Week9_Course

Schema+Design part1 课前预习回顾

Database System - Nankai

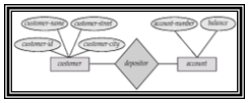
Schema Design and Refinement

- How do we obtain a good design?
- Functional dependencies & keys
- Desirable properties of schema refinement
- Boyce Codd Normal Form (BCNF)
- Third Normal Form (3NF) and 3NF Decomposition
- Fourth Normal Form (4NF)

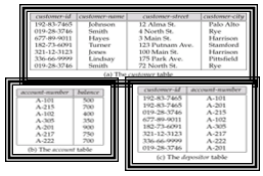
Database System - Nankai

Steps in Building a DB Application

现实世界 → 信息世界 概念模型（不依赖computer） → 机器世界 DBMS支持的数据模型 → 领域信息系统



ER diagram



R = set of R1, R2, ...

well-designed?

Schema Design and Refinement

- We do the following (ER Model to Relation Model)
  - specify all relevant constraints over R
  - implement R in SQL
  - start using it, making sure the constraints always remain valid
- However, R may not be well-designed, thus causing us a lot of problems

Example of Bad Design

Persons with several phones:

Name	SSN	Phone
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

ER图画错了

Problems (also called "Anomalies"异常):

Redundancy = repetition of data
update anomalies = update one item and forget others
= inconsistencies
deletion anomalies = delete many items,
delete one item, loose other information
insertion anomalies = can't insert one item without inserting others

Better Designs Exist

Break the relation into two:

Persons with several phones:

Name	SSN	Phone
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

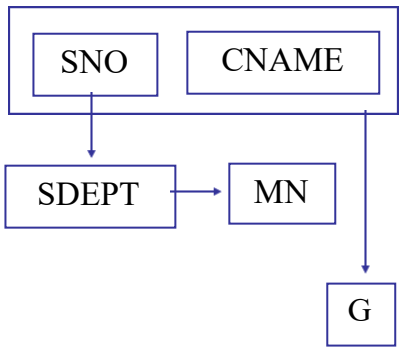
SSN	Name
123-321-99	Fred
909-438-44	Joe

SSN	Phone
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

模式设计的重要性

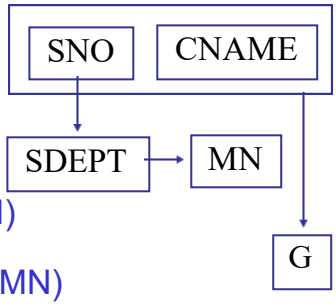
建立一个描述学生情况的数据库，已知对象有：

学生（用学号SNO描述）
系（用系名SDEPT描述）
系负责人（用其姓名MN描述）
课程（用课程名CNAME描述）
成绩（G）



模式设计的重要性

1. S(SNO, SEDPT, MN, CNAME, G)
2. S(SNO, CNAME, G)　D(DEPT, MN)
3. S(SNO, CNAME, G)　SD(SNO, SDEPT, MN)
4. S(SNO, CNAME, G)　SD(SNO, SDEPT)　SM(SNO, MN)
5. S(SNO, CNAME, G)　SD(SNO, SDEPT)　DM(SDEPT, MN)



关键解决插入异常、删除异常和冗余太大的问题

How do We Obtain a Good Design?

- Start with the original db schema R
- Transform it until we get a good design R*
- Desirable properties for R*
  - must preserve the information of R
  - must have minimal amount of redundancy
  - must be dependency-preserving
    - if R is associated with a set of constraints C, then it should be easy to also check C over R*
  - (must also give good query performance)

1. S(SNO, SEDPT, MN, CNAME, G)
2. S(SNO, CNAME, G) D(DEPT, MN)
3. S(SNO, CNAME, G) SD(SNO, SDEPT, MN)
4. S(SNO, CNAME, G) SD(SNO, SDEPT) SM(SNO, MN)
5. S(SNO, CNAME, G) SD(SNO, SDEPT) DM(SDEPT, MN)

Normal Forms(范式)

- DB gurus have developed many normal forms
- Most important ones
  - Boyce-Codd, 3rd, and 4th normal forms
- If R* is in one of these forms, then R* is guaranteed to achieve certain good properties
  - e.g., if R* is in Boyce-Codd NF, it is guaranteed to not have certain types of redundancy
- DB gurus have also developed algorithms to transform R into R* that is in some of these normal forms

Normal Forms (cont.)

