**Lab 4 & 5**

Group Number: SSP4 Group 2

Group Member: Wu Sibing, Dou Maokang, Hu Wenqi, Liu Yanli, Xiao Yang

Assumption: The following schema is based on a school-wide database, which means there will be no circumstances such as staff in different schools having the same Staff\_ID.

**State(Name) Done**

* Key: Name
* Primary Key: Name
* FD: Name → Name
* The relation is in BCNF

**City(City\_Name, State\_Name) Done**

* Key: {City\_Name, State\_Name}
* Primary Key: {City\_Name, State\_Name}
* FD: City\_Name, State\_Name → City\_Name, Sate\_Name
* The relation is in BCNF

**Person(Person\_ID, Name, Schools, Phone, Email, Address, ZIP, City\_Name, State\_Name) Done**

* Key: Person\_ID
* Primary Key: Person\_ID
* FDs:
  + Person\_ID→ Name, School, Phone, Email, Address, ZIP, City\_Name, State\_Name
  + Address → ZIP, City\_Name, State\_Name (violates BCNF)
* Normalization:

1. R1(Address, ZIP, City\_Name, State\_Name)

Key: Address

Primary Key: Address

FD: Address → ZIP, City\_Name, State\_Name

The relation is in BCNF

1. R2(Address, Person\_ID, Name, School, Phone, Email)

Key: Person\_ID

Primary Key: Person\_ID

FD: Person\_ID → Address, Name, School, Phone, Email

The relation is in BCNF

**Staff(Person\_ID, Staff\_ID, Date\_Hired, Position) Done**

* Keys: Person\_ID; Staff\_ID
* Primary Key: Staff\_ID
* FDs:
  + Staff\_ID → Person\_ID, Date\_Hired, Position
  + Person\_ID → Staff\_ID, Date\_Hired, Position
* The relation is in BCNF

**Admin\_Staff(Person\_ID, Admin\_Staff\_ID) Done**

* Keys: Admin\_Staff\_ID
* Primary Key: Admin\_Staff\_ID
* FDs:
  + Admin\_Staff\_ID → Person\_ID
  + Person\_ID → Admin\_Staff\_ID
* The relation is in BCNF

**Technical\_Staff(Person\_ID, Technical\_Staff\_ID, Lab\_Name, Lab\_School) Done**

// Technical\_Staff -- assign → Laboratory; many to one with laboratory

* Keys: Technical\_Staff\_ID
* Primary Keys: Technical\_Staff\_ID
* FDs:
  + Person\_ID → Lab\_Name, Lab\_School
  + Person\_ID → Lab\_Location
  + Technical\_Staff\_ID → Lab\_Name, Lab\_School
  + Technical\_Staff\_ID → Lab\_Location
* The relation is in BCNF

**Laboratory(Lab\_Name, Lab\_School, Location) Done**

* Keys: {Lab\_Name, Lab\_School}
* Primary Key: {Lab\_Name, Lab\_School}
* FD: Lab\_Name, Lab\_School → Location
* The relation is in BCNF

**Equipment(Lab\_Name, Lab\_School, Equipment\_ID, Model\_No, Name, Date\_Purchased) Done**

* Keys: {Lab\_Name, Lab\_School, Equipment\_ID}
* Primary Keys: {Lab\_Name, Lab\_School, Equipment\_ID}
* FD: Lab\_Name, Lab\_School, Equipment\_ID → Model\_No, Name, Date\_Purchased

(Assume two pieces of equipment can have the same Model\_No)

* The relation is in BCNF

**Teaching Laboratory(Lab\_Name, Lab\_School)** // many-to-many with Undergraduate**Done**

* Keys: {Lab\_Name, Lab\_School}
* Primary Key: {Lab\_Name, Lab\_School}
* FD: Not applicable
* The relation is in BCNF

**Experiments(Teaching\_Lab\_Name, Teaching\_Lab\_School, Undergraduate\_Person\_ID, Undergraduate\_Student\_ID, Date, Attendance) Done**

* Key: {Teaching\_Lab\_Name, Teaching\_Lab\_School, Undergraduate\_Student\_ID, Date} (assume one student does not do experiments in different labs on the same date)
* Primary Key: {Teaching\_Lab\_Name, Teaching\_Lab\_School, Undergraduate\_Student\_ID}
* FD: Teaching\_Lab\_Name, Research\_Lab\_School, Undergraduate\_Person\_ID, Undergraduate\_Student\_ID, Date → Attendance

**Research Laboratory(Lab\_Name, Lab\_School) Done**

* Keys: {Lab\_Name, Lab\_School}
* Primary Key: {Lab\_Name, Lab\_School}
* FD: Not applicable
* The relation is in BCNF

