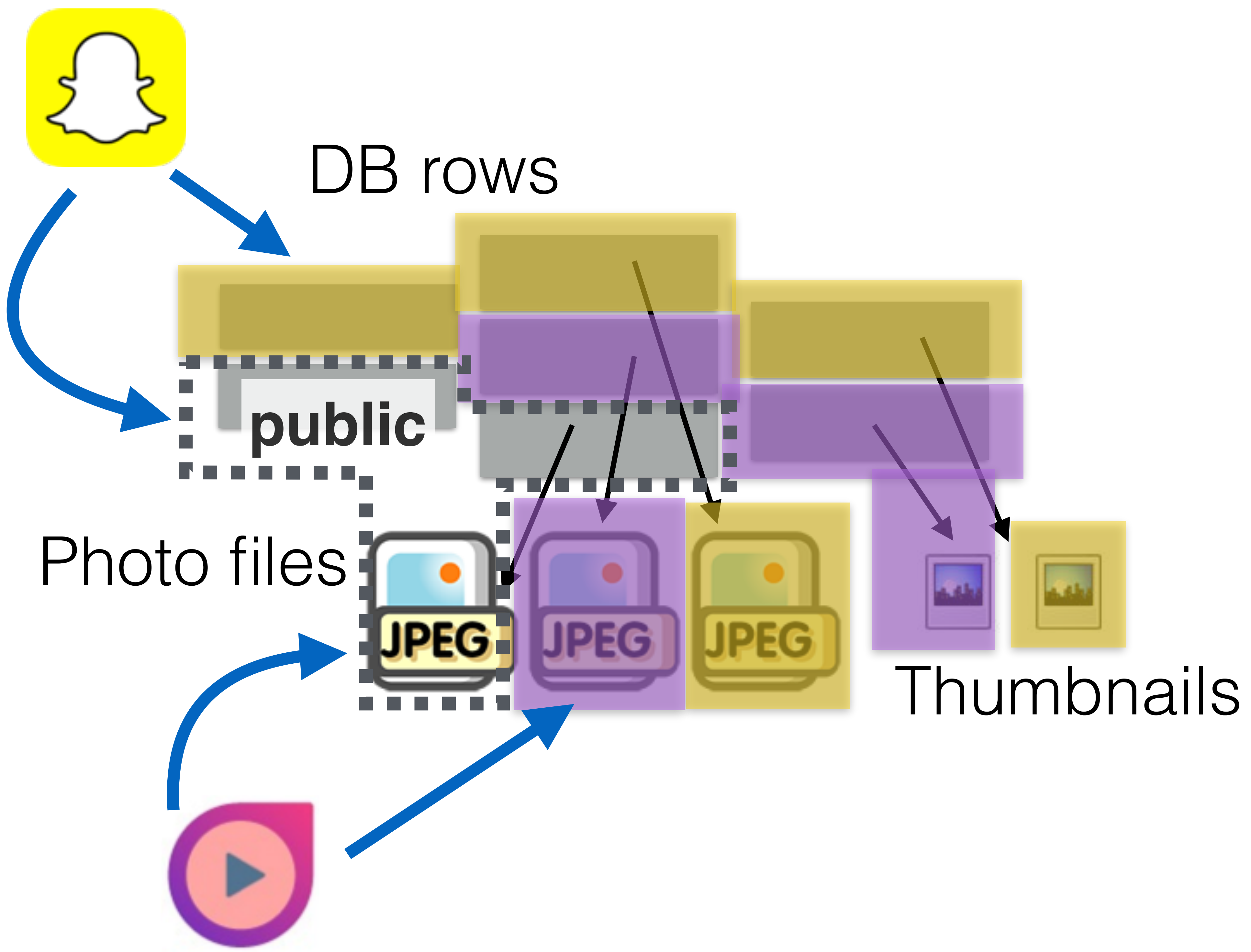
1. Design a data model
    - Organize photos with albums
    - Maintain metadata in database
    - Keep indexes to files



# How would a developer write ad hoc checks?

Example: implement a photo manager

2. Define protection requirements
   - Each app can have its own **private** photos and albums
   - Apps share some **public** photos and albums

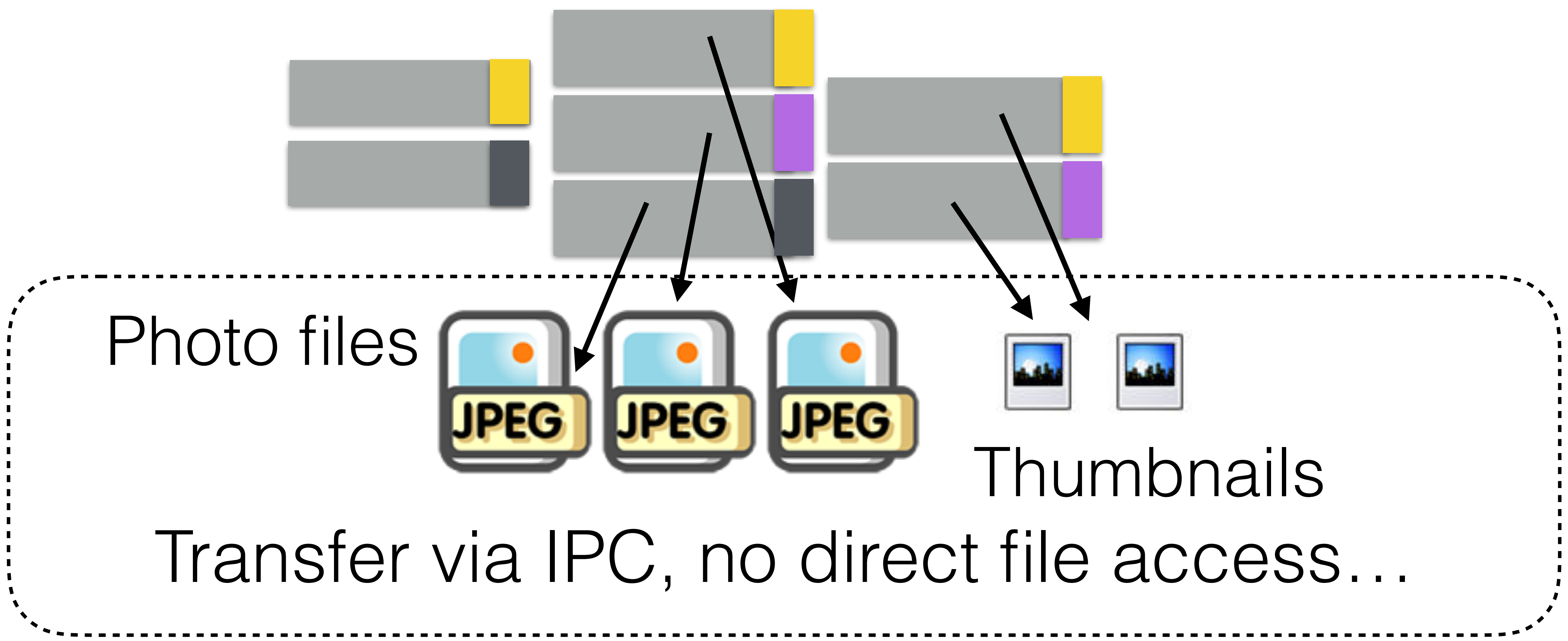


# How would a developer write ad hoc checks?

Example: implement a photo manager

**Problem: ad hoc checks are hard, error-prone**

3. Implement the protection
   - Implement fine-grained permissions — ACL columns in DB, append WHERE clauses in queries
   - Protect files — permission bits not enough for many apps
   - How to change permissions? What is the API?

