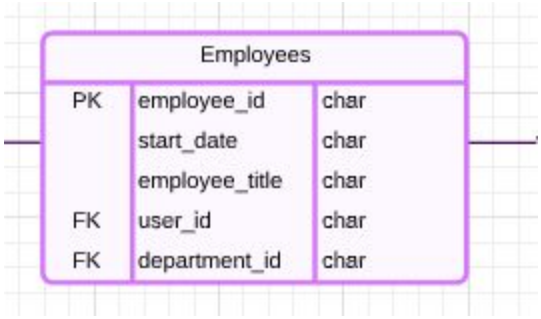
CPS 510 - Assignment 8

Bernstein's Algorithm - Broken down into 4 steps:

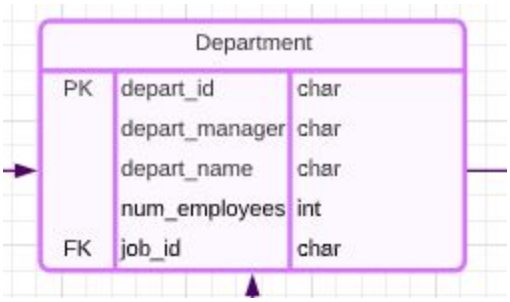
1) Determine all the functional dependencies
2) a) Find and remove redundancies
2) b) Find and remove partial dependencies
3) Find keys
4) Create tables

Step 1 (finding all functional dependencies)

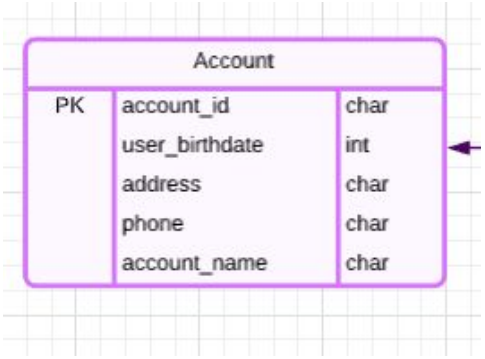
- **employee_id** -> {start_date, employee_title}



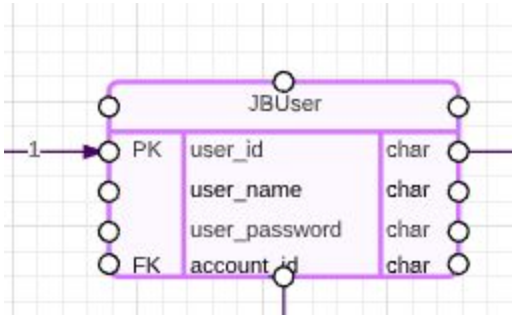
- **depart_id** -> {depart_manager, depart_name, num_employees}



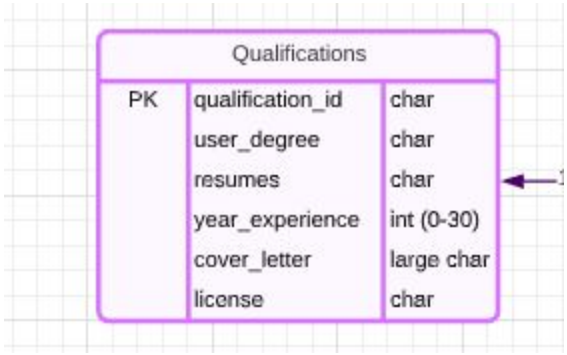
- **account_id** -> {user_birthdate, address, phone, account_name}



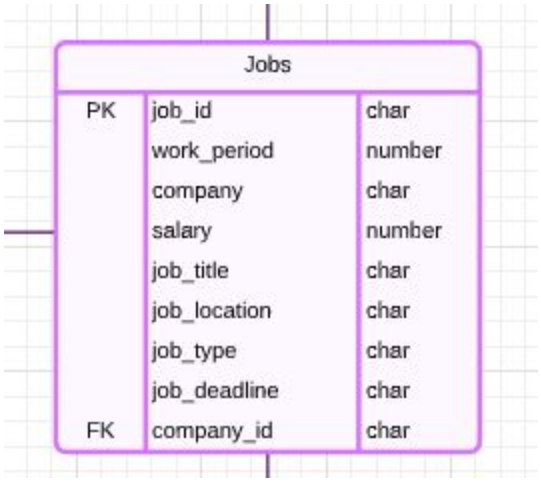
- **user_id** -> {user_name, user_password}



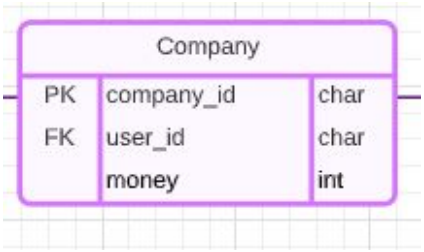
- **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}



