第6讲 函数依赖、范式、分解理论

函数依赖、正则化范式与分解理论

基本内容

1. 函数依赖
2. 完全函数依赖与传递函数依赖
3. 关于函数依赖的公理和定理

重点与难点

概念：函数依赖、部分函数依赖和完全函数依赖、传递函数依赖、 候选键、非主属性、逻辑蕴涵、属性闭包、函数依赖闭包等

方法：掌握判断函数依赖的方法，使用函数依赖的公理和定理进行推导

为什么需要研究函数依赖和关系范式?

不合理的表模式设计会导致一些问题:

1. 插入异常（无法插入必要信息）

如果一个系刚成立(例如AI)，尚无学生，则无法把系号和系主任的信息存入数据库，因为该关系的主键是（学号，课号），该插入操作违反了实体完整性约束。

2. 删除异常（需保留的信息丢失）

如果EE系的学生全部毕业了，那么在删除该系全体学生信息的同时，该系及其系主任的信息也被删除了。

3. 更新异常（不必要的重复更新，也称非受控冗余）

如果CS系主任变更了，那么所有该系学生的记录都需要更新系主任的信息，造成计算冗余和开销。

学号	姓名	年龄	系号	系主任	课号	成绩
S001	张三	19	CS	刘教授	CS101	90
S002	李四	20	CS	刘教授	CS102	85
S003	王五	18	CS	刘教授	CS102	80
S004	赵六	23	EE	胡教授	EE202	75
			AI	马教授		

函数依赖 (Functional Dependency)

函数依赖：关系（表）的属性（列）在语义上的依赖关系

示例： U＝{学号，姓名，年龄，系号，系主任，课号，成绩}

存在如下依赖关系：

- **一个学号对应一个学生姓名/年龄/系号（不考虑双学位）；**
- **一个系只有一名系主任；**
- **一个学生学习一门课程只有一个成绩。**

学号	姓名	年龄	系号	系主任	课号	成绩
S001	张三	19	CS	刘教授	CS101	90
S002	李四	20	CS	刘教授	CS102	85
S003	王五	18	CS	刘教授	CS102	80
S004	赵六	23	EE	胡教授	EE202	75

函数依赖

[Definition] 函数依赖:

设R(U)是属性集合U={A_1,A_2,…,A_n}上的一个关系模式，X, Y是U上的两个子集，若对R(U)的任意一个可能的关系r, r中不可能有两个元组在X上取值相同而在Y上取值不同，则称“X函数决定Y”或“Y函数依赖于X”，记作X → Y

示例：U = {学号，姓名，年龄，系号，系主任，课号，成绩}

学号 → {姓名，年龄，系号}

系号 → 系主任

{学号，课号} → 成绩

设计关系模式时，除给出属性全集外，还需给出数据依赖集合

注：函数依赖的分析取决于对问题领域的限定和分析，取决于对业务规则的正确理解。

例如：问题领域中，学生是没有重名的，则有：“年龄”和“系号”都函数依赖于“姓名”。

而在另一个问题领域中，学生是有重名的，则上述函数依赖是不成立的。

函数依赖

示例：下表就是问题领域，所有可能的数据已经包含在内。则存在的函数依赖有哪些呢?

✓ 下表存在的函数依赖有: A→B, A→C, B→C

属性A	属性B	属性C
1	2	3
4	2	3
5	3	3

“A值相同的元组，B值一定相同”
那么函数依赖A→B成立

✓ 下表存在的函数依赖有: A→C, D→B

属性A	属性B	属性C	属性D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

若A值各不相同呢?
B值无论是否相同，A→B都成立

该定义可扩展到属性组合上

函数依赖的特性(1)

1. 对X→Y，但Y ⊄ X, 则称X→Y为非平凡的函数依赖 (Non-trivial)；

“学号+姓名 → 姓名”是一个函数依赖，但是显然没啥信息量(即平凡的, trivial)

2. 若X→Y，则任意两个元组，若X上值相等，则Y上值必然相等，并且称X为决定因素；

函数依赖的定义如此。

3. 若X→Y，Y→X, 则记作X↔Y；一对一匹配: 例如学号与身份证号

函数依赖的特性(2)