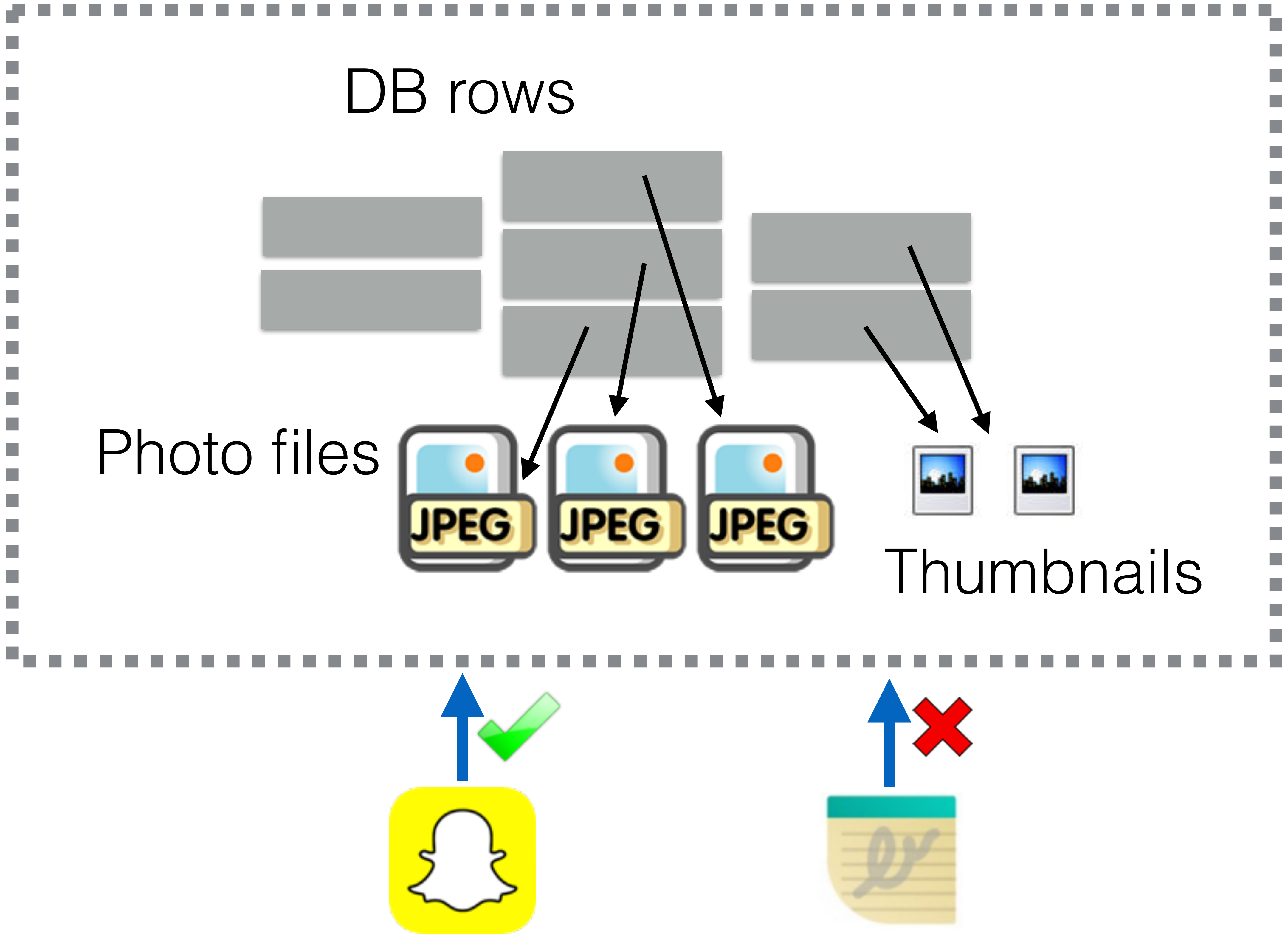


Transfer via IPC, no direct file access…

- What if we want a group of apps to access photos?
- How to hide location info about a photo?

# Reality: all-or-nothing “protection”

- Developers give up fine-grained protection…
- Let apps have access to either all or none of the photos!
- Violates the **principle of least privilege**



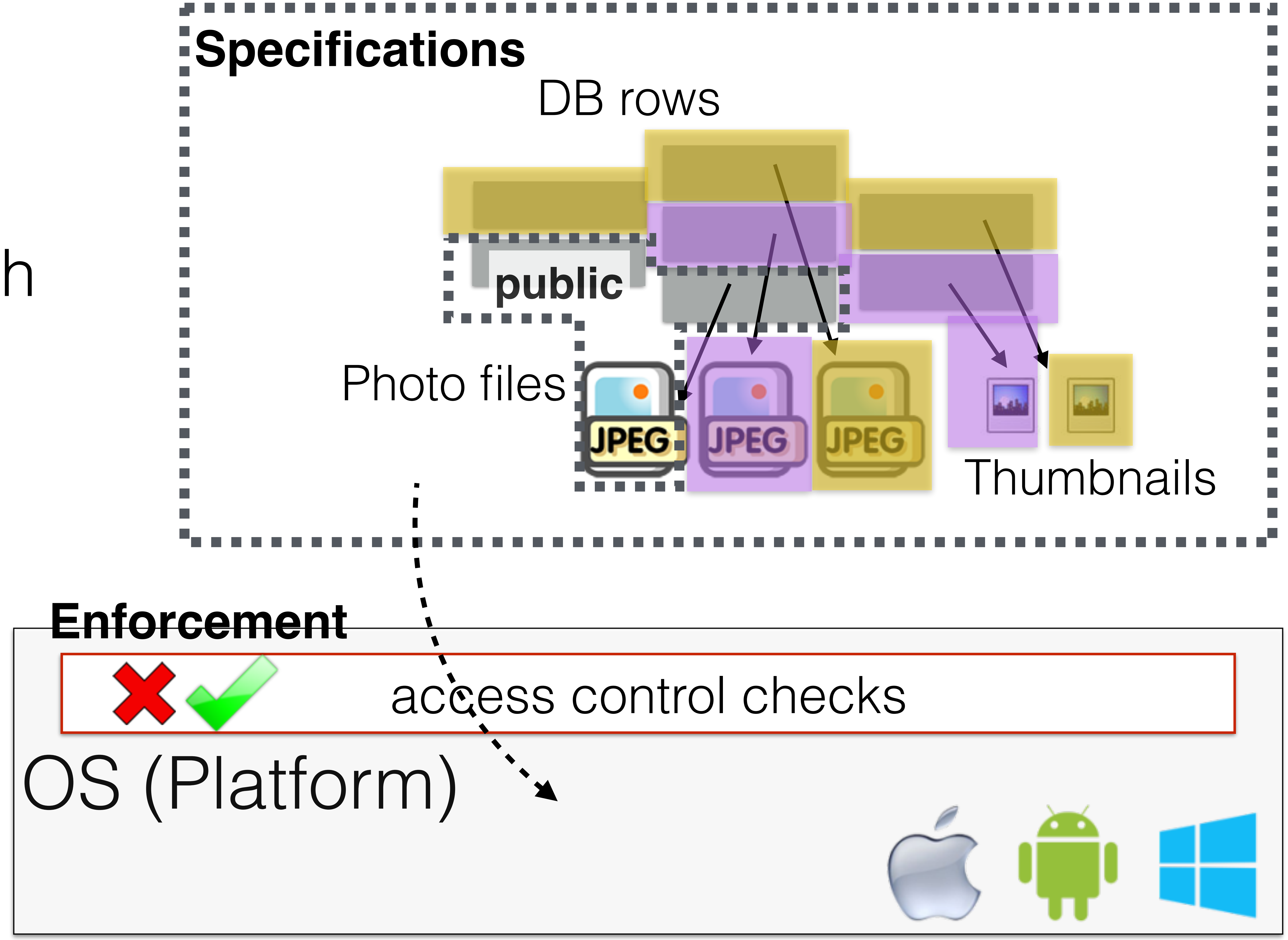
# Reality: apps have insufficient protection

- iOS: Snapchat automatically saves photos to shared gallery
- Android: Dropbox stores files in public external storage
- Firefox OS: email attachments copied to public SD card when opened
- Mistakes in network-based authentication protocols (OAuth):
  - Sun et al. CCS ’12, Viennot et al. SIGMETRICS ’14

