Database System-SQL Introduction

Database System-SQL Introduction-part1

Basic Stuff

Introduction to SQL

- SQL is a very-high-level language
  - User can say “what to do” rather than specify “how to do it”
  - Can avoid specifying a lot of data-manipulation details needed in procedural languages like C++ or Java
- Database management system figures out “best” way to execute query
  - Called “query optimization”

Introduction to SQL

- SQL was developed by IBM in late 1970s
- SQL-92 was endorsed as a national standard by ANSI in 1992
- SQL can be used to:
  - Define database structures
  - Query a database
  - Update database data
- SQL consists of a
  - **Data Definition Language (DDL),** used to define database structures
  - **Data Manipulation Language (DML),** is used to query and update data
- SQL statements are terminated with a semicolon

DDL - Data Definition Language

CREATE (1) Table - define table name, attributes, types

(2) View - define user view of data

(3) Index - create index on nominated attributes

DROP (1) Table - delete table, attributes and values

(2) View - delete user view(s)

(3) Index - delete index, indexes

Some DBMS have an ALTER command to vary attribute characteristics.

DML - Data Manipulation Language

Insert - Add a single row

Delete - Delete rows of data from a table

Update - Amend attribute values in rows

Select - From - Where – Query data

Defining a Database

show databases:

Database
information_schema
mysql
performance_schema
sys

```
CREATE DATABASE IF NOT EXISTS BarDrink
DEFAULT CHARACTER SET utf8mb4
DEFAULT COLLATE utf8mb4_unicode_ci;

SHOW CHARACTER SET;
SHOW COLLATION;
```

After Defining a Database

```
1    #CREATE DATABASE IF NOT EXISTS BarDrink
2    /*DEFAULT CHARACTER SET utf8
3    DEFAULT COLLATE utf8_general_ci;*/
4    show databases;
```

Database
bardrink
information_schema
mysql
performance_schema
sys

C:\ProgramData\MySQL\MySQL Server 8.0\Data\bardrink

Defining a Relation Schema

CREATE TABLE name (list of elements);
Elements: attributes and their types; key declarations; constraints.

CREATE *X* for views, indexes, assertions, triggers.
DROP *X* name deletes the element of kind *X* of that name.

```
Use bardrink;
CREATE TABLE Sells (
        bar   CHAR(20),
        beer  VARCHAR(20),
        price REAL,
        PRIMARY KEY (bar,beer)
 );
```

After Defining a Relation Schema

Show tables;

	Tables_in_bardrink
▶	sells

insert into Sells
values ('Siz','Bud',13);

Select* from sells;

	bar	beer	price
▶	Siz	Bud	13

DROP TABLE Sells;

Basic Types

- char(n)/character(n)
- varchar(n)/charcter varying
- int/integer
- smallint
- numeric(p,d)
- real,double precision

删除数据库对象

- DROP 对象 对象名
  - DROP DATABASE 数据库名1{,数据库名2…...}
  - DROP TABLE 表名1{,表名2…...}
  - DROP VIEW
  - DROP INDEX
  - DROP TRIGGER

delete from 表名

关闭mysql safe mode:
Edit->Preference→SQL editor,去掉勾选
safe udates

Changing Columns

Add/Drop an attribute of relation *R* with
ALTER TABLE *R* ADD/DROP <column>;

Examples

```
ALTER TABLE Bars ADD phone CHAR(16)
                     DEFAULT 'unlisted';

ALTER TABLE Bars DROP COLUMN license;
```

Summary

- SQL Introduction
- DDL & DML
- Defining a Database
- Defining a Relation Schema
- Drop Objects
- Alter Table

多选题 1分

不定项选择

SQL语言具有以下功能

A 关系规范化，数据操纵，数据控制

B 数据定义，数据操纵，数据控制

C 数据定义，关系规范化，数据控制

D 数据定义，关系规范化，数据操纵

提交

多选题 1分

不定项选择

使用CREATE TABLE语句建立的是

A 数据库

B 表

C 视图

D 索引

提交

多选题 1分

不定项选择

若用如下SQL语句创建一个表student:

```
CREATE TABLE student (NO CHAR(4) NOT NULL,
                      NAME CHAR(8) NOT NULL,
                      SEX CHAR(2),
                      AGE INT)
```

可以插入到student表中的是

A ('1031','曾华'，男，23)

B ('1031','曾华'，NULL，NULL)

C (NULL,'曾华'，'男'，23)

D ('1031',NULL，'男',23)

提交

Database System - Nankai

Database System-SQL Introduction-part 2

Integrity Constraints

Declaring Keys

- An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE.
- These each say the attribute(s) so declared functionally determine all the attributes of the relation schema.
- There are a few distinctions to be mentioned later.

