

Exercise walk-through

8.6

Security and data governance

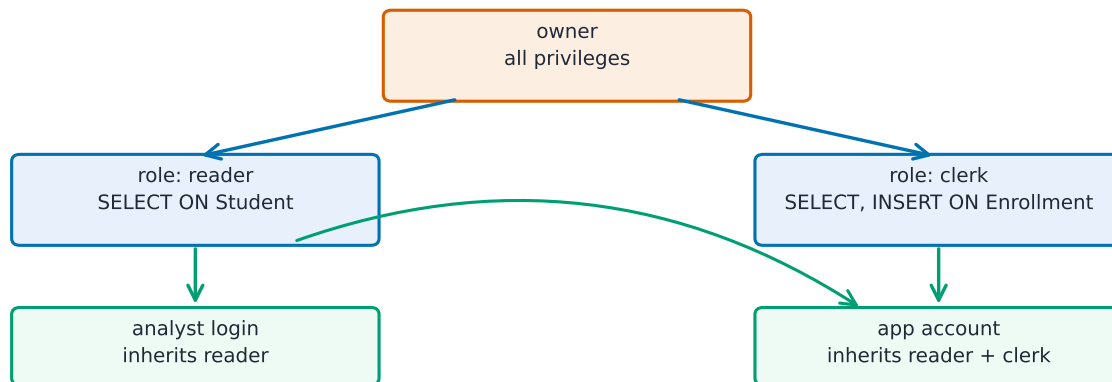
University Computer Science

Security and governance: narrow the grants

- 1 **Inject** Pasting user text lets ' OR '1'='1 match every row - all 8 students on the sample; **parameterised statements** end the class, not the example.
- 2 **Grants** GRANT SELECT ON Student TO reader; SELECT, INSERT ON Enrollment TO clerk - a stolen session is bounded by the grant.
- 3 **Loop** Governance classifies, assigns owners, sets retention, audits access - revoke on role change, never connect the app as owner.

Access model

grants flow down; nothing else flows



least privilege: the app account reads two tables and writes one - a stolen session is bounded by the grant

Grants flow from the owner to roles to users - nothing else flows.

Check yourself

- A many-to-many **relationship** becomes...? (A junction table.)
- Filtering on COUNT(*) belongs in...? (HAVING.)
- Fan-out 100, height 3 covers...? (A million keys.)
- The schedule R1 R2 W2 W1 ends at...? (90 - the deposit lost.)

Exercises 2.1-2.3

2.1 Input ' OR '1'='1 in a pasted query does what on the sample schema? [3]

2.2 Name the two channels a parameterised statement uses. [3]

2.3 Why should the application never connect as the owner? [3]

Exercises 2.1-2.3

2.1 Input ' OR '1'='1 in a pasted query does what on the sample schema? [3]

2.1

- The WHERE becomes always-true: all 8 students match ✓.
-

2.2 Name the two channels a parameterised statement uses. [3]

2.2

- The compiled query text; the bound parameter values ✓.
-

2.3 Why should the application never connect as the owner? [3]

2.3

- Every application bug would run with full owner authority – no bounds on the blast radius ✓.

Exercise 3.1

3.1 The app needs: read Student and Course, insert and delete Enrollment, never touch Department.

Write the three GRANT statements. State what a stolen app session can and cannot do under them. State the extra grant an auditing role needs to read the access log, and why revoke-on-role-change matters. [8]

Exercise 3.1

3.1 The app needs: read Student and Course, insert and delete Enrollment, never touch Department.

Write the three GRANT statements. State what a stolen app session can and cannot do under them. State the extra grant an auditing role needs to read the access log, and why revoke-on-role-change matters. [8]

3.1

- `GRANT SELECT ON Student TO app; GRANT SELECT ON Course TO app; ✓.`
- `GRANT SELECT, INSERT, DELETE ON Enrollment TO app; ✓.`
- Stolen session: can read students and courses, enroll and un-enroll – cannot touch Department, cannot drop or alter anything ✓.
- Auditor: `GRANT SELECT ON access_log TO auditor; ✓.`
- Revoke matters because privileges creep: old roles accumulate rights nobody uses – the audit loop must shrink them ✓.

Exercise 3.2

3.2 A pasted query matches every row; the parameterised version returns only exact-name matches.

Explain why the attack string cannot change the query's structure once parameterised. Give the four governance steps that keep the model true over time. State why encrypting the grade column does not stop a valid login from reading it, and what does. [7]

Exercise 3.2

3.2 A pasted query matches every row; the parameterised version returns only exact-name matches.

Explain why the attack string cannot change the query's structure once parameterised. Give the four governance steps that keep the model true over time. State why encrypting the grade column does not stop a valid login from reading it, and what does. [7]

3.2

- The query is compiled with placeholders: the attacker's text is only ever bound as a value compared for equality – it cannot become SQL ✓.
- Governance: classify the data, assign owners, set retention, audit access ✓.
- Encryption protects data at rest – a valid login reads through the database's own decryption; grants bound what the login may select ✓.