4. 若Y不函数依赖于X，则记作$X \not\rightarrow Y$；

5. $X \rightarrow Y$，有基于模式R的，则要求对任意的关系r成立；有基于具体关系r的，则要求对某一关系r成立；

函数依赖是对任意数据成立，还是只存在于特定数据实例上。

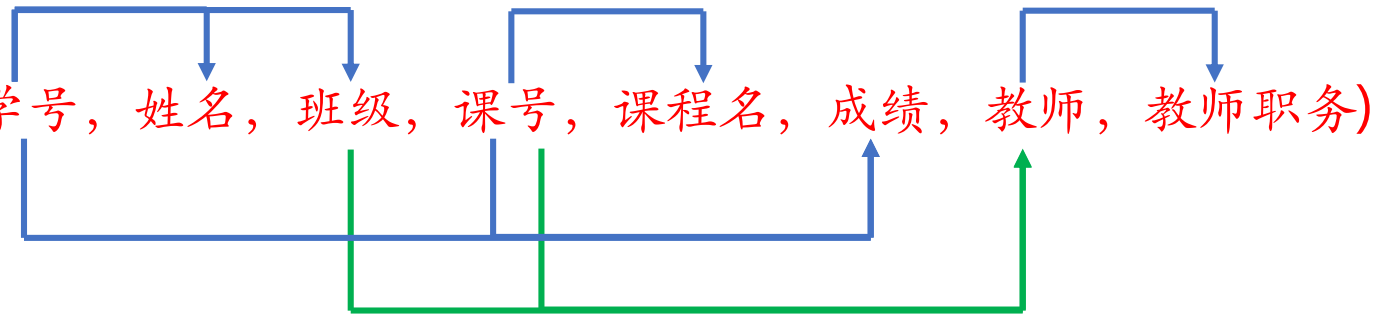
6. 如一关系r的某属性集X, r中根本没有X上相等的两个元组存在，则$X \rightarrow Y$恒成立

X不存在相同值，则X可决定所有属性值，例如学号。所有属性对X有函数依赖

函数依赖

练习：请分析下列属性集上的函数依赖

- 学生(学号，姓名，班级，课号，课程名，成绩，教师，教师职务)



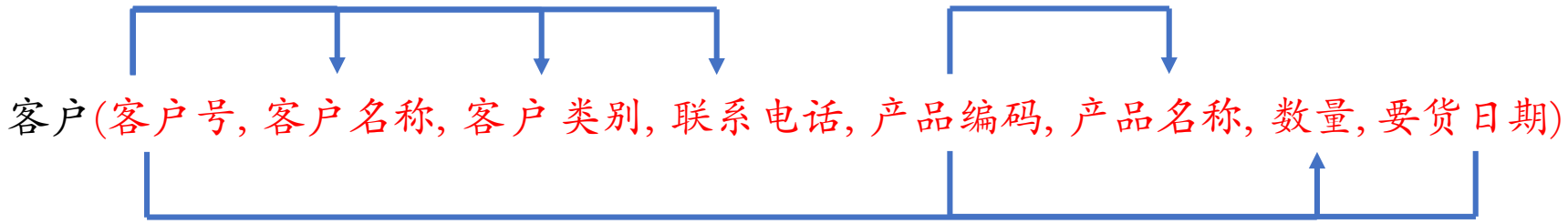
✓学号 → {姓名，班级};
✓教师 → 教师职务
✓**{班级，课号} → 教师**

