EXERCISE

Section A

1. Map ER-Diagram -> Relational Schema

1. Regular Entity Type
    - Tạo 1 Relation (table) cho mỗi Entity.
    - Key attribute thành PK (gạch chân + để đầu tiên trong bảng cho dễ nhìn), nếu composite att là key thì gạch chân hết mấy con của nó.
    - Đưa attributes của Entity đó vào Rel, nếu att là composite att -> chỉ lấy con của att đó vào table rồi bỏ att đó (VD: composite "address" gồm street, city, zip, country => chỉ add street, city, zip, country vào Rel, bỏ address)
2. Weak Entity Type
    - Tạo 1 Relation (table) cho mỗi Entity.
    - FK = PK của parent entity ; partial key cũng là key (để đầu tiên trong bảng cho dễ nhìn)
    - Gạch chân FK này với partial key (2 cái này hợp lại thành PK của Rel)
    - Vẽ mũi tên từ FK chỉ vào PK của parent
    - Đưa attributes của Entity đó vào Rel
3. 1:1 Relationship Làm 1 trong 3 cách dưới đây, cái nào ở trên thì ưu tiên:
    - Cách 1: FK approach
        - Chọn 1 trong 2 Rel (tạm gọi E1, ưu tiên chọn E1 có total participation)
        - Lấy PK của E2 thêm vào bảng của E1 thành FK của E1 (không gạch chân, nhớ vẽ mũi tên, để cuối bảng cho dễn nhìn)
    - Cách 2: Merge Relation
        - Merge 2 Entities với Relation của nó thành 1 bảng luôn (nếu cả 2 đều total participation)
    - Cách 3: Cross reference/relationship relation
        - Tạo 1 Relation (table) mới.
        - Thêm PK của 2 Entities vào, gạch chân cả 2 thành composite PK của Rel, vẽ mũi tên FK.
        - Nếu Relationship có att của riêng nó thì add vô bảng.
4. 1:N Relationship
    - Gọi relation bên phía N là S, phía 1 là T
    - Lấy T[PK] làm S[FK] (không gạch chân, nhớ vẽ mũi tên)
    - Nếu Relationship có att của riêng nó thì add vô bên S
5. N:M Relationship
    - Tạo 1 Relation (table) mới.
    - Thêm PK của cả 2 Entities vào bảng (gạch chân, vẽ mũi tên)
    - Nếu Relationship có att của riêng nó thì add vô bảng
6. Multivalues attributes
    - Tạo 1 Relation (table) mới cho mỗi multivalue att A
    - Thêm PK của Entity chứa nó vào bảng (vẽ mũi tên), thêm nó (A) vào bảng luôn
    - Gạch chân cả 2
7. N-ary Relationship (nói thiệt chưa thấy thầy cho lần nào)
    - Tạo 1 Relation (table) mới
    - Thêm PK của tất cả Entities nối với cái Relationship đó vào (gạch chân + vẽ mũi tên)

    - Nếu Relationship có att của riêng nó thì add vô bảng

Section B

2. SQL Statement

2.1. Create, Delete, Modify TABLE

CREATE

- Create table:

```
CREATE TABLE table_name (
    <col_name_1> <col_type> <constraints>,
    <col_name_2> <col_type> <constraints>,
    ...,
    <col_name_n> <col_type> <constraints>
);
--- Example of constraint: NOT NULL , UNIQUE , ...
```

- Set primary key (3 ways)

```
CREATE TABLE CUSTOMERS (
    CUSTOMER_ID NUMBER(22) PRIMARY KEY, --- 1st, for atomic key without name
    FIRST_NAME VARCHAR2(10) NOT NULL,
    PRIMARY KEY (CUSTOMER_ID, FIRST_NAME), --- 2nd, for composite key without name
    CONSTRAINT PK_name PRIMARY KEY (CUSTOMER_ID, FIRST_NAME) --- 3rd, set name for primary key
);
```

