基本概念

2019年2月27日 10:29

数据库管理系统 (DBMS) + 数据库 (DB) = 数据库系统 (DBS)
在数据库系统出来之前，用的是文件管理系统

原子性：对于一个事件（例如将A账户的钱转到B账户上），要求要么不发生（A的钱还在，B没拿到钱），要么全部发生（事件已经进行完了），防止数据库死机引起的麻烦

数据库分为：物理层（描述数据实际上的存储形式），逻辑层（确定数据库应该保存哪些信息），视图层（相当于应用程序，只用来展示数据库的某个部分的信息，不同类型的用户获取到的信息不一样）

不同的用户通过不同的外部程序，得到相应的数据访问权限，执行不同的操作，不同的外部程序中连接着不同的DBMS的接口，从而实现不同的操作

数据库语言

2019年3月3日 17:18

数据库语言：数据定义语言（DDL），数据操作语言（DML）

数据操作语言：对表进行操作，返回的是一个表

数据定义语言：定义需要存储的数据类型，这些数据类型就成为了元数据（metadata）
有些数据需要一些约束才能存储（比如必须要是整数，必须大于零），这些都在数据定义语言中定义

数据库模型

2019年3月3日 17:20

数据模型：关系模型，实体---联系模型，基于对象的数据模型

关系模型：通过表来表示数据与数据之间的联系，使用DML（如SQL查询语言）来处理数据输出想要的数据

实体---联系模型：同面向对象，在数据库中创造一个实体，通过实体的各个属性来分配数据

基于对象的数据模型：可以看作实体---联系模型的升级版，增加了数据封装，方法等面向对象的特性

存储管理器

2019年3月3日 17:25

就是DBMS，用来连接外部应用和底层数据库的接口

数据独立性

2019年3月5日 16:58

下层的改变可以不影响上层
数据库中的部分改变不会影响

如果程序设计不依赖于物理模式，或者说物理层面上的改变（数据存放位置的改变）不影响到应用程序层面，物理层的改变不会影响到逻辑层，就称为物理数据独立性

物理数据独立性：具体怎么实现存储的方法不需要知道
逻辑数据独立性：这个数据的表由什么样的数据构成的构成的

DBMS：物理和概念的映射，概念和外界的映射
将表转换成硬件中的实际存储是DBMS的功能
针对不同的schema（学生，教授等用户）来做不同的事情

数据的约束

2019年3月5日 17:06

通过数据库设计者来设计数据的约束
通过数据库动态检查，在跑起来的时候检查是否满足这些性质

数据的约束包括：
域约束：一个属性必须对应所有可能取值的域
参照完整性
断言
授权

数据模型

2019年3月5日 17:39

以什么样的方法看待世界

如何表达数据，将数据结构化
数据的操作，用户可以对数据做什么
数据的约束

关系模型：基于表，用各种各样的表组织数据，用表组织
Record
（Record-Based Model）
Entity-Relationship Model，E-R模型：面向对象
Semi-structured data model：XML

数据管理遵循的性质

2019年3月5日 19:17

隔离性：每一个事务之间都是相互隔离的
持久性：引入日志系统，如果电脑死机就会调用日志系统来重新写入（访问磁盘的速度很慢，所以更改数据时都是直接）

原子性（atomicity）：假设A账户转账到B账户，如果数据库宕机，则对这个事件要求要么转账失败，要么转账成功。不能说A的钱成功转到B账户上然而A账户的钱没有扣除
一致性（consistency）：不管转账是否成功，账户上的总额必须是不变的
持久性（durability）：转账成功后，如果系统故障，那么A账户和B账户也应该是转账后的新的余额
隔离性（isolation）：如果两个事务同时进行，那么这两个事务都应该是相互隔离的，即其中一个事务在进行的过程中其结果不会影响到另外一个的操作

静态过程和动态过程

2019年3月6日 15:32

数学的计算：都是静态的

计算机科学的计算：有静态（程序文本是静态的，描述清楚应该怎么算）有动态（执行的过程是动态的，实际上怎么算）

实例和模式

2019年3月6日 15:34

Schema（模式）：用来描述数据库设计的
在逻辑层物理层只有一个schema，视图层有多个
schema提供抽象的观点，不管细节
逻辑schema最重要

实例（instance）：实际上是怎么改变的，数据库中某一个时间点的所有过程叫实例，反映的是
动态的性质
应该有的东西是静态的

数据库的层次

2019年3月6日 15:41

通过数据库实现抽象来看待和处理数据
物理层次：怎么存储在硬盘上（过程化）
数据库系统是通过文件系统实现的
逻辑层次：关心有什么数据，数据间的逻辑，关系（）
视图层（外部层次）：面向终端用户，用户未必用到所有的数据，有些数据是不让用，
每个用户只允许用到数据的一部分，针对不同种类有很多观点（能看到的数据一样），
不针对个体的客户，针对一类客户

物理的改变不会影响外部的
感官
抽象层次的分开，使得数据
和底层有了独立性

数据操作语言

2019年3月6日 15:43

DDL：管静态，定义存什么样的数据类型
所有的限制通过DDL定义的

DDL输出的是元数据（metadata），表示数据的数据（一张标准的表）

DML：管动态：规定数据怎么操作，增删改查
一个数据库的查询语言和DML是被包含关系
Procedural DMLs：写具体操作
Declare DMLs：写我想要什么

Project

2019年3月5日 17:57

要想用户的需求，应该用到什么样的数据库
对象之间的关系要表示清楚
描述清楚物理如何实现

数据库架构

2019年3月12日 16:28

分为两个最主要的部件
存储管理器：将数据存起来，管理数据
查询管理器：管理数据

存储管理器：和文件系统打交道，将用户希望的方式给到用户
将DML转换成底层的操作

存储管理器

2019年3月12日 16:35

权限和完整性的管理：使得不同的用户有不同的权限来管理不同的数据，管理施加在数据中的限制条件
Transaction管理：Transaction指的是逻辑单位，相当于一系列的代码声明一个事务
文件管理：数据库系统中如何管理文件

ACID原理

2019年3月12日 16:51

在数据库事务处理中
A：原子性，要么全做，要么不做
C：一致性，同样的数据，在不同地方不能有两个值，是对于单一数据库而言，而不是对于多个数据库而言
I：隔离性，一个事务在执行过程中最好不要和另外一个事务发生影响
D：持久性，如果没有任何操作，必须保持原有的状态

Transaction管理：在开始的Transaction的时候，需要记录当前的状态，在Transaction完成后，要在确认完成后才变成完成事务的状态（防止电脑死机导致的原子性问题）