# Ideally: separate specification from enforcement

- App **specifies** data model with protection requirement

- Platform **enforces** protection, no ad hoc checks in apps

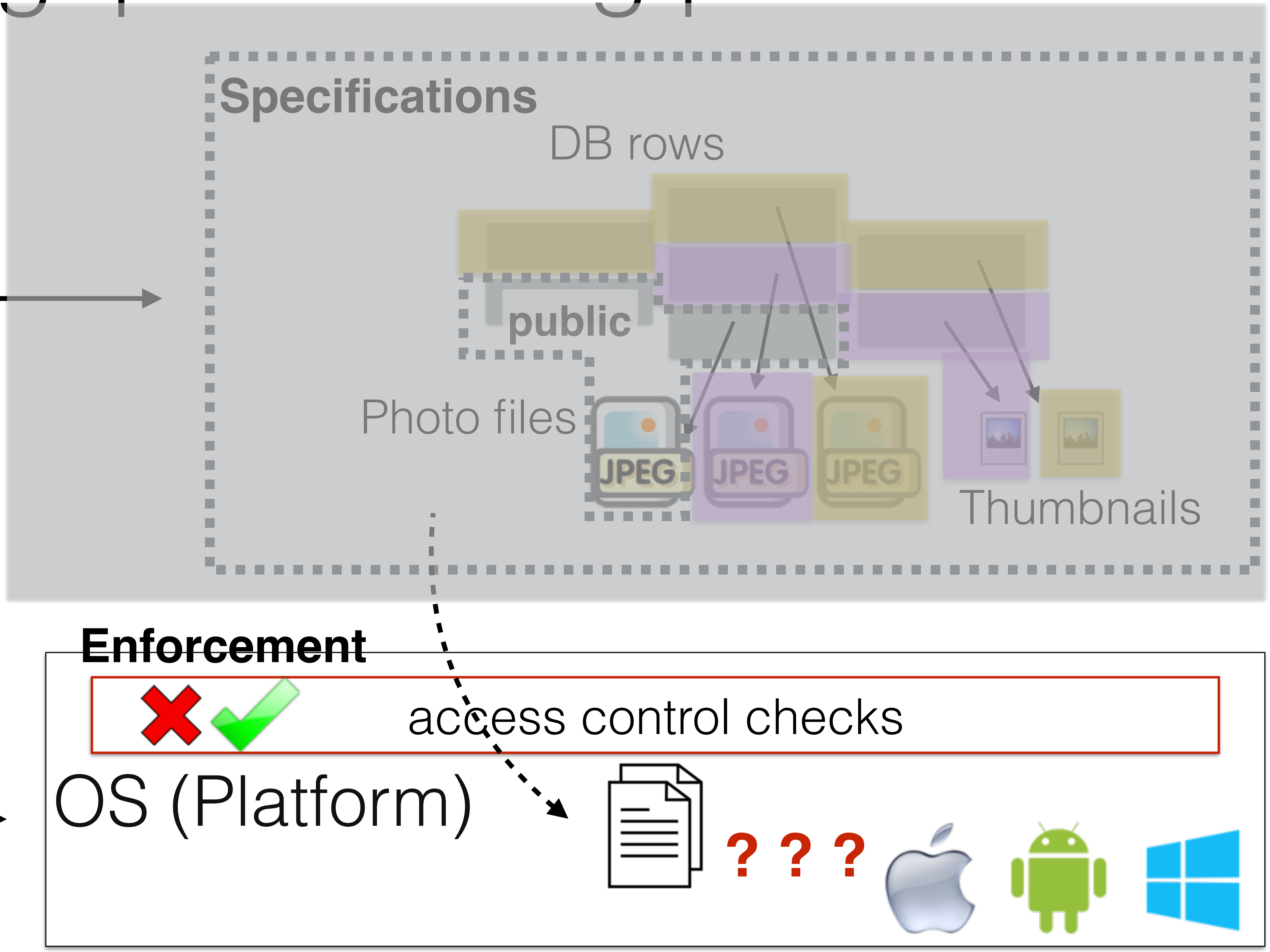


# Problem: semantic gap in existing platforms

Highly **structured** app-level data

**No visibility** to structures

**Unstructured** byte streams



**Specifications**

DB rows

**public**

Photo files

Thumbnails

**Enforcement**

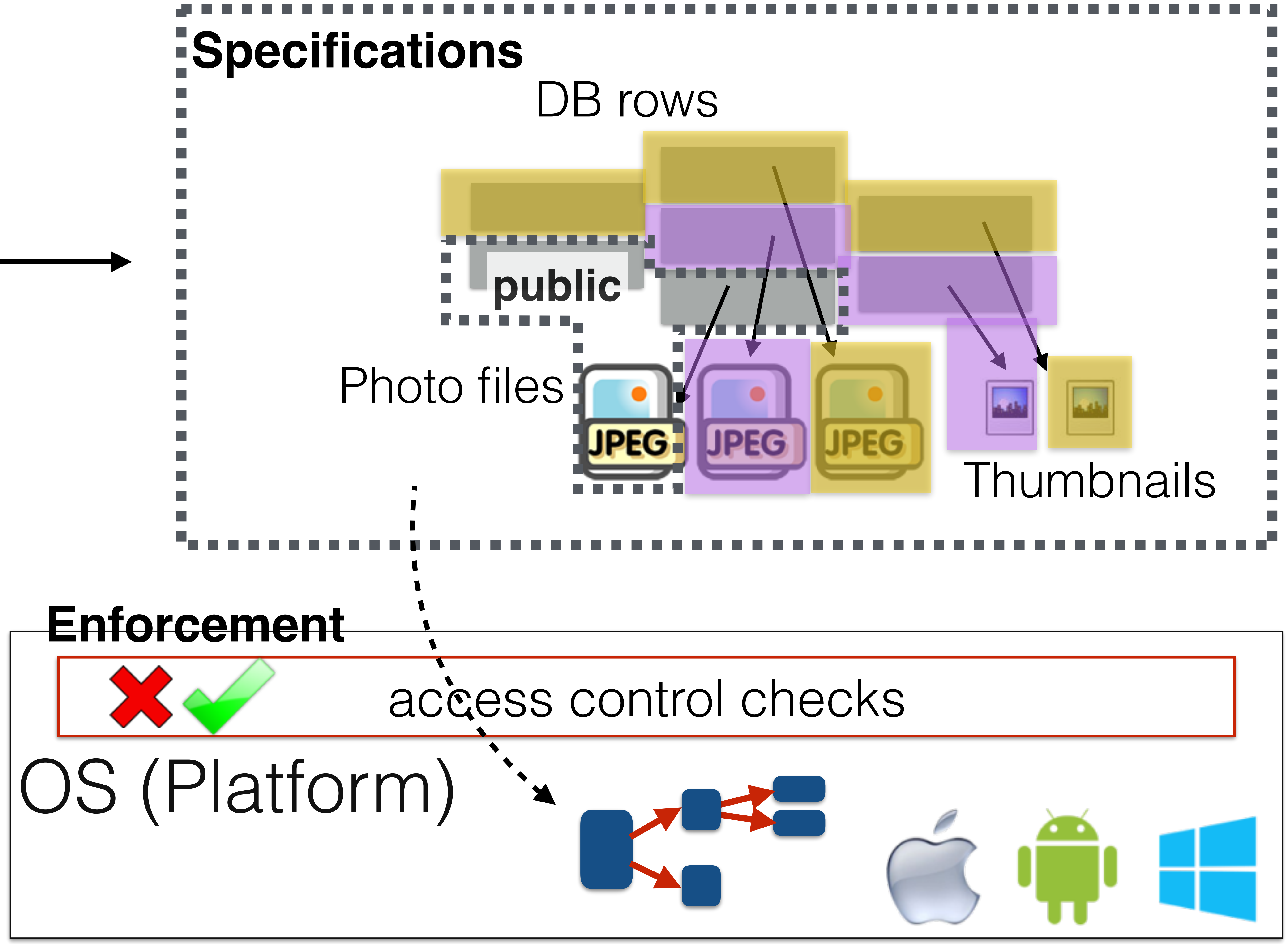
✗ ✓ access control checks

OS (Platform)

# Platform needs to understand structured data

Highly **structured**
app-level data

Platform-level **structured**
abstraction & protection



**Specifications**

DB rows

public

Photo files

Thumbnails

**Enforcement**

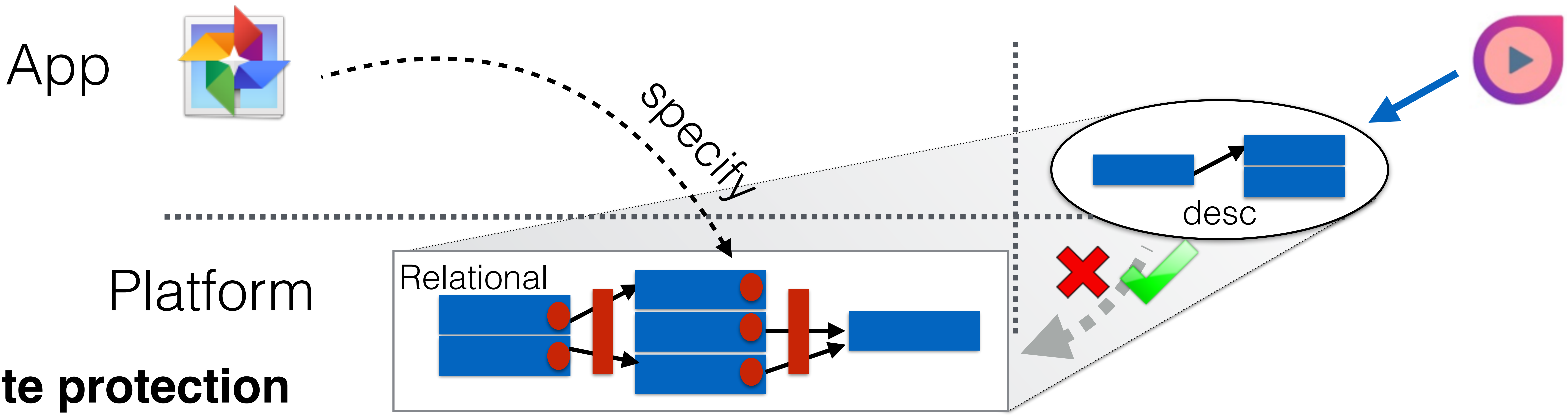
access control checks

OS (Platform)

# Earp

1. Make **relational model** a platform-level abstraction

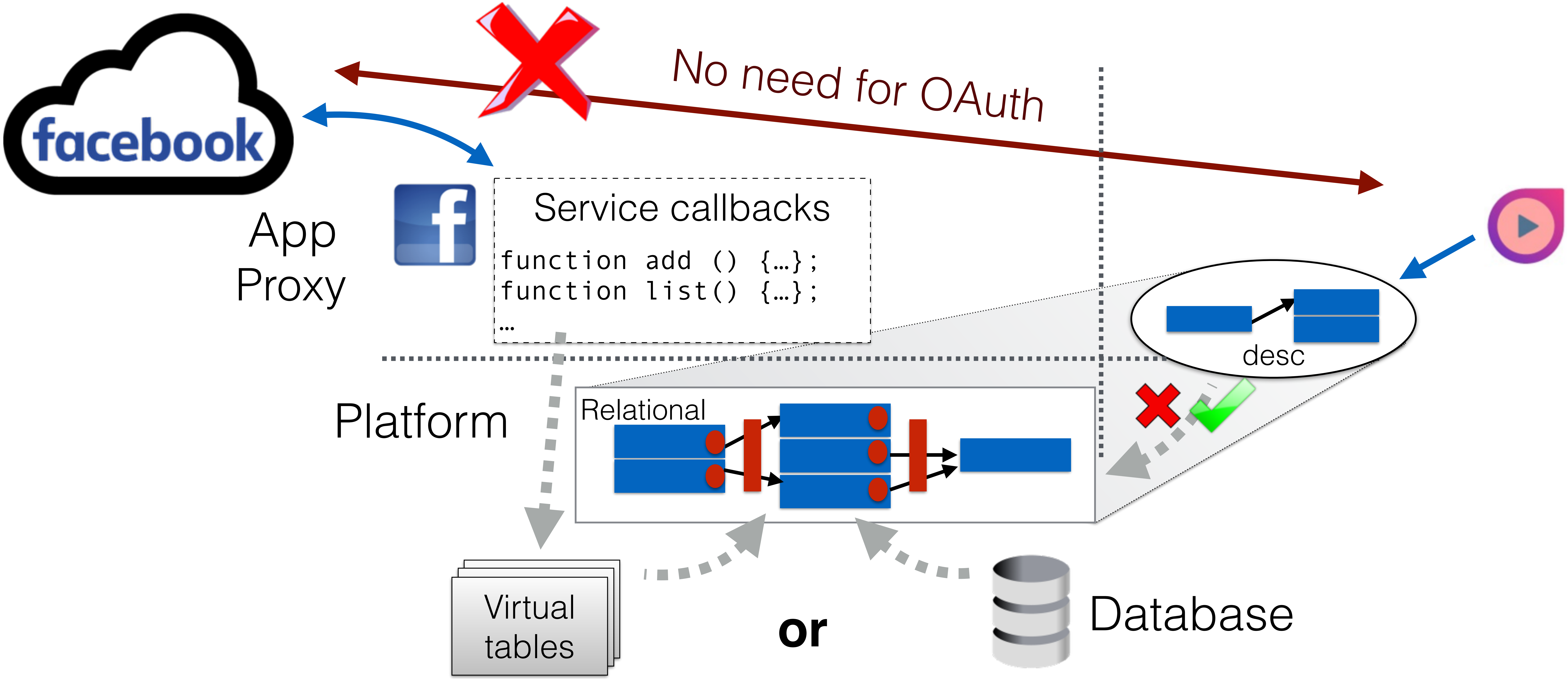
4. Uniform API: **subset descriptor** — capability handle, representing an access control view (but more than just a DB view)