- DB gurus have also discussed trade-offs among normal forms
- Thus, all we have to do is
  - learn these forms
  - transform R into R* in one of these forms
  - carefully evaluate the trade-offs
- Many of these normal forms are defined based on various constraints
  - functional dependencies and keys

Schema Design Motivation : Summary

- Redundancy
- inconsistencies
- update anomalies
- deletion anomalies
- insertion anomalies

多选题 1分

互动交流一 —不定项选择

根据南开大学的教学教务应用场景，你觉
学生（学号、课号、课程名称、成绩）是否

A 没有冗余

B 有冗余，课号冗余

C 有冗余，课程名称冗余

D 有冗余，成绩冗余

提交

Database System - Nankai

多选题 1分

互动交流二 —不定项选择

根据南开大学的教学教务应用场景，你觉得关系模式
学生（学号、课号、课程名称、成绩）存在哪些异常

A 插入异常

B 删除异常

C 更新异常

D 无异常

提交

Database System - Nankai

多选题 1分

互动交流三 —不定项选择

根据南开大学的教学教务应用场景设计的关系
学生（学号、课号、课程名称、成绩），如果你觉得
用如下哪种设计？

A 学生（学号，课程名称，成绩）；课程（课号，课程名程）

B 学生（学号，课号，成绩）；课程（课号，课程名程）

C 学生（学号，课号）；课程（课号，课程名程，成绩）

D 学生（学号，成绩）；课程（课号，课程名程）

提交

Database System - Nankai

单选题 1分

预习效果考察一

根据南开大学的教学教务应用场景，你觉
学生（学号、课号、课程名称、成绩）是不
如果不是好的设计，请用弹幕写明为什么不是

A 是好的设计

B 不是好的设计

提交

Database System - Nankai

单选题 1分

预习效果考察二

某学校设计了如下表，你认为这样设计合理吗？为什么？（弹幕回答）

学生（身份证号，学号，姓名，籍贯，家庭住址）

A 合理

B 不合理

提交

Database System - Nankai

Week9_Course

Database System-Schema+Design part2

Database System - Nankai

Schema Design and Refinement

Why to design and refine DB schema?

What is the key points of schema design ?

预习视频

How to design DB schema?

课堂讲授

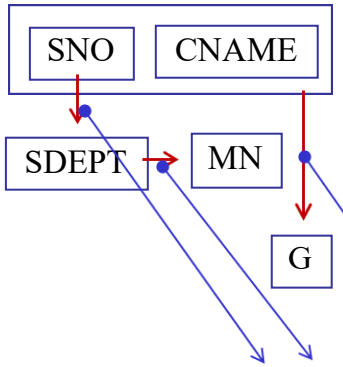
Why to design and refine DB schema?

Persons with several phones:

Name	SSN	Phone
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

- Redundancy
- inconsistencies
- update anomalies
- deletion anomalies
- insertion anomalies

What is the key points of schema design?



1. S(SNO, SEDPT, MN, CNAME, G) ✗
2. S(SNO, CNAME, G) D(DEPT, MN) ✗
3. S(SNO, CNAME, G) SD(SNO, SDEPT, MN) ✗
4. S(SNO, CNAME, G) SD(SNO, SDEPT) SM(SNO, MN) ✗
5. S(SNO, CNAME, G) SD(SNO, SDEPT) DM(SDEPT, MN) ✓

Attack Plan: Functional dependencies ⇨ keys ⇨ normal forms

How to design DB schema?

