Team 01 | class Data Science & AI

Team members:

1. Nguyen Minh Chau 20194420
2. Pham Quang Hieu 20194432
3. Nguyen Viet Hoang 20194434
4. Nguyen Duc Phu 20194447
5. Phan Duc Thang 20194452
6. Nguyen Vu Thien Trang 20194459
7. Phan Manh Tuan 20194461
8. Nguyen Van Thanh Tung 20190090

Exercise 1:

1. Modify this schema so that its relations are in 3NF.

warehouse(warehouse_name, warehouse_address, manager_id#)

warehouse-part(part_no#, warehouse_name#,inventory_date, qty-on-hand)

part(part_no, supplier_name)

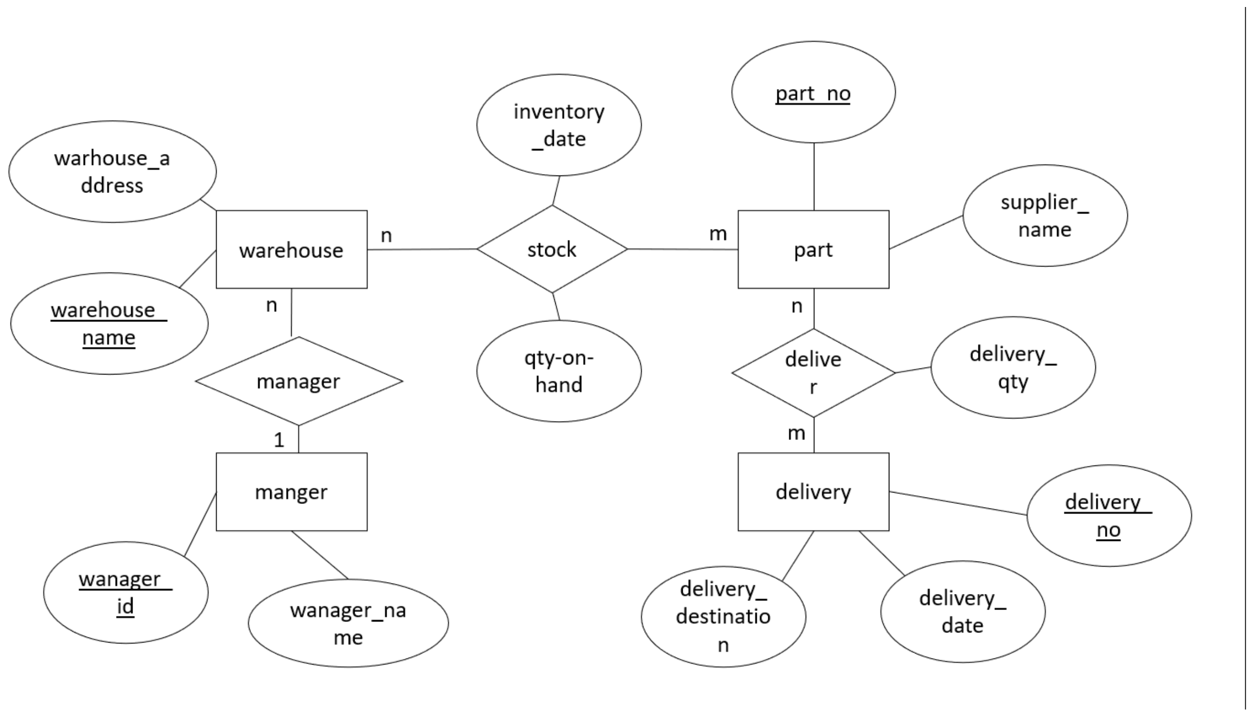
delivery(delivery_no, delivery_date, delivery_destination)

manager(manager_id, manager_name)

delivery_detail(delivery_no#, part_no#, delivery-qty)

Very good

2. Draw the ER diagram of the normalized schema (using UML notations).



Very good

Exercise 2:

AUTHOR(author_id, first_name, last_name)

TITLES(title_id, name, domain_id#)

AUTHOR-TITLE(author_id#, title_id#)

DOMAIN(domain_id, name)

READERS(reader_id, first_name, last_name, address, city_id#, phone)

CITY(city_id, city_name)

BORROWING(borrowing_id, reader_id#, title_id#, date)

HISTORY(reader_id#, title_id#, date_of_borrowing, date_of_returning)

1. Is there any inconsistencies between this schema and its description? If yes, fix the schema accordingly.

Each book corresponds to atmost one domain, hence we should remove the title_id in DOMAIN and add domain_id to TITLLES.