结构

2019年3月12日 17:38

缓冲区存储于动态数据中
存储器存储的是静态数据
缓冲管理器负责将数据写到硬盘中

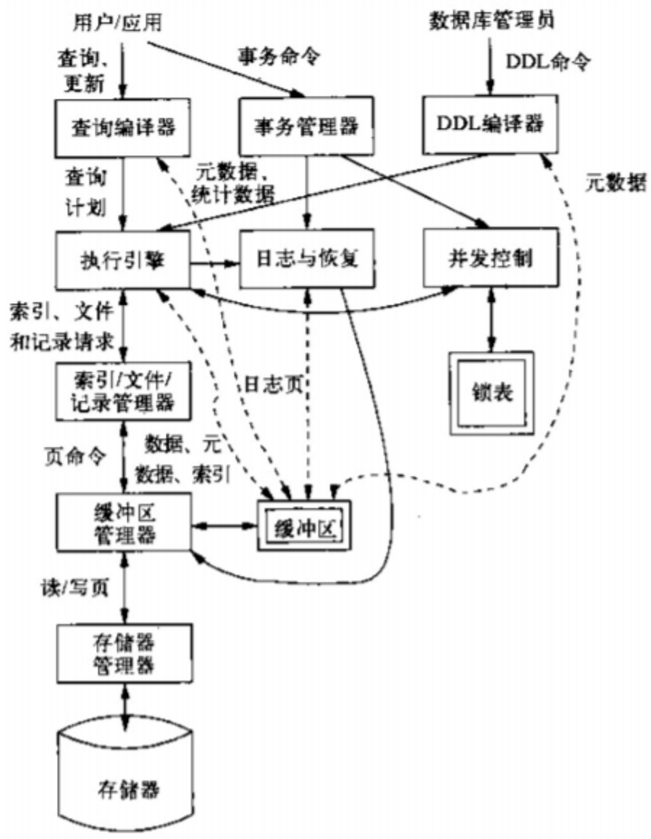


图1-1 数据库管理系统组成

数据库的子领域

2019年3月12日 17:44

1. 数据本身要存起来，能搜索，能更新
2. 需要一个用户（至少是数据库管理员）可访问的目录
3. Transaction管理
4. 并发控制
5. 保持数据库恢复正确状态的功能
6. 权限管理
7. 数据通信（分布式数据库，实现地理通信）
8. 整体性和完整性的管理
9. 增加数据独立性（schema在增删改查时是否影响到老的数据）
10. 生成程序的程序

查询管理器

2019年3月12日 16:40

DDL interpreter
DML compiler
Query evaluaion engine

DDL：数据定义语言，定义数据类型
DML：对数据库执行的增删改查的命令

interpreter：
看到一行源语言命令，就解释一行，然后执行
低效，代价小
compiler：
一次性解释全部命令
高效，代价大

要点

2019年3月19日 17:26

两个不同的表出现相同的属性->逻辑上有联系

关系数据库，生成虚拟的表，存在内存中，从而动态的生成表，存在磁盘中

Schema diagram：多个表组成的关系模型

Schema Diagram for the University Database

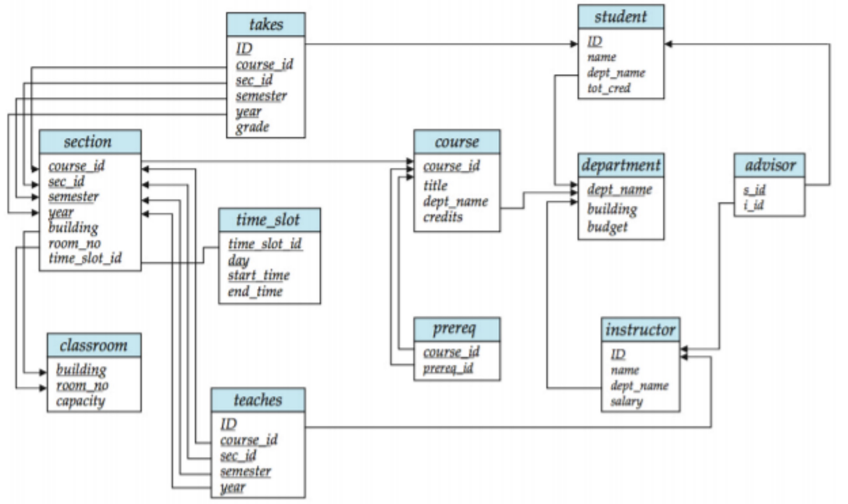


Figure 2.8 Schema diagram for the university database.

关系数据库

2019年3月22日 9:57

就是使用关系模型和各种表（表和表之间产生很多联系）构造的一种数据库

只定义关系的框架
关系实例指的是在一个元组创建完成之后的东西，是一个动态的
定义域：某一列上可以取值的范围
空值（null）：具体的值不确定，这不是一个具体的值

我们要求
所有的数据都是原子性的，不能有一个非简单的数据
项

Table（relation）：就是指表
Column（attribute）：每一列中有多少种元素，又叫属性
Row（tuple）：就是指每一行，也叫元组

Example of a Relation

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

Examples of Relations: Various Concepts [CB-15]

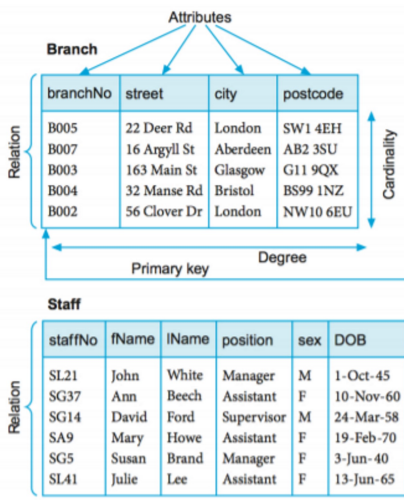


Figure 4.1 Instances of the Branch and Staff relations.

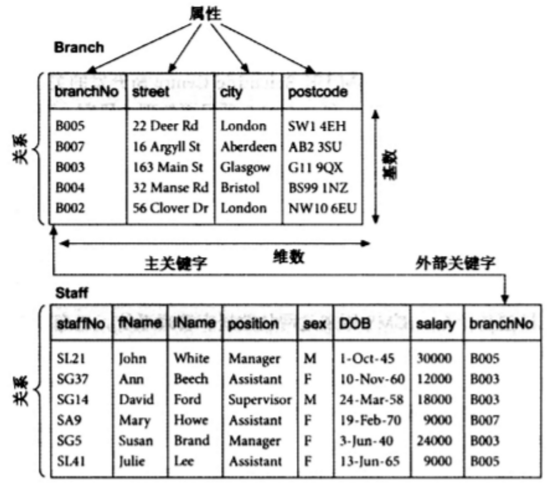
Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Examples of Relations: Various Concepts [CB-15]



