2024 年期中考试

Problem 1

1.1 超键、候选键、主键和外键

给出超键、候选键、主键和外键的定义.

Solution:

- **超键:** 能够唯一标识关系中每个元组的一个属性集.
- **候选键:** 小到不能再小的超键，即其任意真子集都不能构成关系的超键.
- **主键:** 数据库设计者选定的候选键，用于唯一标识关系中的元组.
- **外键:** 一个关系中的属性集，它引用了另一个关系的主键.

1.2 强实体集和弱实体集

给出强实体集和弱实体集的定义.

Solution:

- **强实体集:** 其实例的存在不依赖于任何其他实体类型的实例.
  强实体集拥有自己独立的主键，可以唯一性地标识它的每个实例.
- **弱实体集:** 其实例的存在依赖于其他实体类型的实例.
  弱实体集没有足够的属性来形成主键，因此它的主键通常包括它所依赖的强实体集的主键部分.

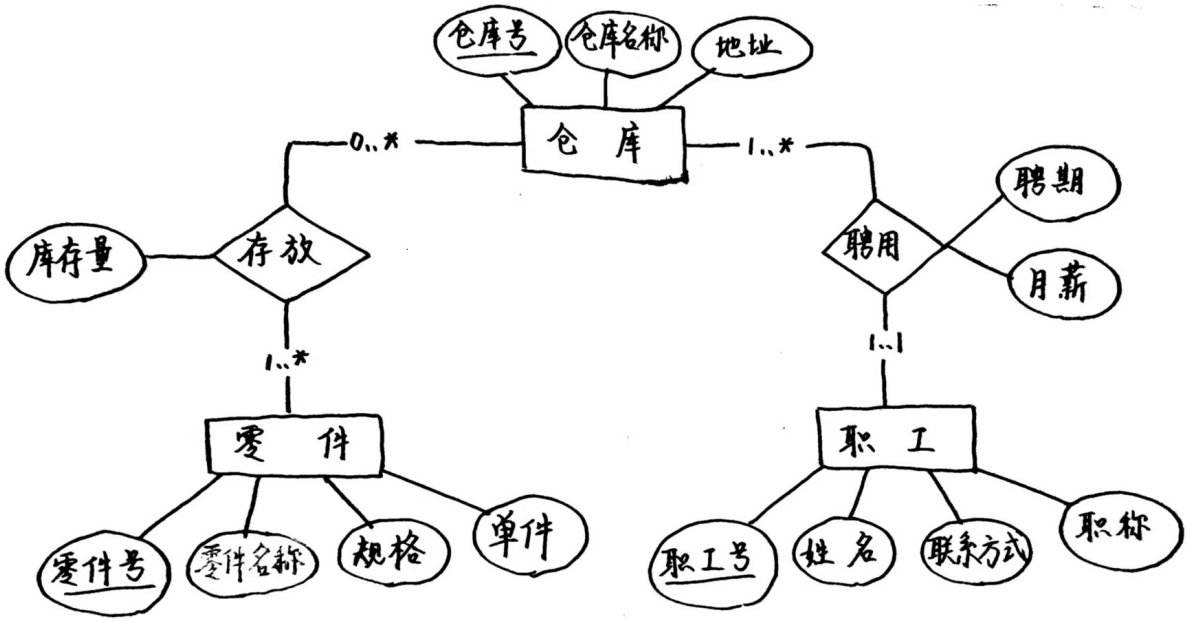
Problem 2

某零件供应链数据库中有三个实体集.
一是 "仓库" 实体集，属性有仓库号，仓库名称，地址等;
二是 "零件" 实体集，属性有零件号，零件名称，规格，单价等;
三是 "职工" 实体集，属性有职工号，姓名，联系方式，职称等.
仓库与零件之间存在 "存放" 关系，每个仓库可以存放多种零件，
每种零件可以存放在多个仓库里，并且零件在仓库存放有一定的库存量;
仓库与职工存在 "聘用" 关系，每个仓库有许多职工，
每个职工只能在一个仓库工作，仓库聘用职工有聘期和月薪.

2.1 ER 图

画出 ER 图，并在图上注明属性、联系的类型.

Solution:



2.2 关系模式

将 ER 图转换成关系模式，并注明主键和外键.