TITLE(title_id, name, domain_id#)

DOMAIN(domain_id, name)

The primary keys in HISTORY is not unique, we should add date_borrowing as another PK

HISTORY(reader_id#, title_id#, date_of_borrowing, date_of_returning)

Very good

2. Are these tables in 3rd Normal Form (3NF)? Why or why not?

In the READERS table, the city_name depends on city_id (which is a non-prime key). Hence, this table is not in 3NF.

Very good

3. If it was not in 3NF, then put it in 3NF

READERS(reader_id, first_name, last_name, address, city_id, city_name, phone)

→ READERS(reader_id, first_name, last_name, address, city_id#, phone), CITY(city_id, city_name)

Very good

4. If two authors work together on writing the same title, can we store that information in the above database? If not, then modify the schema so that it's possible to store more than 1 author per book.

Since now table AUTHOR and TITLES are in n...n relationship, we should add an table AUTHOR-TITLES

AUTHOR(author_id, first_name, last_name)

TITLES(title_id, name, domain_id#)

AUTHOR-TITLE(author_id#, title_id#)

Very good

5. I want to track the people who borrowed a book, so that, in case one book was damaged (pages torn for instance), I can contact the latest person who borrowed the book to check with them what happened. So, when a reader borrows a book, I make an entry in BORROWING table. After he returns the book, I created a trigger that automatically deletes that entry and I make another entry in the HISTORY table. Is this a good idea? Should I have instead one single BORROWING table with a DATE_OF_RETURNING column? Why or why not?

OK

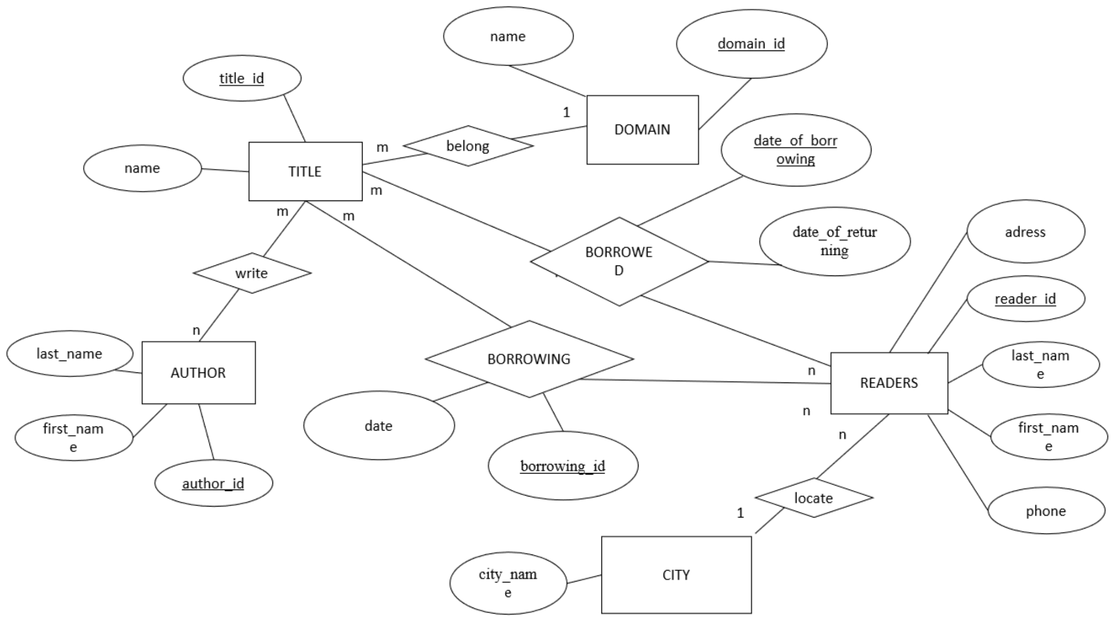
Keeping the two tables HISTORY and BORROWING and using a trigger is a good idea

But, this is not necessarily true ->

Because if we just use the table BORROWING with a DATE_OF_RETURNING, we have to update manually when a reader returns a book, whereas everythings is automatically done with triggers when we use the two tables HISTORY and BORROWING.