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

图 4-1 关系 Branch 和关系 Staff 的实例

关系数据库的状态：

内涵（intension）：关系的结构，定义域，以及一些限制条件，这个是静态的

外延（extension）：指的是表中的元组，是动态的

维数（degree）：指的是属性的数量

基数（cardinality）：代表总共有多少行

键（key）

2019年3月22日 10:17

由来：区分不同元组（因为有些元组在某个属性上有可能有相同的值）
键是一个整个关系的性质

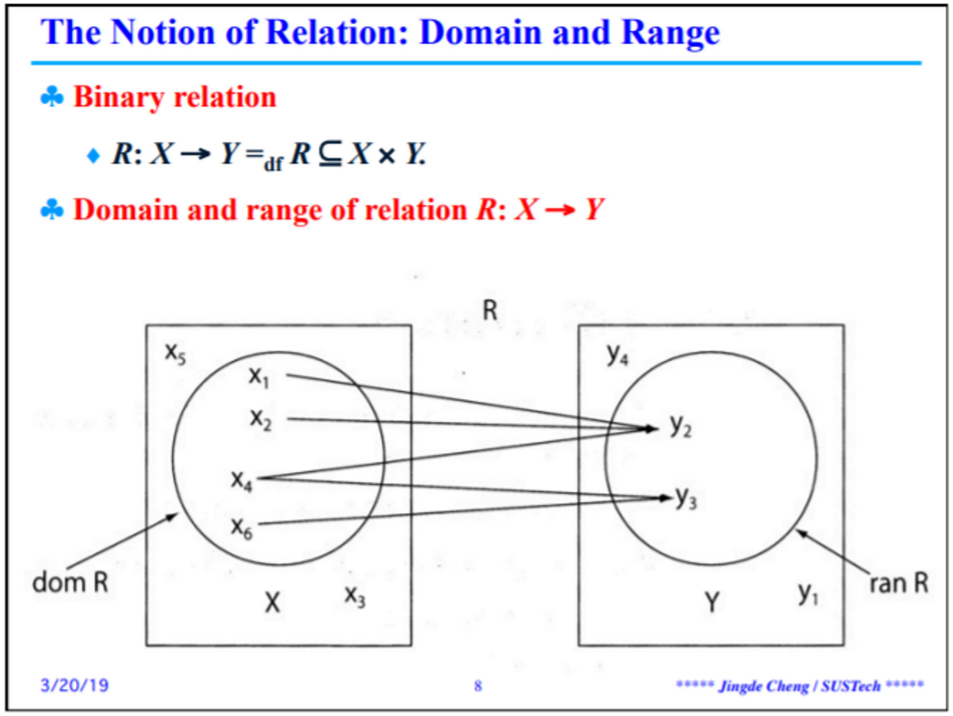
对于一张表，可以有多个候选码
关系中，任意两个不同的元组都不允许同时在码属性
上有相同的值（不论是码，超码，主码，候选码）

超码（superkey）：指的是一个或多个属性的集合，能特定的区分一个元组
候选码（candidate key）：超码可以有多个，例如：可以将学生学号作为一个超码，也可以将学生姓名和学号作为超码（事实上，只要把学号作为超码就够了）。定义其任意真子集都不能成为超码的超码，为候选码，例如，学生的学号就是一个超码，而学生姓名加学号就不是
主码（primary key）：被数据库设计者选中，主要用来在一个关系中区分不同元组的候选码
外码（foreign key）：一个关系模式（r1）可能会在它的属性中包含另外一个关系模式（r2）的主码，这个属性在r1上称作参照r2的外码。关系r1称为外码依赖的参照关系，r2叫外码的被参照关系

集合论相关定义

2019年3月22日 10:06

假设X和Y是两个集合，注意这里的domain和关系数据库中的domain不一样



参照完整性约束

2019年3月22日 10:50

完整性约束：确保数据是精确的

定义域约束：每一个属性都有独立的定义域

实体完整：在关系模型中，主键不能为空

1.设计数据时应当遵从原子性原则，即不能将原本是两个的数据项合并成一个

wrong

peopleid	fullName	lifetime
4	null, Aaliyah	1979, 2001
5	null, Aamani	1973, still alive
6	Willie, Aames	1960, still alive
7	Caroline, Aaron	1952, still alive
8	Quinton, Aaron	1984, still alive
9	Dodo, Abashidze	1924, 1990
10	Diego, Abatantuono	1955, still alive

correct

peopleid	first_name	surname	born	died
4	<null>	Aaliyah	1979	2001
5	<null>	Aamani	1973	<null>
6	Willie	Aames	1960	<null>
7	Caroline	Aaron	1952	<null>
8	Quinton	Aaron	1984	<null>
9	Dodo	Abashidze	1924	1990
10	Diego	Abatantuono	1955	<null>

上图应该修改的地方是应该将full name和life time拆分

2.primary key要尽可能地小

wrong

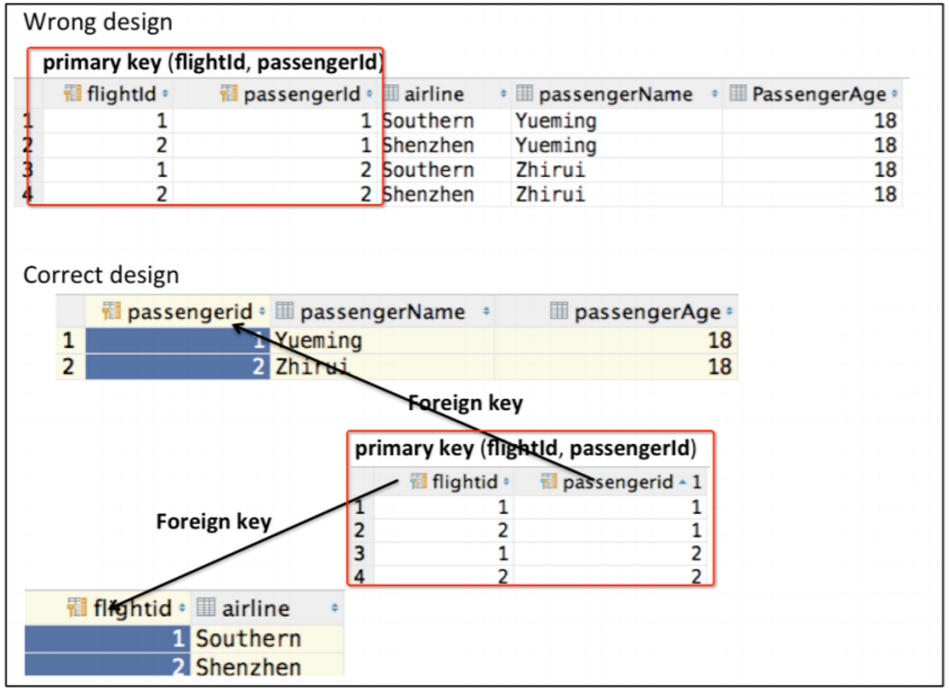
title	year_released	runtime	actor1	actor2
[illegible]	2016	90	Brandon Routh	Yuqi Zhang
[illegible]	2017	112	Andy Lau	Donnie Yen

wrong

	title	year_released	runtime	actor1	actor2
1	Zhiqiu	2016	90	Brandon Routh	Yuqi Zhang
2	Zeoi	2017	112	Andy Lau	Donnie Yen

2.primary key要尽可能地小

当描述多对多的情况时应该新建一个表格，里面存放这两个属性之间的关系



3.当描述一对多的关系时，应当新建一个表格，专门存放这种对应关系

Wrong design

primary key

	flightid	airline	city	province
1	1	Southern	GuangZhou	GuangDong
2	2	Shenzhen	ShenZhen	GuangDong
3	3	Capital	BeiJing	BeiJing
4	4	HK Express	HK	HK
5	5	Cathay Pacific	HK	HK

Correct design

primary key

	flightid	airline	cityid
1	1	Southern	1
2	2	Shenzhen	2
3	3	Capital	3
4	4	HK Express	4
5	5	Cathay Pacific	4

Foreign key

primary key

	cityid	city_name	province
1	1	GuangZhou	GuangDong
2	2	ShenZhen	GuangDong
3	3	BeiJing	BeiJing
4	4	HK	HK

要点

2019年3月26日 16:19

关系代数基本操作：
selection，projection，rename，union，difference，笛卡儿积
笛卡儿积：针对两个集合
重命名对于关系的属性来说没有任何改变
关系代数，数据库中又叫查询
除了命名外，操作的结果都是关系（*），操作的东西也都是关系，将关系表达式进行运算
关系代数表达式中，不能具有很多个select（不能使关系难读）
这六种关系代数都有自己的符号

关系演算和关系代数没有本质上的不同，在表达本质上也没有关系
关系代数表示的是一个过程的关系
关系演算只表达要算什么，怎么算的

这些关系演算都针对所有的关系封
闭

运算都是得到的新关系，新关系是存在内存
中的

R和S表达属性的集合

并和交都是针对属性来做的

聚合函数

2019年4月6日 17:45

针对多重集合，聚集一组的函数，只返回一个值

常见的聚合函数有：
sum，min，max等等

对于消除重复，有distinct关键字
例如：

The total number of instructors who teach a course in the Spring 2010 semester:

$$\mathcal{G}_{\text{count-distinct (ID)}} (\sigma_{\text{semester="Spring"} \wedge \text{year} = 2010} (\text{teaches}))$$