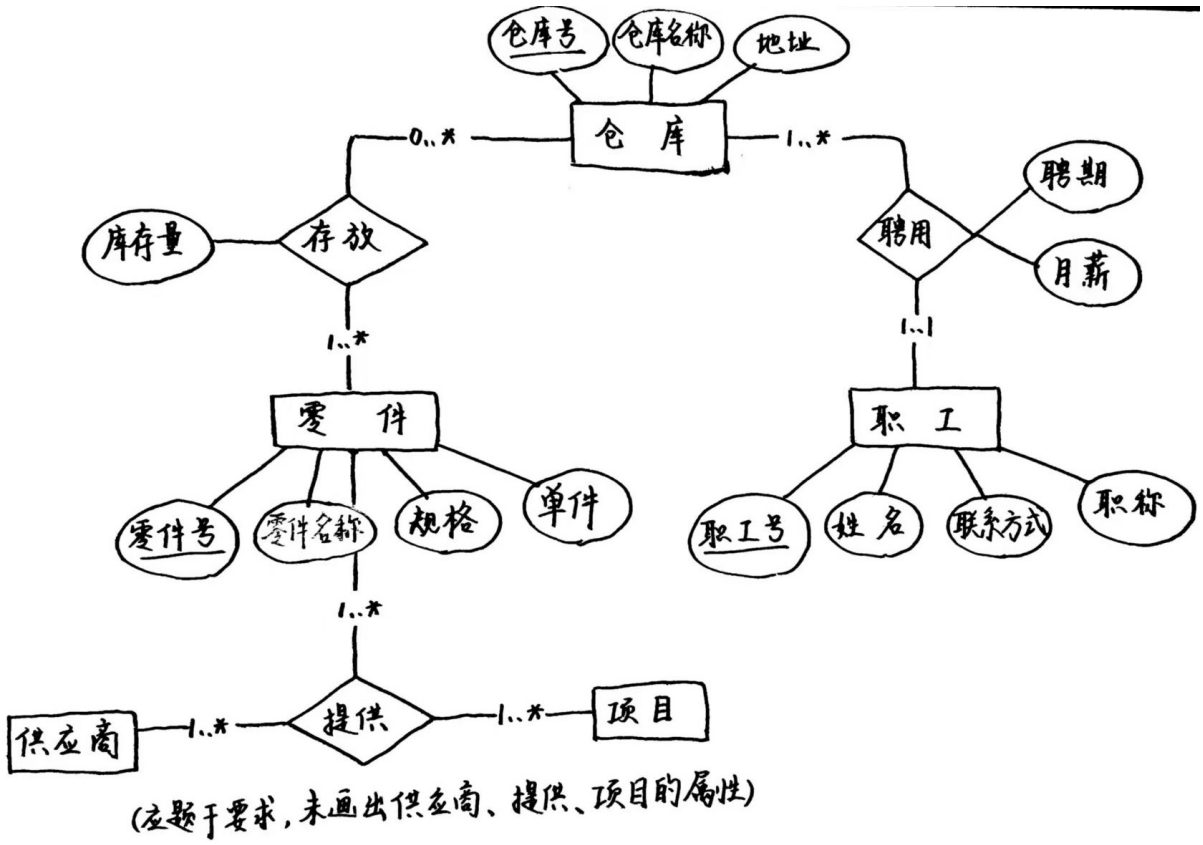
Solution:

$$
\begin{aligned}
&\text{仓库}(\underline{\text{仓库号}},\ \text{仓库名称},\ \text{地址})\\
&\text{零件}(\underline{\text{零件号}},\ \text{零件名称},\ \text{规格},\ \text{单价})\\
&\text{职工}(\underline{\text{职工号}},\ \text{姓名},\ \text{联系方式},\ \text{职称})\\
&\text{存放}(\underline{\text{仓库号}(\mathrm{FK}\to\text{仓库}),\ \text{零件号}(\mathrm{FK}\to\text{零件})},\ \text{库存量})\\
&\text{聘用}(\underline{\text{仓库号}(\mathrm{FK}\to\text{仓库}),\ \text{职工号}(\mathrm{FK}\to\text{职工})},\ \text{聘期},\ \text{月薪})
\end{aligned}
$$

2.3 更新 ER 图

假设数据库中又增加两个实体集: 供应商和项目.
在供应商、项目、零件实体之间存在这样关系:
许多供应商为一些项目提供多种零件，一个项目可以由多个供应商提供多种零件，
一个供应商可以供给多个项目多种零件，每种零件由不同的供应商提供给多个项目.
请画出供应商、项目和零件的 ER 关系图 (不必画出每个实体的属性).