✓课号 →课程名;
✓{ 学号，课号} →成绩

{班级，课号} → 教师;
课号 → 教师;
{学号，课号} → 教师 究竟选哪一个取决于对问题领域的理解

函数依赖

练习：请分析下列属性集上的函数依赖



客户号 →{客户名称，客户类别，联系电话}

产品编码 →产品名称

{客户号，产品编码，要货日期} →数量

本质上,函数依赖是对属性之间取值的一种约束,是一种数据依赖

完全函数依赖与传递函数依赖

[Definition] 部分或完全函数依赖：

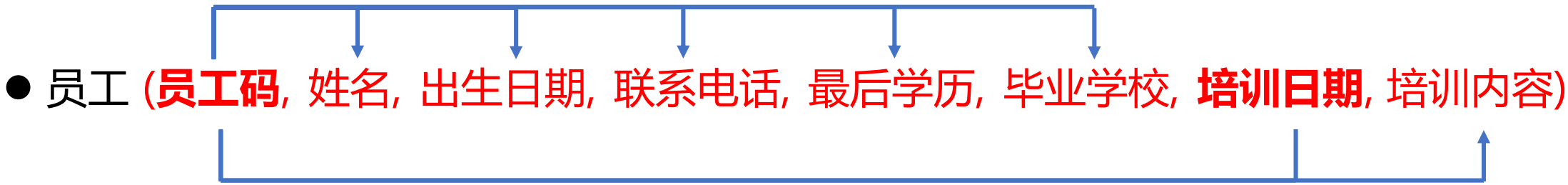
在R(U)中，若$X \rightarrow Y$并且对于X的任何真子集X'都有$X' \not\rightarrow Y$,则称Y完全函数依赖于X,记为：$X \xrightarrow{f} Y$否则称Y部分函数依赖于X, 记为：$X \xrightarrow{p} Y$

示例： U = {学号，姓名，年龄，班号，班长，课程号，成绩}

- {学号，课程号} $\xrightarrow{f}$ U （所有属性的组合依赖于学号+课程号）
- {学号，课程号} $\xrightarrow{p}$ 姓名 （姓名其实只依赖于学号）
- {学号，课程号} $\xrightarrow{f}$ 成绩 （学号和课程号都无法单独决定成绩）

完全函数依赖与传递函数依赖

练习：分析下列模式的完全或部分函数依赖



哪个属性（或属性组合）可以用来唯一地确定一条记录?

- {员工码, 培训日期} $\xrightarrow{f}$ U ;
- {员工码, 培训日期} $\xrightarrow{p}$ { 姓名, 出生日期,... };
- 员工码 $\xrightarrow{f}$ { 姓名, 出生日期,... };

传递函数依赖

[Definition]传递函数依赖

在R(U)中，若X→Y，Y→Z 且Y⊄X, Z⊄Y， Z⊄X, Y↛X, 则称Z传递函数依赖于X。

示例： U＝{学号，姓名，年龄，班号，班长，课号，成绩}

学号→ 班号； 班号→班长

学号→班长

注： “班长”是传递依赖于 “学号”的。

传递依赖存在着非受控冗余! 如下例

示例： 学生(学号，姓名，班级，班主任，职称)

学号→ 班级； 班级→ 班主任；

学号→ 班主任; 学号→ 职称

注： “班主任”是传递依赖于 “学号”的。**类似职称也如此**

学号	姓名	班级	班主任	班主任职称
2003510101	张三	035101	张林	讲师
2003510102	李四	035101	张林	讲师
2003510103	王五	035101	张林	讲师
2003510104	李六	035101	张林	讲师
2003510105	张四	035101	张林	讲师
2003510106	张五	035101	张林	讲师
2003510107	张小三	035101	张林	讲师
2003510108	张小四	035101	张林	讲师
2003510109	李小三	035101	张林	讲师
2003510110	李小四	035101	张林	讲师
2003520201	周三	035202	郑东	副教授
2003520202	赵四	035202	郑东	副教授
2003520203	赵五	035202	郑东	副教授
2003520204	赵六	035202	郑东	副教授
2003520205	钱四	035202	郑东	副教授
2003520206	强五	035202	郑东	副教授
2003520207	梁小三	035202	郑东	副教授
2003520208	梁小四	035202	郑东	副教授
2003520209	王小三	035202	郑东	副教授
2003520210	王小四	035202	郑东	副教授

完全函数依赖与传递函数依赖

练习：分析下列模式的传递函数依赖

商店(商店, 商品, 商品经营部, 经营部经理)

{商店, 商品} →商品经营部；{商店, 商品经营部} → 经营部经理

{商店, 商品} → 经营部经理

学生(学号, 姓名, 班级, 班主任, 课号, 课程名, 成绩, 教师, 教师职务)