使用distinct，就是将相同的ID堪称一个然后相加

对于分类计算，在G前面加上属性即可

The average salary in each department:

$$_{\text{dept_name}}\mathcal{G}_{\text{average (salary)}} (\text{instructor})$$

这个就是在instructor表里面，根据
dept_name分类计算

关系演算

2019年3月26日 16:50

选择（selection）：符号σ
针对行来说
通过施加一些条件（C-condition）来获取一些行
运算的结果的schema（属性）和原来的相同
定义域变了

Projection：符号Π
只是取关系中一部分的属性
运算结果的schema是新选出来的schema
定义域没变

集合并，差，交
规定：
要求有同样属性的schema，不能有不同的schema
在应用的过程中，属性需要保持顺序
（这纯粹是习惯上的一种规定）
并，差交的运算优先级比选择和投影低
交是可以用减法定义的
$R \cap S =_{df} R - (R - S)$

笛卡儿积：符号×
是不对称的
左边的元组全部挑出来，和右边的元组每一个都组合一遍
如果R和S有同样的属性（都有一个id）需要做区分（R.A,S.A）
schema是两个分别关系属性的并
笛卡儿积会导致结果十分的啰嗦
囊括了所有的可能性

改名：符号ρ
1. 将关系的名字换掉
2. 将属性的名字换掉

换名只是关系式中换名

条件连接(theta join)
笛卡儿积中满足条件的连接在一起
条件连接的定义式：$r| >< |_{\theta(C)} s =_{df} \sigma_{\theta(C)}(r \times s)$
不是：$r| >< |_{\theta(C)} s = \sigma_{\theta(C)}(r| >< |s)$

自然连接：|><|

Assignment

自然连接：|><|

有共同值的东西作为考虑的对象

将没有共同值的都扔掉

连接的元组，漂浮的元组

自然连接满足结合律

如果两个关系没有共同的属性，自然连接就变成了笛卡儿积

Assignment

<- 用于嵌套语句

表示暂时形成一个表

半连接，只取左边

除法

不太重要，所以略

外连接

左外连接：先自然连接，丢掉东西；保留左边的数据，对应右边的属性填空值

$r = | >< | s \equiv (r | >< | s) \cup (r - \Pi_R(r | >< | s)) \times \{(null, \ldots, null)\}$

右外连接：同左外连接，定义也可以参照左外连接

全外连接：左连接加上右连接

很重要！！！

关系演算的例子

2019年4月6日 17:16

选择：

也就是选出符合条件的元组

Relational Algebra Operations: Selection

- $\sigma_{\text{dept_name="Physics"}}(\text{instructor})$

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

ID	name	dept_name	salary
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Figure 6.2 Result of $\sigma_{\text{dept_name = "Physics"}}$ (*instructor*).

4/3/19 26 ***** Jingde Cheng / SUSTech *****

投影Projection：

选出相应的列

Relational Algebra Operations: Projection

- $\Pi_{\text{ID, name, salary}}(\text{instructor})$

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

ID	name	salary
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Figure 6.3 Result of $\Pi_{\text{ID, name, salary}}$(*instructor*).

4/3/19 30 ***** Jingde Cheng / SUSTech *****

并：要求两个表有同样的模式

Relational Algebra Operations: Union

- $\Pi_{\text{course_id}}(\sigma_{\text{semeser="Fall"} \wedge \text{year=2009}}(\text{section})) \cup \Pi_{\text{course_id}}(\sigma_{\text{semeser="Spring"} \wedge \text{year=2010}}(\text{section}))$

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A

差Difference：

Relational Algebra Operations: Difference

- $\Pi_{\text{course_id}}(\sigma_{\text{semeser="Fall"} \wedge \text{year=2009}}(\text{section})) - \Pi_{\text{course_id}}(\sigma_{\text{semeser="Spring"} \wedge \text{year=2010}}(\text{section}))$

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C

BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 6.4 The *section* relation.

Relational Algebra Operations: Union

- $\Pi_{course_id}(\sigma_{semeser=\text{“Fall”} \wedge year=2009}(section)) \cup \Pi_{course_id}(\sigma_{semeser=\text{“Spring”} \wedge year=2010}(section))$

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Figure 6.5 Courses offered in either Fall 2009, Spring 2010 or both semesters.

笛卡儿积：直积

Relational Algebra Operations: Cartesian-Product [UW-08]

A	*B*
1	2
3	4

CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 6.4 The *section* relation.

Relational Algebra Operations: Difference

- $\Pi_{course_id}(\sigma_{semeser=\text{“Fall”} \wedge year=2009}(section)) - \Pi_{course_id}(\sigma_{semeser=\text{“Spring”} \wedge year=2010}(section))$

course_id
CS-347
PHY-101

Figure 6.6 Courses offered in the Fall 2009 semester but not in Spring 2010 semester.

重命名：

Relational Algebra Operations: Rename

- $\Pi_{instructor.salary}(\sigma_{instructor.salary<d.salary}(instructor \times \rho_d(instructor)))$
  (those salaries in the instructor relation for which a larger salary appears somewhere in the instructor relation (renamed as d). The result contains all salaries except the largest one)
- $\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary<d.salary}(instructor \times \rho_d(instructor)))$

salary
65000
90000
40000
60000
87000
75000
62000
72000

salary
95000

Figure 6.12 Highest salary in the university.

A	B
1	2
3	4

(a) Relation R

B	C	D
2	5	6
4	7	8
9	10	11

(b) Relation S

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

(c) Result $R \times S$

Figure 2.14: Two relations and their Cartesian product

60000
87000
75000
62000
72000
80000
92000

Figure 6.12 Highest salary in the university.

Figure 6.11 Result of the subexpression $\Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$.

自然连接：将共有的属性连接到一起

Relational Algebra Operations: Natural Join [UW-08]

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

(c) Result $U \bowtie V$

Figure 2.16: Natural join of relations

θ连接：

Relational Algebra Operations: Theta join [UW-08]

A	B	C
1	2	3
6	7	8
9	7	8

(a) Relation U

B	C	D
2	3	4
2	3	5
7	8	10

(b) Relation V

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

(c) Result $U \bowtie V$

Figure 2.16: Natural join of relations

Example 2.15: Consider the operation $U \bowtie_{A<D} V$, where U and V are the relations from Fig. 2.16(a) and (b). We must consider all nine pairs of tuples, one from each relation, and see whether the A component from the U-tuple is less than the D component of the V-tuple. The first tuple of U, with an A component of 1, successfully pairs with each of the tuples from V. However, the second and third tuples from U, with A components of 6 and 9, respectively, pair successfully with only the last tuple of V. Thus, the result has only five tuples, constructed from the five successful pairings. This relation is shown in Fig. 2.17. □

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Figure 2.17: Result of $U \bowtie_{A<D} V$

(c) Result $U \bowtie V$

Figure 2.16: Natural join of relations

Figure 2.16: Natural join of relations

Figure 2.17: Result of $U \bowtie_{A<D} V$

左外连接:

Relational Algebra Operations: Outer Join

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Figure 6.7 The *teaches* relation.

右外连接:

Relational Algebra Operations: Outer Join

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Figure 6.7 The *teaches* relation.