Solution:



Problem 3

考虑以下数据库模式:

$$
\begin{aligned}
&\text{User}(\underline{\text{uid}},\ \text{sex},\ \text{age})\\
&\text{Movie}(\underline{\text{mid}},\ \text{name})\\
&\text{Review}(\underline{\text{uid, mid}},\ \text{score},\ \text{comment})\\
&\text{Friend}(\text{uid1, uid2})
\end{aligned}
$$

其中关系模式 Friend 具有约束 "$\text{uid1} < \text{uid2}$".

3.1 关系代数-1

找出 uid 为 100 的用户未点评的电影的 mid 和 name.

Solution:
所有电影减去 uid 为 100 的用户点评过的电影即可得到未点评的电影:
(从问题的反面出发解决问题)

$$\text{result1} = \prod_{\text{mid,name}}(\text{Movie}) - \prod_{\text{mid,name}}\left(\sigma_{\text{uid}=100}(\text{Movie} \bowtie \text{Review})\right)$$

3.2 关系代数-2 $(\star)$

找出这些用户的 uid, sex 和 age，他们点评了 uid 为 200 的用户点评过的所有电影.

Solution:
考察对除法操作的理解:

$$\text{result2} = \prod_{\text{uid,sex,age}}\left((\text{User} \bowtie \text{Review}) \div \prod_{\text{mid}}\left(\sigma_{\text{uid}=200}(\text{Review})\right)\right)$$

3.3 关系代数-3

找出电影 name 为 "海上钢琴师" 的最高、最低和平均 score.

Solution:
考察对聚合函数的理解:

$$\text{result3} = \prod_{\text{max,min,avg}} \left(g_{\text{max(score) as max, min(score) as min, avg(score) as avg}} \left(\sigma_{\text{name=海上钢琴师}} (\text{Movie} \bowtie \text{Review}) \right) \right)$$

3.4 关系代数-4 $(\star)$

(电影推荐)
找出 uid 为 400 的用户没有点评过，且其朋友的平均 score 最高的电影的 mid 和 name.

Solution:
首先找出 uid 为 400 的用户的所有朋友:

$$\text{uid\_400\_friends} = \rho_{\text{uid}} \left(\prod_{\text{uid2}} (\sigma_{\text{uid1}=400} (\text{Friend})) \right) \cup \rho_{\text{uid}} \left(\prod_{\text{uid1}} (\sigma_{\text{uid2}=400} (\text{Friend})) \right)$$

其次找出电影的 mid, name 和 uid 为 400 的用户的所有朋友对该电影的平均 score:
(考察对分组聚集操作的理解)

$$\text{avg\_score} = \prod_{\text{mid,name,avg}} \left({}_{\text{mid}} g_{\text{avg(score) as avg}} (\text{Movie} \bowtie \text{Review} \bowtie \text{uid\_400\_friends}) \right)$$

然后筛选出 uid 为 400 的用户没有点评过的电影:

$$\text{uid\_400\_not\_reviewed} = \text{avg\_score} - \text{avg\_score} \bowtie \prod_{\text{mid}} (\sigma_{\text{uid}=400} (\text{Review}))$$

最后找出 uid 为 400 的用户没有点评过，且其朋友的平均 score 最高的电影的 mid 和 name;

$$\begin{aligned} \text{result4} = & \prod_{\text{mid,name}} (\text{uid\_400\_not\_reviewed}) \\ & - \prod_{\text{A.mid,A.name}} \left(\rho_{\text{A}} (\text{uid\_400\_not\_reviewed}) \bowtie_{\text{A.avg<B.avg}} \rho_{\text{B}} (\text{uid\_400\_not\_reviewed}) \right) \end{aligned}$$

Problem 4

以下是某购物网站的数据库表，其中下划线部分表示主键.
用户既可以是卖家，也可以是买家.
同一订单最多有 2 条物流信息，status 分别为 "已发货" 和 "已到货".

$$\begin{aligned} &\text{Users}(\underline{\text{uid}},\ \text{phone},\ \text{address}) \\ &\text{Items}(\underline{\text{iid}},\ \text{name},\ \text{uid}(\text{FK} \to \text{Users}),\ \text{price}) \\ &\text{Orders}(\underline{\text{oid}},\ \text{uid}(\text{FK} \to \text{Users}),\ \text{iid}(\text{FK} \to \text{Items}),\ \text{quantity}) \\ &\text{Shipments}(\underline{\text{sid}},\ \text{oid}(\text{FK} \to \text{Orders}),\ \text{status},\ \text{time}) \end{aligned}$$

其中 Items 的 uid 对于买家，Orders 的 uid 对应买家.
(值得注意的是，SQL 保留关键字 `USER` 和 `ORDER`，因此应尽量避免使用它们为表名)

4.1 SQL 查询-1

找出既卖商品又买过商品的用户的 uid, phone 和 address.