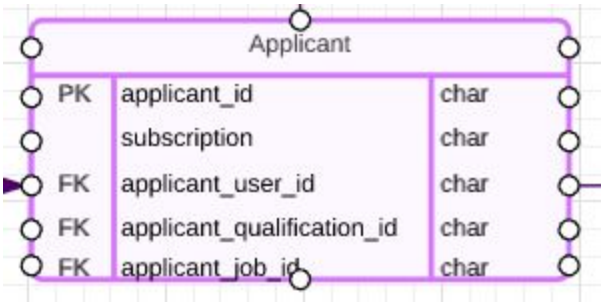
- **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline}



- **company_id** -> {money}



- **applicant_id** -> {subscription}



Step 2a (Break RHS and find redundancies)

Redundancies

- **employee_id** -> {start_date, employee_title, user_id}
    - Reduced list of FD's:
    - **employee_id** -> {start_date}
    - **employee_id** -> {employee_title}
    - **no redundancies**

- **depart_id** -> {depart_manager, depart_name, num_employees}
    - Reduced list of FD's:
    - **depart_id** -> {depart_manager}
    - **depart_id** -> {depart_name}
    - **depart_id** -> {num_employees}
    - **no redundancies**

- **account_id** -> {user_birthdate, address, phone, account_name}
    - Reduced list of FD's:
    - **Account_id** -> {user_birthdate}
    - **Account_id** -> {address, phone}
    - **Account_id** -> {phone}
    - **Account_id** -> {account_name}

- **no redundancies**

- **user_id** -> {user_name, user_password}
  - Reduced list of FD's:
  - **user_id** -> {user_name}
  - **user_id** -> {user_password}
  - **no redundancies**

- **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}
  - Reduced list of FD's:
  - **qualification_id** -> {user_degree}
  - **qualification_id** ->{resume}
  - **qualification_id** ->{years_experience}
  - **qualification_id** ->{cover_letter}
  - **qualification_id** ->{license}
  - **no redundancies**

- **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline}
  - Reduced list of FD's:
  - **job_id** ->{work_period}
  - **job_id** ->{company}
  - **job_id** ->{salary}
  - **job_id** ->{job_title}
  - **job_id** ->{job_location}
  - **job_id** ->{job_type}
  - **job_id** ->{job_deadline}
  - **no redundancies**

- **company_id** -> {money}
  - Reduced list of FD's:
  - **company_id** -> {money}
  - **no redundancies**

- **applicant_id** -> {subscription}
  - Reduced list of FD's:
  - **applicant_id** -> {subscription}
  - **no redundancies**

Step 2b (Minimize LHS, find and remove partial dependencies)

- LHS is already minimized, therefore there are no partial dependencies

Step 3(Find keys) (relational schema)

- **employee_id** -> {start_date, employee_title}
    - Attributes on RHS but not on LHS (cannot be keys)
        - start_date
        - employee_title
    - Possible Keys
        - employee_id
- **depart_id** -> {depart_manager, depart_name, num_employees}
    - Attributes on RHS but not on LHS (cannot be keys)
        - depart_manager
        - depart_name
        - num_employees
    - Possible Keys
        - depart_id
- **account_id** -> {user_birthdate, address, phone, account_name}
    - Attributes on RHS but not on LHS (cannot be keys)
        - user_birthdate
        - address
        - phone
        - account_name
    - Possible Keys
        - account_id
- **user_id** -> {user_name, user_password}
    - Attributes on RHS but not on LHS (cannot be keys)
        - user_name
        - user_password
    - Possible Keys
        - user_id
- **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}
    - Attributes on RHS but not on LHS (cannot be keys)
        - user_degree
        - resume
        - years_experience
        - cover_letter

        - license
    - Possible Keys
        - qualification_id
- **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline}
    - Attributes on RHS but not on LHS (cannot be keys)
        - work_period
        - company
        - salary
        - job_title
        - job_location
        - job_type
        - job_deadline
    - Possible Keys
        - job_id
- **company_id** -> {money}
    - Attributes on RHS but not on LHS (cannot be keys)
        - money
    - Possible Keys
        - company_id
- **applicant_id** -> {subscription}
    - Attributes on RHS but not on LHS (cannot be keys)
        - subscription
    - Possible Keys
        - applicant_id

Step 4(Make tables)

R1(employee_id, start_date, employee_title) with FD: **employee_id** -> {start_date, employee_title}

R2(depart_id, depart_manager, depart_name, num_employees) with FD: **depart_id** -> {depart_manager, depart_name, num_employees}

R3(account_id, user_birthdate, address, phone, account_name) with FD: **account_id** -> {user_birthdate, address, phone, account_name}

