

DBS\_MidSem2025

Exported on: 2/22/2026, 6:21:56 PM

User

Answer (i)

Gemini

Based on the requirements provided in the images, here is the Role-Based Access Control (RBAC) configuration for the first part of your assignment.

### i. Role-Hierarchy and Permission-Assignment Relations

To accurately represent the department's needs, we define the hierarchy where higher roles inherit permissions from lower roles, and then assign specific permissions to the base roles.

#### Permission-Assignment Relation (PA)

We first map the fundamental permissions to the specific course roles.

| Role     | Permissions Assigned         |
|----------|------------------------------|
| Fac Grad | read 123, read 456, read 789 |
| 123TA    | read 123, write 123          |
| 456TA    | read 456, write 456          |
| 789TA    | read 789, write 789          |
| 123CI    | submit 123                   |
| 456CI    | submit 456                   |
| 789CI    | submit 789                   |

#### Role-Hierarchy Relation (RH)

The hierarchy is defined by the  $\geq$  symbol, where  $r_1 \geq r_2$  means  $r_1$  inherits all permissions of  $r_2$ .

- Faculty/Grad Base Inheritance:** All faculty and graduate students must be able to read all grades.
  - Fac Grad is the base role containing the "read all" permissions.
- Course Instructor (CI) Inheritance:** A Course Instructor can do everything a TA can do, plus submit final grades.
  - $123CI \geq 123TA$
  - $456CI \geq 456TA$
  - $789CI \geq 789TA$
- Cross-Course Visibility:** Since every TA or CI is also either a faculty member or a graduate student (and all faculty/graduates can read all grades):
  - $123TA \geq FacGrad$
  - $456TA \geq FacGrad$
  - $789TA \geq FacGrad$
  - $123CI \geq FacGrad$
  - $456CI \geq FacGrad$
  - $789CI \geq FacGrad$

**Note:** By setting the CIs to inherit from TAs, and TAs to inherit from Fac Grad, we satisfy the requirement that a CI can read, write, and submit, while a standard faculty member (in the Fac Grad role) can only read.

Would you like me to proceed with the **Static Separation of Duty (SSD)** relations for part (ii)?

User

Answer (ii)

Gemini

To address part (ii), we need to define the **Static Separation of Duty (SSD)** relations. These constraints are applied at the time of **role assignment** (UA) to prevent a user from being assigned to a combination of roles that would violate the department's security policies.

ii. Static Separation-of-Duty (SSD) Relations

According to the requirements, we must capture the constraints regarding graduate students and the limitations on how many roles they can hold.

1. Graduate Student Role Limit

The prompt states: "No graduate student can be a TA for more than two courses." This is a classic *k-out-of-n* SSD constraint. It ensures that no single user (who is a graduate student) can be assigned more than two TA roles.

- SSD-1: ({123TA, 456TA, 789TA}, 3)

**Logic:** This means a user can be assigned to any 2 of these roles, but being assigned to 3 (all of them) is strictly forbidden.

2. Role-Based Exclusion (Graduate vs. CI)

The prompt states: "No graduate student can be a Course instructor for any course." Since the system handles three courses, a graduate student cannot hold any of the CI roles. We can define this as a set of mutual exclusions between the "Graduate" status and the "CI" roles.

- SSD-2: ({Grad, 123CI}, 2)
- SSD-3: ({Grad, 456CI}, 2)
- SSD-4: ({Grad, 789CI}, 2)

**Logic:** If a user is assigned the "Grad" role, they are statically barred from being assigned any of the CI roles.

3. Faculty vs. Graduate Disjointness

The prompt states: "Graduate students and faculty are disjoint groups." This means a user cannot be both a Faculty member and a Graduate student simultaneously.