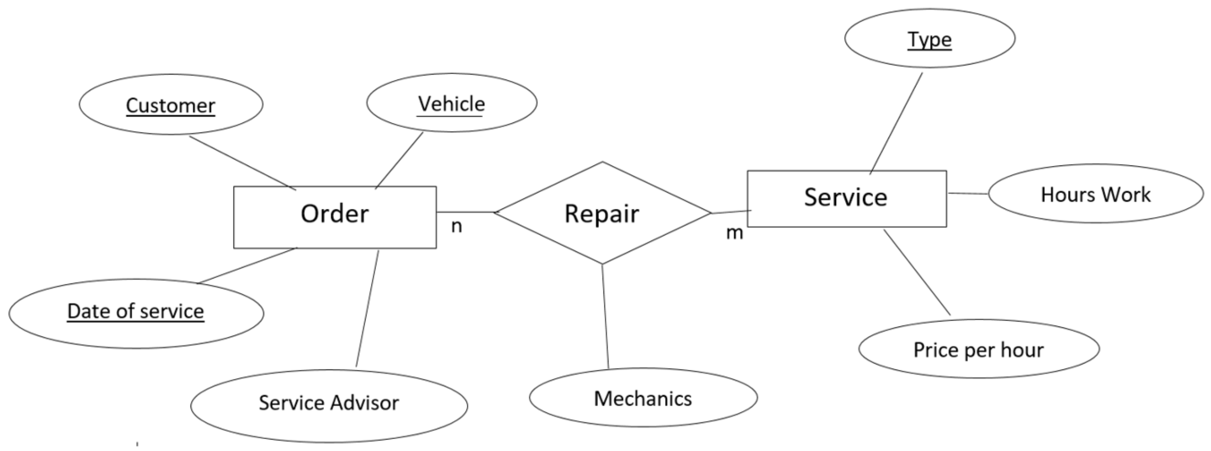
6. Draw the final ER diagram of the database.

Very good, knowing that the relationship « BORROWED » will be implemented in the database as the table « HISTORY »



Exercises 3:

1. **Give the ER diagram corresponding to this database, using Chen's notations**



Very good

Error in ORDER_DETAIL: in your schema, the three attributes customer, vehicle and date are a reference (foreign key) to ORDER. Therefore all these attributes should be followed by #.

2. Map it into a relational schema.

REPAIR_ORDER(customer, vehicle, date, advisor_name)

ORDER_DETAIL(customer, service#, vehicle, date, mechanic)

SERVICE(service, num_hrs, price_per_hour)