R4(user_id, user_name, user_password) with FD: **user_id** -> {user_name, user_password}

R5(qualification_id, user_degree, resume, years_experience, cover_letter, license) with FD: **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}

R6(job_id, work_period, company, salary, job_title, job_location, job_type, job_deadline) with FD: **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline}

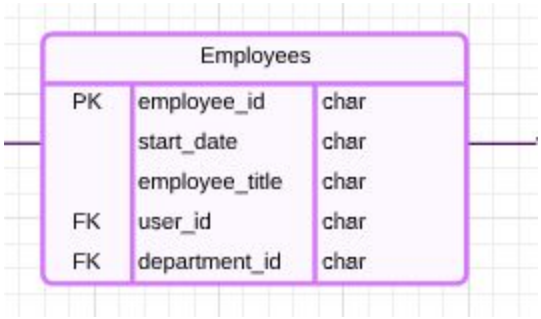
R7(company_id, money) with FD: **company_id** -> {money}

R8(application_id, subscription) with FD: **applicant_id** -> {subscription}

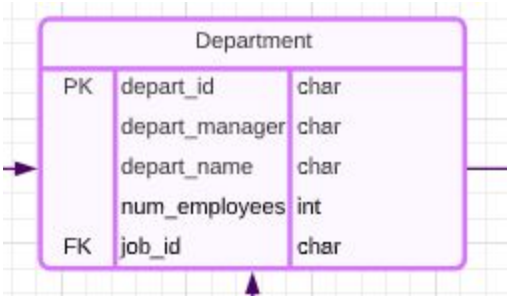
BCNF (Boyce Codd Normal Form)

Step 1 (finding all functional dependencies)

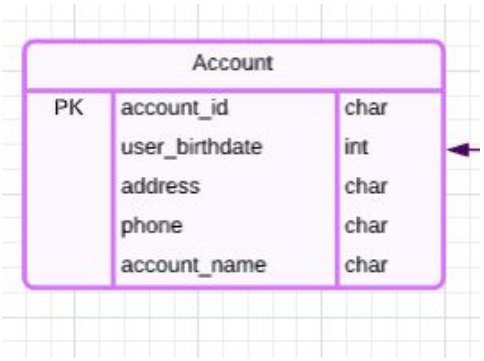
- **employee_id** -> {start_date, employee_title, user_id, department_id}



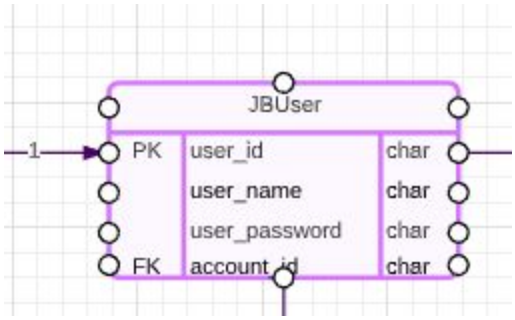
- **depart_id** -> {depart_manager, depart_name, num_employees, job_id}



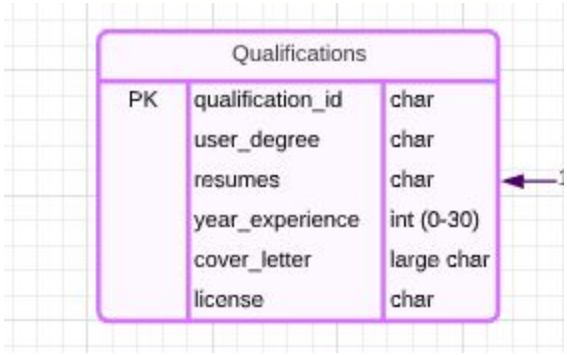
- **account_id** -> {user_birthdate, address, phone, account_name}



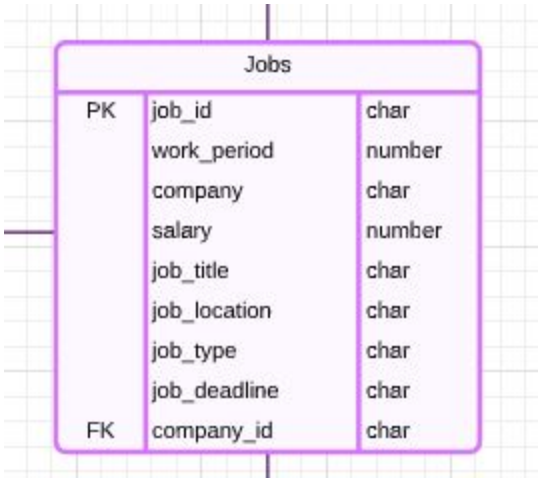
- **user_id** -> {user_name, user_password,account_id}