Solution:
我们只需查询 Users 表中既在 Items 表中出现过，又在 Orders 表中出现过的用户:

```sql
SELECT uid, phone, address
FROM Users, Items, Orders
WHERE Users.uid = Items.uid AND Users.uid = Orders.uid;
```

4.2 SQL 查询-2

找出这些用户的 uid，他们购买了 uid 为 300 的用户卖的所有商品.

Solution:

```sql
SELECT Users.uid
FROM Users
WHERE NOT EXISTS( -- 若差集为空，则代表用户 Users.uid 购买过用户 300 贩卖的所有商品
    (
        SELECT Items.iid -- 用户 300 贩卖的所有商品
        FROM Items
        WHERE Items.iid = 300
    ) EXCEPT (
        SELECT Orders.iid -- 用户 Users.uid 购买过的所有商品
        FROM Orders
        WHERE Orders.uid = Users.uid
    )
);
```

4.3 SQL 查询-3

找出用户 uid 和他购买的商品 iid 和 price，
并且这些商品的 price 大于该用户购买的所有商品的平均 price.

Solution:

```sql
SELECT O1.uid, O1.iid, I1.price
FROM Orders AS O1, Items AS I1
WHERE O1.iid = I1.iid AND I1.price > (
    SELECT AVG(I2.price) -- 统计用户 O1.uid 购买的所有商品的平均 price 的子查询
    FROM Orders AS O2, Items AS I2
    WHERE O2.iid = I2.iid AND O1.uid = O2.uid
);
```

4.4 SQL 查询-4 $(\star)$

找出 uid 为 600 的卖家的最小、最大和平均送货时间.

Solution:
首先找出 uid 为 600 的卖家的物流 status 为 "已到货" 的订单的 oid 和 time:

```sql
SELECT O.oid, S.time
FROM Orders AS O
    JOIN Shipments AS S ON O.oid = S.oid
    JOIN Items AS I ON O.iid = I.iid
WHERE I.uid = 600 AND S.status = "已到货";
```

然后以上述结果为 `tmp` 表，计算最小、最大和平均送货时间:

```
WITH tmp AS (
    SELECT O.oid, S.time
    FROM Orders AS O
        JOIN Shipments AS S ON O.oid = S.oid
        JOIN Items AS I ON O.iid = I.iid
    WHERE I.uid = 600 AND S.status = "已到货"
)
SELECT
    MIN(S.time - t.time) AS min_delivery,
    MAX(S.time - t.time) AS max_delivery,
    AVG(S.time - t.time) AS avg_delivery
FROM Shipments AS S
    JOIN tmp AS t ON S.oid = t.oid
WHERE S.status = '已发货';
```

Problem 5

设关系模式 $R=(A,B,C,D,E)$ 上有函数依赖集 $\mathcal{F}=\{A\to B,\ BC\to D,\ C\to E,\ E\to A\}$ 成立.

5.1 属性闭包算法

判断函数依赖 $AC\to DE$ 是否在函数依赖集闭包 $\mathrm{cl}(\mathcal{F})$ 上成立.

Solution:
对 $\mathrm{result}=\{A,C\}$ 应用属性闭包算法:

- 注意到函数依赖 $A\to B$ 成立且 $A\subseteq\{A,C\}$，因此可置 $\mathrm{result}=\{A,B,C\}$
- 注意到函数依赖 $BC\to D$ 成立且 $BC\subseteq\{A,B,C\}$，因此可置 $\mathrm{result}=\{A,B,C,D\}$
- 注意到函数依赖 $C\to E$ 成立且 $C\subseteq\{A,B,C,D\}$，因此可置 $\mathrm{result}=\{A,B,C,D,E\}$

因此 $\mathrm{cl}(A,C)=(A,B,C,D,E)$
这意味着函数依赖 $AC\to DE$ 在函数依赖集闭包 $\mathrm{cl}(\mathcal{F})$ 上成立.

5.2 无损分解

将 R 分解为 $R_1=(A,C,D)$ 和 $R_2=(A,B,C,E)$ 是否是无损分解?