- Set foreign key (2 ways) + enforce referential integrity constraints (2 ways)

```
CREATE TABLE CUSTOMERS (
    CUSTOMER_ID NUMBER(22) PRIMARY KEY,
    DEPARTMENT_ID NUMBER(22) NOT NULL,
    FOREIGN KEY DEPARTMENT_ID REFERENCES DEPARTMENT(DEPARTMENT_ID), --- 1st, for composite key without name
    CONSTRAINT FK_name FOREIGN KEY DEPARTMENT_ID REFERENCES DEPARTMENT(DEPARTMENT_ID), --- 2nd, set name for foreign key
    FOREIGN KEY DEPARTMENT_ID
        REFERENCES DEPARTMENT(DEPARTMENT_ID)
        ON DELETE SET NULL, --- If references is deleted, set null
    FOREIGN KEY DEPARTMENT_ID
        REFERENCES DEPARTMENT(DEPARTMENT_ID)
        ON DELETE CASCADE, --- If references is deleted, delete whole rows that store the deleted references
);
```

```

```

DELETE

```sql
DROP TABLE table_name;
```

MODIFY

- Add, delete constraints of TABLE:

```sql
--- Add
ALTER TABLE table_name
    ADD PRIMARY KEY (col_name);
ALTER TABLE table_name
    ADD CONSTRAINT PK_name PRIMARY KEY (col_name);
ALTER TABLE table_name
    ADD FOREIGN KEY (col_name) REFERENCES ref_table(ref_col);
ALTER TABLE table_name
    ADD CONSTRAINT FK_name FOREIGN KEY (col_name) REFERENCES ref_table(ref_col);
--- Delete
ALTER TABLE table_name
    DROP CONSTRAINT key_name;
```

- Add, delete, modify columns:

```sql
--- Add & Delete
ALTER TABLE table_name
    ADD column_name datatype constraints;
ALTER TABLE table_name
    DROP COLUMN column_name;
--- Rename
ALTER TABLE table_name
    RENAME COLUMN old_name to new_name;
--- Modify
ALTER TABLE table_name
    MODIFY COLUMN column_name datatype constraints;
```

2.2. Create, Delete, Modify ROWS

```sql
--- Insert row by row
INSERT INTO table_name --- Must insert all cols
    VALUES (val1, val2, ...);
INSERT INTO table_name(col1, col2, ..., coln)
    VALUES (val1, val2, ..., valn); --- Nhét đủ số cột mình ghi là được
--- Insert data from another table
```

```sql
INSERT INTO table_name
    SELECT * FROM other_table;
INSERT INTO table_name(col1, col2)
    SELECT other1, other2 FROM other_table;
--- Update rows
UPDATE table_name
    SET column1 = value1, column2 = value2, ...
    WHERE condition;
--- Drop rows
DELETE FROM table_name WHERE condition;
--- Drop all rows
    --- Solution 1: Faster, but cannot be rolled back
TRUNCATE TABLE table_name;
    --- Solution 2: Safer with referential integrity, can be rolled back
DELETE FROM table_name;
```

2.3. Queries

2.4. Transactions: COMMIT, ROLLBACK, SAVEPOINTS

```sql
COMMIT;
ROLLBACK;

SAVEPOINT savepoint_name;
RELEASE SAVEPOINT savepoint_name;
ROLLBACK TO savepoint_name;
```

- `COMMIT`: makes permanent change to DB.
- `ROLLBACK`: rollback to latest commit, double rollback doesn't do anything (rollback not behave like undo).
- `SAVEPOINT`: create savepoint
  - Can create as many savepoints as wish.
  - Can Rollback to any savepoint, as long as that savepoint hasn't been released (deleted).
  - All savepoints are released automatically after `COMMIT`.

---

3. Relational Algebra

Note: Relational algebra create a **new relation** (like queries), original relations are not modified.