学号→ 班级； 班级→ 班主任； {学号， 课号， 班级} → 教师；教师→ 教师职务

学号→ 班主任； **{学号, 课号， 班级} → 教师职务**

员工(员工码, 姓名, 部门, 部门经理)

员工码→部门；部门→部门经理

员工码 → 部门经理

函数依赖与传递函数依赖

练习：分析下列模式的传递函数依赖

图书(书号, 书名, 出版日期, 出版社, 书架号, 房间号, 管理员)

客户(客户号, 客户名称, 类别, 联系电话, 产品编码, 产品名称, 数量, 要货日期)

答案略。

候选键、主键与非主属性

[Definition]候选键

设K为R(U)中的属性或属性组合，若$K \xrightarrow{f} U$，则称K为R(U)上的候选键 (Candidate Key)。

- 其值能够用来唯一地识别每一条记录的非冗余属性（或属性组合）即为候选键。
- 去掉其中任何属性，该属性组合将出现重复值。
- 一个关系（表）中可能有多个候选键。

说明：

(1)可任选一候选键作为R的主键(Primary Key)；

(2)包含在任一候选键中的属性称主属性(Prime Attribute),其他属性称 非主属性；

(3)若K是R的一个候选键，$S \supseteq K$, 则称S为R的一个超键(Super Key)。

候选键、主键与非主属性

[Definition] 主键 (Primary Key)

关系R(U)中被选中用来作为每条记录独特唯一识别信息的候选键即为主键 (Primary Key)。

- 每个关系必须有一个主键，在定义关系模式(schema)时同时定义。通常用下划线标记。
- 主键中任何列不能包含缺失值(NULL)。
- 通常可以添加一个取值不重复，无实际意义的ID或编码列作为代理主键(Surrogate Key) 。

示例：U = {学号，姓名，年龄，班号，班长，课号，成绩 }

{学号，课号} $\xrightarrow{f}$ U，{学号，课号} 同时为候选键、主键、超键

练习：找候选键与非主属性

学生(学号, 年龄, 家庭住址, 课程号, 成绩, 授课教师, 教师职务)

候选键是??，非主属性是??

邮编(城市名, 街道, 邮政编码) 候选键是??，非主属性是??

学生(学号, 姓名, 所属系别, 系主任)

候选键是??，非主属性是??

找候选键需要先找出函数依赖集合

函数依赖的公理系统

- 在设计关系模式时，数据库设计者只能给出一部分函数依赖，无法（也没必要）给出全部函数依赖。
- 在使用关系数据库规范化理论评估关系模式的规范化程度时，仅知道数据库设计者明确给出的函数依赖可能是不够的。
- 使用函数依赖的公理系统，可以在数据库设计者给出函数依赖的基础上推导出其他成立的函数依赖，帮助优化数据库设计，找到合适的主键。
- 逻辑推断：根据给出的函数依赖推导出隐含的函数依赖
- 属性闭包：找出某个属性（组合）可以决定的所有属性集合

函数依赖的公理系统

[Definition]逻辑蕴涵

设F是关系模式R(U)中的一个函数依赖集合，X,Y是R的属性子集，如果从F中的函数依赖能够推导出$X \rightarrow Y$，则称F逻辑蕴涵$X \rightarrow Y$，或称$X \rightarrow Y$是F的逻辑蕴涵。

记作$F \models X \rightarrow Y$。

说明：

设F是关系模式R(U)的函数依赖集，$X \rightarrow Y$是一个函数依赖，若对R中的每个满足F的关系r，能够用逻辑推理的方法推出r也满足$X \rightarrow Y$，则称$F \models X \rightarrow Y$。

例如：F={学号→身份证号，身份证号→年龄}，$F \models$ 学号→年龄

函数依赖的公理

如何用已知函数依赖推导出其他隐含的函数依赖?