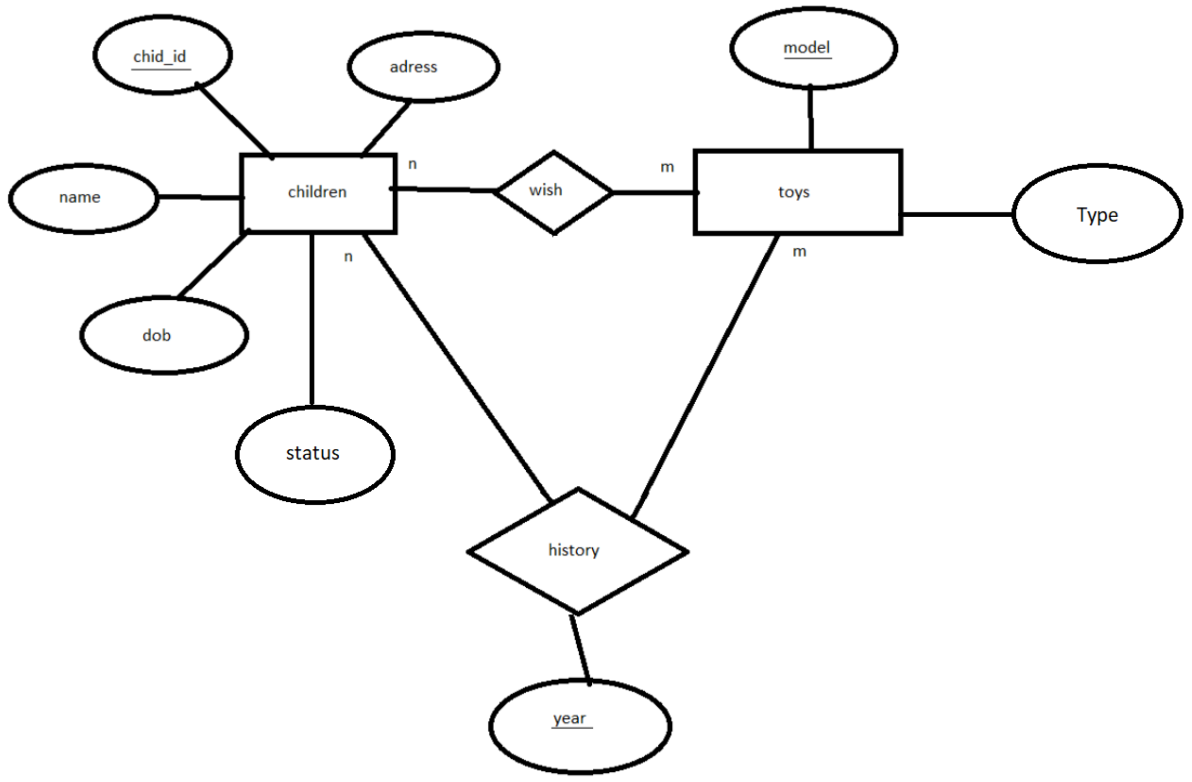
3. Verify that the relational schema you obtain is in 3NF

It is in 3NF OK

Exercise 4:

1. Give the ER diagram corresponding to this database, using UML notations

This model is OK. Its only drawback is that it does not keep a history of which toys the child wanted in past years (only which toys the kid got), and which year he/she has been nice or naughty. But this information was not required in the text so it's OK. Just have a look at Group 5's solution though, and the one I propose, both are slightly better because they keep a history.



2. Map it into a relational schema

OK

CHILD(<u>child_id</u>, name, address, dob, status)

TOY(<u>toy_id</u>, type, model)

WISHLIST(<u>child#, toy_id#</u>)

HISTORY(<u>child#, toy_id#, year</u>)

3. Verify that the relational schema you obtain is in 3NF

OK

This schema is in 3NF

Exercise 5:

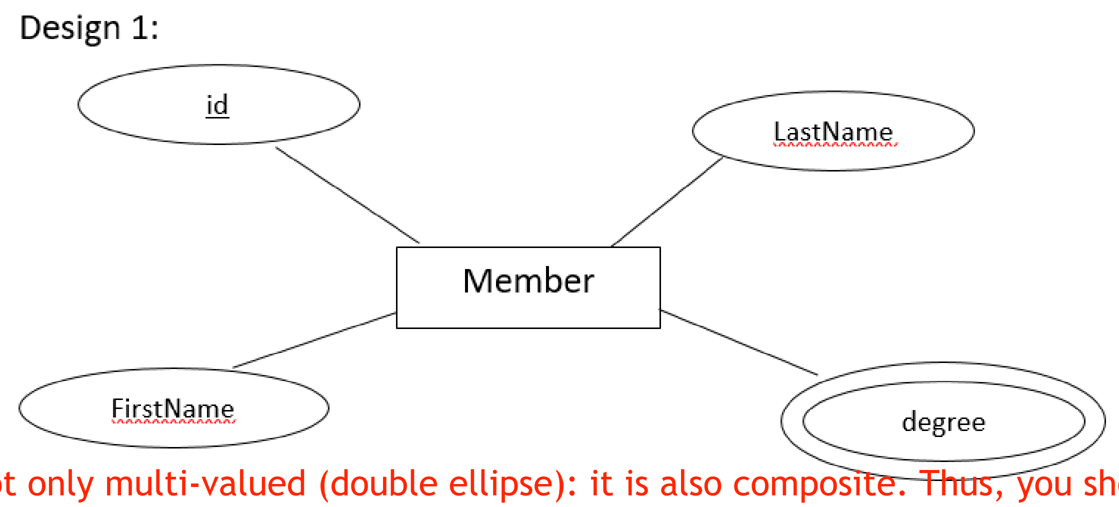
1. What are the main drawbacks of the above-mentioned database design?