- SSD-5: ({Fac, Grad}, 2)

Summary Table of SSD Constraints

| Constraint Name | Role Set              | Threshold ( <i>k</i> ) | Description                              |
|-----------------|-----------------------|------------------------|--|
| Max TA Roles    | {123TA, 456TA, 789TA} | 3                      | Prevents a user from holding 3 TA roles. |

| Constraint Name    | Role Set                    | Threshold ( $k$ ) | Description                                      |
|--------------------|-----------------------------|-------------------|--|
| Grad-CI Exclusion  | {Grad, 123CI, 456CI, 789CI} | 2                 | A user in the Grad role cannot hold any CI role. |
| Group Disjointness | {Fac, Grad}                 | 2                 | A user cannot be both Faculty and Graduate.      |



Export to Sheets



**Note:** We do **not** apply SSD to Faculty members regarding TA/CI roles because the prompt explicitly says "there are no limits on how they are assigned to courses."

Would you like me to wrap this up with the **Dynamic Separation-of-Duty (DSD)** relation for part (iii)?

User

(iii)

Gemini

To complete the set, here are the **Dynamic Separation-of-Duty (DSD)** relations. These constraints differ from SSD because they allow a user to be *assigned* to multiple roles, but they prevent a user from *activating* those roles simultaneously in the same session.

iii. Dynamic Separation-of-Duty (DSD) Relations

The prompt specifically identifies a "potential for confusion with final grades" and dictates that the system must not permit anyone to perform in **more than one Course-instructor (CI) role** at any given time (e.g., login session).

1. Course-Instructor Role Activation Constraint

Since faculty members are allowed to be assigned to any number of courses (no limits on assignment), they might be the CI for CSE 123, CSE 456, and CSE 789 all at once. However, to prevent the mentioned confusion, they can only "act" as the instructor for one at a time.

- **DSD-1:** ({123CI, 456CI, 789CI}, 2)

**Logic:** A user may have all three CI roles in their user profile, but the system will block a session from activating more than one. If they are logged in as 123CI , they must close that session or switch roles to act as 456CI .

2. Implementation Summary

While SSD (Part ii) handled the *eligibility* (e.g., Graduate students can't be CIs), DSD handles the *workflow*.

| Constraint Name      | Role Set              | Threshold ( <i>k</i> ) | Scope               |
|----------------------|-----------------------|------------------------|---------------------|
| Single-CI Activation | {123CI, 456CI, 789CI} | 2                      | Per Session / Login |



Export to Sheets



**Note:** We do **not** need a DSD for TA roles or the "Fac Grad" role, as the prompt only explicitly restricts the "Course-instructor role" from being performed in multiples simultaneously.

Summary of the complete RBAC Model:

- **Part (i):** Established the **Inheritance** (CI > TA > Fac/Grad) so permissions flow correctly.
- **Part (ii):** Established **Assignment** limits (Grads can't be CIs, and max 2 TA roles).
- **Part (iii):** Established **Activation** limits (Only 1 CI role active at a time).

Would you like me to help you draft a brief explanation of why these specific DSD/SSD choices satisfy the "no permissions or restrictions other than those explicitly mentioned" rule for your lab report?

**User**

answer (i)

**Gemini**

To address the database security requirements for **SPIELBERG** in the purchasing department, we must provide him with access to manage the inventory and the financial/production aspects of the films and studios.

**i. SPIELBERG (Purchasing Department)****Assumptions:**

- The purchasing department is responsible for acquiring new stock and managing studio relationships.
- They need to modify **ITEM** (inventory/price) and **STUDIO** (contact info) but only need to view **FILM** details to make buying decisions.
- Since this is a specific department role, he is granted propagation privileges (**WITH GRANT OPTION**) so he can delegate these tasks to subordinates within the purchasing team.

**SQL Commands****SQL**

```
-- Granting permission to view film details
GRANT SELECT ON FILM TO SPIELBERG WITH GRANT OPTION;

-- Granting permission to manage studio information
GRANT SELECT, INSERT, UPDATE ON STUDIO TO SPIELBERG WITH GRANT OPTION;

-- Granting permission to manage inventory and pricing
GRANT SELECT, INSERT, UPDATE, DELETE ON ITEM TO SPIELBERG WITH GRANT OPTION;
```

**Explanation of Choices**

- **FILM Table:** SPIELBERG needs **SELECT** to identify which movies are available, their ratings, and directors to decide what the store should purchase.
- **STUDIO Table:** He needs **INSERT** and **UPDATE** to add new production partners or update their addresses and websites.
- **ITEM Table:** This is the core of purchasing. He must be able to add new stock items ( **INSERT** ), adjust prices based on market value ( **UPDATE** ), and remove discontinued items ( **DELETE** ).
- **Grant Propagation:** Using **WITH GRANT OPTION** allows SPIELBERG, as a primary user in purchasing, to manage the access levels of other clerks in his department without needing the Database Administrator for every minor change.