**Assign(Graduate\_Person\_ID, Graduate\_Student\_ID, Lab\_Name, Lab\_School) Done**

（Note: Research Lab -- Assign -- Graduate）

* Keys: {Graduate\_Person\_ID, Lab\_Name, Lab\_School}, {Graduate\_Student\_ID, Lab\_Name, Lab\_School}
* Primary Key: {Graduate\_Student\_ID, Lab\_Name, Lab\_School}
* FD: Not applicable
* The relation is in BCNF

**Professor(Person\_ID, Fields\_of\_Expertise) Done**

* Key: Person\_ID
* Primary Key: Person\_ID
* FD: Professor\_Persion\_ID → Fields\_of\_Expertise
* The relation is in BCNF

**Timetable(Prof\_ Person\_ID, Date\_Time, Class) Done**

* Keys:{Prof\_ Person\_ID, Date\_Time, Class}
* Primary Key: {Prof\_ Person\_ID, Date\_Time, Class}
* FD: Not applicable; assume that a course can happen on many time slots
* The relation is in BCNF

**Course(ID, Name, Date, Prof\_Person\_ID) Done**

* Key: ID (by assumption)
* Primary Key: ID
* FDs:
  + Course\_ID → Name, Date, Prof\_Person\_ID (assume you cannot find two records with the same Course\_ID but different (prof, date) combinations)
  + Course\_ID → Prof\_Field\_of\_Expertise
* The relation is in BCNF

Note:

splitting Take to 2 binary relationships: Student -- take -- Course, Course -- teach → Professor

**Take(Person\_ID, Student\_ID, Course\_ID, Take\_Date) Done**

Key: {Student\_ID, Course\_ID}

Primary: {Student\_ID, Course\_ID}

FD:

Student\_ID, Course\_ID → Take\_Date

The relation is in BCNF

**Stakeholder(Person\_ID, Domain) Done**

* Keys: Person\_ID
* Primary Key: Person\_ID
* FD: Stakeholder\_Person\_ID → Domain
* The relation is in BCNF

**Comments/Suggestions(Stakeholder\_Person\_ID, Date, Topic) Done**

* Keys: {Stakeholder\_Person\_ID, Date, Topic} (assume the same person can comments on different topics on the same date)
* Primary Key: {Stakeholder\_Person\_ID, Date, Topic}
* FD: Stakeholder\_Person\_ID, Date, Topic → Stakeholder\_Person\_ID, Date, Topic
* The relation is in BCNF

**Student(Person\_ID, Student\_ID, Majors\_Minors, Admission\_Date, takecourse\_Prof\_Person\_ID) Done**

* Keys: Person\_ID; Student\_ID
* Primary Key: Student\_ID
* FDs:
  + Person\_ID → Majors\_Minors, Admission\_Date, takecourse\_Prof\_Person\_ID
  + Student\_ID → Majors\_Minors, Admission\_Date, takecourse\_Prof\_Person\_ID
* The relation is in BCNF

**Undergraduate(Person\_ID, Undergraduate\_Student\_ID) Done**

* Keys: Undergraduate\_Student\_ID
* Primary Key: Undergraduate\_Student\_ID
* FDs:
  + Person\_ID → Undergraduate\_Student\_ID
  + Undergraduate\_Student\_ID → Person\_ID
* The relation is in BCNF

**Graduate(Student\_Person\_ID, Graduate\_Student\_ID, Prof\_ Person\_ID, Supervise\_Topic) Done**

// Graduate -- supervise → Professor; many to one

* Keys: {Graduate\_Student\_ID, Prof\_Person\_ID}
* Primary Keys: {Graduate\_Student\_ID, Prof\_Person\_ID}
* FDs:
  + Graduate\_Student\_ID, Prof\_ Person\_ID → Prof\_ Person\_ID, Supervise\_Topic
  + Graduate\_Student\_ID → Prof\_Field\_of\_Expertise
* The relation is in BCNF

Tips:

-- A child table references a Parent table's unique key as a foreign key,

-- Person(Person\_ID, Name, Schools, Phone, Email, Address, ZIP, City\_Name, State\_Name)

--CREATE TABLE Person (

-- Person\_ID int not null Primary Key,

-- Name varchar(255),

-- School varchar(255),

-- Phone varchar(255),

-- Email varchar(255),

-- Addresses varchar(255), -- to avoid datatye address

-- ZIP varchar(255),

-- City\_Name varchar(255),

-- State\_Name varchar(255),

--);