Declaring Single-Attribute Keys

- Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute.
- Example:

```
CREATE TABLE Beers (
        name    CHAR(20) UNIQUE,
        manf    CHAR(20)
);
```

Declaring Single-Attribute Keys

insert into Beers
values ('Bud','AM ');

```
15 •  insert into Beers
16    values ('Bud','AM');
17 •  insert into Beers
18    values ('Bud','AM');
19
20
```

Output

Action Output

#	Time	Action
7	21:02:44	CREATE TABLE Beers (nameCHAR(20) UNIQUE, manfCHAR(20))
8	21:07:16	insert into Beers values ('Bud','AM')
9	21:07:16	insert into Beers values ('Bud','AM')

声明为UNIQUE的属性是否可以出现空值？是否允许多个空值？

声明为PRIMARY KEY的属性是否可以出现空值？

Declaring Multiattribute Keys

- A key declaration can also be another element in the list of elements of a CREATE TABLE statement.
- This form is essential if the key consists of more than one attribute.
  - May be used even for one-attribute keys.

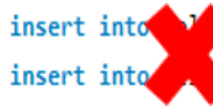
Declaring Keys: Example

```
CREATE TABLE Sells (
        bar   CHAR(20),
        beer VARCHAR(20),
        price REAL,
        UNIQUE(bar,beer) );
```

is different from:

```
CREATE TABLE Sells (
        bar CHAR(20)          UNIQUE,
        beer VARCHAR(20)      UNIQUE,
        price REAL );
```

```
insert into sells values ('A','B',1);
insert into sells values ('A','C',2);
insert into sells values ('D','B',1);
```



```
insert into  values ('A','B',1);
insert into  values ('A','B',2);
```

Example: Multiattribute Key

- The bar and beer together are the key for Sells:

```
CREATE TABLE Sells (
        bar             CHAR(20),
        beer            VARCHAR(20),
        price   REAL,
        PRIMARY KEY (bar, beer)
    );
```

Required Distinctions

- **Standard SQL requires these distinctions:**
  1. There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes.
  2. No attribute of a PRIMARY KEY can ever be NULL in any tuple. But attributes declared UNIQUE may have NULL.

PRIMARY KEY Versus UNIQUE

- The SQL standard allows DBMS implementers to make their own distinctions between PRIMARY KEY and UNIQUE.
  - Example: some DBMS might automatically create an *index* (data structure to speed search) in response to PRIMARY KEY, but not UNIQUE.

Foreign Keys

- Consider Relation Sells(bar, beer, price).
- We might expect that a beer value is a real beer -- - something appearing in Beers.name .
- A constraint that requires a beer in Sells to be a beer in Beers is called a *foreign-key* constraint.

Expressing Foreign Keys

- Use the keyword REFERENCES, either:
  1. Within the declaration of an attribute, when only one attribute is involved.

     REFERENCES <relation> (<attribute>)

  2. As an element of the schema, as:

     FOREIGN KEY (<list of attributes>)

     REFERENCES <relation> (<attributes>)
- Referenced attributes must be declared PRIMARY KEY or UNIQUE.

Example: With Attribute

```
CREATE TABLE Beers (
        name    CHAR(20) PRIMARY KEY,
        manf    CHAR(20) );
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20) REFERENCES Beers(name),
        price   REAL );
```

Example: As Element

```
CREATE TABLE Beers (
        name    CHAR(20) PRIMARY KEY,
        manf    CHAR(20) );

CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price   REAL,
        FOREIGN KEY(beer) REFERENCES
          Beers(name));
```

Enforcing Foreign-Key Constraints

Name	Manf
Bud	Anheuser-Busch
…	…

Beers

Bar	Beer	Price
Joe's	Bud	5.00
Sally's	Bud	4.00

Sells

insert into sells
values ('Joe','Bud',5.0);

Actions Taken

- The three possible ways to handle beers that suddenly cease to exist are:
  1. *Default* : Reject the modification.
  2. *Cascade* : Make the same changes in Sells.
     - Deleted beer: delete Sells tuple.
     - Updated beer: change value in Sells.
  3. *Set NULL* : Change the beer to NULL in Sells.