**Next Step:** Would you like me to provide the SQL commands and explanations for **SHYAMALN** in the shipping department or **MARSHALL** in customer service?

**User**

Answer (ii)

**Gemini**

To address the database security requirements for **SHYAMALN** in the shipping department, the privileges focus on the logistics of getting orders to customers.

**ii. SHYAMALN (Shipping Department)****Assumptions:**

- The shipping department needs to know *what* to ship ( **ITEM** and **CART** ) and *where* to ship it ( **SHOPPER** ).
- They do not need access to sensitive financial information like credit card numbers or internal studio contracts.
- Unlike a department manager, a shipping clerk typically does not need to delegate permissions, so **WITH GRANT OPTION** is omitted.

**SQL Commands****SQL**

```
-- Granting permission to view shipping addresses and names only
GRANT SELECT (SHNUM, FNAME, LNAME, SHIPADDR) ON SHOPPER TO SHYAMALN;

-- Granting permission to view what is in the customer's cart for fulfillment
GRANT SELECT ON CART TO SHYAMALN;

-- Granting permission to view item details to verify the correct format/title
GRANT SELECT ON ITEM TO SHYAMALN;
GRANT SELECT (FNUM, TITLE) ON FILM TO SHYAMALN;
```

**Explanation of Choices**

- **SHOPPER Table:** Access is restricted to specific columns. SHYAMALN needs the address and name to print shipping labels, but for security and privacy, he is **denied access** to the **CRCARD** column.
- **CART Table:** This is necessary to see the **QUANTITY** and specific **INUM** (item number) that needs to be packed for a specific shopper.
- **ITEM and FILM Tables:** Shipping needs to verify that they are putting the correct "Format" (e.g., DVD vs. Blu-ray) in the box. **SELECT** on these tables allows for inventory verification without allowing SHYAMALN to change prices or film metadata.
- **No Grant Propagation:** Since SHYAMALN is an operational user in shipping rather than an administrator, he is not given the authority to pass these privileges to others.

**Next Step:** Would you like the final set of SQL commands for **MARSHALL**, the manager of the Customer Service department?

**User**

Answer (iii)

**Gemini**

To address the database security requirements for **MARSHALL**, the manager of the customer service department, the privileges are designed to facilitate customer support, dispute resolution, and team management.

**iii. MARSHALL (Customer Service Manager)****Assumptions:**

- As a manager, Marshall needs broad visibility to resolve customer issues but does not need access to internal production costs or studio contracts.
- He requires access to `SHOPPER` and `CART` details to track orders and verify accounts.
- Since he is a manager, he is granted propagation privileges (**WITH GRANT OPTION**) to delegate viewing tasks to his customer service representatives.

**SQL Commands****SQL**

```
-- Access to customer profiles for support (excluding sensitive credit card data)
GRANT SELECT (SHNUM, FNAME, LNAME, SHIPADDR) ON SHOPPER TO MARSHALL WITH GRANT OPTION;

-- Access to view order history and current carts for troubleshooting
GRANT SELECT ON CART TO MARSHALL WITH GRANT OPTION;

-- Access to view item and film details to answer customer queries about products
GRANT SELECT ON ITEM TO MARSHALL WITH GRANT OPTION;
GRANT SELECT ON FILM TO MARSHALL WITH GRANT OPTION;

-- Access to view actor/role info to assist customers looking for specific content
GRANT SELECT ON ACTOR TO MARSHALL WITH GRANT OPTION;
GRANT SELECT ON ROLES TO MARSHALL WITH GRANT OPTION;
```