There is too much wasted storage (many null values and the same type of degree is repeated multiple times).

True. And also,
- the attribute 'degree' is not simple: it is composite -> not in 1NF
- What if an employee has 4 degrees (for instance 1 Bachelor, 1 Master and 2 PhDs??)

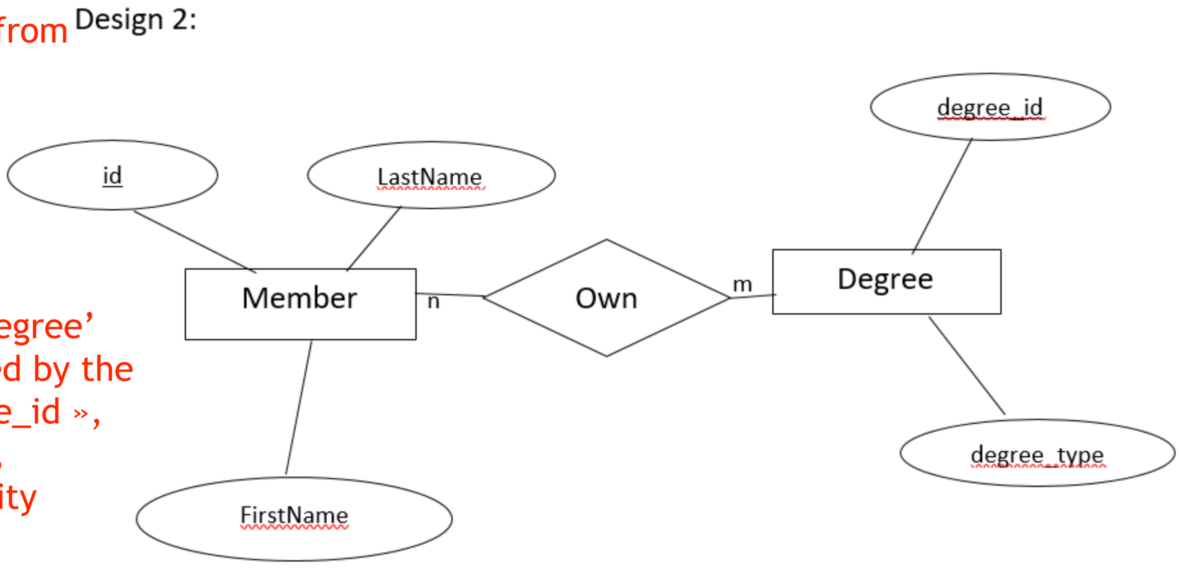
2. Give two ER diagrams which solve these drawbacks, using Chen's notations.

The first one:



Here degree is not only multi-valued (double ellipse): it is also composite. Thus, you should decompose the composite attribute 'degree' into the attributes « degree_id », « level », « year », « major », « university ».
See the correction from Group 2

The second one:



Here the entity 'degree' should be described by the attributes « degree_id », « level », « year », « major », university

3. Now, let's say that we want to store as well the grade (GPA) of the faculty members, for each degree. Which of the two above designs is the best for that purpose?

That's right -> Design 2 is better to store grade (by adding GPA attribute to the OWN relationship)

That's not right -> FACULTY_DEGREE(<u>fac_id#, degree_id</u>, year, major, university, GPA)

(year, major, univ. are attributes of DEGREE)

4. Map it into a relational schema (with the grade)

MEMBER(id, facFirstName, facLastName)

MEMBER_DEGREE(member_id#, degree_id#, year, major, university, GPA)

DEGREE(degree_id, degree_type)

Missing attributes for degree, too many attributes for MEMBER_DEGREE (see above)

5. Verify that the relational schema you obtain is in 3NF.

Schema is in 3NF

OK