Example: Cascade

Name	Manf
~~Bud~~	~~Anheuser-Busch~~
…	…

Beers

Bar	Beer	Price
~~Joe's~~	~~Bud~~	~~5.00~~
~~Sally's~~	~~Bud~~	~~4.00~~

Sells

```
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price   REAL,
        FOREIGN KEY(beer) REFERENCES    Beers(name)
                ON DELETE CASCADE);
```

Example: Cascade

Name	Manf
Budweiser	Anheuser-Busch
…	…

Beers

Bar	Beer	Price
Joe's	Budweiser	5.00
Sally's	Budweiser	4.00

Sells

```
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price   REAL,
        FOREIGN KEY(beer) REFERENCES    Beers(name)
                ON UPDATE CASCADE);
```

Example: Set NULL

Name	Manf
~~Bud~~	~~Anheuser-Busch~~
...	...

Beers

Bar	Beer	Price
Joe's	NULL	5.00
Sally's	NULL	4.00

Sells

```
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price   REAL,
        FOREIGN KEY(beer) REFERENCES    Beers(name)
                ON DELETE SET NULL);
```

Example: Set NULL

Name	Manf
Budweiser	Anheuser-Busch
…	…

Beers

Bar	Beer	Price
Joe's	NULL	5.00
Sally's	NULL	4.00

Sells

```
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price   REAL,
        FOREIGN KEY(beer) REFERENCES    Beers(name)
                ON UPDATE SET NULL);
```

Other Declarations for Attributes

- Two other declarations we can make for an attribute are:
  1. NOT NULL means that the value for this attribute may never be NULL.
  2. DEFAULT <value> says that if there is no specific value known for this attribute’s component in some tuple, use the stated <value>.

Other Declarations for Attributes

```
CREATE TABLE Sells (
        bar   CHAR(20),
        beer  VARCHAR(20)  DEFAULT 'HouseBeer',
        price REAL      NOT NULL,
        PRIMARY KEY (bar,beer)
 );
```

```
insert into sells(bar,price)
values ('456',8.9);
select * from sells;
```

	bar	beer	price
▶	456	HouseBeer	8.9

insert into sells(bar)
values ('B');

Attribute based Check: Example

```
CREATE TABLE Sells (
        bar CHAR(20),
        beer CHAR(20)   CHECK (beer IN
        (SELECT name  FROM Beers)),
        price REAL  CHECK (price <= 5.00));
```

Check on beer is like a foreign-key constraint, except:

Check occurs only when the ‘beer’ attribute of a ‘Sells’ tuple is inserted or updated.

Not when a ‘Beers’ tuple is deleted or updated.

Tuple Based Check: Example

Only Joe's Bar can sell beer for more than $5.

```
CREATE TABLE Sells (
        bar         CHAR(20),
        beer  CHAR(20),
        price       REAL,
        CHECK (bar = 'Joe''s Bar' OR
                price <= 5.00));
```

```
ALTER TABLE Sells ADD CHECK(…);
```

```
ALTER TABLE Sells ADD CHECK (bar='Joe’ s Bar' or beer <> 'Miller');
```

example relation schema

Beers(name, manf)

Bars(name, addr, license)

BarDrink

Drinkers(name, addr, phone)

Likes(drinker, beer)

Sells(bar, beer, price)

Frequents(drinker, bar)

```
create table Beers(
        name varchar(20) primary key,
        manf varchar(40) not null);
create table Bars(
        name char(20) primary key,
        addr varchar(40) not null,
        license int not null);
create table Drinkers(
        name char(10) primary key,
        addr varchar(40),
        phone char(8));
```

