Comp23111 – Cwk1 - Database Design

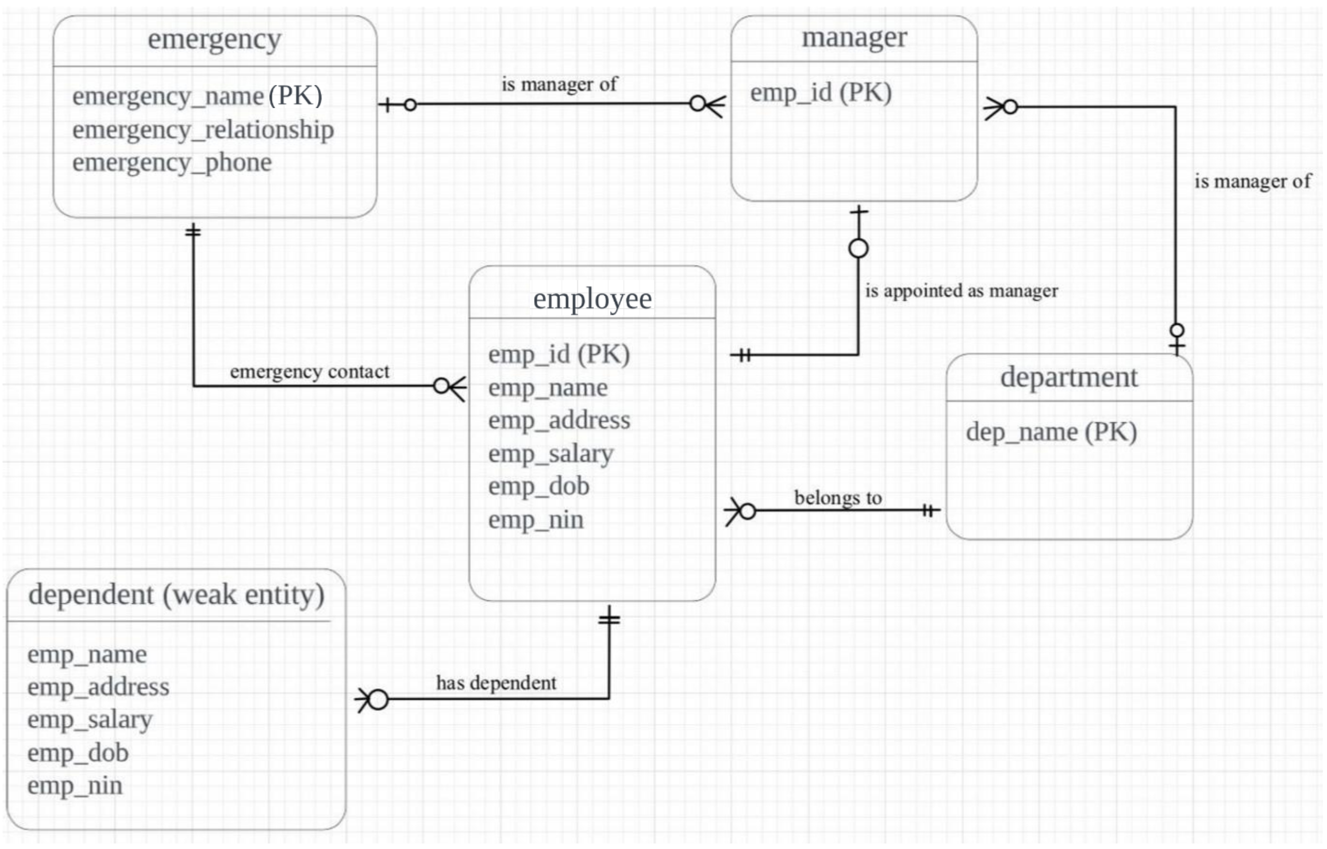
Yuwei Sang

11/11/2022

Contents

Part1: ERD

- For this section, I'll show my entity relationship diagram (ERD) which is in Crow's Foot notation. It describes the relationship between the information of employees of COMP23111 company "Kilburnazon". Here is my ERD.



- Explain my thoughts

  Our subject is the employee, and the information we use has the employee as the starting point. All the information in the diagram is employee information, including id, name, address, salary, etc.

  In the middle, the emp_id in [employee] is primary Key which uses attributes to identify the entity. That is, if we know the emp_id, we know which employee it is. The dep_name in [department] and the emergency_name in [emergency] is the same role, that if we know dep_name and emergency_name, we know which department employees in and who we need contact in an emergency.