[Armstrong's Axioms A1~A3] （阿姆斯特朗公理系统）

设R(U)是属性集U={A_1,A_2,…,A_n}上的一个关系模式，F为R(U)的一组函数依赖，记为R(U, F), 则有如下规则成立：

[A1]自反律(Reflexivity rule)：若Y⊆X⊆U, 则X→Y被F逻辑蕴涵。

[A2]增广律(Augmentation rule)：若X→Y∈F, 且Z⊆U, 则XZ→YZ 被F逻辑蕴涵。

[A3]传递律(Transtivity rule)：若X→Y∈F, 且Y→Z, 则X→Z被F逻辑蕴涵。

关于函数依赖的公理和定理

[引理2]由Armstrong‘s Axiom可推出如下结论：

- (a)合并律(Union Rule)：若X→Y且X→Z, 则X →YZ。
- (b)伪传递律(Pseudo Transitivity)：若X→Y且WY→Z, 则XW→Z。
- (c)分解律(Decomposition Rule)：若X→Y且Z⊆Y, 则X→Z。

利用以上的公理系统，我们可以

（1）推导出每个属性（组合）X所能决定的所有属性（属性闭包）。

（2）推导出关系模式上隐含的函数依赖（函数依赖闭包）。

属性闭包 (Attribute Closure)

定义: **属性(组合)闭包**

给定关系模式R(U, F)，U为属性集合，F为一个函数依赖集合，X⊆U是一个属性（或属性组合），X关于F的属性闭包X^+_F是根据F可推导出的所有能被X决定的属性集合。

- **X^+_F中的属性包括X本身**
- **X^+_F中的属性可以完全满足F的所有函数依赖规则**
- **$Y \in X^+_F$ ⟺ X→Y可由F中函数依赖推导得出。**

例子： R= (U, F), U = (A,B,C,D), F ={A→B, B→C, C→D}， 求$(AB)^+_F$

- AB→A, AB→B (自反律)
- B→C 推出 AB→C (增广律)
- AB→C, C→D 推出 AB→D (传递律)

$(AB)^+_F$ = {A,B,C,D} = **U**

属性闭包 (Attribute Closure)

属性闭包的意义:

- **若$X^+_F = U$，则X为R的一个超键(Super Key)**
- **若$X^+_F = U$，且X的任何真子集$X' \subset X$都有$X'^+_F \neq U$，则X为一个候选键，即X可以决定其他所有属性的值，且不包含冗余信息。**
- **求取属性闭包可以帮助确定候选键和主键。**

例子：R= (U, F), U = (A,B,C,D), F ={A→B, B→C, C→D}， 求$(AB)^+_F$

因为$(AB)^+_F = \{A,B,C,D\} = U$, 所以 (A,B) 是R的一个超键。候选键?

因为$(A)^+_F = \{A,B,C,D\} = U$，所以(A,B)包含冗余信息，非候选键。A为候选键。

属性闭包计算方法

输入：给定的关系属性集U, 函数依赖集F，待求属性闭包的子集X⊆U

输出：X关于F的属性闭包F^+_X

步骤：

1. 初始化，F^+_X=X
2. 对从F^+_X中选取每一个属性A, 如果

   （1）在F中存在形如A→B的函数依赖 （2）B不在当前F^+_X中

   则将B加入到F^+_X, 即$F^+_X = F^+_X \cup B$
3. 重复步骤2直到没有新的属性可以添加到F^+_X
4. 输出F^+_X

属性闭包计算方法

示例： 已知 R(U, F), U={A, B, C, D, E}, F={AB → C, B → D, C → E, EC → B, AC → B}。
求：$(AB)^+$

1. $(AB)^+ = \{A, B\}$
2. 根据AB →C 可以添加C, $(AB)^+ = \{A, B, C\}$
3. 根据B → D可以添加D, $(AB)^+ = \{A, B, C, D\}$
4. 根据C →E 可以添加E, $(AB)^+ = \{A, B, C, D, E\}$。 //AC → B但是B已在闭包内。
5. 没有新属性可以添加 //EC → B 但B已在闭包内，不能再添加。
6. 结束。输出$(AB)^+ = \{A, B, C, D, E\}$

函数依赖集合闭包

[Definition]函数依赖闭包(注意和属性闭包区分)