Operation	Symbol Name	Symbol + Syntax	Equivalent SQL statement	Type	Note
select	sigma	$\sigma_{condition}(R)$	SELECT * FROM R WHERE condition	Unary	

Operation	Symbol Name	Symbol + Syntax	Equivalent SQL statement	Type	Note
project	pi	$\pi_{att1, att2, ...}(R)$	SELECT att1, att2, ... FROM R	Unary	
rename	rho	$\rho_{name(name1, name2, ...)}(R)$	SELECT * AS name1, name2, ... FROM R name	Unary	Có thể chỉ đổi tên table/cols
join	theta	$R \underset{join condition}{\Join} S$	SELECT * FROM R JOIN S ON join condition	Binary	R có n cols, S có m cols thì result table có n+m cols, giữ luôn duplicated cols
right outer join		R ⟖ S	RIGHT JOIN		
left outer join		R ⟕ S	LEFT JOIN		
full outer join		R ⟗ S	FULL JOIN		
division		$R(A, B) \div S(B)$		Binary	A = (all columns of R) \ B. Return rows of R if for every A, it contains all B in S.B *(Ex: Display professors who teach every course)*
union		$R \cup S$	UNION	from set theory	
intersection		$R \cap S$	INTERSECT	from set theory	
difference (or minus)		$R - S$	MINUS	from set theory	
cartesian product		$R \times S$		from set theory	Tưởng tượng như nhân vector thành matrix

Operation	Symbol Name	Symbol + Syntax	Equivalent SQL statement	Type	Note
aggregate function		$\mathcal{F}_{function(att)}(R)$			

Section C

4. Functional Dependencies & Normalization

Functional Dependency

- Định nghĩa dễ hiểu: B functional dependent on A nếu
  - att (hoặc set các atts) A có thể làm key cho att (các atts) B
  - Nếu 2 records của A giống nhau => correspond records của B cũng phải giống nhau
- Điều kiện để là candidate key:
  - Is SK: F^+ contains all atts of R
  - Is minimal
- Detect candidate key (tạm gọi set C)
  1. Viết tất cả các atts của R đó ra vào set S1 (candidate key 1 attribute).
     - Att nào không xuất hiện ở LHS của các FD -> gạch khỏi S1
     - Xét F^+ của từng att trong S1 -> là candidate key thì thêm vào C, không phải candidate key thì gạch
  2. Viết tất cả atts bị gạch trong S1 vào set Sn (candidate key n atts, 2 <= n <= tổng số att ban đầu trong S1)
     - Dùng combination lấy subset n att từ Sn
     - Xét F^+ của từng subset => thêm vào C nếu là candidate key (nhớ kiểm tra với candidate key trong C để xem có minimal ko)

Normalization

1. Categories: 1NF, 2NF, 3NF, BCNF
   - 1NF: Không có att nào là composite/multivalue
   - 2NF: Non-key atts phải depend on
     - **all candidate keys** của relation
     - **all components** của mỗi candidate key *(remove partial dependency)*
   - 3NF: Non-key att phải **directly** dependent on candidate key *remove transitive dependency* For each FD X->A in R:
     1. X is SK , or
     2. A is part of any candidate key
   - BCNF: là 3NF nhưng bỏ điều kiện số 2
2. Normalized method Tách bảng cho tới khi nào thỏa điều kiện thì thôi.

---

5. Transaction Procesing

1. Table (để trình bày)
    - Bao nhiêu cái T thì vẽ bấy nhiêu cột
    - Viết operation trong S theo từng cột (**Note**: mỗi hàng chỉ viết 1 operation) *Ex: r1(x) -> viết r(x) vào cột T1 ; w3(y) -> viết w(y) vào cột T3*
2. Precedence Graph
    - Vẽ 1 node (như trong automata) cho mỗi T
    - Dò từng operation từ trên xuống dưới, for each operation A, check các operation B nếu:
        - B phía ở dưới A (trên A thì kệ)
        - B nằm ở khác cột với A
        - A và B cùng modify 1 variable *Ex: A r(x) - B w(x)* => Nếu A hoặc B là write operation w() *(conflict)* => Vẽ 1 edge từ node A chỉ đến node B
3. Equivalent Serial Schedule
    - Nếu Precedence Graph có cycle => Schedule is not serializable, No squivalent serial schedule
    - Không có cycle:
        - Kiểm tra từng node trong precedence graph
        - Node nào không có mũi tên nào chỉ vào nó => xóa nó & mũi tên gắn với nó khỏi precedence graph => Note lại nó
        - Loop bước trên tới khi hết node trong precedence graph => Dãy noted lại là equivalent serial schedule *(1 Schedule có thể có nhiều cái ESS)*

THEORIES