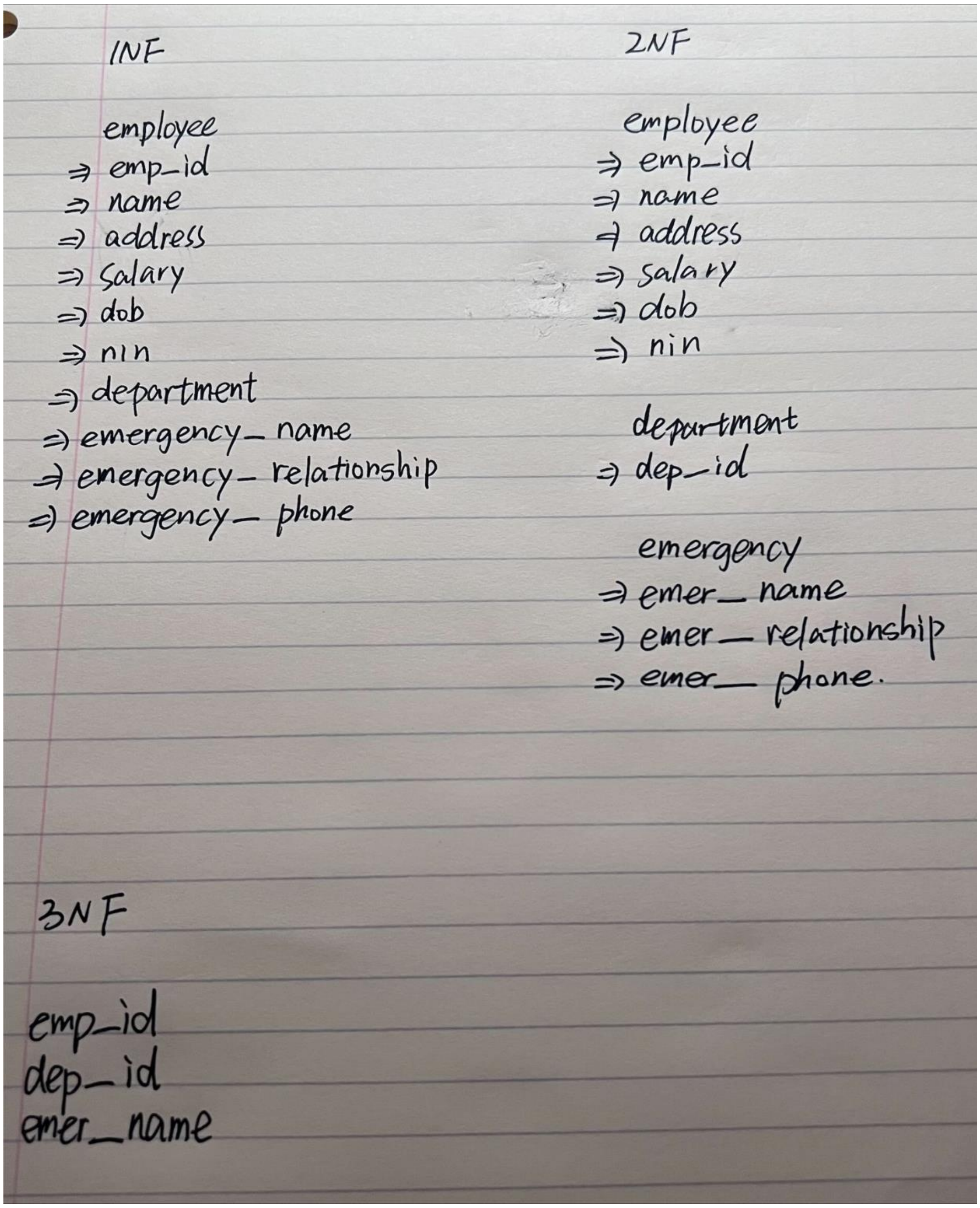
  When we know the emp_id, we can not only know the employee's name, but we can also know all other information about this employee, so the emp_id is like a manager. It manages everything. That’s why I set a [manager]. It is manager of [emergency] and [department].

  In addition to the emp_id, other types of information for employee are weak entity, so I set a [dependent].

  For the [emergency], it can be an independent whole, I set it up as a separate part. There is only one single emergency contact for each employee. It is same for [department].

Part2: Normalisation

- For this section, I'll show my 3NF. It is used for data normalization which is ordered to remove duplicate data and enhance data uniformity. Here is my 3NF.



1NF

employee
⇒ emp_id
⇒ name
⇒ address
⇒ Salary
⇒ dob
⇒ nin
⇒ department
⇒ emergency_ name
⇒ emergency_ relationship
⇒ emergency_ phone

2NF

employee
⇒ emp_id
⇒ name
⇒ address
⇒ salary
⇒ dob
⇒ nin

department
⇒ dep_id

emergency
⇒ emer_ name
⇒ emer_ relationship
⇒ emer_ phone.

