Group 3:

- Pham Cat Vu - 20194465
- Vu Quoc Viet - 20194464
- Pham Dinh Gia Dung - 20194428
- Pham Van Cuong - 20194421
- Pham Viet Thanh - 20194454
- Phung Quoc Viet - 20194463
- Nguyen Tuan Dung - 20194427

Exercise 1:

Functional dependencies:

- manager_id => manager_name | 1st
- warehouse_name => warehouse_address, manager_id | 2nd
- part_no, warehouse_name => inventory_date, qty_on_hand | 3rd
- part_no => supplier_name |4th
- delivery_no => delivery_date, delivery_destination | 5th
- delivery_no, part_no => delivery_qty | 6th

1. The relation is already in 1NF. (No calculated and non-atomic attributes)

Finding primary key using Amstrong's Axiom:

- warehouse_name,part_no,delivery_no => warehouse_address, manager_id, manager_name, inventory_date, qty_on_hand, supplier_name,delivery_date, delivery_destination, delivery_qty

The 2nd,3rd,4th,5th,6th fd is causing Partial Dependency
The 1st is causing Transitive Dependency

Decompose to 2NF:

WAREHOUSE (delivery-no, warehouse-name, part-no, manager-id, manager-name, warehouse-address, inventory-date, qty-on-hand, supplier-name, delivery-date, delivery_destination, delivery-qty)

=> WAREHOUSE(part-no, delivery-no, warehouse_name#, inventory-date, qty-on-hand, supplier-name, delivery-date, delivery_destination, delivery-qty)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)

=> WAREHOUSE(delivery-no, warehouse-name#, part-no#, inventory-date, qty-on-hand, delivery-date, delivery_destination,delivery-qty)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)
PART(part_no, supplier_name)

=> WAREHOUSE(delivery-no, part-no#, warehouse-name#, delivery-date, delivery_destination, delivery-qty)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)
PART(part_no, supplier_name)

=> WAREHOUSE(delivery-no, warehouse-name#, part-no#, delivery-date, delivery_destination)
DELIVERY_QTY(delivery_no#, part_no, delivery_qty)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)
PART(part_no, supplier_name)

=> WAREHOUSE(delivery-no, warehouse-name#, part-no#)
DELIVERY(delivery-no, delivery-date, delivery_destination)
DELIVERY_QTY(delivery_no#, part_no, delivery_qty)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)
PART(part_no, supplier_name)

Decompose to 3NF:

=> WAREHOUSE(delivery-no, warehouse-name#, part-no#)
DELIVERY(delivery-no, delivery-date, delivery_destination)
DELIVERY_QTY(delivery_no#, part_no, delivery_qty)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id, manager_name)
PART(part_no, supplier_name)

=> WAREHOUSE(delivery-no#, warehouse-name#, part-no#)
DELIVERY(delivery-no, delivery-date, delivery_destination)
DELIVERY_QTY(delivery_no#, part_no, delivery_qty)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE_NAME(warehouse_name, warehouse_address, manager_id#)
PART(part_no, supplier_name)
MANAGER(manager_id, manager_name)

Conclusion:

MANAGER(manager_id, manager_name)
PART(part_no, supplier_name)
PART_DETAIL(warehouse_name#, part_no#, inventory-date, qty-on-hand)
WAREHOUSE(warehouse_name, warehouse_address, manager_id#)
DELIVERY(delivery-no, delivery-date, delivery_destination)
DELIVERY_DETAIL(delivery_no#, part_no#, delivery_qty)

Very good

2. Diagram

Very good

Exercise 2:

1.
In domain table:
Title_id violates the pk constraint
=> remove TITLE_ID in DOMAIN and add DOMAIN_ID to TITLES table as FK

In history table:
One reader can borrow a book multiple times, so having reader_id and title_id as pk is not sufficient.
=> add pk constraint to date_of_borrowing

Very good

2.
These original tables are in 2nd NF because
There's no calculated, non-atomic values (1NF)
There's no partial dependency (2NF)
City ID => City_Name cause transitive dependency (violates 3NF)

Very good

3.Create another relation CITY (City ID , city name)

Very good

4.No, because then the author_ID attribute wouldn't be atomic

Create another Table:
AUTHOR_GROUP (AuthorID#, TitleID#)

Final schema:
author(author_id, first_name, last_name)
title(title_id, name, domain_id#)
author_group(author_id#, title_id#)
domain(domain_id, name)
reader(reader_id, first_name, last_name, city_id#)
city(city_id, city_name)
borrowing(borrowing_id, reader_id#, title_id#, date)
history(reader_id#, title_id#, date_of_borrowing, date_of_returning)