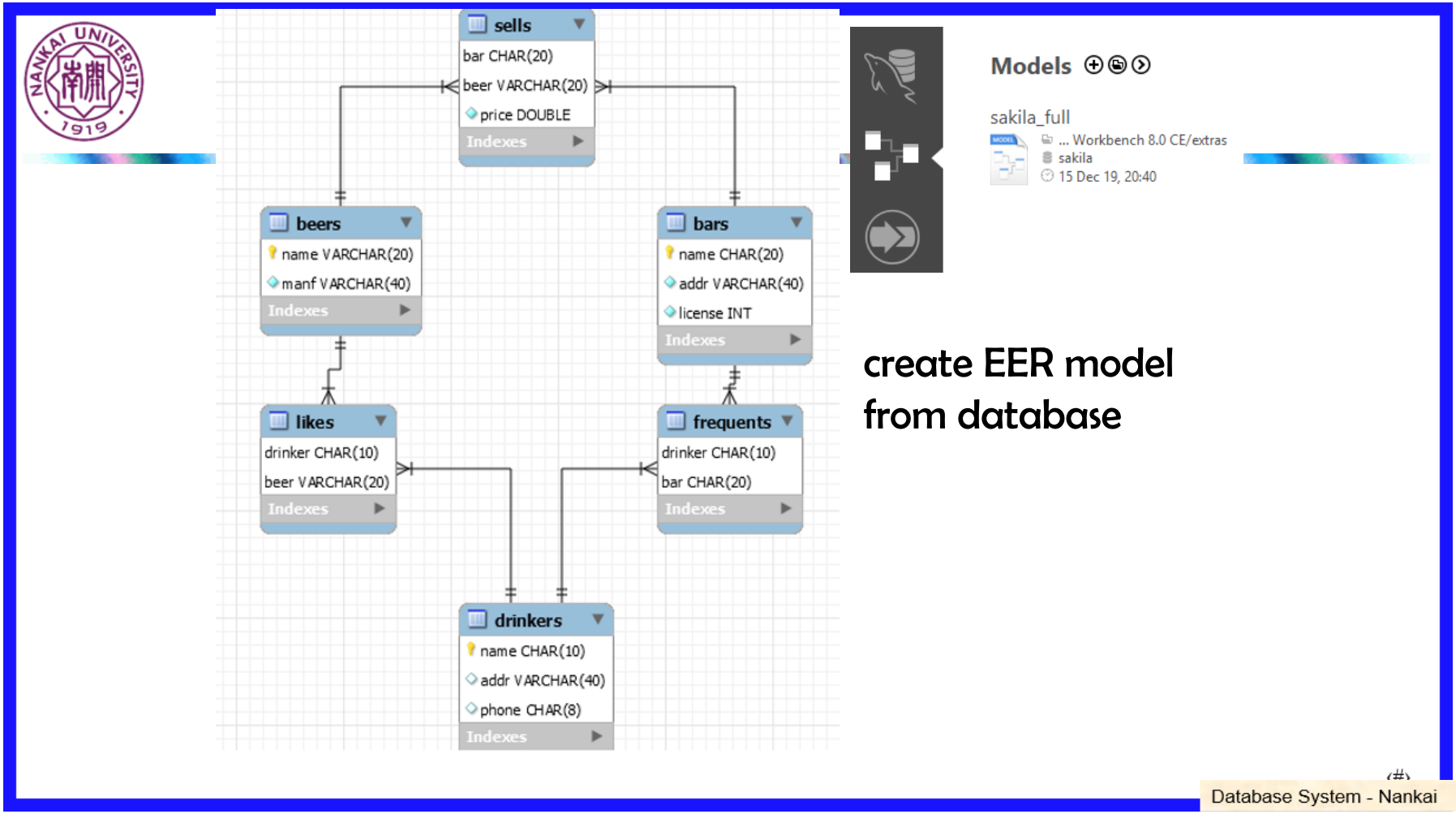
```
Create table Frequents(
    drinker char(10)  references Drinkers(name),
    bar  char(20) references Bars(name),
    primary key(drinker,bar));

create table Likes(
    drinker char(10) references Drinkers(name),
    beer varchar(20) references  Beers(name),
    primary key (drinker,beer));

create table Sells (
     bar   char(20) references Bars(name),
     beer  varchar(20) references Beers(name),
     price REAL NOT NULL,
     primary key (bar,beer));
```

(#)

Database System - Nankai

```
Create table Frequents(
    drinker char(10),
    bar  char(20) ,
    foreign key(drinker) references Drinkers(name),
    foreign key(bar) references Bars(name),
    primary key(drinker,bar));
create table Likes(
    drinker char(10) ,
    beer varchar(20) ,
    foreign key(drinker) references Drinkers(name),
    foreign key(beer) references  Beers(name),
    primary key (drinker,beer));
create table Sells (
    bar   char(20) ,
    beer  varchar(20) ,
    price REAL NOT NULL,
    foreign key (bar) references Bars(name),
    foreign key (beer) references Beers(name),
    primary key (bar,beer));
```