-- R1(Address, ZIP, City\_Name, State\_Name)

CREATE TABLE Person\_R1 (

Addresses varchar(255) NOT NULL PRIMARY KEY, -- to avoid datatye address

ZIP varchar(255),

City\_Name varchar(255),

State\_Name varchar(255),

);

-- R2(Address, Person\_ID, Name, School, Phone, Email)

CREATE TABLE Person (

Person\_ID int not null Primary Key,

Name varchar(255),

School varchar(255),

Phone varchar(255),

Email varchar(255),

Addresses varchar(255), -- to avoid datatye address

);

CREATE TABLE Laboratory (

Name varchar(255) not null,

School varchar(255) not null,

Location varchar(255),

CONSTRAINT KEY1 Primary Key(Name,School),

);

CREATE TABLE Equipment (

Name varchar(255),

Model\_No int,

Date\_purchased varchar(255),

ID int not null,

Lab\_Name varchar(255) not null,

Lab\_School varchar(255) not null,

CONSTRAINT KEY2 Primary Key(ID, Lab\_Name, Lab\_School),

CONSTRAINT KEY20 foreign key(Lab\_Name, Lab\_School) references Laboratory(Name, School),

);

CREATE TABLE Professor(

Person\_ID int not null primary key foreign key references Person(Person\_ID),

Fields\_of\_Expertise varchar(255),

);

-- Student(Person\_ID, Student\_ID, Majors\_Minors, Admission\_Date, takecourse\_Prof\_Person\_ID)

CREATE TABLE Student (

Person\_ID int NOT NULL FOREIGN KEY REFERENCES Person(Person\_ID),

Student\_ID int NOT NULL PRIMARY KEY,

Majors\_Minors varchar(400),

Admissin\_Date date,

takecourse\_Prof\_Person\_ID int FOREIGN KEY REFERENCES Professor(Person\_ID),

);

-- Undergraduate(Person\_ID, Undergraduate\_Student\_ID)

CREATE TABLE Undergraduate (

Person\_ID int FOREIGN KEY REFERENCES Person(Person\_ID),

Undergraduate\_Student\_ID int NOT NULL PRIMARY KEY FOREIGN KEY REFERENCES Student(Student\_ID),

);

-- Graduate(Student\_Person\_ID, Graduate\_Student\_ID, Prof\_ Person\_ID, Supervise\_Topic)

CREATE TABLE Graduate (

Person\_ID int FOREIGN KEY REFERENCES Person(Person\_ID),

Graduate\_Student\_ID int NOT NULL PRIMARY KEY FOREIGN KEY REFERENCES Student(Student\_ID),

Prof\_Person\_ID int FOREIGN KEY REFERENCES Professor(Person\_ID),

Supervise\_Topic varchar(400),

);

CREATE TABLE Experiments (

Teaching\_Lab\_Name varchar(255) not null,

Teaching\_Lab\_School varchar(255) not null,

Undergraduate\_Person\_ID int not null foreign key references Person (Person\_ID),

Undergraduate\_Student\_ID int not null foreign key references Student (Student\_ID),

Dates date,

Attendance INT,

CONSTRAINT KEY3 Primary Key(Teaching\_Lab\_Name, Teaching\_Lab\_School, Undergraduate\_Person\_ID,Undergraduate\_Student\_ID),

CONSTRAINT KEY30 foreign key(Teaching\_Lab\_Name, Teaching\_Lab\_School) references Laboratory(Name, School),

);

CREATE TABLE Teaching\_Laboratory (

Lab\_Name varchar(255) not null,

Lab\_School varchar(255) not null,

CONSTRAINT KEY40 foreign key(Lab\_Name, Lab\_School) references Laboratory(Name, School),

CONSTRAINT KEY4 Primary Key(Lab\_Name, Lab\_School),

);

CREATE TABLE Research\_Laboratory (

Lab\_Name varchar(255) not nulL,

Lab\_School varchar(255) not null,

CONSTRAINT KEY5 Primary Key(Lab\_Name, Lab\_School),

CONSTRAINT KEY50 foreign key(Lab\_Name, Lab\_School) references Laboratory(Name, School),

);

CREATE TABLE Assign (

Graduate\_Person\_ID int not null foreign key references Person (Person\_ID) ,

Graduate\_Student\_ID int not null foreign key references Student(Student\_ID) ,

Lab\_Name varchar(255) not null,

Lab\_School varchar(255) not null,

CONSTRAINT KEY6 Primary Key(Graduate\_Person\_ID, Graduate\_Student\_ID,Lab\_Name,Lab\_School),

