# 综合作业一

（1）查询所有供应商名称。

π 供应商名 (供应商)

（2）查询在“武汉”工作的职工所有信息。

σ 城市 = '武汉' (职工 ⨝ 仓库)

（3）查询与职工“范不闲”有业务往来的所有供应商，输出供应商名。

π 供应商名 (σ 姓名 = '范不闲' (职工) ⨝ 订购单 ⨝ 供应商)

（4）查询目前没有业务的职工，输出职工号。

π 职工号 (职工) - π 职工号 (订购单)

（5）查询和所有供应商都有业务的职工，输出职工姓名和工资。 这个问题有点复杂，因为它要求找出和每个供应商都至少有一笔交易的职工。这可以通过以下方式来实现：

π 姓名, 工资 (职工) ⋈ (

π 职工号 (订购单) ÷ π 供应商号 (供应商)

)

（1）查询所有职工姓名。

SELECT 姓名 FROM 职工;

（2）查询在“武汉”工作的员工的平均工资。

SELECT AVG(工资) AS 平均工资

FROM 职工

JOIN 仓库 ON 职工.仓库号 = 仓库.仓库号

WHERE 仓库.城市 = '武汉';

（3）查询与职工“范不闲”有业务往来的所有供应商，输出供应商名。

SELECT DISTINCT 供应商.供应商名

FROM 供应商

JOIN 订购单 ON 供应商.供应商号 = 订购单.供应商号

JOIN 职工 ON 订购单.职工号 = 职工.职工号

WHERE 职工.姓名 = '范不闲';

（4）将在“儋州”工作的所有员工工资增加 10%。

UPDATE 职工

SET 工资 = 工资 \* 1.1

WHERE 仓库号 IN (

SELECT 仓库号

FROM 仓库

WHERE 城市 = '儋州'

);

（5）创建视图“各城市员工统计”，包括：城市、员工数量。

CREATE VIEW 各城市员工统计 AS

SELECT 仓库.城市, COUNT(\*) AS 员工数量

FROM 职工

JOIN 仓库 ON 职工.仓库号 = 仓库.仓库号

GROUP BY 仓库.城市;

（6）删除没有业务的职工数据。

DELETE FROM 职工

WHERE 职工号 NOT IN (

SELECT DISTINCT 职工号

FROM 订购单

);

# 综合作业二

（1）1、最终的最小函数依赖集为：

* A→C
* A→E
* CB→G
* BD→A

2、经过分析，候选码为 BD 或 BG。

3、R 最高属于第一范式 (1NF)。

4、**将此模型按照模式分解的要求分解为 3NF**

R1(B, D, A) - 包含 BD→A

* R2(A, C, E) - 包含 A→C 和 A→E
* R3(C, B, G) - 包含 CB→G

（2）1、候选码为 Q 和 SZ

2、由于存在 Q→W，这是一个非主属性对候选码的部分依赖，所以 R 不满足 2NF。因此，R 最高属于 1NF。

3、为了将 R 分解为 3NF 并保持函数依赖和无损连接性，我们可以应用第三范式分解算法，即 BCNF 分解或合成方法。这里我们将采用合成方法，确保每个非平凡的函数依赖 X → Y 都在一个关系中，并且 X 是该关系的超键。

根据 F，我们可以创建以下关系：

* R1(P, Q, X) - 包含 PQ→X
* R2(S, Z, Q, W) - 包含 SZ→QW 和 Q→W
* R3(Q, W, P) - 包含 QW→P

现在，R2 中仍然存在 Q→W 的部分依赖，所以我们需要进一步分解 R2：

* R2a(S, Z, Q) - 移除 W 后的结果
* R2b(Q, W) - 包含 Q→W

最终，我们得到了以下 3NF 关系模式：

* R1(P, Q, X)
* R2a(S, Z, Q)
* R2b(Q, W)
* R3(Q, W, P)

这个分解保持了原始的函数依赖，并且保证了无损连接性，因为 Q 是一个公共属性，它可以在需要时用来重新组合这些表。

# 综合作业三

（1）**企业数据库设计**

**（1）E-R 图模型**

对于企业的数据库，我们可以设计如下的 E-R 图：

* **单位** (Unit) 实体包含属性：单位名、电话。
* **职工** (Employee) 实体包含属性：职工号、姓名、性别。每个职工隶属于一个单位，并且在一个工程中工作。
* **工程** (Project) 实体包含属性：工程名、地点。每个工程有多个职工参加。
* **设备** (Equipment) 实体包含属性：设备号、设备名、产地。这些设备由供应商提供给各个工程。
* **供应商** (Supplier) 实体包含属性：姓名、电话。每个供应商可以为多个工程供应不同设备。

关系：

* **隶属** (Belongs\_to) 关系连接 **单位** 和 **职工**，表示职工隶属于一个单位。
* **参与** (Participates\_in) 关系连接 **职工** 和 **工程**，表示职工在一个工程中工作。
* **供应** (Supplies) 关系连接 **供应商** 和 **设备**，表示供应商为工程提供设备。此关系也涉及到 **工程** 实体，因为设备是供应给特定工程的。