**Explanation of Choices**

- **SHOPPER and CART Tables:** These are essential for identifying the customer and seeing what they have ordered. Restricting `CRCARD` access maintains security while allowing Marshall to confirm shipping addresses or order quantities.
- **FILM, ITEM, ACTOR, and ROLES Tables:** Customer service often needs to help users find movies based on a specific actor or check if a certain format (like Blu-ray) is in stock. `SELECT` access across these tables empowers the department to provide comprehensive information.
- **Grant Propagation:** As a department head, the `WITH GRANT OPTION` allows Marshall to grant `SELECT` privileges to his staff so they can handle routine tickets without his direct intervention for every query.
- **Excluded Tables:** Marshall is not granted access to the `STUDIO` table or the `SALARY` column in the `ROLES` table, as these are internal business/financial details unnecessary for customer support.

**Next Step:** These three parts complete the security setup for Question 2. Would you like me to review the full set of SQL commands for any specific edge cases, like how to revoke these privileges if a user changes

departments?

## User

Answer Q3) (a)

## Gemini

To answer **Question 3(a)**, we need to compare Mandatory Access Control (MAC) and Discretionary Access Control (DAC) through the lens of data leakage prevention.

### Q3. a) Mandatory Access Controls (MAC) vs. Discretionary Access Controls (DAC)

A scenario where **Mandatory Access Control (MAC)** prevents a breach that **Discretionary Access Control (DAC)** cannot is a **Trojan Horse attack involving unauthorized information flow**.

#### The Scenario: Unauthorized Information Leakage

Imagine two users in an organization: **Alice** (who has access to a "Secret" project file) and **Bob** (who does not have access to that file).

##### 1. Under Discretionary Access Control (DAC):

- Alice owns a file named `Secret_Project.doc`. Because she is the owner, she has the "discretion" to grant permissions to others.
- A malicious actor tricks Alice into running a seemingly harmless program (a Trojan Horse).
- Since the program runs with Alice's permissions, it can silently create a new file, copy the contents of `Secret_Project.doc` into it, and then execute a command to give **Bob** read access to this new copy.
- **The Result:** Bob now has the secret information. DAC failed because it allows the "discretionary" transfer of permissions by users or their software.

##### 2. Under Mandatory Access Control (MAC):

- The system uses labels: Alice and the `Secret_Project.doc` are labeled **Secret**, while Bob is labeled **Unclassified**.
- The system enforces the **Bell-LaPadula** "No Write Down" (Simple Security Property) and "No Read Up" rules.
- Even if Alice is tricked into running the same Trojan Horse, the program is constrained by the system-wide security policy.
- When the Trojan tries to write the secret data into a file that Bob can read (an "Unclassified" file), the system blocks the action because a **Secret** process is forbidden from writing to an **Unclassified** object.
- **The Result:** The breach is prevented. MAC overrides Alice's personal permissions and enforces a global policy that prevents data from flowing from a high security level to a lower one.

**Next Step:** Would you like to move on to part **(b)** to design the **lattice model** for the medical information system?



**User**

Answer b part (i)

**Gemini**

To capture the medical information system's policy in a lattice model that prevents information flow from **patient records** to **prescriptions**, we use the concept of security classes and the **Bell-LaPadula** model properties.

**Q3. b) i. Lattice Model for Information Flow Control**

To prevent information flow from  $A$  to  $B$  in a lattice, we must ensure that the security label of  $A$  is **not lower than or equal to** the security label of  $B$ . In this specific case, we want to prevent information from flowing from **Patient Records (PR)** to **Prescriptions (PX)**.

**1. Security Classes (Labels)**

We define a set of labels representing the sensitivity of the data:

- $L_{PR}$ : The security class for Patient Records (High Sensitivity).
- $L_{PX}$ : The security class for Prescriptions (Lower Sensitivity relative to Records).

**2. The Lattice Structure**

We define a partial ordering  $\leq$  such that  $L_{PX} < L_{PR}$ .