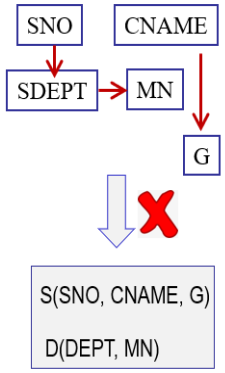
- Motivation
- Functional dependencies & keys
- Reasoning with FDs and keys
- Desirable properties of schema refinement
- Various normal forms and the trade-offs
  - BCNF, 3rd normal form, 4th normal form, etc.
- Putting all together: how to design DB schema

课前预习视频

三周课程讲授

Schema Design and Refinement

- How do we obtain a good design?
- Functional dependencies & keys
- Desirable properties of schema refinement
- Boyce Codd Normal Form (BCNF)
- Third Normal Form (3NF) and 3NF Decomposition
- Fourth Normal Form (4NF)



Functional Dependencies

- A form of constraint (hence, part of the schema)
- Finding them is part of the database design
- Used heavily in schema refinement

Definition:

If two tuples agree on the attributes $A_1, A_2, \ldots A_n$

then they must also agree on the attributes $B_1, B_2, \ldots B_m$

Formally : $A_1, A_2, \ldots A_n \rightarrow B_1, B_2, \ldots B_m$

Example of functional dependencies

S(SNO, SDEPT, MN, CNAME, G)

FDs:

SNO→SDEPT; (SNO,CNAME)→G; (SNO,SDEPT)→ MN;

SDEPT→ MN; (SNO, SDEPT, MN, CNAME) →G; ……

注意：在数据库开发过程中，函数依赖（FDs）一般是用户给出的应用场景数据之间的约束描述。如果从已有数据中找寻FDs，属于知识发现或数据挖掘。

Examples

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	John	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	lawyer

- EmpID ⟶ Name, Phone, Position
- Position ⟶ Phone
- but Phone ↛ Position

In General

- To check A → B, erase all other columns

…	A	…	B	
	X1		Y1	
	X2		Y2	
	…		…	

- check if the remaining relation is many-one (called ***functional*** in mathematics)

Relation Keys

- After defining FDs, we can now define keys
- Key of a relation R is a set of attributes that
  - functionally determines all attributes of R
  - none of its subsets determines all attributes of R
- Superkey
  - a set of attributes that contains a key
- We will need to know the keys of the relations in a DB schema, so that we can refine the schema

Example:

R(A,B,C,D,E)

Key → {A,B,C,D,E}

Key的子集 ↛ {A,B,C,D,E}

Key的超集是Superkey

Example of keys

S(SNO, SDEPT, MN, CNAME, G)

FDs: SNO→SDEPT; (SNO,CNAME)→G; SDEPT→ MN

求Keys: SNO → (SNO, SDEPT, MN, CNAME, G)?

(SNO, MN) → (SNO, SEDPT, MN, CNAME, G)?

(SNO,CNAME) → (SNO, SDEPT, MN, CNAME, G)?

(SDEPT, CNAME) → (SNO, SDEPT, MN, CNAME, G)?

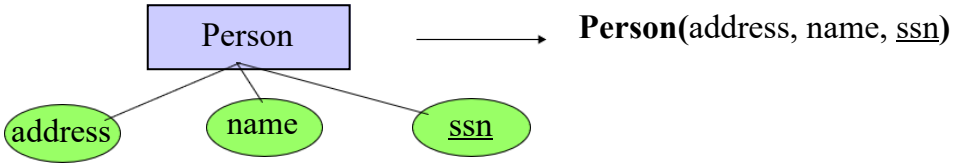
Superkeys: (SNO,CNAME) , (SNO,CNAME,SDEPT) , ……

Finding the Keys of a Relation

Given a relation constructed from an E/R diagram, what is its key?