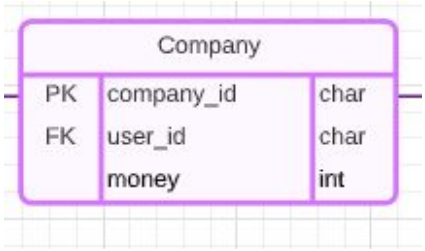
- **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}



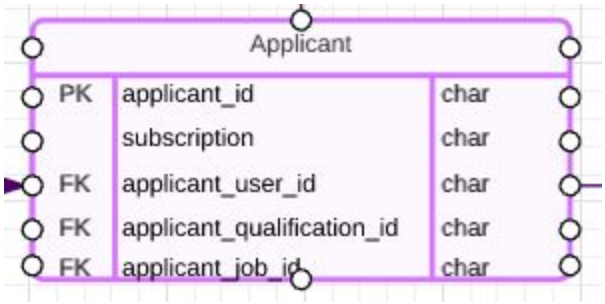
- **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline, company_id}



- **company_id** -> {money, user_id}



- **applicant_id** -> {subscription, applicant_user_id, applicant_qualifications_id, applicant_job_id}



Step 2 (make sure that the left hand side are keys if not decompose)

Consider the relation schema

- R1(employee_id,start_date,employee_title)

with FD

- **employee_id** -> {start_date, employee_title}

This schema has one candidate key

- employee_id

Therefore this schema is in BCNF

Consider the relation schema

- R2(depart_id, depart_manager, depart_name, num_employees)

with FD

- **depart_id** -> {depart_manager, depart_name, num_employees}

This schema has one candidate key

- depart_id

Therefore this schema is in BCNF

Consider the relation schema

- R3(account_id, user_birthdate, address, phone, account_name)

with FD

- **account_id** -> {user_birthdate, address, phone, account_name}

This schema has one candidate key

- account_id

Therefore this schema is in BCNF

Consider the relation schema

- R4(user_id, user_name, user_password)

with FD

- **user_id** -> {user_name, user_password}

This schema has one candidate key

- user_id

Therefore this schema is in BCNF

Consider the relation schema

- R5(qualification_id, user_degree, resume, years_experience, cover_letter, license)

with FD

- **qualification_id** -> {user_degree, resume, years_experience, cover_letter, license}

This schema has one candidate key

- qualification_id

Therefore this schema is in BCNF

Consider the relation schema

- R6(job_id, work_period, company, salary, job_title, job_location, job_type, job_deadline)

with FD

- **job_id** -> {work_period, company, salary, job_title, job_location, job_type, job_deadline}

This schema has one candidate key

- job_id

Therefore this schema is in BCNF

Consider the relation schema

- R7(company_id, money)

with FD

- **company_id** -> {money}

This schema has one candidate key

- company_id

Therefore this schema is in BCNF

Consider the relation schema

- R8(application_id, subscription)

with FD

- **applicant_id** -> {subscription}

This schema has one candidate key

- **applicant_id**

Therefore this schema is in BCNF

Step 3 (final BCNF schema for R)

R1(employee_id,start_date,employee_title)
R2(depart_id, depart_manager, depart_name, num_employees)
R3(account_id, user_birthdate, address, phone, account_name)
R4(user_id, user_name, user_password)
R5(qualification_id, user_degree, resume, years_experience, cover_letter, license)
R6(job_id, work_period, company, salary, job_title, job_location, job_type, job_deadline)
R7(company_id, money)
R8(application_id, subscription)

Creating BCNF with Functional Dependencies for at least one table:

Relationship Used: Employees and Department

Combined Table (with FD):

emp_id	emp_nationality	emp_name	emp_department	emp_departNum

1000	Chinese	Edmond	Purchasing	200
1000	Chinese	Edmond	HR	15
1001	Korean	Brian	R&D	45
1002	Indian	Ashwin	Technical Support	100

Functional Dependencies:

1) emp_id -> {emp_nationality, emp_name}
2) emp_department -> emp_departNum

Candidate key:

{emp_id, emp_department}

BCNF Conversion as Follows:

Table 1 (Employee Table):

emp_id	emp_nationality	emp_name
1000	Chinese	Edmond
1000	Chinese	Edmond
1001	Korean	Brian
1002	Indian	Ashwin

Table 2 (Department Table):

emp_dept	emp_dept_num
Purchasing	200
HR	15
R&D	45
Technical Support	100

Emp_dept_combined_mapping table:

emp_id	emp_departNum
1000	Purchasing
1000	HR
1001	R&D
1002	Technical Support

Functional Dependencies:

1) emp_id -> {emp_nationality, emp_name}
2) emp_department -> dept_num

Candidate Keys:

Table 1: emp_id

Table 2: emp_dept

Table 3: {emp_id, emp_dept}