Perfect — since `departments` has `id`, `name`, and `total_members`, we can reflect that in the schema and adjust the design accordingly.

Here’s the **revised schema** with that extra field.

🧱 Final Database Design (with `departments.total_members`)

1. `departments`

Field	Type	Notes
id	INT (PK)	Unique department ID
name	VARCHAR	e.g., HR, Admin, IT
total_members	INT	Cached count of users in this dept

> ⚠ Note: `total_members` should be updated whenever a user is added, removed, or transferred to/from that department. You can manage this with:

- Triggers (at DB level)
- Or in application logic (service layer)

---

2. `users`

Field	Type	Notes
id	INT (PK)	
name	VARCHAR	
email	VARCHAR	Unique login
password	VARCHAR	Hashed password
role	ENUM	LEADERSHIP, TEAMLEAD, EXECUTIVE
department_id	INT (FK)	Foreign key to departments(id)
created_at	TIMESTAMP	

3. `rooms`

Field	Type	Notes
id	INT (PK)	
name	VARCHAR	
capacity	INT	Optional
location	VARCHAR	Optional

4. `meetings`

Field	Type	Notes
id	INT (PK)	Meeting ID
title	VARCHAR	
description	TEXT	Optional
start_time	DATETIME	
end_time	DATETIME	
room_id	INT (FK)	From rooms
host_id	INT (FK)	From users, only LEADERSHIP/TEAMLEAD
created_at	TIMESTAMP	

5. `meeting_attendees`

Field	Type	Notes
id	INT (PK)	
meeting_id	INT (FK)	
user_id	INT (FK)	

🔄 Example Workflow (to visualize logic)

1. **Leadership/TeamLead logs in**
2. Creates meeting → chooses time + room
   - Room conflict check runs:

     ```
     sqlCopyEditSELECT * FROM meetings WHERE room_id = ? AND (
       (start_time < ? AND end_time > ?) OR
       (start_time < ? AND end_time > ?) OR
       (? BETWEEN start_time AND end_time)
     )
     ```
3. Chooses attendees → all roles allowed
4. On user registration → increment `departments.total_members`
5. On user transfer/deletion → update member counts

🔄 Backend Suggestion (Handling `total_members`)

If you're using **Java/Spring Boot**, update `total_members` in:

- `UserService.createUser()`
- `UserService.deleteUser()`
- `UserService.updateUserDepartment()`

Same applies in Express.js: use service/controller layer logic.

Would you like an **ER Diagram** or **API endpoint plan** next?