Database System - Nankai


sells
bar CHAR(20)
beer VARCHAR(20)
price DOUBLE
Indexes
beers
name VARCHAR(20)
manf VARCHAR(40)
Indexes
bars
name CHAR(20)
addr VARCHAR(40)
license INT
Indexes
likes
drinker CHAR(10)
beer VARCHAR(20)
Indexes
frequents
drinker CHAR(10)
bar CHAR(20)
Indexes
drinkers
name CHAR(10)
addr VARCHAR(40)
phone CHAR(8)
Indexes
Models
sakila_full
... Workbench 8.0 CE/extras
sakila
15 Dec 19, 20:40
create EER model
from database
Database System - Nankai

Summary

- Declaring Keys
- Foreign Keys
- Enforcing Foreign Key Constraints
- Other Constraints
- Example Relation

多选题 1分

互动交流——不定项选择

假设有如下SQL语句

```
create table students(
    name varchar(20) unique,
    sno varchar(20) unique
);
```

则说明

A 属性name中不允许出现重复值

B 属性sno中不允许出现重复值

C 属性name和sno中不允许出现重复的值组

D 这是错误的写法

提交

多选题 1分

互动交流二——不定项选择

假设有如下SQL语句

```
create table students(
    name varchar(20) primary key,
    sno varchar(20) primary key
);
```

则说明

A 属性name为主键

B 属性sno为主键

C 属性name和sno共同构成主键

D 这是错误的写法

提交

多选题 1分

互动交流三——不定项选择

对于一个表来说，unique和primary key的区别是

A 没有区别

B unique可以出现空值，primary key不可以出现空值

C unique可以有多个，primary key 只可以有一个

D unique可以出现多个空值，primary key只可以出现一个空值

提交

多选题 1分

互动交流四—不定项选择

假设有如下SQL语句

```
create table students(
    name varchar(20),
    sno varchar(20),
    unique(name,sno)
);
```

则说明

A 属性name中不允许出现重复值

B 属性sno中不允许出现重复值

C 属性name和sno中不允许出现重复的值组

D 这是错误的写法

提交

多选题 1分

互动交流五—不定项选择

假设有如下SQL语句

```
create table students(
    name varchar(20),
    sno varchar(20),
    primary key (name,sno)
);
```

则说明

A 属性name中不允许出现空值

B 属性sno中不允许出现空值

C 属性name和sno中不允许出现重复的值组

D 这是错误的写法

提交

多选题 1分

互动交流六——不定项选择

假设有表Students(SID, name)和表Courses(CID, name)，想创建表Enrollments说明一位学生选了一门课，则正确的SQL语句是

A
```
create table Enrollments(
    SID varchar(10),
    CID varchar(10)
);
```

B
```
create table Enrollments(
    SID varchar(10) references Students(SID),
    CID varchar(10) references Courses(CID),
    primary key (SID, CID)
);
```

C
```
create table Enrollments(
    SID varchar(10),
    CID varchar(10),
    primary key (SID, CID),
    foreign key (SID) references Students (SID),
    foreign key (CID) references Courses (CID)
);
```

D
```
create table Enrollments(
    SID varchar(10),
    CID varchar(10),
    primary key (SID, CID),
    foreign key SID references Students (SID),
    foreign key CID references Courses (CID)
);
```

提交

讲解-多属性构成外键

```
create table department
    (dept_name      varchar (20),
     building       varchar (15),
     budget         numeric (12,2),
     primary key (dept_name));

create table course
    (course_id      varchar (7),
     title          varchar (50),
     dept_name      varchar (20),
     credits        numeric (2,0),
     primary key (course_id),
     foreign key (dept_name) references department);

create table instructor
    (ID             varchar (5),
     name           varchar (20) not null,
     dept_name      varchar (20),
     salary         numeric (8,2),
     primary key (ID),
     foreign key (dept_name) references department);

create table section
    (course_id      varchar (8),
     sec_id         varchar (8),
     semester       varchar (6),
     year           numeric (4,0),
     building       varchar (15),
     room_number    varchar (7),
     time_slot_id   varchar (4),
     primary key (course_id, sec_id, semester, year),
     foreign key (course_id) references course);

create table teaches
    (ID             varchar (5),
     course_id      varchar (8),
     sec_id         varchar (8),
     semester       varchar (6),
     year           numeric (4,0),
     primary key (ID, course_id, sec_id, semester, year),
     foreign key (course_id, sec_id, semester, year) references section,
     foreign key (ID) references instructor);
```