Rules:

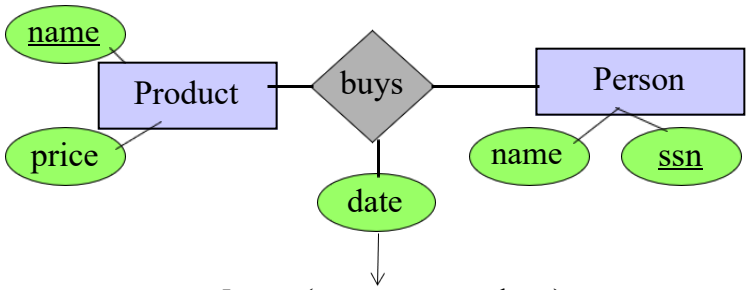
1. If the relation comes from an entity set, the key of the relation is the set of attributes which is the key of the entity set.



Finding the Keys

Rules:

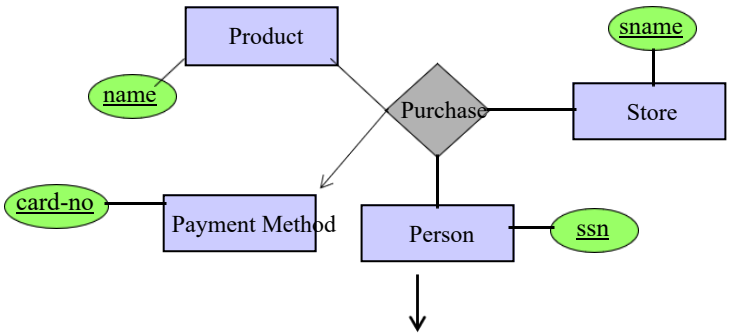
2. If the relation comes from a many-many relationship, the key of the relation is the set of all attribute keys in the relations corresponding to the entity sets



buys(name, ssn, date)

Finding the Keys

But: if there is an arrow from the relationship to E, then we don't need the key of E as part of the relation key.



Purchase(name , sname, ssn, card-no)

Finding the Keys

More rules:

- Many-one, one-many, one-one relationships
- Multi-way relationships
- Weak entity sets

(Try to find them yourself, check book)

Definition of FDs & keys：Summary

- Functional Dependencies $A_1, A_2, \ldots A_n \rightarrow B_1, B_2, \ldots B_m$
- Keys and Superkeys
- How to find one key (relation) from E/R diagram

多选题 1分

互动交流一 —不定项选择

在南开大学学生管理应用背景下，关系表学生(学号, 学院，辅导员，年级）是否存在以下

A 学号 → 年级

B 辅导员 → 学号

C 学院 → 辅导员

D 辅导员 → 学院

E （学号，年级） → 辅导员

提交

Database System - Nankai

互动交流二

请同学们用弹幕回答

- If AB→C then A→C, B→ C ?
- If A→BC then A→B, A→C ?
- If A→B and B→C then A→C ?

If (学号、课号)→成绩 then 学号→成绩，课号→成绩 ?

If 学号→系 and 系→系主任 then 学号→系主任 ?

Database System - Nankai

单选题 1分

互动交流三

what is its key?

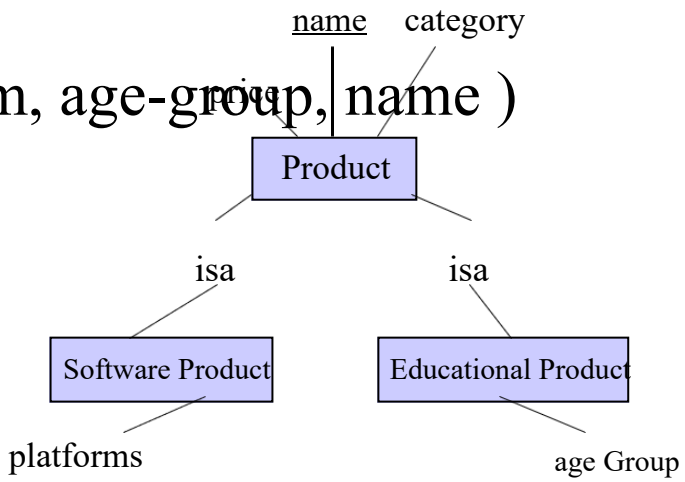
seproduct (platform, age-group, name)