**（2）转换为关系模式**

根据上述 E-R 模型，我们得到以下关系模式：

* **单位** (Unit)
  + 单位名 (UnitName, PK)
  + 电话 (Phone)
* **职工** (Employee)
  + 职工号 (EmployeeID, PK)
  + 姓名 (Name)
  + 性别 (Gender)
  + 单位名 (UnitName, FK)
* **工程** (Project)
  + 工程名 (ProjectName, PK)
  + 地点 (Location)
* **设备** (Equipment)
  + 设备号 (EquipmentID, PK)
  + 设备名 (EquipmentName)
  + 产地 (Origin)
* **供应商** (Supplier)
  + 供应商名 (SupplierName, PK)
  + 电话 (Phone)
* **参与** (Participates\_in)
  + 职工号 (EmployeeID, PK, FK)
  + 工程名 (ProjectName, PK, FK)
* **供应** (Supplies)
  + 供应商名 (SupplierName, PK, FK)
  + 设备号 (EquipmentID, PK, FK)
  + 工程名 (ProjectName, PK, FK)

**（3）SQL 创建表命令**

sql

深色版本

CREATE TABLE Unit (

UnitName VARCHAR(50) PRIMARY KEY,

Phone VARCHAR(15) NOT NULL

);

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Gender CHAR(1) CHECK (Gender IN ('M', 'F')),

UnitName VARCHAR(50),

FOREIGN KEY (UnitName) REFERENCES Unit(UnitName)

);

CREATE TABLE Project (

ProjectName VARCHAR(50) PRIMARY KEY,

Location VARCHAR(100) NOT NULL

);

CREATE TABLE Equipment (

EquipmentID INT PRIMARY KEY,

EquipmentName VARCHAR(50) NOT NULL,

Origin VARCHAR(50) NOT NULL

);

CREATE TABLE Supplier (

SupplierName VARCHAR(50) PRIMARY KEY,

Phone VARCHAR(15) NOT NULL

);

CREATE TABLE Participates\_in (

EmployeeID INT,

ProjectName VARCHAR(50),

PRIMARY KEY (EmployeeID, ProjectName),

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),

FOREIGN KEY (ProjectName) REFERENCES Project(ProjectName)

);

CREATE TABLE Supplies (

SupplierName VARCHAR(50),

EquipmentID INT,

ProjectName VARCHAR(50),

PRIMARY KEY (SupplierName, EquipmentID, ProjectName),

FOREIGN KEY (SupplierName) REFERENCES Supplier(SupplierName),

FOREIGN KEY (EquipmentID) REFERENCES Equipment(EquipmentID),

FOREIGN KEY (ProjectName) REFERENCES Project(ProjectName)

);

（2）**高校赛事管理系统设计**

**（1）E-R 图模型**

对于高校赛事管理系统的数据库，我们可以设计如下的 E-R 图：

* **学院** (College) 实体包含属性：学院编号、学院名称、学院人数。
* **运动员** (Athlete) 实体包含属性：运动员编号、姓名、性别、年龄。每个运动员隶属于一个学院。
* **项目** (Event) 实体包含属性：项目编号、项目名称、比赛场地。
* **成绩** (Result) 实体记录运动员在特定项目中的成绩和名次。

关系：

* **隶属** (Belongs\_to) 关系连接 **学院** 和 **运动员**，表示运动员属于一个学院。
* **参加** (Participates\_in) 关系连接 **运动员** 和 **项目**，表示运动员参加多个项目，而每个项目有多名运动员参加。这个关系还涉及到 **成绩** 实体，用于记录具体的成绩和名次。

**（2）转换为关系模式**

根据上述 E-R 模型，我们得到以下关系模式：

* **学院** (College)
  + 学院编号 (CollegeID, PK)
  + 学院名称 (CollegeName)
  + 学院人数 (NumberOfStudents)
* **运动员** (Athlete)
  + 运动员编号 (AthleteID, PK)
  + 姓名 (Name)
  + 性别 (Gender)
  + 年龄 (Age)
  + 学院编号 (CollegeID, FK)
* **项目** (Event)
  + 项目编号 (EventID, PK)
  + 项目名称 (EventName)
  + 比赛场地 (Venue)
* **参加** (Participates\_in)
  + 运动员编号 (AthleteID, PK, FK)
  + 项目编号 (EventID, PK, FK)
  + 名次 (Rank)
  + 成绩 (Score)

**（3）SQL 创建表命令**

sql

深色版本

CREATE TABLE College (

CollegeID INT PRIMARY KEY,

CollegeName VARCHAR(100) NOT NULL,

NumberOfStudents INT NOT NULL

);

CREATE TABLE Athlete (

AthleteID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Gender CHAR(1) CHECK (Gender IN ('M', 'F')),

Age INT NOT NULL,

CollegeID INT,

FOREIGN KEY (CollegeID) REFERENCES College(CollegeID)

);

CREATE TABLE Event (

EventID INT PRIMARY KEY,

EventName VARCHAR(100) NOT NULL,

Venue VARCHAR(100) NOT NULL

);

CREATE TABLE Participates\_in (

AthleteID INT,

EventID INT,

Rank INT,

Score DECIMAL(6,2),

PRIMARY KEY (AthleteID, EventID),

FOREIGN KEY (AthleteID) REFERENCES Athlete(AthleteID),

FOREIGN KEY (EventID) REFERENCES Event(EventID)

);