Pages from Database System Concepts - Figure 3.1 SQL data definition for part of the university database

Database System - Nankai

多选题 1分

互动交流七—不定项选择

假设表A的主键是a,表B的外键b参照了A上的a，则以下说法正确的是

A 在对B表进行更新时，如果b上的值是A中的a不存在的值，则此操作被拒绝

B 在对表A进行更新时，如果a上的值是B中的b不存在的值，则此操作被拒绝

C 在对A表进行更新时，如果对a的更新会使B中b出现a中不存在的值，则此操作默认被拒绝

D 在对B表进行更新时，如果对b的更新会使A中a出现b中不存在的值，则此操作默认被拒绝

提交

多选题 1分

互动交流八—不定项选择

假设表A的主键是a,表B的外键b参照了A上的a，则以下创建表B的SQL语句中正确的是？请弹幕说出正确选项的含义是什么

A

```
create table B(
    ID varchar(10) primary key,
    b varchar(10) references A (a)
        on delete cascade
        on update set null
);
```

B

```
create table B(
    ID varchar(10) primary key,
    b varchar(10) references A (a)
        on delete cascade
        on update cascade
        on delete set null
        on update set null
);
```

C

```
create table B(
    ID varchar(10) primary key,
    b varchar(10) references A (a)
        on delete cascade
);
```

D

```
create table B(
    ID varchar(10) primary key,
    b varchar(10) references A (a)
        on update cascade
);
```

提交

多选题 1分

互动交流九—不定项选择

以下说法正确的是

A 外键约束可以转换成基于属性的check约束，这两种约束的效果是一样的

B 基于属性的约束可以转换为基于元组的约束

C 在创建表后，可以通过ALTER TABLE增加表上的约束

D 基于属性的约束可能会由于约束中其他值的改变导致违反约束

提交

讲解

```
create table B(
    ID varchar(10) primary key,
    b varchar(10) check (b IN (select a from A))
);
```

Attribute based check

```
create table B(
    ID varchar(10) primary key,
    b varchar(10),
    check (b IN (select a from A))
);
```

Tuple based check

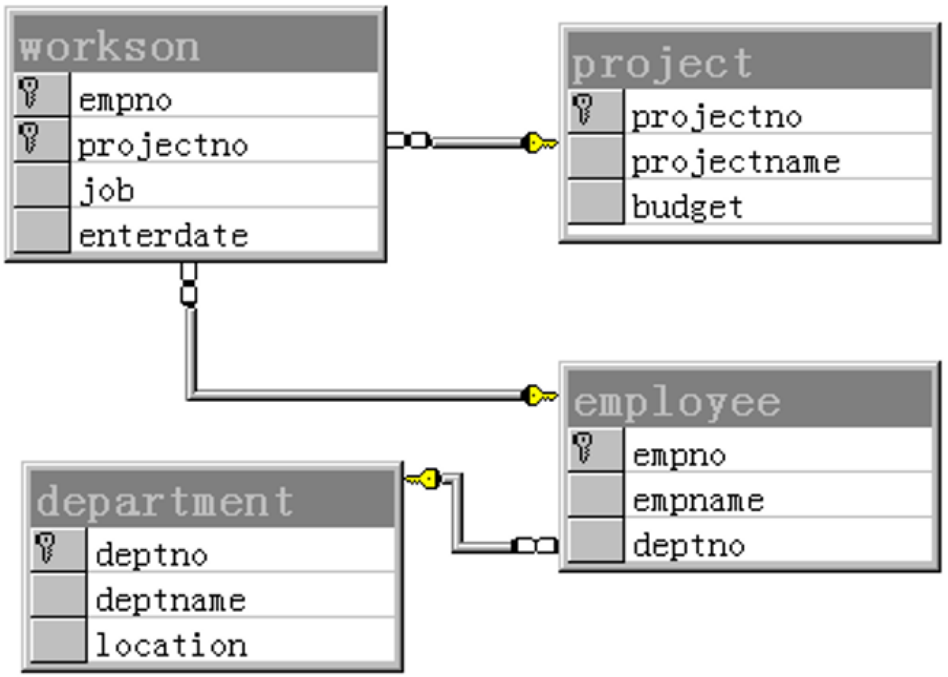
alter table B add check (b in (select a from A));