- In this hierarchy, **Prescriptions** are at the bottom and **Patient Records** are at the top.
- The lattice is defined as  $(\{L_{PX}, L_{PR}\}, \leq)$ .

**3. Enforcing the Policy**

By applying the **Bell-LaPadula** properties, we prevent the flow:

- **Simple Security Property (No Read Up)**: A subject with clearance  $L_{PX}$  (like a Nurse) cannot read an object with label  $L_{PR}$  (Patient Records).
- **$\star$ -Property (No Write Down)**: A subject acting with clearance  $L_{PR}$  (like a Doctor) can read  $L_{PR}$ , but the system prevents them from writing that information into an object labeled  $L_{PX}$  (Prescriptions).

**Result:** This structure ensures that while a doctor can access both, the highly sensitive data in the **Patient Records** can never "leak" or be written into the **Prescription** files, which are accessible to a wider group (Nurses).

---

**Next Step:** Would you like to move to **part (ii)** to discuss which specific security model (Bell-LaPadula vs. Biba) is most appropriate for this medical scenario?

**User**

part (ii)

**Gemini**

To address part (ii), we need to select the most appropriate security model for the medical information system and sketch its implementation.

**Q3. b) ii. Choice of Security Model and Implementation**

**Most Appropriate Model:** The **Bell-LaPadula (BLP)** model.

**Why?**

The primary requirement of this policy is **confidentiality** and the prevention of unauthorized **information flow** (ensuring nurses learn nothing about patient records).

- **Confidentiality Focus:** BLP is specifically designed to prevent the leakage of sensitive information from a higher security level to a lower one.
- **Read/Write Restrictions:** It perfectly maps to the requirement: doctors (High Clearance) can read and write patient records, while nurses (Low Clearance) are restricted from "reading up" into those records but can interact with prescriptions.

**Sketch of the Security Model**

The model is built on two fundamental properties that govern how subjects (Doctors, Nurses) interact with objects (Records, Prescriptions).

**1. The Subjects and Objects**

- **Subjects:**
  - **Doctor ( $S_D$ ):** Clearance level = **High**.
  - **Nurse ( $S_N$ ):** Clearance level = **Low**.
- **Objects:**
  - **Patient Records ( $O_{Rec}$ ):** Classification = **High**.
  - **Prescriptions ( $O_{Pre}$ ):** Classification = **Low**.

**2. The Governing Rules**

- **Simple Security Property (No Read Up):**
  - A Nurse ( $S_N$ ) cannot read Patient Records ( $O_{Rec}$ ) because  $Low < High$ .
  - A Doctor ( $S_D$ ) **can** read both because their clearance is  $\geq$  the object classifications.
- **$\star$ -Property (No Write Down):**
  - A Doctor ( $S_D$ ) is prevented from writing information from a Patient Record into a Prescription file ( $O_{Pre}$ ).
  - This ensures that sensitive medical history cannot be accidentally or maliciously leaked into the prescription database where nurses could see it.

| Subject       | Object: Records (High) | Object: Prescriptions (Low) |
|---------------|------------------------|-----------------------------|
| Doctor (High) | Read / Write           | Read / Write*               |
| Nurse (Low)   | No Access              | Read / Write                |



Export to Sheets



*\*Note: In a strict BLP implementation, a High-level subject writing to a Low-level object is typically restricted to prevent leakage, as defined by the  $\star$ -Property.*

**Next Step:** Would you like to move to **part (iii)** to see how to augment this model to prevent a doctor from prescribing medication for herself?

User

part (iii)

Gemini

To prevent a doctor from prescribing medication for herself, you must augment the model by incorporating a **Dynamic Separation of Duty (DSD)** or a **Binding of Duty** constraint.

Q3. b) iii. Augmenting the Model to Prevent Prescription Abuse

The standard Bell-LaPadula model handles **who** can see **what** (Clearance), but it doesn't naturally handle **who** can perform **which specific action** on a **specific instance** of data (like their own record).

To prevent this abuse, we introduce the following mechanisms:

1. Object-Based Constraints (Owner-Target Conflict)

The system must be augmented with a rule that checks the identity of the **Subject (Doctor)** against the identity of the **Target (Patient)**.

- **The Rule:** A Subject *S* cannot execute a `Write` (Prescribe) operation on Object *O* if  $Identity(S) == Identity(O)$ .
- **Effect:** Even though the doctor has "High" clearance to write prescriptions, the system checks the metadata of the patient record. If the record belongs to the doctor herself, the action is blocked.

2. Two-Person Integrity (Dual Control)

For high-risk actions like prescriptions, you can implement a **Binding of Duty** constraint:

- A prescription is only valid if it is initiated by one authorized subject and **countersigned** by a different authorized subject.
- In this scenario, if a doctor attempts to prescribe for herself, she would still require a second, independent doctor to authorize the transaction, effectively preventing self-prescription.

3. Formal Requirement (Clark-Wilson Model Integration)

While Bell-LaPadula handles confidentiality, the **Clark-Wilson Integrity Model** is better suited for preventing fraud.

- **Separation of Duties:** It requires that any "Transformation Procedure" (like writing a prescription) must be performed by a user in a way that prevents them from completing a sensitive task entirely on their own.

Summary of Augmented Policy

| Policy Layer           | Mechanism        | Purpose  |
|------------------------|------------------|--|
| Confidentiality (BLP)  | Security Lattice | Prevents Nurses from seeing Records.                       |
| Integrity (Constraint) | Identity Check   | Prevents Doctor <i>X</i> from writing to Record <i>X</i> . |

| Policy Layer    | Mechanism          | Purpose   |
|-----------------|--------------------|---|
| Integrity (DSD) | Dual Authorization | Ensures a second person must verify the prescription. |



Export to Sheets



**Next Step:** Would you like to move on to the final part, **Question 3(c)**, to find the sequence of queries needed to bypass the database tracker and determine the size of Erika's loan?

**User**

Answer Q3) (c)

**Gemini**

To determine the size of Erika's loan in a database that only reveals results when **4 to 7 people** are covered, we can use a **Tracker Attack**. A tracker is a set of conditions ( $T$ ) that splits the database into a group that meets the requirement ( $4 \leq |T| \leq 7$ ) and a group that does not.

Since we are allowed **4 queries**, we will use a "General Tracker" approach to isolate Erika's specific value by calculating the difference between a group including her and a group excluding her.

**The Strategy**

We will define a tracker  $T$  that covers a valid number of people (e.g., all students in a specific department). Let's assume **DeptID 10** has 5 students, one of whom is Erika.

1. **Query 1: Get the sum for the tracker group (which includes Erika)**  $Q_1 =$   
`SELECT SUM(Loan) FROM Students WHERE DeptID = 10`  
  - *Result:* Sum of loans for 5 people (Valid, since  $5 \in [4, 7]$ ).
2. **Query 2: Get the count for the tracker group**  $Q_2 =$   
`SELECT COUNT(*) FROM Students WHERE DeptID = 10`  
  - *Result:* 5 (Valid).
3. **Query 3: Get the sum for the tracker group EXCLUDING Erika**  $Q_3 =$   
`SELECT SUM(Loan) FROM Students WHERE DeptID = 10 AND SNAME <> 'Erika'`  
  - *Result:* Sum of loans for 4 people (Valid, since  $4 \in [4, 7]$ ).
4. **Query 4: Calculate Erika's Loan**  $\text{Erika's Loan} = Q_1 - Q_3$   
  - By subtracting the sum of the 4 other people from the total sum of the 5 people, the remaining value is exactly Erika's loan amount.

**Why this works**

The "obvious" query for Erika alone failed because the count (1) was outside the allowed range of 4-7. However, by embedding Erika within a larger "tracker" set ( $T$ ) and then querying the set  $T - \{\text{Erika}\}$ , we ensure both queries fall within the **4-7 person safety window**, even though the difference between them reveals the restricted individual data.

**Next Step:** Since you are working on security analysis and LaTeX documents for your M-Tech CSIS program, would you like me to help format these SQL queries or the logic explanation into an **IEEEtran LaTeX** block for your lab report?