Solution:
要判断 R 分解为 R_1 和 R_2 是否为无损分解，
即验证 $R_1\cap R_2$ 是否为 R_1 或 R_2 的超键，
因此只需检查函数依赖 $R_1\cap R_2\to R_1$ 和 $R_1\cap R_2\to R_2$ 是否至少有一个在 R 上成立.
根据 5.1 问可知 $R_1\cap R_2=(A,C)\to(A,C,D)=R_1$ 在 R 上成立，
故上述分解是无损分解.

5.3 依赖保持

将 R 分解为 $R_1=(A,B,C,D)$ 和 $R_2=(A,C,E)$ 是否保持依赖?

Solution No.1:
注意到 $R_1=(A,B,C,D)$ 上成立函数依赖集 $\mathcal{F}_1=\{A\to B,BC\to D\}$.
注意到 $R_2=(A,C,E)$ 上成立函数依赖集 $\mathcal{F}_2=\{C\to E,E\to A\}$.
根据 $\mathcal{F}_1\cup\mathcal{F}_2=\mathcal{F}=\{A\to B,\ BC\to D,\ C\to E,\ E\to A\}$

可知 $\mathrm{cl}(\mathcal{F}_1 \cup \mathcal{F}_2) = \mathrm{cl}(\mathcal{F})$，这意味着上述分解保持依赖.

Solution No.2:
逐一检验 $\mathcal{F}$ 中的每个函数依赖在 R 分解成 R_1, R_2 后是否被保持:

- ① $\mathrm{cl}(A \cap R_1) \cap R_1 = AB$ 包含 B，函数依赖 $A \to B$ 被保持.
- ② $\mathrm{cl}(BC \cap R_1) \cap R_1 = BCD$ 包含 D，函数依赖 $BC \to D$ 被保持.
- ③ $\mathrm{cl}(C \cap R_2) \cap R_2 = ACE$ 包含 E，函数依赖 $C \to E$ 被保持.
- ④ $\mathrm{cl}(E \cap R_2) \cap R_2 = AE$ 包含 A，函数依赖 $E \to A$ 被保持.

因此上述分解保持依赖.

5.4 最小覆盖

判断函数依赖集 $\mathcal{F}$ 是否为它自己的最小覆盖.

- 若 $A \in \alpha$ 且 $\mathcal{F}$ 逻辑蕴含 $(\mathcal{F} - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$，
  则属性 A 在函数依赖 $\alpha \to \beta$ 的左侧属性 α 中无关.
- 若 $B \in \beta$ 且 $\mathcal{F}$ 逻辑蕴含 $(\mathcal{F} - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - B)\}$，
  则属性 B 在函数依赖 $\alpha \to \beta$ 的右侧属性 β 中无关.

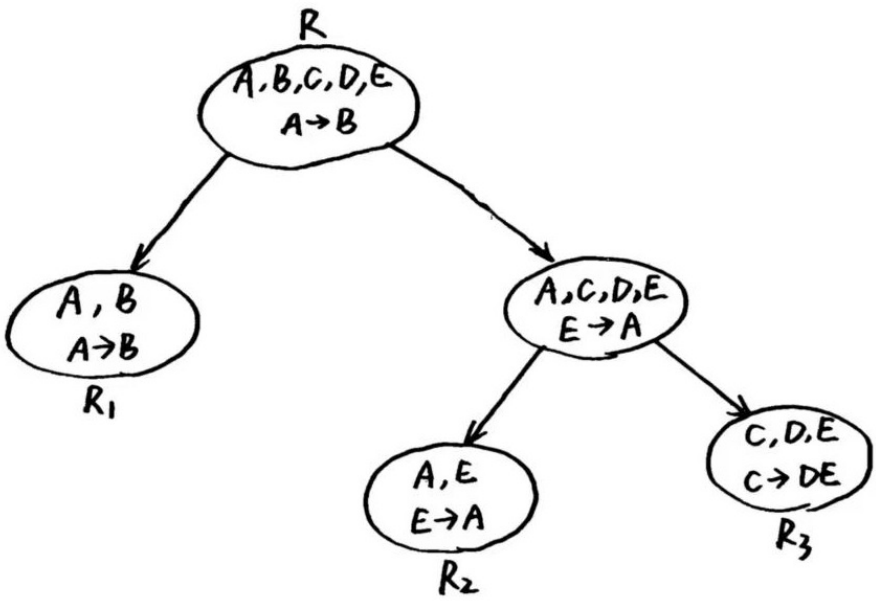
Solution:
注意到函数依赖集 $\{A \to B,\ C \to D,\ C \to E,\ E \to A\}$
被 $\mathcal{F} = \{A \to B,\ BC \to D,\ C \to E,\ E \to A\}$ 逻辑蕴含，
因此属性 B 在函数依赖 $BC \to D$ 的左侧属性集 BC 中是无关的.
这表明函数依赖集 $\mathcal{F}$ 不是它自己的最小覆盖.