被F逻辑蕴涵的所有函数依赖集合称为F的闭包(Closure)，记作 F^+。

说明：

若F^+=F, 则说F是一个全函数依赖族(函数依赖完备集)。

示例： 设U=(A,B,C), F={A → B, B → C}, 则F^+的组成如下，即由如下形式的X → Y 构成：

(1)X包含A, 而Y任意， 如A → A, A → B, A → C,

AB → A, AB → B, AB → C, AC → A, AC → B, AC → C

ABC → A/B/C

(2)X包含B但不含A且Y不含A, 如B → B, B → C, BC → B, BC → C

(3) 其他: C → C

注： 蓝色为平凡函数依赖（即左端包含右端）

等价函数依赖集

- 在设计关系模式时，不同设计者会给出（表面上）不同的函数依赖集。
- 如果这些表面上不同的函数依赖集能够逻辑蕴含相同的函数依赖，那么它们本质上是一样的，因此是等价的。

[Definition] 等价函数依赖集和最小函数依赖集

- 对R(U)上的两个函数依赖集合F、G，如果$F^+=G^+$，则称F和G是等价的。
  也就是说：从F和G开始进行逻辑推导，最终可以得到同样的函数依赖集
- 如果$F^+ \subseteq G^+$，则称G覆盖F (即F中每个函数依赖都被G逻辑蕴含)

- 所有等价的函数依赖集中，最小依赖集是不存在任何冗余信息的那个（些）。
- 找到最小函数依赖集可以帮助简化函数依赖定义。

等价函数依赖集合闭包

示例：R(U, F), U={A, B, C, D}，以下函数依赖集合闭包是否等价？是否互相覆盖？

- $F_1 = \{AB \rightarrow C, AB \rightarrow D, D \rightarrow C\}$
- $F_2 = \{ABD \rightarrow C, AB \rightarrow D, D \rightarrow C\}$
- $F_3 = \{A \rightarrow C, B \rightarrow D\}$

非平凡闭包(左右无交集)：

$F_1^+ = \{AB \rightarrow C,\ AB \rightarrow D, D \rightarrow C, AD \rightarrow C, BD \rightarrow C, ABD \rightarrow C\}$

$F_2^+ = \{ABD \rightarrow C, AB \rightarrow D, D \rightarrow C, AD \rightarrow C, BD \rightarrow C, AB \rightarrow C\}$

$F_1^+ = F_2^+$

$F_3^+ = \{A \rightarrow C, B \rightarrow D, AB \rightarrow C, AB \rightarrow D, AD \rightarrow C, BC \rightarrow D\ ABC \rightarrow D, ABD \rightarrow C\}$

F1与F2等价，与F3均不互相覆盖。

函数依赖的最小覆盖

[引理]：每个函数依赖集F可被一个其右端至多有一个属性的函数依赖之集 G覆盖。

最小覆盖：满足下面三个条件则F是最小覆盖

（1）右端均为单属性

（2）不存在冗余函数依赖，即去掉任意函数依赖将改变其函数依赖闭包。

（3）任何函数依赖左部不存在冗余，即去掉任意一个依赖的任意左边属性，会变其函数依赖闭包。

上一个例子中的非平凡闭包(左右部无交集)：

$F_1 = \{AB \rightarrow C,\ AB \rightarrow D, D \rightarrow C\}$ //违反(2) 存在冗余的函数依赖AB→C

$F_2 = \{ABD \rightarrow C, AB \rightarrow D, D \rightarrow C\}$ //违反(3) 左部存在冗余属性 ABD→C

与F_1,F_2等价的最小覆盖为$\{AB \rightarrow D, D \rightarrow C\}$

总结

1. 为什么要研究函数依赖与关系范式？（三类数据操作异常）
2. 函数依赖的定义、性质以及分类（部分依赖，完全依赖，传递依赖）
3. 函数依赖的判定
4. 函数依赖的公理系统
    1. Armstrong公理系统的6个推导规则
    2. 属性闭包的定义和计算方法
    3. 函数依赖集合闭包的定义和计算方法
    4. 函数依赖的覆盖、等价、最小覆盖等关系。