2. **Integrate protection requirements** with the data model — annotated relational schema

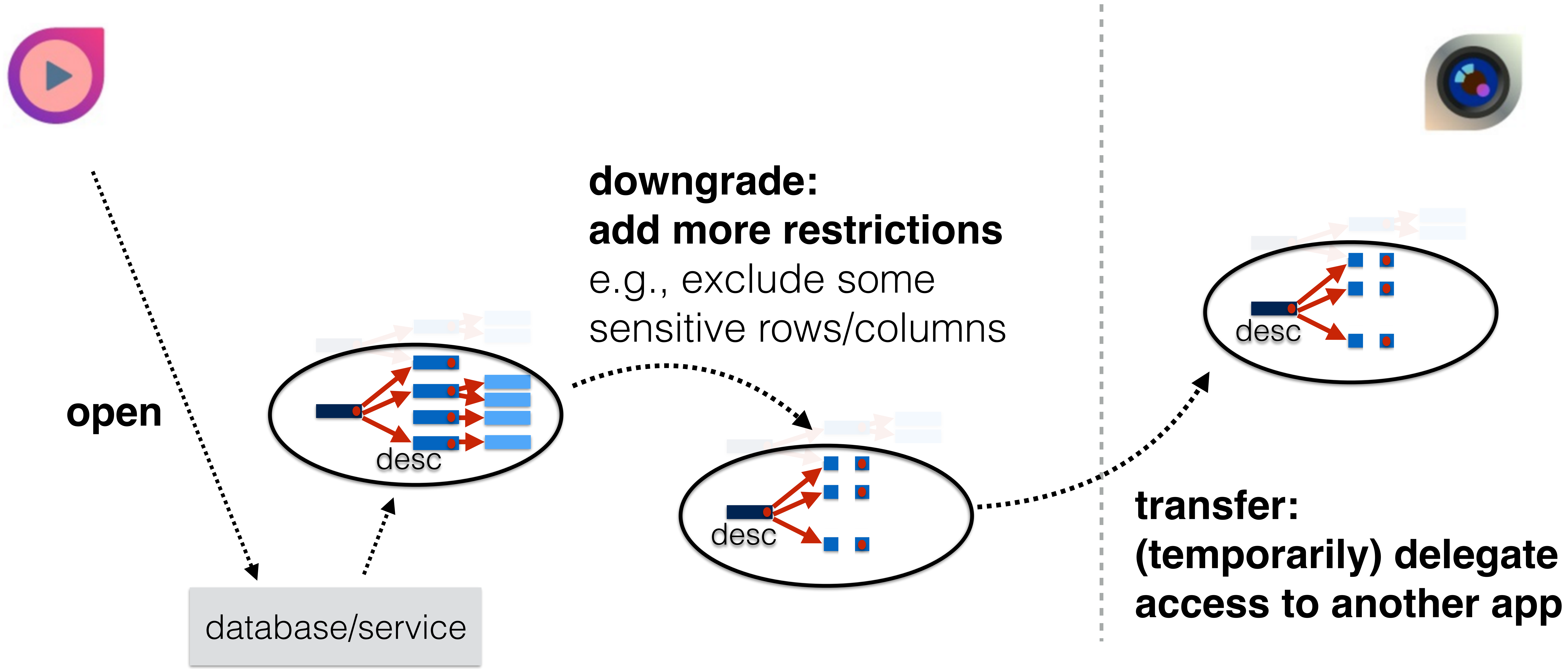
3. **Platform enforces** protection for the app