A platform

B age-group

C name

D (name, platforms)



提交

System - Nankai

互动交流四

某学校设计了如下表，你觉得哪些属性集合可以作为Keys?

学生（身份证号，学号，姓名，籍贯，家庭住址）

（请用弹幕进行回答）

Database System - Nankai

Week9_Course

Database System-Schema+Design part3

Functional dependencies & keys(cont)

Reasoning with FDs

1) closure of FD sets
2) closure of Attribute sets

Closure of FD sets

（函数依赖集合的闭包）

- Given a relation schema R & a set S of FDs
  - is the FD f logically implied by S?
- Example
  - R = {A,B,C,G,H,I}
  - S = A ➔B, A ➔C, CG ➔ H, CG ➔ I, B ➔ H
  - would A ➔ H be logically implied?
  - yes (you can prove this, using the definition of FD)
- Closure of S: S^+ = all FDs logically implied by S
- How to compute S^+?
  - we can use Armstrong's axioms

Armstrong's Axioms

- Reflexivity rule
  - A1A2...An ➔ a subset of A1A2...An
- Augmentation rule
  - A1A2...An ➔ B1B2...Bm, then
    A1A2...An C1C2..Ck ➔ B1B2...Bm C1C2...Ck
- Transitivity rule
  - A1A2...An ➔ B1B2...Bm and
    B1B2...Bm ➔ C1C2...Ck, then
    A1A2...An ➔ C1C2...Ck

Additional Rules

- Union rule
  - X ➔ Y and X ➔ Z, then X ➔ YZ
  - (X, Y, Z are sets of attributes)
- Decomposition rule
  - X ➔ YZ, then X ➔ Y and X ➔ Z
- Pseudo-transitivity rule
  - X ➔ Y and YZ ➔ U, then XZ ➔ U
- These rules can be inferred from Armstrong's axioms

Inferring S+ using Armstrong's Axioms

- $S^+ = S$
- Loop
  - foreach f in S, apply reflexivity and augment. rules
  - add the new FDs to S^+
  - foreach pair of FDs in S, apply the transitivity rule
  - add the new FD to S^+
- Until S^+ does not change any further

Closure of a Set of Attributes

Given a set of attributes $\{A_1, \cdots, A_n\}$ and a set of dependencies S.

Problem: find all attributes B such that:

any relation which satisfies S also satisfies: $A_1, \cdots, A_n \rightarrow B$

The **closure** of $\{A_1, \cdots, A_n\}$, denoted $\{A_1, \cdots, A_n\}^+$,
is the set of all such attributes B

We will discuss the motivations for attribute closures soon

Algorithm to Compute Closure

Start with X={A1, ⋯, An}.

Repeat until X doesn't change **do**:

if $B_1, B_2, \cdots, B_n \rightarrow C$ is in S, **and**
$B_1, B_2, \cdots, B_n$ are all in X, **and**
C is not in X

then

add C to X.

R(A,B,C,D,E,F)

FDs: {A B → C
A D → E
B → D
A F → B}

Closure of {A,B}:
X = {A, B, C, D, E}

Closure of {A, F}:
X = {A, F, B, D, C, E}

How to find key?