Relational Algebra Operations: Outer Join

ID	course_id	sec_id	semester	year	name	dept_name	salary
10101	CS-101	1	Fall	2009	Srinivasan	Comp. Sci.	65000
10101	CS-315	1	Spring	2010	Srinivasan	Comp. Sci.	65000
10101	CS-347	1	Fall	2009	Srinivasan	Comp. Sci.	65000
12121	FIN-201	1	Spring	2010	Wu	Finance	90000
15151	MU-199	1	Spring	2010	Mozart	Music	40000
22222	PHY-101	1	Fall	2009	Einstein	Physics	95000
32343	HIS-351	1	Spring	2010	El Said	History	60000
33456	*null*	*null*	*null*	*null*	Gold	Physics	87000
45565	CS-101	1	Spring	2010	Katz	Comp. Sci.	75000
45565	CS-319	1	Spring	2010	Katz	Comp. Sci.	75000
58583	*null*	*null*	*null*	*null*	Califieri	History	62000
76543	*null*	*null*	*null*	*null*	Singh	Finance	80000
76766	BIO-101	1	Summer	2009	Crick	Biology	72000
76766	BIO-301	1	Summer	2010	Crick	Biology	72000
83821	CS-190	1	Spring	2009	Brandt	Comp. Sci.	92000

Relational Algebra Operations: Outer Join

ID	name	dept_name	salary	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
33456	Gold	Physics	87000	*null*	*null*	*null*	*null*

15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
33456	Gold	Physics	87000	*null*	*null*	*null*	*null*
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
58583	Califieri	History	62000	*null*	*null*	*null*	*null*
76543	Singh	Finance	80000	*null*	*null*	*null*	*null*
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Figure 6.17 Result of *instructor* ⟗ *teaches*.

58583	*null*	*null*	*null*	*null*	Califieri	History	62000
76543	*null*	*null*	*null*	*null*	Singh	Finance	80000
76766	BIO-101	1	Summer	2009	Crick	Biology	72000
76766	BIO-301	1	Summer	2010	Crick	Biology	72000
83821	CS-190	1	Spring	2009	Brandt	Comp. Sci.	92000
83821	CS-190	2	Spring	2009	Brandt	Comp. Sci.	92000
83821	CS-319	2	Spring	2010	Brandt	Comp. Sci.	92000
98345	EE-181	1	Spring	2009	Kim	Elec. Eng.	80000

Figure 6.18 Result of *teaches* ⟕ *instructor*.

全外连接就是将左外连接∪右外连接得到的结果

SQL查询语句

2019年3月26日 20:02

查询语句的关键字是：select，这个属于DML的一部分
就是说，只要有查询，select是一个必不可少的关键字

关键字：

Select

语法格式：SELECT 列名称 FROM 表名称

列名称可以有很多个，用逗号隔开

如果想要查找全部，用*

Distinct

语法格式：SELECT DISTINCT 列名称 FROM 表名称

有些时候，不同的元组会有相同的值

如果直接用select，就会将相同的值重复的列出

如果加上distinct，相同的值就合并成一个

Where

语法格式：

SELECT 列名称 FROM 表名称 WHERE 列 运算符 值

and，or，not

用在where中的限制符，连接条件用

In

在where中规定多个值

SELECT * FROM Persons

WHERE LastName IN ('Adams','Carter')

指的是选取lastname为Adams或Carter的

Like

在where中搜索列中的指定模式的

SELECT * FROM Persons

WHERE City LIKE 'N%'

指的是在city中搜索前缀是N的字符串

having最后做过滤
where最先做过滤

对于通配符：

%	替代一个或多个字符
_	仅替代一个字符
[charlist]	字符中任何一个单一字符
[!charlist]	不属于字符中的任何一个单一字符

eg：

```
SELECT * FROM Persons
WHERE City LIKE '[ALN]%'
```

指的是在city中查找开头为A或L或N的城市

常用函数

count(*)

用来计算每一行的数量，放在select后面

如果count里面填了列的名称，就会检索该列中非NULL的行的个数

round（column_name,decimal）

舍入小数点的位数，放在select后面

Max(),min(),avg()

用来计算这一列中的最大/最小/平均值

Group by

通过某个属性进行分组，如果这条语句使用了其他函数如max（）

放在from后面使用

则就相当于对相同属性的值分别进行函数调用

select * from Cities

WHERE City LIKE 'N%'

指的是在city中搜索前缀是N的字符串

like后面接的是通配符

Is Null,is not Null

在where中判断一个属性是不是NULL

||

将两个字符串连接在一起，放在select后面

对于

select first_name || ' ' ||surname|| ' was born in ' || born from people where born > 2000;

结果是表格里面的值为select后面显示的字符串

As

将定义的列重命名，放在select后面

对于

select first_name || ' ' ||surname|| ' was born in ' || born as born_info from people where born > 2000;

就是将行名称变成born_info

case... when... when ... else... end

相当于if-else，放在select后面，表示选择出满足条件的东西

```
select born, died,
case
    when died is null
    then 'alive and kicking'
    else 'passed away'
end as status
from people limit 10;
```

这条语句就是说新创建一列，叫status，如果这行的值died是null，就赋值alive，否则就是passed away

通过某个属性进行分组，如果这条语句使用了其他函数如max（）

放在from后面使用

则就相当于对相同属性的值分别进行函数调用

Having

用来做一些限制条件的，是最后再进行的操作

由于where是先进行检索，所以当使用函数后（再where后执行）就

不会再限制值，这时候就可以用having来函数进行限制

case相当于if。。else基于表达式

Sum(

Case ...(属性)

When '某个值' then 赋值

Else

end)

使用了case关键字就相当于声明了一个属性，最好是通过as关键字来重命名

Cast ...as float强制转化成float

Limit 限制只查询前几行

Offset 代表跳过前面的几条

Inner join...on代表在进行自然连接时看准什么属性来合并

逻辑

2019年4月2日 17:38

命题变量，p1，p2
连接词，与或非等等
x1个体变量，具体到人，物
c1个体常数

SQL语句

2019年4月12日 19:52

sql语句中允许缺省

SQL语句包括：

数据定义语言（DDL）

数据操作语言（DML）

完整性（integrity）

视图定义（view definition）

事务控制（transaction control）

嵌入式SQL和动态SQL：如何将SQL嵌入到C，java中

授权

子查询

2019年4月24日 17:47

集合测试（sub-query）
关键字，对于这些关键字，前面都是用where来限定，后面都是跟一个集合
Some，all，is null，exists，not exists，unique
可以说，some等同于存在，all等同于任意
=some等同于in，但是！=some不等同于not in
>some只要大于集合中的某一个即可
>all要求大于集合中的全部元素
!= all等同于not in
代表选出存在于exists集合后面的元素
B where not exists A代表B中不包含A的部分
Select A where unique B返回的是B中没有出现重复的元素且在A中出现的元素

sql语句中对查询的结果返回的是元素还是集合没有进行严格的限定
实际中，查询的结果既可以看作是集合，也可以看作元素

对于子查询，可以放在select语句后面，不过此时返回的应该是一个集合

基本类型

2019年4月9日 16:49

char（n）：代表固定长度的字符串
varchar（n）：可变长度的字符串，最多的长度是n
int：整数类型
smallint，bigint：int能表示的范围进行增加或缩小
numeric（p，d）：表示小数，代表这个数有p个数字（包括符号位），其中最多d位数字在小鼠后面
real，double precision：浮点数
float（n）：精度至少为n的浮点数

null空值是一个值，代表未知

对于比较短的字符，直接用char
中文字符也是算一个字符

布尔类型
本质是三个值的逻辑
true，false，unknown
unknown参与运算，结果一定是unknown

日期
不太重要

创建表格
Create schema[Name|authorization CreatorIdentifier]
指定schema的Name，指定谁来创建的

删除schema

Drop schema Name[Restrict|cascade]