# Unify storage and inter-app services



```
function add () {…};
function list() {…};
…
```

# Subset descriptors are flexible

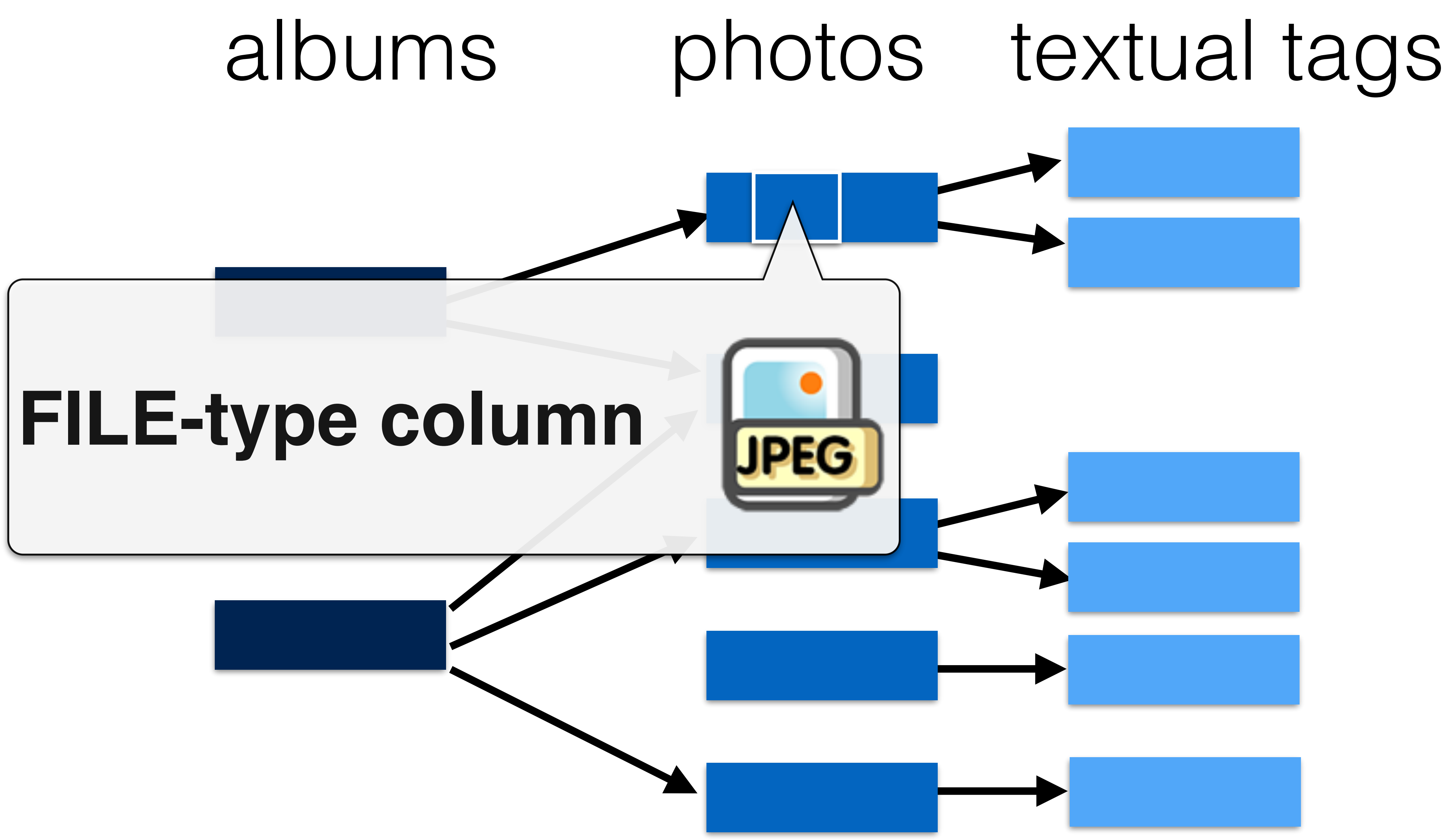


**downgrade:**
**add more restrictions**
e.g., exclude some
sensitive rows/columns

**transfer:**
**(temporarily) delegate**
**access to another app**

# Photo manager example revisited

objects in different tables

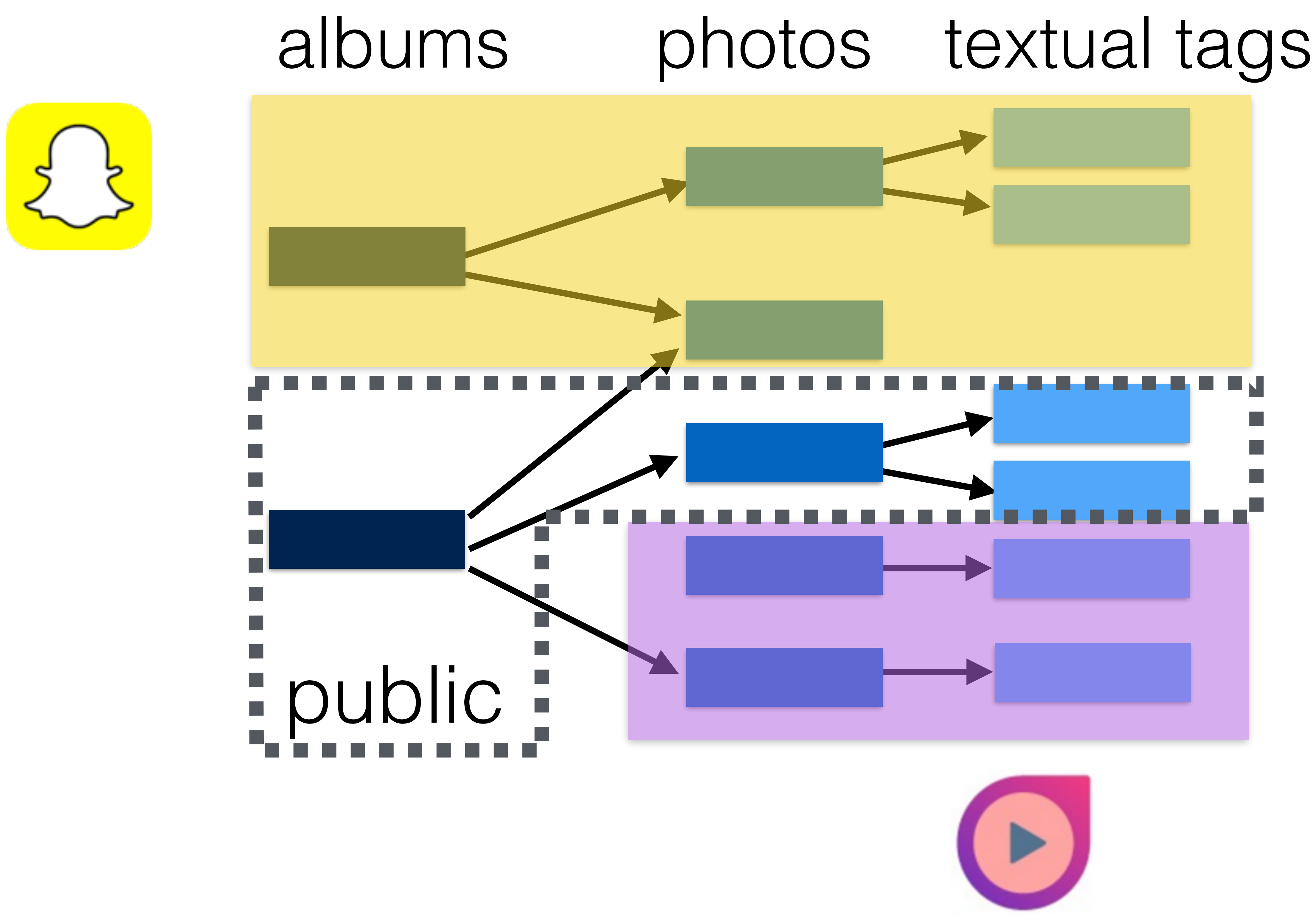


Operations:

- View photos directly
- View photos in an album
- Search photos with a certain tag

# Photo manager example revisited

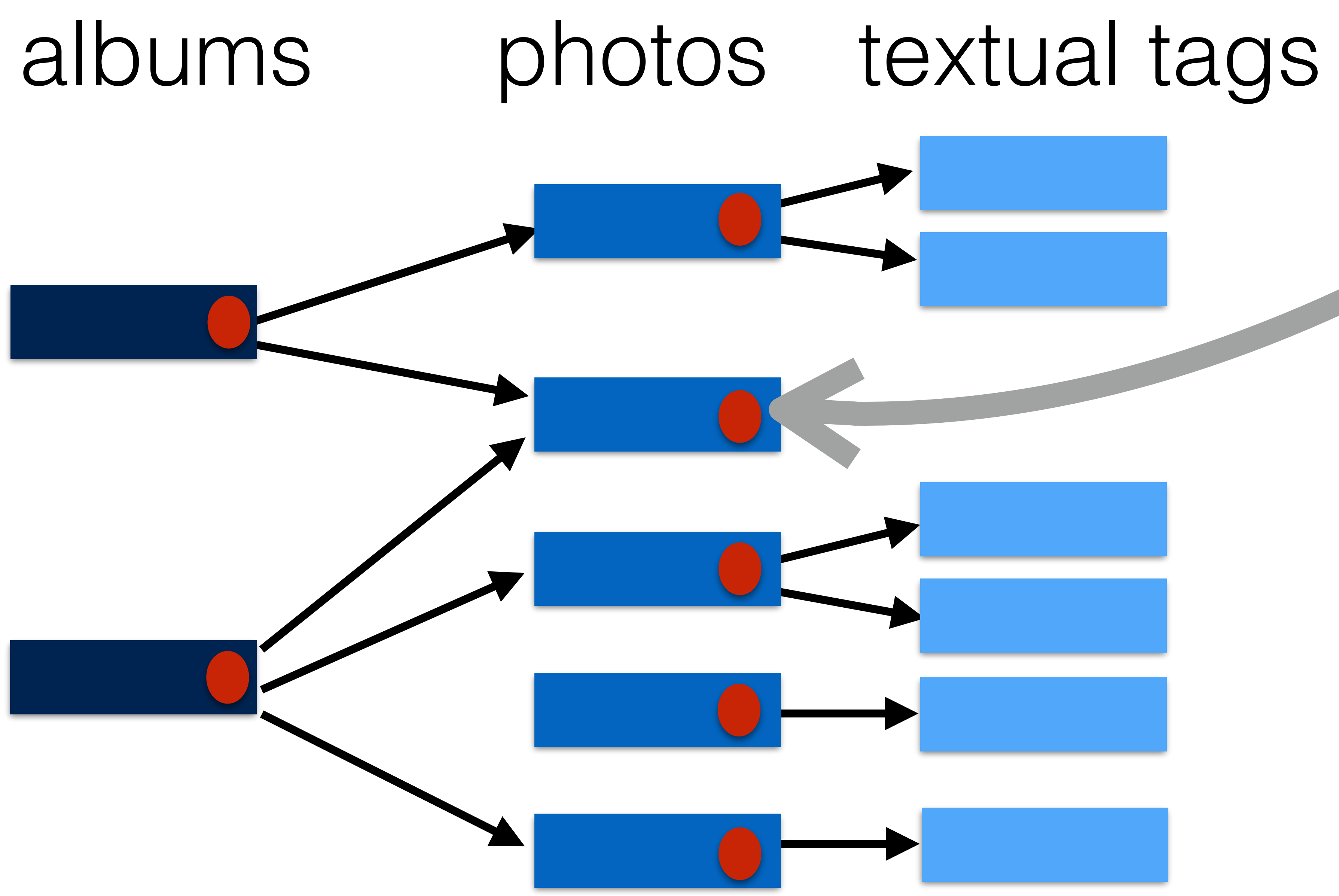
objects in different tables



Protection requirements:

- Each app has its own **private** photos and albums
- Apps share **public** photos and albums

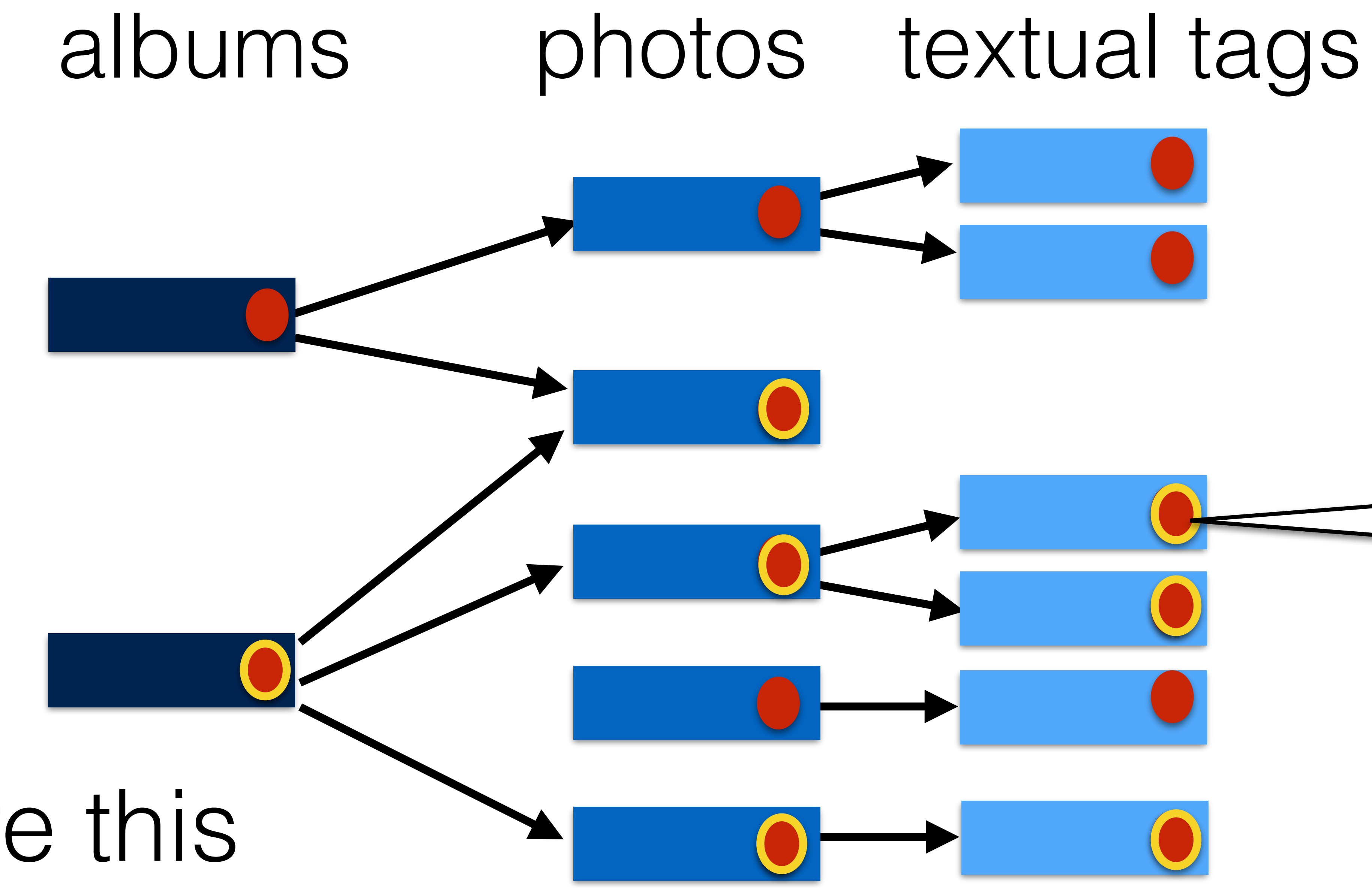
# Specify protection in data model #1



Per-object permissions (per-row ACLs)

# Fine-grained permissions are insufficient

Problem with permissions only: sharing collections of data.