Key of a relation R is a set of attributes that

- functionally determines all attributes of R
- none of its subsets determines all attributes of R

复杂度高的求解方法(暴力求解)：计算一个关系表所有属性组合$X_1, X_2, \cdots\cdots, X_n$的闭包，找出其闭包包含R所有属性的属性组合$X_{i1}, X_{i2}, \cdots\cdots, X_{im}$，这些属性组合都是超键（superkeys），如果$X_{ij} \subset X_{is}$，去掉X_{is}，剩余属性组合均为键（keys）。

求键的启发式算法：①不在任何函数依赖右边出现的属性必是键的一部分；

②不在函数依赖左部出现的属性不是键的一部分。

（一定要先使用① 后再使用②）

启发式算法举例

R(A,B,C,D,E,F）

FDs：{AB →C， AD→E，B→D，AF→B}

①A、F属性没有在函数依赖的右边出现，它必是键的一部分，先计算$(A,F)^+$，如果它不是key，再继续计算$(A,F,\cdots)^+$

R(A,B,C,D,E）

FDs：{AB →CE， E→AB，C→D}

①所有属性都在函数依赖的右边出现了，②左边只出现了A、B、C和E，则不需要计算$(D,\cdots)^+$，D肯定不在keys中出现

Database System - Nankai

启发式算法举例

S(SNO, SDEPT, MN, CNAME, G)

FDs: SNO→SDEPT; (SNO,CNAME)→G; SDEPT→ MN

Keys: (SNO,CNAME)

Address（street，city，zip）

FDs: （street，city ）→zip; zip → city;

Keys: （street，city ）和（street，zip ）

Usage for Attribute Closure

- Test if X is a superkey
  - compute X^+, and check if X^+ contains all attrs of R
- Check if X ➔ Y holds
  - by checking if Y is contained in X^+
- Another way to compute closure S^+ of FDs
  - for each subset of attributes X in relation R, compute X^+
  - for each subset of attributes Y in X^+, output the FD X ➔ Y

How to find keys: Summary

- closure of FD sets S^+: all FDs logically implied by S
- closure of attributes sets $(A1, \cdots, An)^+$: All B $\{A1, \cdots, An \rightarrow B\}$
- algorithm to find keys 启发式算法

多选题 1分

互动交流一 一不定项选择

Two sets of functional dependencies(FD's) *F* a
follow from the ones in *F*, and all FD's in *F* foll
the following three sets of functional dependenc

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

(a) Are sets F_1 and F_2 equivalent? A YES B NO

(b) Are sets F_2 and F_3 equivalent? C YES D NO

(c) Are sets F_1 and F_3 equivalent? E YES F NO

提交

Database System - Nankai

单选题 1分

互动交流二

The following three questions refer to a relation R(A, B, C, D,E) with functional dependencies A→B, BC→D, and E→C.

Which of the following functional dependency does not necessarily hold in R?

A AC → D

B AE → C

C BC → B

D CE → D

提交

单选题 1分

互动交流三

The following three questions refer to a relation R(A, B, C, D,E) with functional dependencies A→B, BC→D, and E→C.

The number of keys of R is:

A 1

B 2

C 8

D 11

提交

填空题 6分

互动交流四

R(ABCD), FD={B→D,AB→C}, Keys: [填空1] {A,B}

R(ABCDE),FD={AB→CE,E→AB,C→D} keys: [填空2] {A,B}{E}

R(ABCD),F={B→D,D→B,AB→C} } keys: [填空3] {A,B}

R(ABC),F={A→B,B→A,A→C} } keys: [填空4] {A},{B}

R(ABC),F={A→B,B→A,C→A} } keys: [填空5] {C}

R(ABCD),F={A→C,CD→B} } keys: [填空6] {A,D}

提交 se System - Nankai

主观题 10分

往年的期末考题

Consider a relation R = (A, B, C, D, E) with

FD' s A→C, CD→B, B→E and E→D

(a) What is the attribute closure of CD? {C,D,B,E}

(b) Of the following FDs, circle the ones that are implied by the functional dependencies given above:

i. AB→D ii. CD→A iii. B→D iv. AD→C

(c) List all keys for R.

{A,B} {A,D} {A,E}

提交 se System - Nankai