3NF

emp_id
dep_id
emer_name

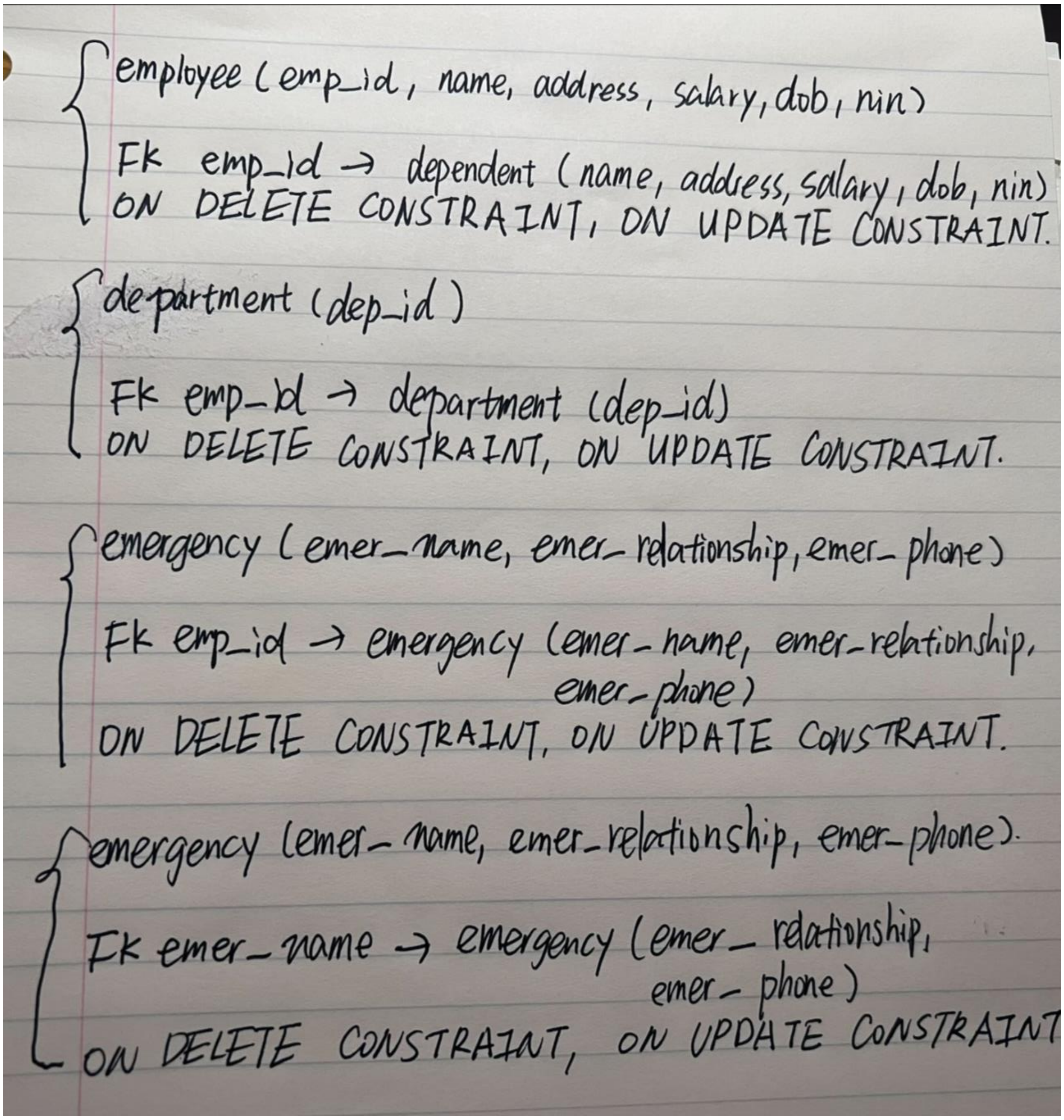
- For 1NF, each column in the data table must be the smallest unit that cannot be split, that is, ensure the atomicity of each column. That’s why I set all things together.

  For 2NF, after satisfying 1NF, it is required that all columns in the table, must depend on the primary key, and no column can have no relationship with the primary key, that is, a table describes only one thing. So, I separated the topic to 3 parts, each part just describes one topic.

  For 3NF, the second paradigm (2NF) must be satisfied first, requiring that: each column in the table is only directly related to the primary key and not indirectly related, (each column in the table can only depend on the primary key). For this, I just put the primary key (PK) together.

Part3: Relational schema

- For this section, I'll show my schema.



employee (emp_id, name, address, salary, dob, nin)

FK emp_id → dependent (name, address, salary, dob, nin)
ON DELETE CONSTRAINT, ON UPDATE CONSTRAINT.

department (dep_id)

FK emp_id → department (dep_id)
ON DELETE CONSTRAINT, ON UPDATE CONSTRAINT.

emergency (emer_name, emer_relationship, emer_phone)

FK emp_id → emergency (emer_name, emer_relationship, emer_phone)
ON DELETE CONSTRAINT, ON UPDATE CONSTRAINT.

emergency (emer_name, emer_relationship, emer_phone).

FK emer_name → emergency (emer_relationship, emer_phone)
ON DELETE CONSTRAINT, ON UPDATE CONSTRAINT

- When we know the emp_id we can know all the other information and make changes to it, delete it etc.

When we know the dep_id, we can know the corresponding department.
When I know the emer_name, we can know the relationship between the corresponding contact and the employee and get their phone number.
After we know the data, we can manage them.