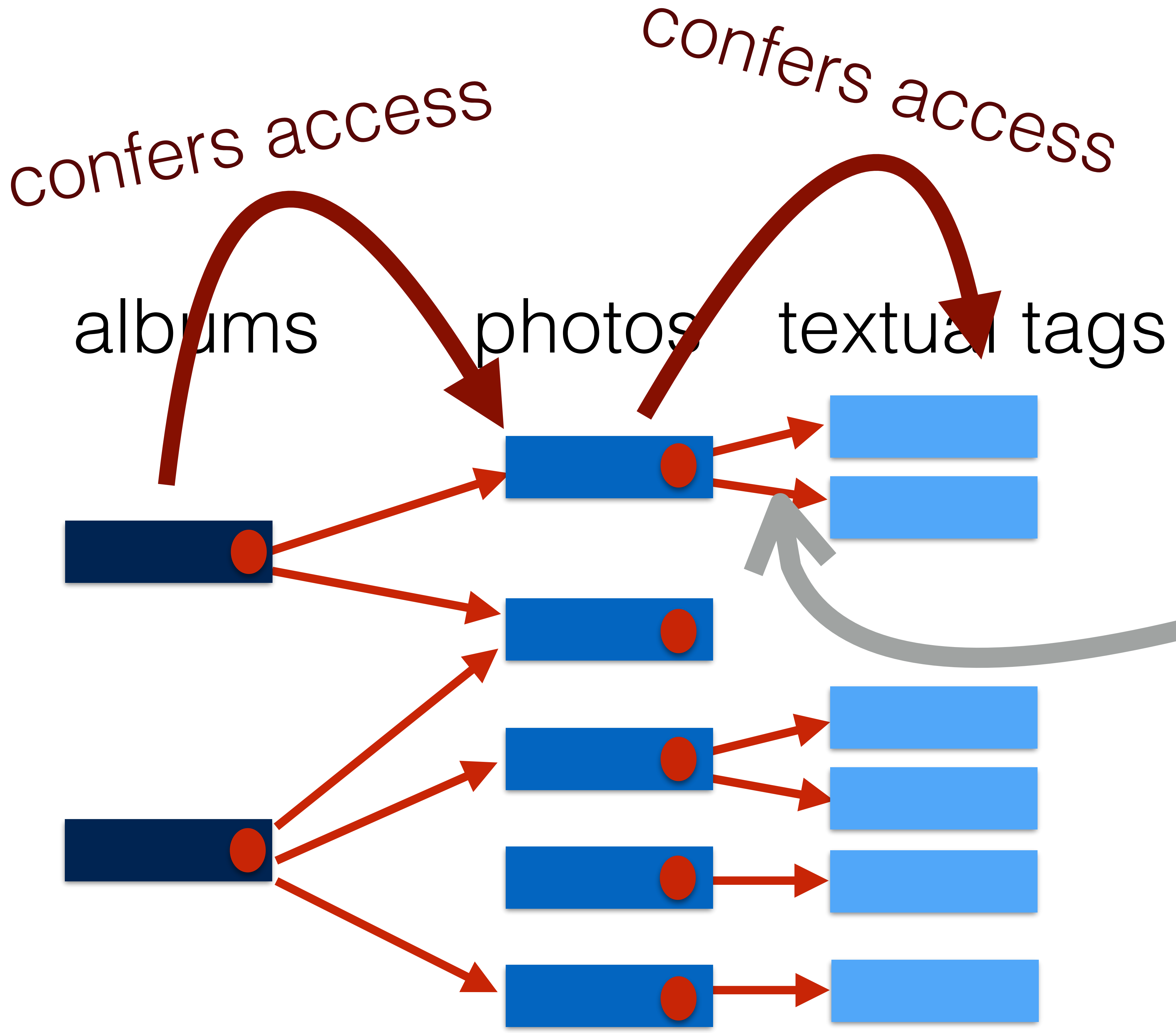


Share this album?

Need to **transitively** updating ACLs of many objects!

- Complicated permission management
- Consistency challenge

# Specify protection in data model #2



Capability relationships:

Cross-table relationships can confer access rights, in **one direction** (red arrows).

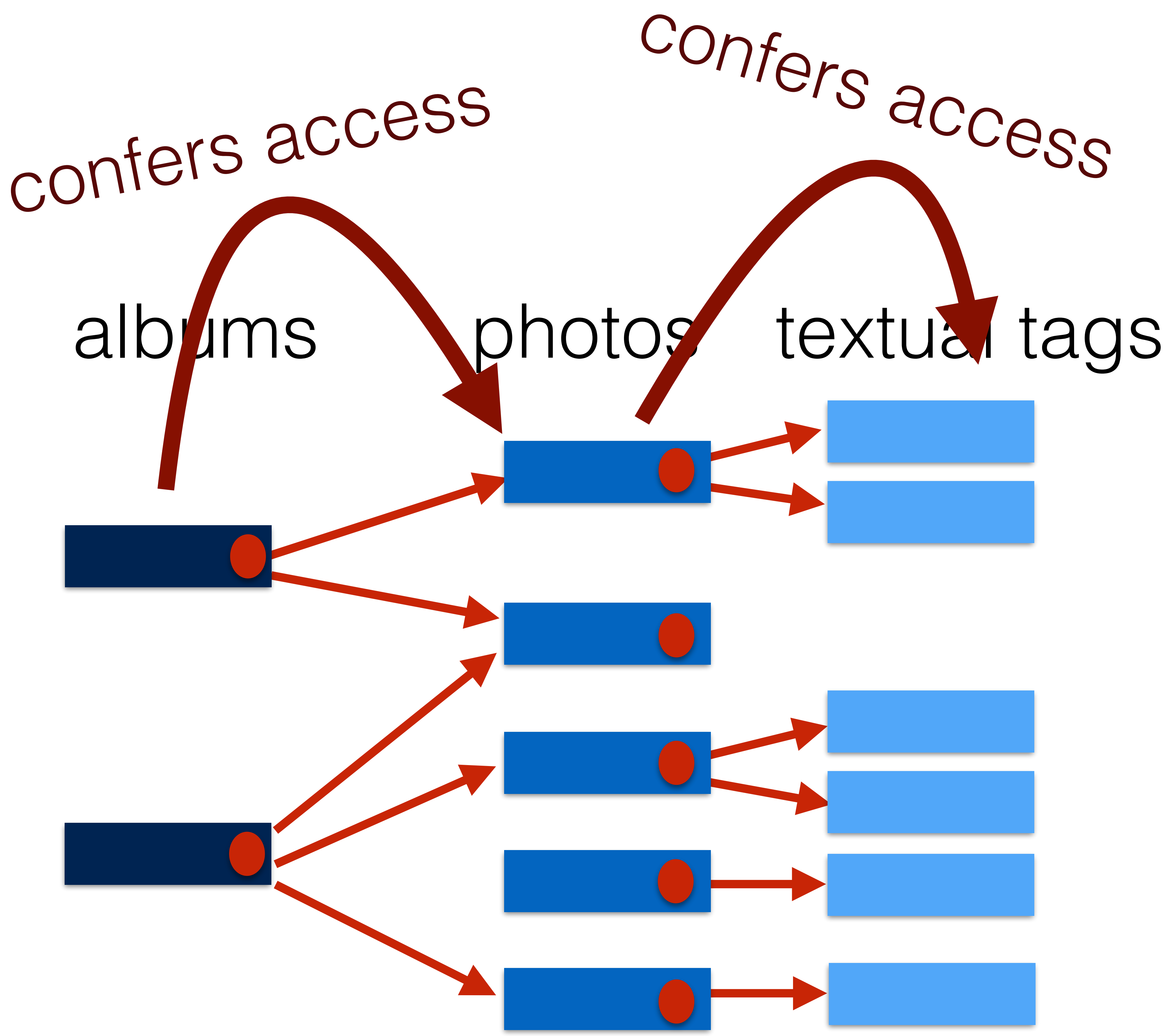
- Avoid transitively updating ACLs
- Achieve flexible access control with **simple ACLs**

# Done!

Data model is specified.

Let the platform enforce protection!