Changes of A may cause violation on B' s check constraint

Database System-SQL Introduction-part3

SQL Queries & SQL Injection

样本数据库结构


workson
empno
projectno
job
enterdate
project
projectno
projectname
budget
employee
empno
empname
deptno
department
deptno
deptname
location

样本数据库

Department

deptno	deptname	location
d1	开发部	天津
d2	财务部	北京
d3	市场部	广东

Project

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

Workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

(#)

Database System - Nankai

SQL Queries

Principal form:

SELECT desired attributes

FROM relations

WHERE condition about tuple variables;

Example

Beers(name, manf)

- What beers are made by Anheuser-Busch?

```
SELECT   name
FROM          Beers
WHERE   manf = 'Anheuser-Busch';
```

- Note: single quotes for strings.

  where manf in ('A co.','B co.')

Example

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

	empname
▶	冯新
	刘国风

部门d1中所有员工的姓名?

```
select empname
from employee
where deptno='d1';
```

```
where deptno in ( 'd1' );
```

Star as List of All Attribute

When there is one relation in the FROM clause, * in the SELECT clause stands for "all attributes of this relation."

Beers(name, manf)

```
SELECT  *
FROM    Beers
WHERE   manf = 'Anheuser-Busch';
```

name	manf
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch

Renaming columns

If you want the result to have different attribute names, use "AS <new name>" to rename an attribute. ("AS" could be omitted)

Beers(name, manf)

```
SELECT  name AS beer
FROM    Beers
WHERE   manf = 'Anheuser-Busch';
```

beer
Bud
Bud Lite
Michelob

Expressions as Values in Columns

Any expression that makes sense can appear as an element of a SELECT clause.

Sells(bar, beer, price)

```
SELECT bar, beer, price*120 AS priceInYen
FROM   Sells;
```

bar	beer	priceInYen
Joe's	Bud	300
Sue's	Miller	360
…	…	…

Note: no **WHERE** clause is OK.

- If you want an answer with a particular string in each row, use that constant as an expression.

Likes(drinker, beer)

```
SELECT  drinker, 'likes Bud' AS whoLikesBud
FROM    Likes
WHERE  beer = 'Bud';
```

drinker	whoLikesBud
Sally	likes Bud
Fred	likes Bud
…	…

Example

Sells(bar, beer, price)

Find the price Joe's Bar charges for Bud.

```
SELECT   price
FROM     Sells
WHERE    bar = 'Joe''s Bar' AND beer = 'Bud';
```

Note: two single-quotes in a character string represent one single quote.

Comparison operators: =, <>, <, >, <=, >=

Conditions in **WHERE** clause can use logical operators.(AND, OR, NOT)

Remember: SQL is ***case insensitive.*** (保留字，关系名，属性名)

Only inside quoted strings does case matter. (各系统不同)

Example

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

	empname
▶	徐唱
	李静

d2部门中所有工号小于10000的员工姓名?

```
select empname
from employee
where empno<10000 and
deptno='d2';
```

SQL Injection

- SQL Injection 的起因通常是因为程序利用字符串构造的方式执行
- 正常查询网站数据时，将攻击资料库的指令藏于查询指令中
- 攻击者可穿透防火墙，绕过身份认证机制，取得资料库使用权限，进而窃取资料或窜改、破坏资料

www.sample.com V.S. www.sample.com?testid=23

Database System - Nankai

影响的系统

- 通过ASP、PHP与JSP等程序代码，攻击破坏各种SQL资料。
- 影响的系统包括MSSQL、MySQL、Oracle、Sybase和DB2等。

SQL Injection原理

- 一般输入帐号密码的网站SQL语法:

  select * from member where UID =' "& request("ID") &" '

  and Passwd =' "& request("Pwd") & " '

- 如果正常使用者帐号A123456789， 密码1234

  select * from member where UID ='A123456789'

  and Passwd='1234'

- 输入的帐号和密码等信息会取代ASP(or PHP、JSP)中的变量，并由两个单引号(' ')所包住

SQL Injection原理

- 若攻击者已知系统中已有一个Admin的管理者帐号，则输入Admin ‘-- ，即可不需输入密码而进入资料库

select * from member where UID ='Admin'-- ' And Passwd =' '

- 注: -- 单行注释（符号后的任何叙述被当作注释）
  (以上面为例，And子句将被SQL视为注释)

帐号： ‘ or 1=1 --，密码任意输入

select * from member where UID ="or 1=1 --And Passwd =' '

Summary

- SQL Queries
- SQL Injection

多选题 1分

互动交流——不定项选择

SQL语言的一次查询的结果是一个?