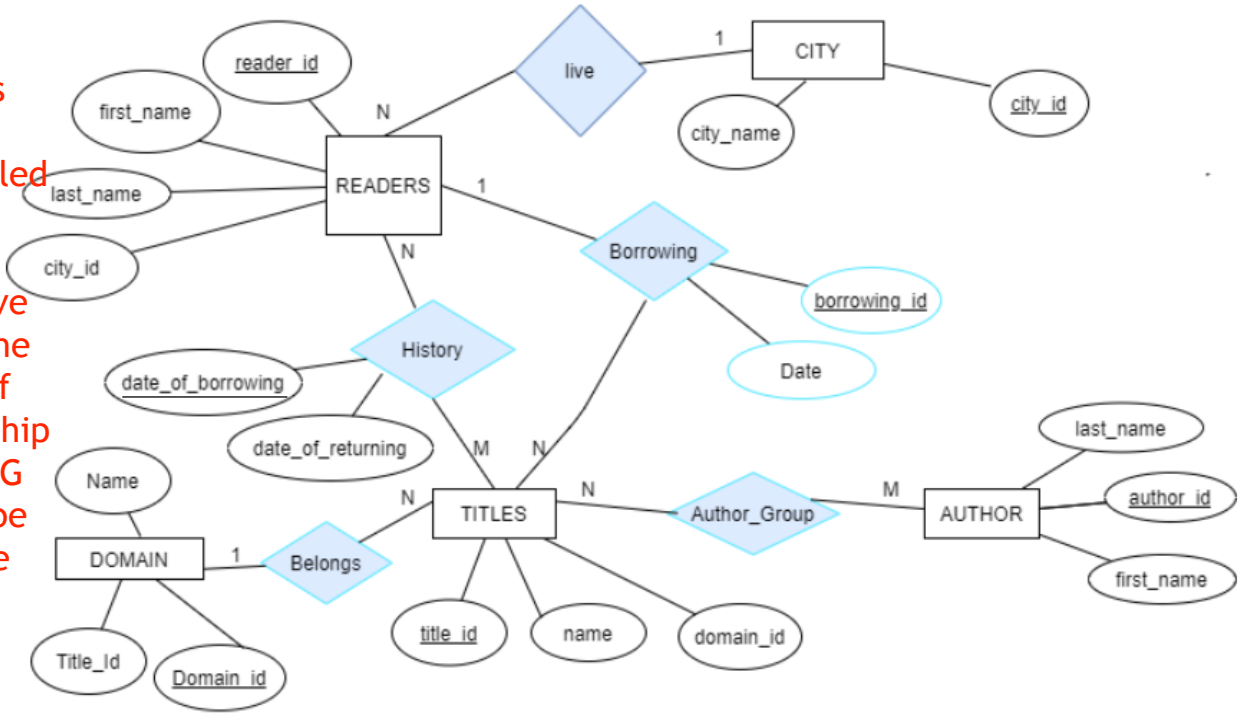
Very good

5.The first one, Because the 2nd one would create too much redundancy =>
Wasting storage space as the database contains duplicate information, slower query

The first one is indeed better, but your argument is unclear

6.

The entity DOMAIN does not have an attribute called Title_id any more…
Also, you have an error in the cardinality of the relationship « BORROWING »: it should be M on the side of READERS (more than 1 reader can borrow the same title).



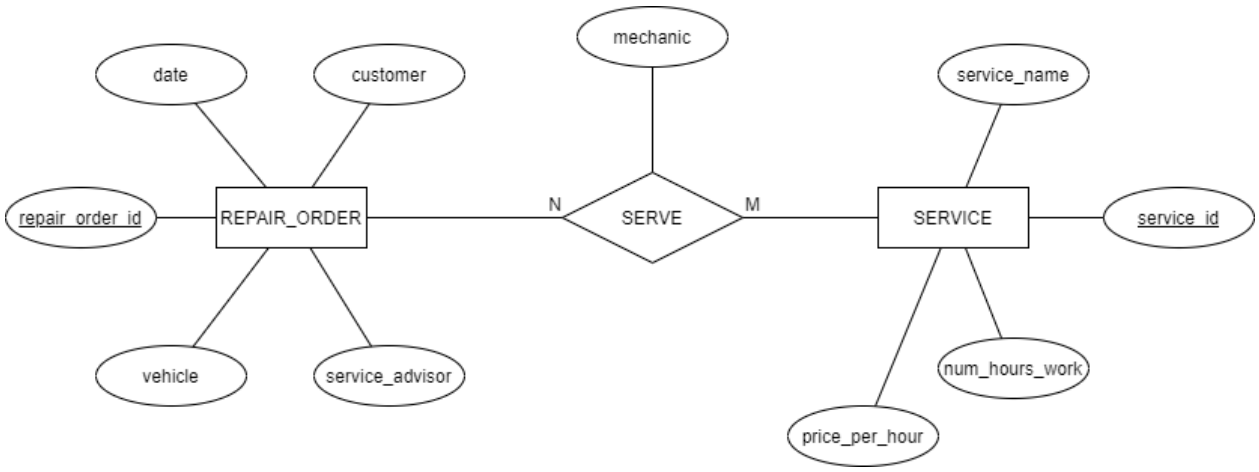
Exercise 3:

Functional dependencies:

- repair_order_id => customer, vehicle, date, service_advisor
- service_id => service_name, num_hours_work, price_per_hour
- repair_order_id, service_id => mechanic

1. Diagram:

Very good



2. **Relational schema:**

REPAIR_ORDER(repair_order_id, customer, vehicle, date, service_advisor)
SERVICE(service_id, service_name, num_hours_work, price_per_hour)
SERVICE_DETAIL(repair_order_id#, service_id#, mechanic)

Very good

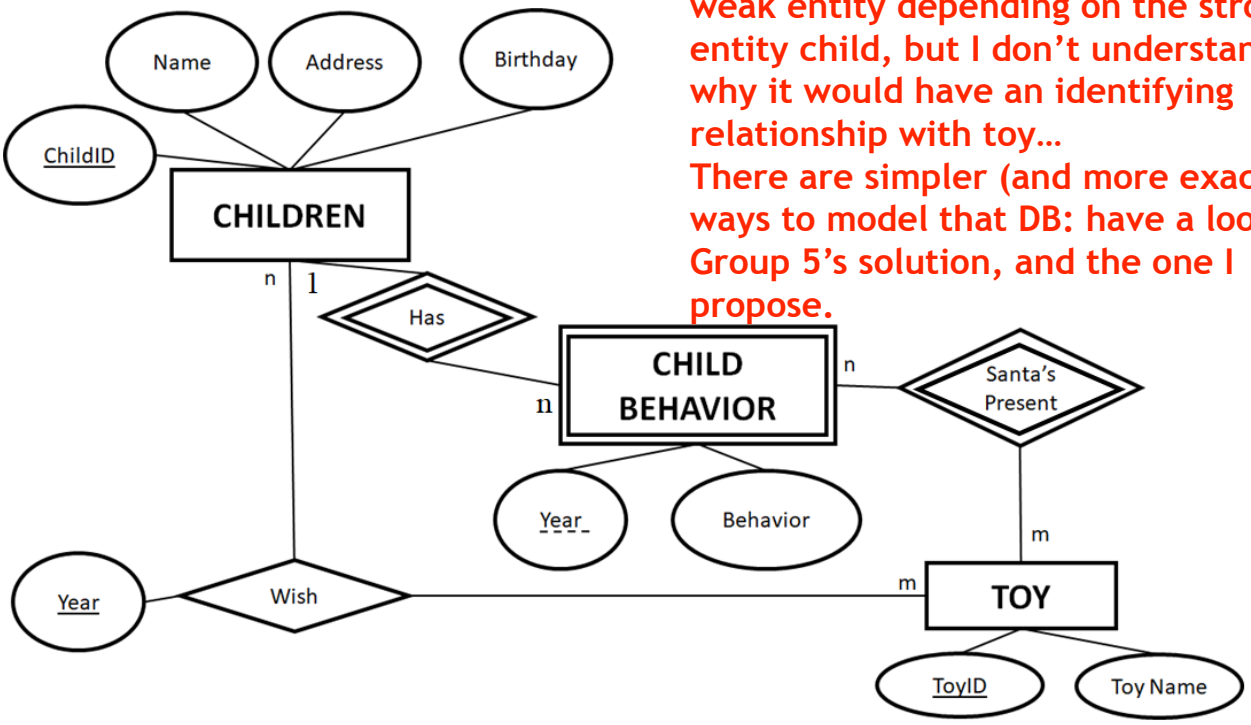
3. The schema is in 1NF (no calculated or non-atomic attribute)
   The schema is in 2NF (no partial dependency)
   The schema is also in 3NF (no transitive dependency)

Very good

Exercise 4:

It is a very interesting ER diagram, as you use weak entities. « Child behavior » can indeed be seen as a weak entity depending on the strong entity child, but I don't understand why it would have an identifying relationship with toy…
There are simpler (and more exact) ways to model that DB: have a look at Group 5's solution, and the one I propose.

1. ER diagram



2. CHILDREN(child_id, name, address, birthday)
   SANTAS_PRESENT(toy_id#, child_id#, year#)
   WISH(child_id#, toy_id#, year)
   CHILD_BEHAVIOR(child_id#, year, behavior)
   TOY(toy_id, toy_name)
3. Functional dependencies:
   - Child_id -> Name, address, birthday
   - Child_id, year -> behaviour
   - Toy_id -> Toy_name

This schema is in 1NF because there is no calculated, non-atomic attribute.