# But there is an efficiency challenge

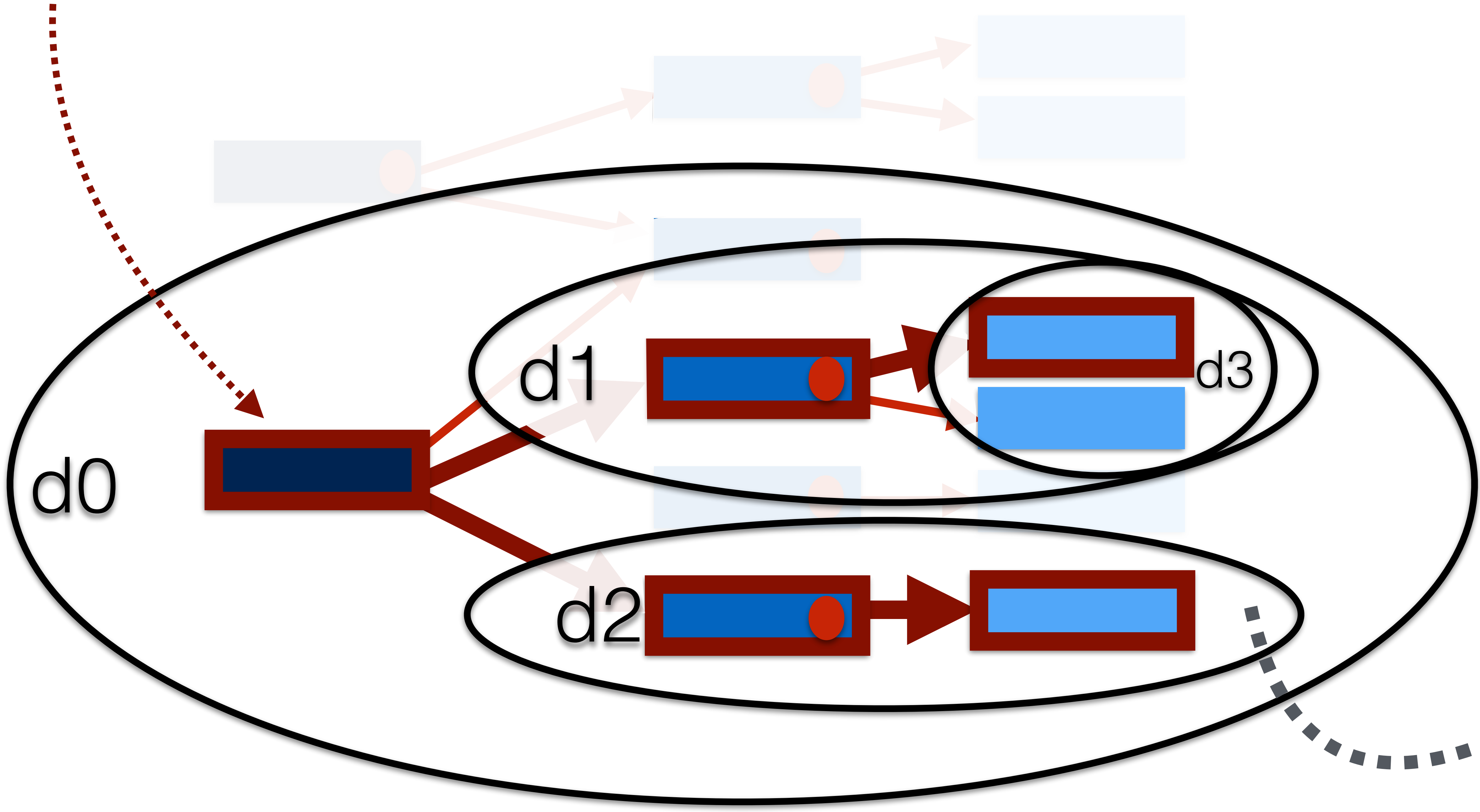


Capability relationships make access rights on one object may **depend** on other objects

Cross-table checks for every access?

# Minimize cross-table checks with descriptors

successful query proves access

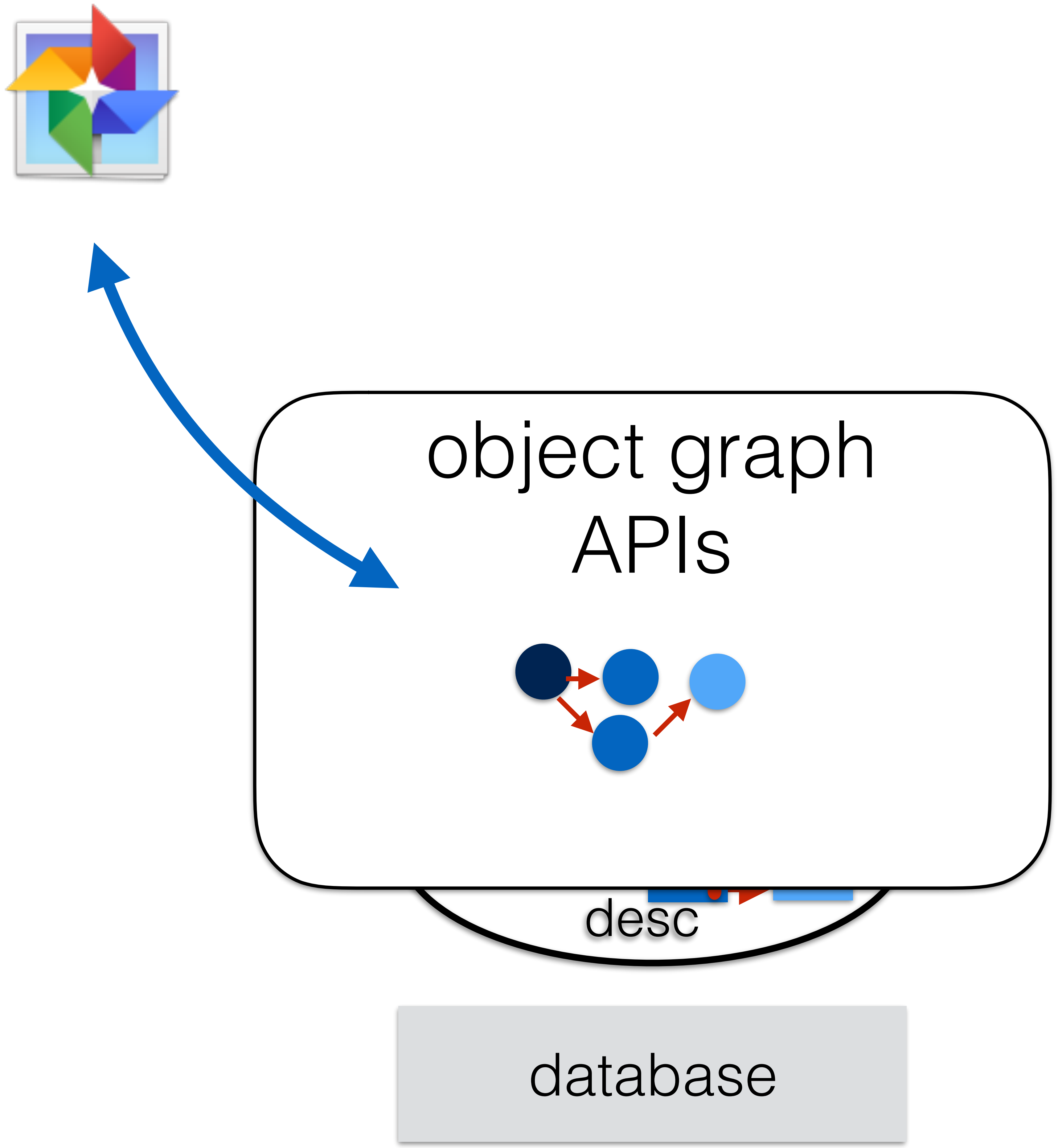


Solution: “**buffer**” computed access rights in descriptors

- E.g., **derive** fine-grained descriptors based on query results

Directly allow access to the photo

# Making it simpler to use



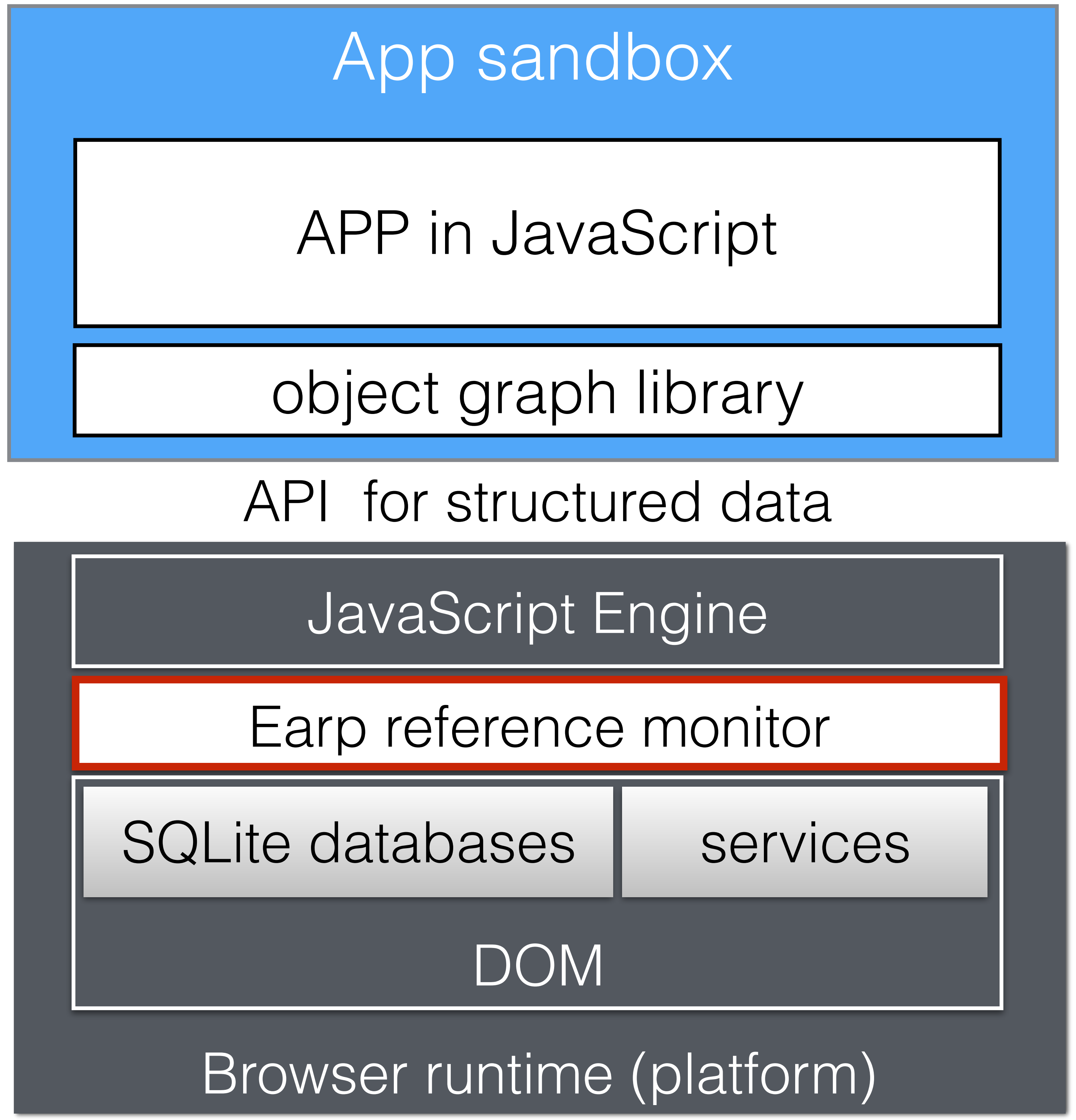
- Simple high-level APIs that hide details about descriptors
- Automates descriptor creation and management