删除table实际上指的是删除逻辑关系

基本模式定义

2019年4月12日 20:04

Primary key和foreign key都可以是多个属性，但是foreign key必须多个属性对应关系s上primary key的每一个值

Create table xxx(
Shuxing1 leixing
Shuxing2 leixing
Primary key(Shuxing1));
其中，创建模式内的括号上面填写各个属性，下面填写各个属性的限制
Primary key：主码，要求每一个主码都是非空且唯一，而且每一个关系都要求有主码
Foreign key xxx references：外键，用于连接各个关系，foreign key一定要对应关系s在主码属性上的取值
    如果关系s中的primary key没有对应的值但是该表插入了，则数据库就会报错
    有了foreign key上面的对应关系，就叫完整性

插入语句

2019年4月9日 17:39

Insert into table values（……）

Insert into tablename [columnList]

select……;

通过select语句从数据库中选择一组数据作为插入的值

查询

2019年4月12日 20:22

Select distinct（指明去除重复）/all（指明不去除重复）xxx
From xxx
where

select子句中的属性可以含有算术表达式

运算顺序先是from，然后where，最后select

自然连接：
如果from关键字后面有多个关系，则就可以看成是笛卡儿积
自然连接使用natrual join关键字
eg：r natrual join s，使得具有相同属性的两个关系进行自然连接
R Join s using xx 代表通过特定的属性对r和s进行自然连接

集合运算：
∪∩ –分别用于对新生成的表进行的操作，注意这两个表之间要有相同的schema
交-intersect
并-union
补-except
Except all 重复减去重复的值，如果有的属性个数为负，就不显示

Not in 前面接属性，后面接括号，代表找出其属性都不在括号里面的元组

拆选出来的元组允许重复
在DBMS中，数据库是不会做笛卡儿积的（开销太大）

程序执行顺序

1. from确定从那个表中筛选，如果有多个表，则使用笛卡儿积
2. where，确定筛选的条件
3. Group by，确定分组

4. having，确定条件
5. select，挑属性
6. Order by，排序

With ，相当于重命名一个子查询的表格
Select …
From…
With ans as xx
下一次就可以直接用ans代替子查询的表格，减少了时间复杂度，如果有ans后面可以加上括号，代表将表里面的属性进行重命名

Exist，放在where里面，执行多次
In，放在where里面，通常只执行一次，执行大的就只能这样

exist使用情况：from小，exist大
in使用情况：from大，in小

Not in 后面的查询结果只要有一个null，就不返回任何元组
所以not in后面的常量不能有null，但是in后面可以有null
但是exist不会出现这种问题，其实质是看有没有结果
in/not in后面选出来的是一个集合

select可以选出一个值，也可能选出一个数字，结果

用max选出多个最大值

字符串转义，‘用 ’’ 来代替，写两次
百分号，一个或者多个字符
下划线，只有一个字符串
\代表

A left join B on A.a = B.b
如果有A left join B，还要和C进行自然连接
那么就必须要将A和B进行左连接，再和C做左连接

Union all最好在明确知道不会有重复时使用
如果使用union，就一定会加上一个去重操作

intersect和except都可以用join改写
intersect可以用inner join改写
except可以用outer join改写（保留null的可能性）
join来改写可以有更好的性能

聚合函数

2019年4月23日 16:31

select里面选用的是一个值，也可以看作是一个集合
sql中的语言处理的内容可以看成集合也可以看成值

some，every，

当聚集函数有空值时，忽略掉

where应用到元组上，having应用到group上
最后执行select的计算
当使用group by时，将这四个属性都相同的学生分成一组

聚集函数是给定一个集合，算出结果，并不在乎谁贡献了这个结果

with。。As
With max_budget(value) as (select max(budget) from department)
指的是选出这个表，并命名为max_budget

窗口函数

2019年4月23日 19:24

返回多个值，在窗口函数内也可以用聚合函数
窗口函数可以用来代替group by，并且只用一个嵌套来实现
适用于对于一个或者多个表的相关数据显示

Over (partition by xxx)
显示的是最大的发布年份，并将这个最大年份返回到每一行中
over（）代表所有行都加了一个参数

Rank() over(order by xxx desc)通过什么来进行排序，数字相同，并列排名跳过
Descrank：并列相同排名不跳过
Row_number：并列排名可能不同（排名不保证顺序）

lag（属性，上面几行）over（partition by xxx order by xxx）
指的是获取上一行的数据，根据group by 和 order by来执行

Generate_series(a,b，c)
用来生成需要a代表最小的值，
b代表下界，c代表步长
左闭右闭

length（）
显示字符串长度

Splite_part(源字符串，分隔符，显示分出来第几个)
将字符串分成很多个字符串
substr（源字符串，起始位置，终止位置）

更新

2019年4月9日 17:40

Update xxx set salary = salary * 1.5
指的是让salary上涨1.5倍

Alter table r add constrain cn cd；代表对r增加限制条件，限制条件类型是cd，cn是限制的条件名称
Alter table r drop constrain cn

Alter table r add A D；
指的是在关系表r中添加一个新的属性A，他是D类型
Drop table r
指的是删除被去掉关系的所有信息，包括这个表

删除

2019年4月9日 17:50

安全的办法：先删除数据，然后再删除表

Alter table r drop A
指的是在r中删除A属性

Delete from student；
指的是从student关系中删除所有元组

View

2019年4月30日 16:43

是实实在在列出来的，不是物理中用来存储的表
就是定义一个select中的表，执行后即可使用这个view
view是通过select语句定义的
可以通过view来设置权限（设置了全部的属性，除了一些这个管理员不能看到的权限）
管理员通过操作view来对这个对象进行修改

可以通过view来定义其他的view

view相当于是一个函数，写一次就能直接调用
数据库记住的是view如何算出来，这样就不会因为原来表的改变而改变
每用一次view，就有一次计算（可以通过其他方式来优化）
用于追求准确率的地方
效率低

view允许修改，但是不能涉及到其物理文件的结构中

Materialized view
这个表存在cache中，
代表隔一段时间查询一次，用于搜索引擎等不太追求准确率的地方，效率高

view可以删除
有两个条件：无条件消除和对正在用的人无影响的消除

优点：
数据独立性（之前的数据不管怎么改，都不会影响view）
实时性，（每使用一次view，更新）

信息安全性（通过只选取一部分的表，只给顾客看指定量的东西）
降低复杂性（用了view，可以引入层次的表达）
方便
个性化（针对某一个用户提供特殊的需要）
数据完整性（通过检查保证数据完整性）

缺点：
开销太大（每查一次，就要更新一次）
更新局限
结构局限

Transaction

2019年4月30日 17:32

事务：要么不做，要么全做完

体现原子性

正常结束：如果提交没问题，效果就永久的留在数据库中

回溯：将transaction做的东西都删去，回溯到transaction开始之前的状态

Set transaction（多个人产生了并发控制）

指定只读，可读可写等

并行控制要求高，则效率低

Integrity Constraints

2019年4月30日 17:41

外键；
不允许空值
不允许重复
Check clause，在定义关系时，都要有谓词表达式（限制某种特定的关系）

外键关系：
Referential integrity：指的是外键不是另外一个的主键
Access control：自主访问控制，强制性的访问控制
所有者可以将权限转移给指定的用户
Mandatory asscees control：将用户和文件的安全性分级（所有高级权限可以看低级权限，低级的却不行）