事实上，$\mathcal{F}$ 的最小覆盖 $\mathcal{F}_c$ 可以是 $\mathcal{F}_c = \{A \to B,\ C \to D,\ C \to E,\ E \to A\}$，
或者说 $\mathcal{F}_c = \{A \to B,\ C \to DE,\ E \to A\}$

5.5 Boyce-Codd 范式

判断 R 是否符合 BCNF.
若符合请给出判断依据; 若不符合，请按照 BCNF 分解算法给出无损分解后的关系模式.

Solution:
注意到 R 上成立函数依赖 $A \to B$，但 A 的闭包 $\mathrm{cl}(A) = AB$，说明 A 不是 R 的超键.
因此 R 不符合 BCNF.
根据 BCNF 分解算法，我们有如下无损分解:



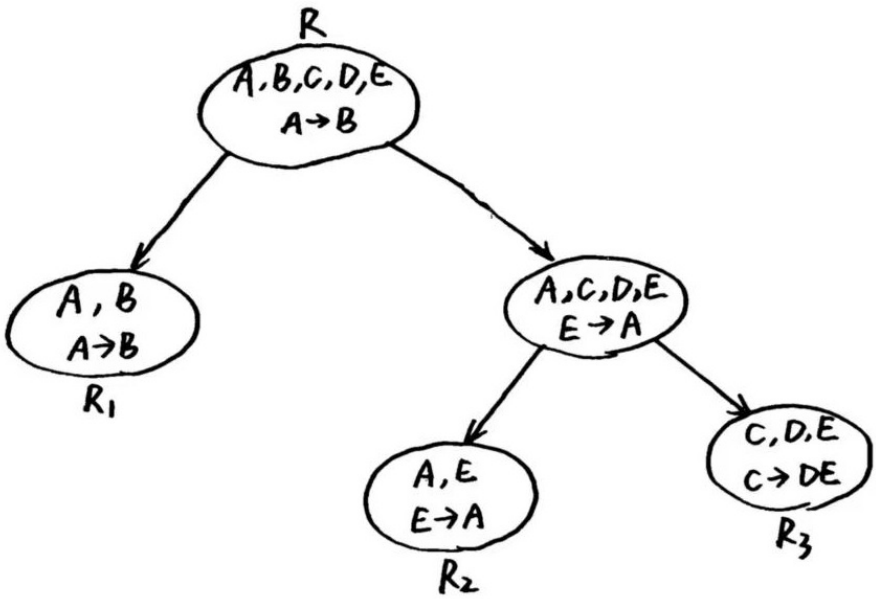
$$R=\{A,B,C,D,E\} \quad \Rightarrow \quad \begin{cases} R_1=\{A,B\} \\ R_2=\{A,E\} \\ R_3=\{C,D,E\} \end{cases}$$

5.6 第三范式

判断 R 是否符合 $3\mathrm{NF}$.
若符合请给出判断依据; 若不符合，请按照 $3\mathrm{NF}$ 分解算法给出保持依赖且无损分解后的关系模式.

Solution:
根据 5.4 问可知 $\mathcal{F}$ 的最小覆盖 $\mathcal{F}_c$ 可以是 $\mathcal{F}_c=\{A\to B,\ C\to DE,\ E\to A\}$.
容易推得 $\mathrm{cl}(C)=(A,B,C,D,E)$，可知 C 是唯一的候选键.

注意到 R 上成立函数依赖 $A\to B$，但 A 的闭包 $\mathrm{cl}(A)=AB$，说明 A 不是 R 的超键;
同时差集 $B-A=B$ 也不属于候选键 C，因此 R 不符合 $3\mathrm{NF}$.
根据 BCNF 分解算法，我们有如下保持依赖的无损分解:



$$R=\{A,B,C,D,E\} \quad \Rightarrow \quad \begin{cases} R_1=\{A,B\} \\ R_2=\{A,E\} \\ R_3=\{C,D,E\} \end{cases}$$

其中由于 $R_3=\{C,D,E\}$ 包含候选键 C，故无须生成 $R_4=\{C\}$.

The End