# Implementation: browser-based platform

A modified Firefox OS:

- Apps written purely in Web code (HTML5, JavaScript)
- Structured APIs implemented in the platform (browser)

Paper discusses ways to apply Earp innovations to Android



# List of Earp apps

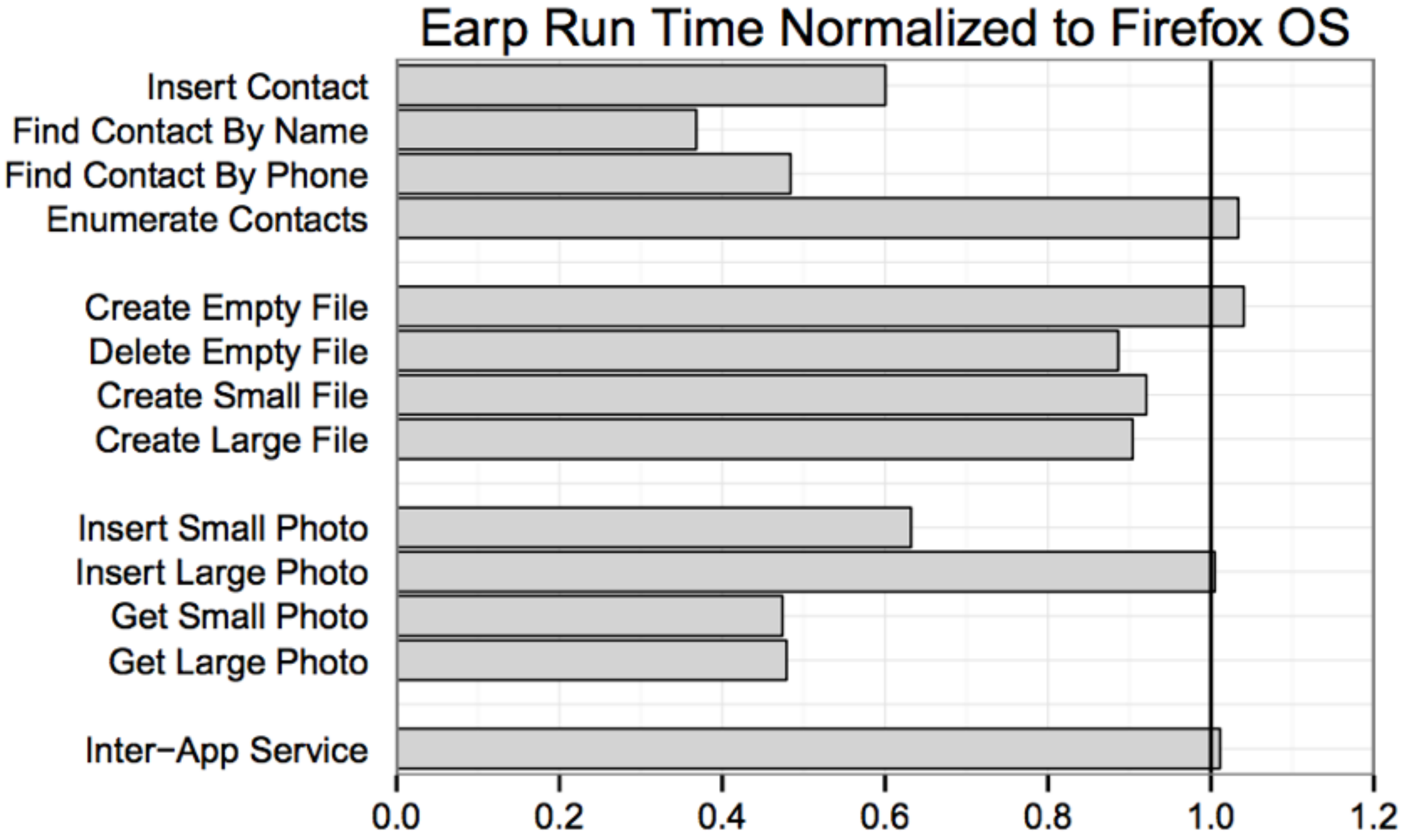
## Local apps

- Photo manager
- Contacts
  - Access control based on categories and data fields
- Email
  - Temporary, restricted access to attachments

## Proxies for remote services

- Egg-based social service
- Google Drive
  - Per-app private folders

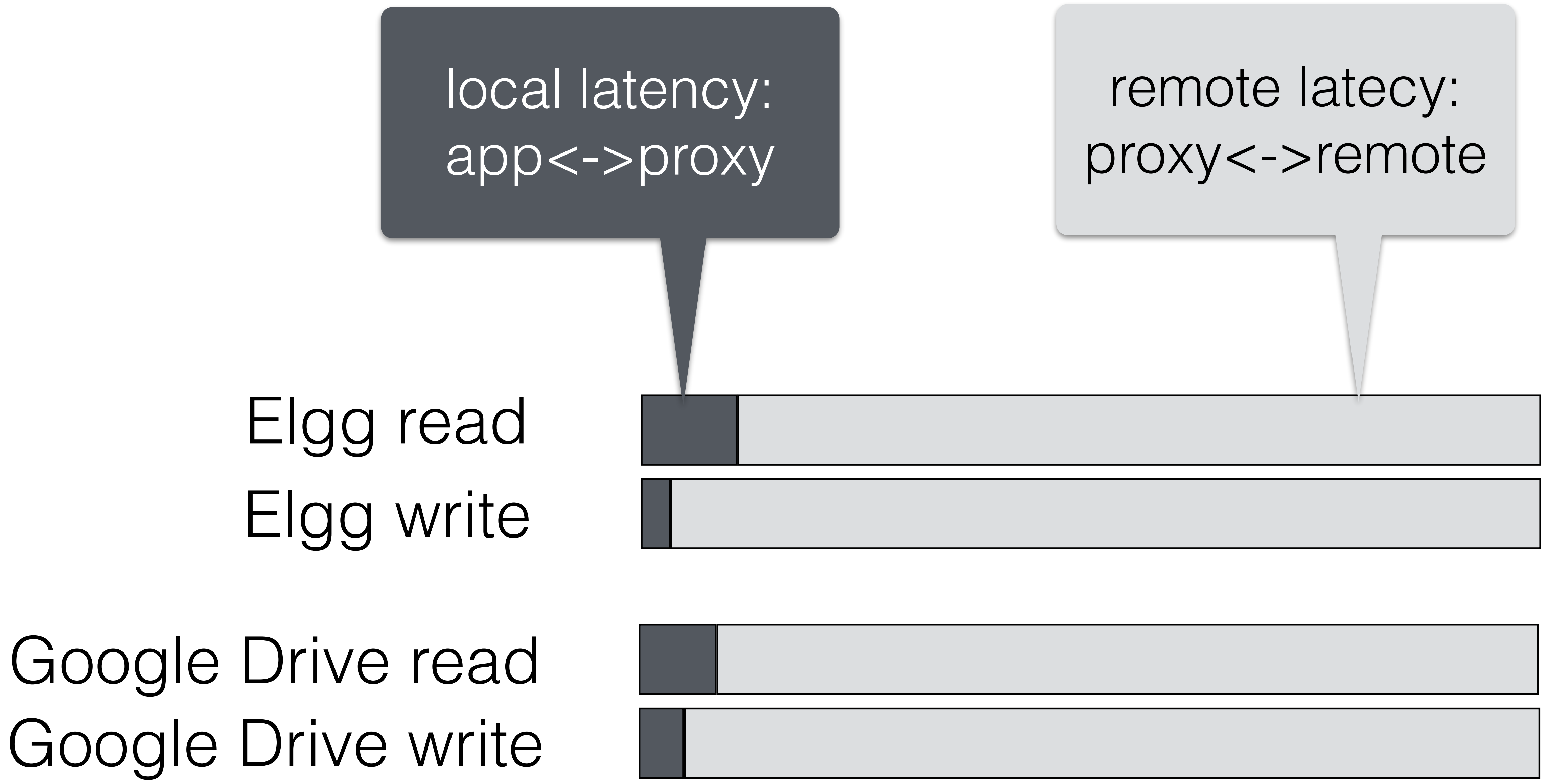
# Expressive access control can be efficient



Microbenchmarks: mostly outperforms baseline (Firefox OS)

- Earp apps directly use SQLite, and access control is efficient
- Firefox OS apps use IndexedDB (built on top of SQLite)

# Expressive access control can be efficient



Macrobenchmarks for remote services

- Local proxies add 2% - 8% latency

# Conclusion

- Inconsistent data abstractions in existing platforms
    - App: inter-related, structured data objects
    - Platform: unstructured byte streams
- Earp provides structured data as a platform-level abstraction
    - Principled storage, sharing, and protection