Auditing

- book 42's replacement cost changed from 20.00 to 200.00 and a member was billed 200.00 for losing it
  - who changed it?
  - when?
  - from where?
  - what was it before?
- *auditing* records who did what so you can answer those questions later
  - accountability – people know their changes are recorded
  - detection – unusual activity shows up
  - investigation / forensics – reconstructing what happened after an incident

Audit Trails

```
CREATE TABLE BOOK_AUDIT (
  audit_id    INT          NOT NULL AUTO_INCREMENT,
  book_id     INT          NOT NULL,
  changed_by  INT          NOT NULL,  -- librarian's member_id, from the session
  changed_at  DATETIME     NOT NULL DEFAULT CURRENT_TIMESTAMP,
  client_ip   VARCHAR(45)  NULL,
  old_cost    DECIMAL(6,2) NULL,
  new_cost    DECIMAL(6,2) NULL,
  PRIMARY KEY (audit_id)
);
```

- an *audit trail* captures who, what, when, from where, and before and after
- write the audit row in the same transaction as the change so both change and audit info commit or neither does

Generating Audit Records

- application writes audit rows or log entries
  - application has to record
  - application knows the app user
- trigger on the table
  - database records automatically
  - database doesn't know the app user itself – must be passed from the application
- DBMS audit features
  - records statements per database account
  - doesn't know the app user

Protecting the Audit Trail

```
GRANT INSERT ON demo_library.BOOK_AUDIT TO 'library_app'@'localhost';
GRANT SELECT ON demo_library.BOOK_AUDIT TO 'auditor'@'localhost';
```

- audit trail is only useful if it can't be quietly edited
  - application can add rows but not change or delete
- audit data is sensitive itself
  - records who did what, and often personal data
- policy: decide how long to keep the data and who may read it