用户权限转移：grant
权限回收：revoke

还可以设置view的权限
还可以设置schema的权限

主键不允许有重复

数据库设计

2019年5月7日 16:27

1. 搞清楚用户的需求，出发点，关于用户要求的描述，描述要准确
2. 概念的设计，将要求变成抽象的东西，使用ER图来抽象
3. 将数据本身分析清楚，将要求变成一系列的transaction
4. 逻辑设计，转换成schema，关系模型
5. 使用DBMS实现数据库，实现在计算机系统中

概念层次
逻辑层次
物理层次

用户需要的功能
大的数据库，可能用户有重叠的要求，也可能有分离的要求
Reqiurement specification techniques
方法有：
对全体用户设计一个schema，然后合并
对每一个用户设计一个schema，然后合并（重叠就会有啰嗦）

关系数据库简单，有许多的开发功能

缺陷：
冗余，使得同样的数据，出现了两次或以上，导致数据的不一致
忽略掉某些数据，导致某些功能难以提供
不准确性，理解错用户的目的

DBS planning
说清楚目的
不要画蛇添足，一定不要管什么
数据库设计
top-down的方法来做

数据模型：
少一个正确性

使用权重来选择不同的DBMS

ER模型

2019年5月7日 17:37

Entity sets
Relationship sets
Attributes区分实体和个体
ER模型只表达数据关系是啥，并没有表达操作
静态的

Entity 就是表达一个实体，可以被区分的
Entity set 每一个实体的集合，这个集合有共同性质，是一个具体的概念
Attribute 通过attribute来区分entity set中的entity（通过attribute的值来区分），以及区分不同的entity set（通过attribute的名称来区分），有取值范围
Relationship 多个entity的联系，只要有relationship，就一定牵扯到两个或以上的entity
Relationship set 两个或以上set之间的关系
Recursive relationship set 两个集合之间有多个关系，此时要使用不同的角色来说明这件事
Descriptive
Degree 衡量关系的大小

属性
简单属性
符合属性：有子属性
Primitive attribute
Derived attribute：可以推导出来的属性
空值：有这个值，只不过不确定

ER模型

2019年5月14日 16:23

由一对一，一对多，多对多
参与：某一个集合中的每个元素是否都和别的表有交集
每个实体的个体之间要能够区别，通过值来区别，有些时候只需要一部分的值就可以区分（例如id）
Super key：将多个entity通过primary key连接在一起，这张图表就是super key

弱实体机，属性所有并起来都不能区分
强实体集

先从实体出发，考虑人和物之间的联系
表示分为两种，用联系来表示，用实体来表示

属性只能一一对应
实体集则可以多对多

ER模型：设计数据库的top-down的方式
在设计表格时，使用子属性来缩进
属性可能有多值的情况
还有的属性是导出的属性，可以被计算出来的属性
有的实体之间可能有多种联系

1. 正确理解现实世界中的内容
2. 避免冗余
3. 实体之间联系越少越好，但是要足够

要点

2019年5月28日 16:25

冗余，更新，消除（原本不该消除的也消除了）

设计数据库要求技术无损和依赖性保存

无损分解：关系数据库的时候要分解（要求各个子关系进行自然连接时能正确表示原来的总关系），如果是有损分解就会损坏原来的语义
依赖性保存：限制：是将传递闭包中拿出来的限制F，如果F‘+ = F+，则说保持了原来的分解
F+指的是F的传递性闭包

测试BCNF
将不符合BCNF的变成BCNF
如果第一点过了，就不用进行第二点了
算法流程：不断地判断是否为BCNF，如果是，输出，如果不是，分解变成BCNF
BCNF不一定既能满足无损又保持依赖性

当BCNF不能解决问题时，用3NF（保持依赖性且无损的）
无用属性：把这些属性拿掉也无关（FC），比F好处理

4NF
一个老师会属于很多个系，很多个地址
方法：老师的名字单独分成一个列，其系名称，街道，城市分为一个表
这样会产生很多冗余
如果这些属性互相不依赖，就会产生冗余
最好的方法是：名称分成一个表，街道和城市分成一个表
多值依赖（MVD）：
假设R是若干属性的集合，为了确定是依赖关系是无关的
确定左边时，右边的值并不是规律性的出现，这叫两者不产生依赖性
平凡的：不会产生依赖性
用来定义范式为多值依赖
每一个4NF一定是BCNF
重复性的数据太多的时候才会用到4NF

3NF算法:

DBS1

2019年6月6日 14:04

Database system：what is it and why study it

1. data和information的区别
   data在被翻译之前是没有意义的
   information对于接收者来说是有意义的，是一个概念
2. DB和DBMS的概念
   DB是物理存储数据的集合
   DBMS是提供一些方法将这些数据变成一个集合
   DB + DBMS = DBS
3. 基于文件的系统及其缺点
4. 数据库系统，分层实现操作

Database system concepts and architecture

1. Data abstraction概念
   三个层次：
   物理层：存数据实际上如何储存的
   逻辑层：什么数据存在物理层中，关系是什么
   视图层：提供多种view（表格）
2. instance，schema，各个层schema的作用
3. 每一层之间都有独立性
4. 数据模型
5. DDL和DML
   DDL用来定义schema

DML用来对数据进行操作

6. 各个层次的数据库设计
7. Functional components of database system
8. Transaction management及其ACID属性

DBS2

2019年6月6日 15:41

Relation databases and the relational model

1. 二维关系，定义域，值域
2. reflexive，connected，transitive，symmetric，antisymmetric
3. Relation = table，attribute = column，tuple = row
4. key：一整个关系的属性

   superkey：能唯一表示这个元组的多个属性

   candidate key：superkey的最小子集

   Primary key：candidate key的一种，其属性值很少改变

   Foreign key：

   R1中的某个属性是R2中的primary key

   要求R1中出现的某种外键在R2中也要出现

   Relation schema
5. Integrity constraint
6. Views in the relation model：相当于一张张表

DBS3

2019年6月7日 4:25

Relational Algebra and Relational Calculus

1. 关系代数性质（关系操作之后还是关系）
2. Relational algebra operation

    selection：选择的是满足条件的行

    projection：选择不同属性输出

    交（intersection），并（union），差（difference）：要求R和S两个关系的schema相同，必须是排好序的（这样属性的顺序就不会搞混）

    Direct product：就是笛卡儿积

    Rename operation：注意是对一整个关系进行重命名，而不是属性

    Natrual join：将有相同属性的值连接

    Theta join：有条件的直积

    assignment：将一个变量赋值为一个关系

    Outer join：

        left outer join：右边没有连上的元组，用null填充

        Right outer join：左边没有连上的元组，用null填充

        Full outer join：左右都没有连上的元组，用null填充

    其他操作符不太常见，这里略
3. 聚合函数：操作的是属性
4. 逻辑：relational algebra和relational calculus之间是逻辑等价的，只不过表达的方式不一样

    Relational calculus表达的是需要完成什么

    Relational algebra表达的是完成这个的过程
5. 等价的式子：relational algebra（no extention），tuple relational calculus（safety），domain relational calculus（safety）

