Here's a conceptual overview of an **Asset Liability Management (ALM) System ER Diagram**, tailored for enterprise-level financial applications like the ones you've been exploring:

🧩 Key Entities and Relationships

Entity	Attributes	Relationships
Asset	asset_id (PK), name, type, value, acquisition_date, depreciation_rate	Linked to Asset Category, assigned to Branch, tracked in Transfer History
Liability	liability_id (PK), name, type, amount, due_date, interest_rate	Associated with Department or Branch
Branch	branch_id (PK), name, location	Manages Assets and Liabilities
Department	department_id (PK), name, head	Oversees Liabilities and Employee assignments
Employee	employee_id (PK), name, contact, department_id (FK), branch_id (FK)	Assigned Assets, involved in Transfers
Asset Assignment	assignment_id (PK), asset_id (FK), employee_id (FK), date_assigned, condition	Tracks which employee holds which asset
Transfer History	transfer_id (PK), asset_id (FK), from_employee_id (FK), to_employee_id (FK), date	Logs asset movement between employees
User	user_id (PK), username, role, contact	Encodes and manages system data
Asset Category	category_id (PK), name, description	Categorizes Assets

🔗 Relationship Highlights

- **One-to-Many**: A branch can manage many assets and liabilities.
- **Many-to-One**: Multiple employees can belong to one department.
- **Many-to-Many**: Assets can be reassigned across employees over time.
- **One-to-One**: Each asset belongs to one category.

🛠️ Design Notes

- Use **foreign keys** to enforce referential integrity (e.g., `employee_id` in `Asset Assignment`).
- Include **derived attributes** like `net_asset_value` or `liability_risk_score` for reporting.
- Consider **audit trails** for compliance—especially in `Transfer History`.