A 数据项

B 记录

C 表

提交

Database System - Nankai

多选题 1分

互动交流二——不定项选择

在SQL语言中，查询学生的全部信息使用如下哪一条命令

A SELECT * FROM 学生

B SELECT ALL FROM 学生

C SELECT 学号，姓名 FROM 学生

提交

Database System - Nankai

多选题 1分

互动交流三——不定项选择

对于以下数据库表，
想要查询除部门d1
外所有员工的姓名可
以使用的SQL语句是

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

A
```
select empname
from employee
where deptno <> 'd1';
```

B
```
select empname
from employee
where deptno='d2' or
deptno = 'd3';
```

C
```
select empname
from employee
where deptno in
('d2','d3');
```

D
```
select empname
from employee
where deptno not in ('d1');
```

提交

Database System - Nankai

主观题 10分

互动交流四

在以下表中，如何查找所有职员的工号?

Workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

提交

Database System - Nankai

讲解

Workson

empno	projectno	job	enterdate
2581	p3	分析员	98-10-15
9031	p1	管理员	98-4-15
9031	p3	职员	97-11-15
10102	p1	分析员	97-1-10
10102	p3	管理员	99-1-1
18316	p2	职员	98-2-15
25348	p2	<NULL>	98-6-1
28559	p1	<NULL>	98-8-1
28559	p2	职员	99-2-1
29346	p1	职员	98-1-4
29346	p2	<NULL>	97-12-15

```
select empno
from workson
where job= '职员';
```

empno
9031
18316
28559
29346

主观题 10分

互动交流五

假设如下左侧展示了关系，并且假设人民币兑加元汇率为：1加元=5人民币，则什么样的SQL查询语句可以得到右侧显示的关系？

Project

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

	projectno	projectname	budget_in_CAD
▶	p1	网络布线	24000.0000
	p2	软件升级	19000.0000
	p3	系统开发	37120.0000

提交

Database System - Nankai

讲解

Project

projectno	projectname	budget
p1	网络布线	120000
p2	软件升级	95000
p3	系统开发	185600

	projectno	projectname	budget_in_CAD
▶	p1	网络布线	24000.0000
	p2	软件升级	19000.0000
	p3	系统开发	37120.0000

```
select projectno, projectname, budget/5 as budget_in_CAD
from project;
```

主观题 10分

互动交流六

假设如下左侧展示了关系，则什么样的SQL查询语句可以得到右侧显示的关系?

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

	empno	empname	dept
▶	18316	冯新	部门d1
	28559	刘国风	部门d1

提交

Database System - Nankai

讲解

Employee

empno	empname	deptno
2581	徐唱	d2
9031	李静	d2
10102	王闻刚	d3
18316	冯新	d1
25348	张风	d3
28559	刘国风	d1
29346	赵东生	d2

	empno	empname	dept
▶	18316	冯新	部门d1
	28559	刘国风	部门d1

```
select empno,empname, '部门d1' as dept
from employee
where deptno='d1';
```

多选题 1分

互动交流七——不定项选择

假设想要对数据库为MySQL的某网站进行SQL Injection，并假设已知系统中有一个Administrator的管理者账号，则在用户处输入什么可以随意输入密码进入资料库?

A Admin'--

B Administrator'--

C or 1=1'#

D 'or1=1#

提交

Database System - Nankai

讲解

Admin'-- → select * from member where UID ='Admin'-- ' And Passwd =' '

Administrator'-- →

select * from member where UID ='Administrator'-- ' And Passwd =' '

or 1=1'# → select * from member where UID ='or 1=1' # ' And Passwd =' '

'or1=1# → select * from member where UID = ' 'or 1=1 # ' And Passwd =' '

Administrator'# →

select * from member where UID ='Administrator'# ' And Passwd =' '