这里讲的relational calculus都是不能使用聚合函数的

DBS4-1

2019年6月7日 5:05

SQL：Structured Query Language

1. DDL，DML
2. SQL数据类型，记住null是一个值，代表这个值未知
3. Schema creation/deletion
4. Relation creation/deletion/modification
5. Data loading
6. select语句：closed in relations
7. 元素的重复：sql中允许元组重复
8. 子查询：in，not in
9. some：greater than at least one等价于some
10. exists：关系A存在于关系B中
11. Not exists：relation A contains relation B等价于not exists（B except A）
12. unique：将子查询中不重复的元组输出
13. with：重命名

DBS4-2

2019年6月7日 5:47

SQL：Structured Query Language

1. join……on……

DBS4-3

2019年6月7日 6:17

SQL：Structured Query Language

1. Virtual relations（view）：存在内存中，不是存储在硬盘上的
    a. Create view（virtual relation）
    b. The set of tuples in a view:定义view时，数据库保存的是语句，而不是view产生的结果
    c. Materialized view：更改数据时，view上面的数据也进行了更改
    d. View modification以及修改的条件：一般的，不允许对view进行修改，除了特殊情况
    e. Advantage/disadvantage of view
    f. Transaction
2. Transaction：一堆查询，或者序列的语句
    a. Commit：使得数据库的更新永久化
    b. rollback：对数据不做修改
    c. Isolation level：
        i. Dirty read，non-repeatable read，phantom read
3. Constraint on a single relation：
    a. check：规定了数据必须在命题p的范围内
4. Referential integrity：
    a. Referential integrity要求比foreign key低，referential integrity不要求其中一个属性一定是primary key，而且可以有多个相同属性的元组出现
5. Access control：
    a. Discretionary access control
    b. Mandatory access control
6. Authorization identifiers：设置不同数据库管理员的权限

7. ownership：确定是谁负责的这个表格
8. privileges：每一个使用者都有不同的使用级别，可以实现什么操作
9. Authorization：使用语句时，先看看有没有进行授权
10. Granting/Revoke/Transfer privileges：是对特定的属性进行授权，而不是特定的元组
11. Dynamic SQL和Embedded SQL

DBS5

2019年6月7日 17:51

Database Design

1. 不同DBS的设计阶段
2. DBS planning：让不同阶段的开发尽可能地有效
3. DBS definition：定义DBS的范围还有需求
4. User views：从user的角度
5. DBS design：top-down，bottom-up

DBS5_1

2019年6月8日 18:38

Relational database design

1. Functional dependency
    a. 定义了什么是好的设计
2. 1NF：定义数据时所有的attribute要是原子性的
3. Functional dependency
    a. 对实例中所有的元组对t_1和t_2，只要$t_1[\alpha] = t_2[\alpha]$，则$t_1[\beta] = t_2[\beta]$
    b. 函数依赖是使用大量的数据进行判定，而不是给定的一小部分数据
4. BCNF
    a. 令F为r（R）上的函数依赖集，F^+为F的传递闭包
    b. BCNF的条件是：对F^+中所有形如$\alpha \to \beta$的函数依赖，都有以下成立：
        i. $\alpha \to \beta$是平凡的函数依赖（必然成立的函数依赖）
        ii. α是模式R的超码
    c. 变成BCNF：
        i. 存在至少一个非平凡的函数依赖$\alpha \to \beta$，其中α不是R的超码，则将其分解为：
            1) $(\alpha \cup \beta)$
            2) $R - (\beta - \alpha)$
5. 第三范式
    a. 令F为r（R）上的函数依赖集，F^+为F的传递闭包
    b. BCNF的条件是：对F^+中所有形如$\alpha \to \beta$的函数依赖，至少一项成立：
        i. $\alpha \to \beta$是平凡的函数依赖（必然成立的函数依赖）
        ii. α是模式R的超码
        iii. $\beta - \alpha$中的每个属性A都包含于R的候选码中
6. 无损分解
    a. 令R_1和R_2为R上的无损分解，如果函数依赖中至少有一个属于F^+

i. $R_1 \cap R_2 \rightarrow R_1$

ii. $R_1 \cap R_2 \rightarrow R_2$

则这种情况就是无损分解

7. 保持依赖

   a. 将R分解成n个关系R_i，令F_i为R_i上的关系依赖集，令$F' = \cup_{i=1}^{n} R_i$，如果$F'^{+} = F^{+}$，则称为保持依赖的分解

8. 4NF

   a. Multivalued dependency：$\alpha \rightarrow\rightarrow \beta$，关系r（R）的任意实例中，对r中任意一对满足$t_1[\alpha] = t_2[\alpha]$的元组对，r中都存在元组$t_3$和$t_4$，使得$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha], t_3[\beta] = t_1[\beta]$
   $t_3[R - \beta] = t_2[R - \beta], t_4[\beta] = t_2[\beta], t_4[R - \beta] = t_2[R - \beta]$

DBS5_2

2019年6月8日 18:37

The entity-relationship model

1. 采用的是top-down的方式
2. Entity and entity-set：
    a. Entity 指的是一个世界上存在的真实的物体，entity-set指的是这一系列拥有相同类型属性的集合
3. Attribute：每一个实体都使用attribute表示
4. Relationship and relationship set
    a. Relationship：多个实体之间的关系
    b. relationship set：多个相同类型的关系的集合
    c. Relationship instance：现实中各个实体之间的关系
    d. Recursive relationship set：同一个实体集之间相互作用
5. Role 指的是实体中的实例，属于的是哪一个实体
6. Attribute
    a. Descriptive attribute：关系的属性
    b. Domain of attribute：每一个attribute之间都有的取值范围
    c. Simple attribute：不再能被分解的
    d. Composite attribute：可以再被分解的
    e. single-valued attribute：一个attribute有单个值
    f. multiple-valued attribute：有多个取值
    g. Primitive attribute：不能从其他值推出来的
    h. Derived attribute：可以被其他值推出来的
7. Constraint
    a. Mapping cardinalities：
        i. 指的是两个或多个实体映射的关系（一对一，多对多等）
        ii. 可以限制每一个关系的数量

    b. Participation：
        i. total：E中每一个实体至少参与了一个关系
        ii. partial：E中只有一些参与了关系
8. Keys for relationship
    a. Individual relationship：
        i. 若R没有descriptive attribute，则这个关系的每一个primary key称为individual relationship
        ii. R有descirptive attribute，则这个关系的每一个primary key加上这些attribute称为individual relationship
9. Weak/Strong entity sets
    a. Weak entity sets：没有足够的attributes来形成一个primary key
        i. 处理方法：构造一个新的id集合
    b. strong：与a相反
10. Specialization
    a. 一个实体下面有多个子集
    b. 使用多个实体，采用继承的关系进行特化
    c. 是top-down的方式
11. Generlization
    a. bottom-up的方式

DBS5_3

2019年6月8日 18:37

The entity-relationship mode

1. 每一个关系都要用key来唯一的表示
2. 设计要点
    a. 实体集之间可以通过多个联系来实现，但是不要对每一个实体集都加上一个可能的联系
    b. 能用attribute就尽量不用relation来表示
    c. 将composite attribute分解成单一的attribute
    d. 通过主码的设置来调整一对一，多对一
    e. 弱实体集必须和owner entity set联系起来才有意义，将弱实体集与其表示实体集相连的关系叫identifying relationship，设置discirminator来表示。
    f. 弱实体集的primary key由owner entity set的主码加上discirminator联合表示
    g. 多对一，且多的部分是全部参与，则多的部分合并
    h. 一般情况下，连接弱实体集和强实体集的联系集是冗余的
    i. 不能用一个实体集的主码作为另一个实体集的属性
    j. 三元联系可以通过增加一个实体的方式来实现三元拆分成多个二元