CONSTRAINT KEY60 foreign key(Lab\_Name, Lab\_School) references Laboratory(Name, School),

);

CREATE TABLE Timetable (

Prof\_Person\_ID int not null foreign key references Person(Person\_ID),

Date\_Time date not null,

Class varchar(255) not null,

CONSTRAINT KEY7 Primary Key(Date\_Time, Class,Prof\_Person\_ID),

);

CREATE TABLE Course (

ID int not null PRIMARY KEY,

Name varchar(255),

Dates date,

Prof\_Person\_ID int,

);

-- Take(Person\_ID, Student\_ID, Course\_ID, Take\_Date)

CREATE TABLE Takes (

Person\_ID int not null FOREIGN KEY REFERENCES Person(Person\_ID),

Student\_ID int not null,

Course\_ID int not null,

Take\_Date date,

CONSTRAINT KEY8 Primary Key(Person\_ID, Student\_ID, Course\_ID),

);

-- State(Name)

CREATE TABLE States

(Name varchar(400) NOT NULL PRIMARY KEY

);

-- City(City\_Name, State\_Name)

CREATE TABLE City

(City\_Name varchar(400) NOT NULL,

State\_Name varchar(400) NOT NULL FOREIGN KEY REFERENCES States(Name),

CONSTRAINT Key9 PRIMARY KEY (State\_Name, City\_Name)

);

-- Staff(Person\_ID, Staff\_ID, Date\_Hired, Position)

CREATE TABLE Staff

(Person\_ID int FOREIGN KEY REFERENCES Person(Person\_ID),

Staff\_ID int NOT NULL PRIMARY KEY,

Date\_Hired date,

Position varchar(400)

);

-- A child table references a Parent table's unique key as a foreign key,

-- the column(s) in the child table that references should also have a unique key on it.

-- Admin\_Staff(Person\_ID, Admin\_Staff\_ID)

CREATE TABLE Admin\_Staff

(Person\_ID int FOREIGN KEY REFERENCES Person(Person\_ID),

Admin\_Staff\_ID int PRIMARY KEY

);

-- Technical\_Staff(Person\_ID, Technical\_Staff\_ID, Lab\_Name, Lab\_School)

CREATE TABLE Technical\_Staff

(Person\_ID int FOREIGN KEY REFERENCES Person(Person\_ID),

Technical\_Staff\_ID int NOT NULL PRIMARY KEY,

Lab\_Name varchar(400),

Lab\_School varchar(400)

);

-- Stakeholder(Person\_ID, Domain)

CREATE TABLE Stakeholder (

Person\_ID int NOT NULL PRIMARY KEY FOREIGN KEY REFERENCES Person(Person\_ID),

Domain varchar(400),

);

-- Comments/Suggestions(Stakeholder\_Person\_ID, Date, Topic)

CREATE TABLE Comments\_n\_Suggestions (

Stakeholder\_Person\_ID int NOT NULL FOREIGN KEY REFERENCES Person(Person\_ID),

Dates date NOT NULL,

Topic varchar(400) NOT NULL,

CONSTRAINT Key10 PRIMARY KEY (Stakeholder\_Person\_ID, Date, Topic)

);

Queries

1. Find all Stakeholders who belong to the “public” domain.

2. Find all Stakeholders who have provided at least five comments or suggestions.

3. Find Graduates who are supervised by more than one professor and assigned to more than one research laboratory.

4. Find all Professors who teach more than one courses in the semester.

5. List all the Equipment belonging to a particular Laboratory.

6. Find all Undergraduates who have not attended at least one laboratory experiments.

7. List all Graduates who are doing research and taking courses in the semester.

-- query

-- Q1: 1. Find all Stakeholders who belong to the “public” domain.

SELECT Name

FROM Person, Stakeholder

WHERE Person.Person\_ID = Stakeholder.Person\_ID AND Stakeholder.Domain = 'public';

-- Q2: 2. Find all Stakeholders who have provided at least five comments or suggestions.

SELECT Name

FROM Person AS P, Stakeholder AS S, Comments\_n\_Suggestions AS CNS

WHERE P.Person\_ID = S.Person\_ID AND S.Person\_ID = CNS.Stakeholder\_Person\_ID

GROUP BY S.Person\_ID

HAVING COUNT(CNS.Topic) >= 5;

-- Q3. Find Graduates who are supervised by more than one professor and assigned to more than one research laboratory.