This schema is in 2NF because there is no partial dependency.
This schema is in 3NF because there is no transitive dependency.

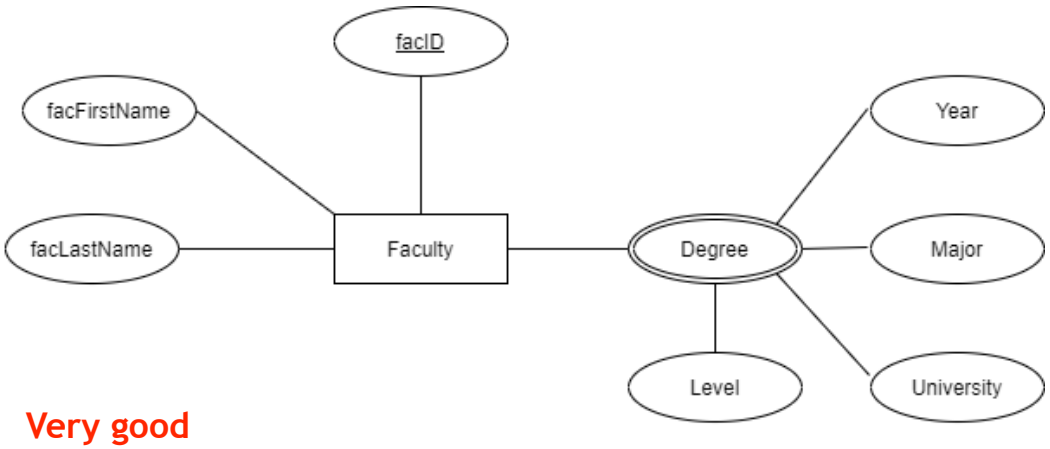
Exercise 5:

1. **Main drawbacks:**

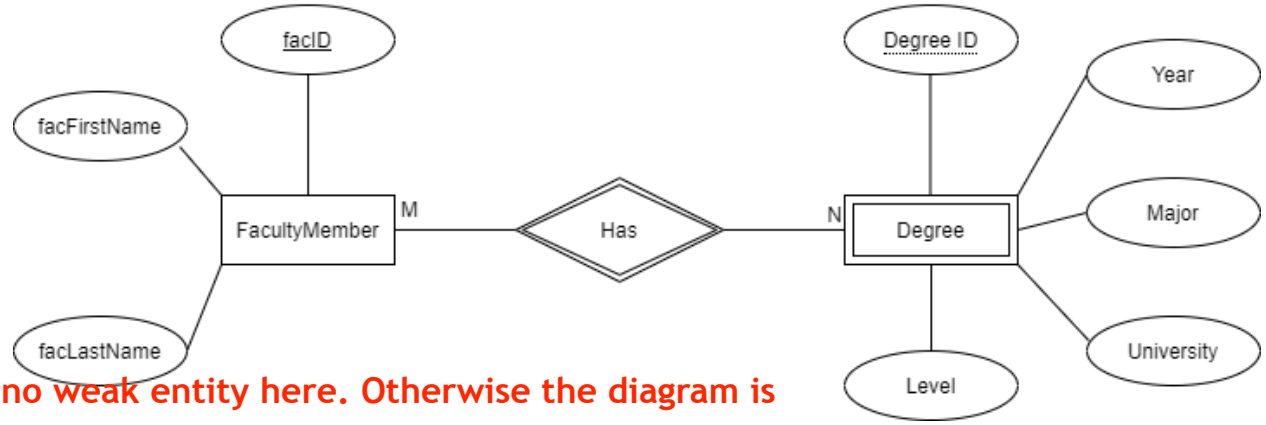
- This is memory-inefficient because records which contain null values also take up space.
- In reality, each member may have many degrees in the same level, while it is stated here that each member has no more than 3.
- Each degree is described by the same properties (level, year, major, university) but not in the same order. This can cause mismatch data when database users insert these records into the database.

Very good

2. **Diagram:**



Very good



There is no weak entity here. Otherwise the diagram is good.

3. The second diagram is best suited for storing GPA of the faculty members in order to save up storage space/improve query speed.

Very good.

4. FACULTY_MEMBER(faculty_member_id, first_name, last_name)
   DEGREE(degree_id, year, major, university, level)
   DEGREE_DETAIL(faculty_member_id#, degree_id#, GPA)

Very good.

5. Yes. Because there is no calculated or non-atomic attribute, no partial dependency, and no transitive dependency.

Very good.