Method 1(subquery)

SELECT G1.Person\_ID

FROM Graduate AS G1, Person AS P1

WHERE P1.Person\_ID = G1.Person\_ID AND P1.Person\_ID IN (

SELECT A2.Graduate\_Person\_ID

FROM Person AS P2, Assign AS A2

WHERE P2.Person\_ID = A2.Graduate\_Person\_ID

GROUP BY A2.Graduate\_Person\_ID

HAVING COUNT(Lab\_Name) > 1

)

GROUP BY G1.Person\_ID

HAVING COUNT(G1.Prof\_Person\_ID) > 1;

Method 2(INTERSECT)

SELECT DISTINCT G.Graduate\_Student\_ID

FROM Graduate AS G

-- WHERE P.Person\_ID = G.Person\_ID

GROUP BY G.Graduate\_Student\_ID

HAVING COUNT(G.Prof\_Person\_ID) > 1

INTERSECT

SELECT DISTINCT G.Graduate\_Student\_ID

FROM Graduate AS G, Assign AS A

WHERE G.Person\_ID = A.Graduate\_Person\_ID

GROUP BY G.Graduate\_Student\_ID

HAVING COUNT(A.Lab\_Name) > 1;

-- Q4. Find all Professors who teach more than one courses in the semester.

SELECT Name

FROM Person

WHERE Person.Person\_ID IN (

SELECT T.Prof\_Person\_ID

FROM Timetable AS T, Professor AS P

WHERE T.Prof\_Person\_ID = P.Person\_ID

GROUP BY T.Prof\_Person\_ID

HAVING COUNT(\*) > 1

);

-- Q5. List all the Equipment belonging to a particular Laboratory.

-- Assume Lab Name = "Software Lab 2"

SELECT DISTINCT Name

FROM Equipment AS E

WHERE E.Lab\_Name = 'Software Lab 2';

-- Q6. Find all Undergraduates who have not attended at least one laboratory experiments.

SELECT Name

FROM Person AS P, Experiments AS E

WHERE P.Person\_ID = E.Undergraduate\_Person\_ID AND E.Attendance = 0;

-- Q7. List all Graduates who are doing research and taking courses in the semester.

Method 1

SELECT DISTINCT P.Person\_ID, Name

FROM Person AS P, Graduate AS G, Assign as A, Takes as T

WHERE P.Person\_ID = G.Person\_ID AND

G.Person\_ID = A.Graduate\_Person\_ID AND

P.Person\_ID = T.Person\_ID;

Method 2

SELECT DISTINCT P.Person\_ID, Name

FROM Person AS P, Graduate AS G, Assign AS A

WHERE P.Person\_ID = G.Person\_ID AND A.Graduate\_Person\_ID = P.Person\_ID;

INTERSECT

SELECT DISTINCT P.Person\_ID, Name

FROM Person AS P, Graduate AS G, Takes AS T

WHERE P.Person\_ID = T.Person\_ID;

每个表格15个，1 - 4 ddl：周一中午十二点前；5 & 1.5 ddl：周一晚上23:59前

【1】-- 吴思冰

Professor，Stakeholder，Comments\_n\_Suggestions

<https://docs.google.com/spreadsheets/d/1tMcldi08hceP8JoNEu6ZmQqL7bisxKcDAcnvReiNdiI/edit?usp=sharing>

【2】-- 窦茂康

Student，Undergraduate，Graduate，

<https://docs.google.com/spreadsheets/d/125uEC73ZkSL3qlk74H51fBSz0vzqAu2Nhbw4cq2ofvE>

【3】-- lili

Staff，Admin\_Staff，Technical\_Staff

<https://docs.google.com/spreadsheets/d/1NCfJMzGEqAcF01FYbw_EVnx93vgK-QzLoARQAjlqk3k/edit?usp=sharing>

【4】-- 胡文琦

Laboratory，Teaching\_Laboratory，Research\_Laboratory，Equipment

在这个表格里！

<https://drive.google.com/file/d/1f6YhVWWik_hDWh_A2YvzcDgwr2mAhojR/view?usp=sharing>

【5】-- 肖阳

Experiments，Assign，Timetable，Course，Takes

<https://docs.google.com/spreadsheets/d/107Nz5vgHXBsqqy2oEBKQFVrh6IwkJdqcYOU40YvOLo0/edit?usp=sharing>

【1.5】-- 吴思冰 （这个最后估摸着是需要大家的帮助，得等1 - 3全部写好我才能combine）

-- R1(Address, ZIP, City\_Name, State\_Name)

Person\_R1

-- R2(Address, Person\_ID, Name, School, Phone, Email)

Person

States，City

【Final Insertion】 -- 玉树临风英俊潇洒风流倜傥的豆豆

|  |  |
| --- | --- |
|  | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |