目录

一、概述

1. 数据库、数据库管理系统、数据库系统的基本概念

(1) 数据库(DataBase，简称 DB)：数据库是长期储存在计算机内的、有组织的、可共享的数据集合。(数据库中的数据按一定的数据模型组织、描述和储存,具有较小的冗余度、较高的数据独立性和易扩展性,并可为各种用户共享。)

(2) 数据库系统(DataBase System，简称 DBS)：数据库系统是由数据库、数据库管理系统(及其应用开发工具)、应用程序和数据库管理员组成的存储、管理、处理和维护数据的系统。

(3) 数据库管理系统(DataBase Management System，简称 DBMS)：数据库管理系统是位于用户与操作系统之间的一层数据管理软件,用于科学地组织和存储数据、高效地获取和维护数据。

(DBMS 的主要功能包括数据定义功能、数据操纵功能、数据库的运行管理功能、数据库的建立和维护功能。)

2. 数据库作为数据存储方法的特点,优缺点

(1)特点

①数据抽象 ②可靠性 ③高效

(2)优缺点

优点(PPT)

①有效地控制数据冗余

②同一个数据可以表现为不同的信息

③共享

④完整性、一致性、安全性高

优点(课本)

①数据结构化

②数据的共享性高、冗余度低且易扩充

③数据独立性高

④数据由数据库管理系统统一管理和控制

缺点

①需要购买 DBMS 数据库管理系统,有花费

②需要满足一定的硬件设备

③需要花时间和精力去学习使用和维护 DBMS

二、关系模型

1. 数据模型、关系模型概念及其意义

(1)数据模型的概念

数据模型是数据库中用来对现实世界进行抽象的工具,是数据库中用于提供信息表示和操作手段的形式构架。

(2)数据模型的作用

一般地讲,数据模型是严格定义的概念的集合。这些概念精确描述了系统的静态特性、动态特性和完整性约束条件。(因此数据模型通常由数据结构、数据操作和完整性约束三部分组成。)

(3)关系模型的概念

关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。(在用户观点下,关系模型中数据的逻辑结构是一张二维表,它由行和列组成。)(关系模型基于关系的数学概念,该关系在物理

上表示为表格。)

(用二维表格结构表示实体集,外键表示实体间联系的数据模型称为关系模型。)

(4)关系模型的意义

用关键码而不是用指针导航数据，表格简单，用户易懂，编程时并不涉及存储结构、访问技术等细节。

2. 关系模型的数据结构、数据操作和完整性约束三要素

(1)数据结构

关系模型的数据结构非常简单,只包含单一的数据结构——关系。在用户看来,关系模型中数据的逻辑结构是一张扁平的二维表。

(2)数据操作

关系操作的特点是集合操作方式,即操作的对象和结果都是集合。

(3)完整性约束

①1 实体完整性

实体完整性要求每个表都有唯一标识符,每一个表中的主键字段不能为空或者重复的值。

②2 参照完整性

参照完整性要求关系中不允许引用不存在的实体。设定相应的更新删除插入规则来更新参考表。

例如表 sutdent(有 id,username,password),表 sutdent_info(有 id,name,age,sex)。其中表 sutdent_info 参照了表 student。id 作为外键。那么当 student 表删除一行时,表 student_info 对应的 id 那一列将被删除或者置空(根据设定的规则而定)。同样,表 student_info 增加一行,其中的 id 必须等于 student 表中的 id。

③ 用户定义的完整性

用户自定义完整性是针对某一具体关系数据库的约束条件,它反映某一具体应用所涉及的数据必须满足的语义要求。

3. 关系模型特点、优缺点

(1)优点

① 关系模型与非关系模型不同,它是建立在严格的数学概念的基础上的。

② 关系模型的概念单一,数据结构简单、清晰,用户易懂易用。(无论实体还是实体之间的联系都用关系表示。对数据的检索和更新结果也是关系(即表))

③ 关系模型的存取路径对用户透明,从而具有更高的数据独立性、更好的安全保密性,也简化了程序员的工作和数据库开发建立的工作。

(2)缺点

① 对“现实世界”实体的表达能力比较弱

② 语义过载

③ 不能很好的支持业务规则

④ 由于存取路径对用户透明,查询效率往往不如非关系数据模型。因此,为了提高性能,必须对用户的查询请求进行优化,增加了开发数据库管理系统的难度。(课本)

4. 关系代数对数据查询的表达

(1)基本概念

- 超码 bai(super key):在关系中能唯一标识元组的属性集称为关系模式的超键。
- 候选码(candidate key):不含有多余属性的超键称为候选键。
- 主码(primary key):用户选作元组标识的一个候选键称为主键。

- 备选码(alternate key)
- 外码(foreign key)

(2)关系代数(作业 1)

并、差、笛卡尔积、投影和选择 5 种运算为基本的运算。其他 3 种运算,即交、连接和除,均可以用这 5 种基本运算来表达。

①选择

表 2.5 条件表达式中的运算符

运算符		含义
比较运算符	>	大于
	≥	大于等于
	<	小于
	≤	小于等于
	=	等于
	<>	不等于
逻辑运算符	¬	非
	∧	与
	∨	或

②投影

投影之后可能会取消某些元组(因为可能会出现重复行)

③连接 $R \underset{A\theta B}{\bowtie} S = \sigma_{r[A]\theta S[B]}(R \times S)$

相等连接 $R \underset{A=B}{\bowtie} S=\{\overparen{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A]=t_s[B]\}$

自然连接 $R \bowtie S=\{\overparen{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[X]=t_s[X]\}$

外连接

左外连接

右外连接

④除

给定关系R(X,Y)和S(Y,Z)，其中X，Y，Z为属性组。 R中的Y与S中的Y可以有不同的属性名，但必须出自相同的域集。R与S的除运算得到一个新的关系P(X)，P是R中满足下列条件的元组在X属性列上的投影：元组在X上分量值x的象集Y_x包含S在Y上投影的集合。记作：

三、结构化查询语言 SQL

SQLfunctions Verb	SQLfunctions Verb
Data query SELECT	Data query SELECT
Data define CREATE, DROP, ALTER	Data define CREATE, DROP, ALTER
Data manipulation INSERT, UPDATE, DELETE	Data manipulation INSERT, UPDATE, DELETE
Data control GRANT, REVOKE	Data control GRANT, REVOKE

Data type Declarations	Data type Declarations
boolean BOOLEAN	boolean BOOLEAN
character CHAR VARCHAR	character CHAR VARCHAR
bit BIT BIT VARYING	bit BIT BIT VARYING
exact numeric NUMERIC DECIMAL INTEGER SMALLINT	exact numeric NUMERIC DECIMAL INTEGER SMALLINT
approximate numeric FLOAT REAL DOUBLE PRECISION	approximate numeric FLOAT REAL DOUBLE PRECISION
datetime DATE TIME TIMESTAMP	datetime DATE TIME TIMESTAMP
interval INTERVAL	interval INTERVAL
large objects CHARACTER LARGE OBJECT BINARY LARGE OBJECT	large objects CHARACTER LARGE OBJECT BINARY LARGE OBJECT

1. 基本查询,多表连接查询,自连接,分组与聚集函数,子查询

Condition Predicates(谓词)	Condition Predicates(谓词)
comparison Comparison operators(=,>,<,<>,<=,>=,!=)	comparison Comparison operators(=,>,<,<>,<=,>=,!=)
range BETWEEN AND, NOT BETWEEN AND	range BETWEEN AND, NOT BETWEEN AND
set membership IN, NOT IN	set membership IN, NOT IN
pattern match LIKE, NOT LIKE	pattern match LIKE, NOT LIKE

null IS NULL, IS NOT NULL	null IS NULL, IS NOT NULL
logical operators AND, OR,NOT	logical operators AND, OR,NOT

- x=null(结果为 null)
- 模糊匹配:_表示单个通配符,%表示任意多个通配符。

```
select sno,sname,age,dno
from student
where sname='钱\_%' escape '\';
```

- 使用 DISTINCT 消除重复项
- T AND U = U　F AND U = F　U AND U = U　T OR U = T

  F OR U = U　U OR U = U　NOT U = U
- 排序——order by

  ASC(升序)—ascending order, default order

  DESC(降序)—descending order

  把 null 值当成最大值

```
select *
from sc
order by sno,score desc;
```

- 聚集函数(COUNT, SUM, AVG, MAX, MIN)

  空值不计算在平均值，求和中

```
select count(distinct sno)
from sc;
```

- 分组(GROUP BY)

  限制条件(HAVING) (空值不参与运算)
- 子查询

  in (返回结果为多个值或单个值)

  = (返回结果为单个值)
- 多值集合比较

  >ANY, <ANY, >=ANY, <=ANY, =ANY, <>ANY/!=ANY

  >ALL, <ALL, >=ALL, <=ALL, =ALL(no meaning), <>ALL/!=ALL

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>=MIN

ALL	- -	NOT IN	<MIN	<=MIN	>MAX	>=MAX

- 连接运算（多表查询）
  - N 个表做连接运算，至少有 n- 1个条件，否则会产生笛卡尔积。
  - 自连接

```
Select c2.cname
From course c1, course c2
Where c1.cpno=c2.cno
    And c1.cname=’数据库系统’;
```

  - 外连接

左外连接

```
Select s.sno, s.sname, d.dname
From student s left outer join department d
 on s.dno=d.sno;
```

右外连接

```
Select s.sno, s.sname, d.dname
From student s right outer join department d
 on s.dno=d.sno;
```

全外连接

```
Select s.sno, s.sname, d.dname
From student s full outer join department d
 on s.dno=d.sno;
```

```
select col1,FUN(col2)
from R
where F
(子查询中没有必要排序)order by col1 desc,col2(默认升序)
group by col1
having


select
from R
where x IN (select
        from R
        where)
```

返回一个值：单行单列——=
返回多个值：多行多列——in

不相关子查询，如：查询某一门课的先修课的课程名

```
select R1.col1,R2.col2
from R1,R2(自然连接)
where R1.col1=R2.col1

select col1,col2
from R1,R2
where R1.col1=R2.col2(等值连接)


ISO
select
from R1 natural join R2

select
from R1 join R2 using col1

select
from R1 join R2 on R1.col1=R2.col2

select
from R1 left outer join R2 on R1.col1=R2.col2

select
from R1 full outer join R2 on R1.col1=R2.col2
```

总结
1、查询结果和查询条件在同一个关系中——单表查询

2、查询结果在一个关系中，但查询条件用到了另外一个关系——子查询，连接

3、查询结果不在一个关系——连接

- EXISTS and NOT EXISTS

  相关子查询（执行效率低，尽量用不相关子查询）

  ♂ 可以用 IN/NOT IN 替代的查询

  Select sname

  From student

  Where not exists

  (select *

From sc

Where sno=student.sno and cno='001');

② 不可以用 IN/NOT IN 替代的查询（对全称量词的表达- - >转化为存在量词、蕴含式）

- 集合运算的并(UNION)、交(INTERSECT)、差(EXCEPT)

2. 创建表,表创建后增加、删除约束

(1)创建表

- 创建表的同时增加约束

```
--DROP TABLE department CASCADE;
--DROP TABLE student CASCADE;
--DROP TABLE course CASCADE;
--DROP TABLE sc CASCADE;

--department(dNo,dName,officeRoom,homepage)
--student(sNo,sName,sex,age,dNo)
--course(cNo,cName,cPNo,credit,dNo)
--sc(sNo,cNo,score,recordDate)

CREATE TABLE department(
  dNo           CHAR(2)         NOT NULL UNIQUE,
  dName         VARCHAR(20),
  officeRoom    VARCHAR(40),
  homePage      VARCHAR(80),
  PRIMARY KEY(dNo)
);

CREATE TABLE student(
  sNo           CHAR(6)         NOT NULL UNIQUE,
  sName         VARCHAR(20)     NOT NULL,
  sex           CHAR(2)         CHECK (sex IN('男','女')),
  age           INT,
  dNo           CHAR(2),
  PRIMARY KEY(sNo),
  FOREIGN KEY (dNo) REFERENCES  department(dNo)
);

CREATE TABLE course(
  cNo           CHAR(6)         NOT NULL UNIQUE,
  cName         VARCHAR(20)     NOT NULL,
  cPNo          CHAR(6),
  credit        INT,
  dNo           CHAR(2),
  PRIMARY KEY(cNo),
  FOREIGN KEY (cPNo) REFERENCES  course(cNo),
  FOREIGN KEY (dNo)  REFERENCES  department(dNo)
);

CREATE TABLE sc(
  sNo           CHAR(6)         NOT NULL,
  cNo           CHAR(6)         NOT NULL,
  score         INT,
  recordDate    date            DEFAULT current_date,
  PRIMARY KEY(sNo,cNo),
  FOREIGN KEY (sNo) REFERENCES  student(sNo),
  FOREIGN KEY (cNo) REFERENCES  course(cNo)
);
```

- 创建表的同时不增加约束

```
CREATE TABLE press(
        pno char(8) not null,
        pname varchar(100),
        address varchar(100),
        tele varchar(50)
);

CREATE TABLE book(
        bno     char(6) not null,
        bname   varchar(100),
        author  varchar(100),
        ISBN    varchar(50) not null,
        pno     char(10),
        edition varchar(50),
        pubDate varchar(50)
);
```

(2)增加约束

```
alter table press add unique(pno);
alter table press add primary key(pno);

alter table book add unique(bno);
alter table book add unique(ISBN);
alter table book add primary key(bno);
alter table book add foreign key(pno) references press(pno);
```

(3)删除约束

```
ALTER TABLE table_name
DROP CONSTRAINT MyUniqueConstraint;
```

3. 插入数据,更新数据,删除数据

(1)插入数据

```
INSERT INTO department VALUES('01','信息学院','行政楼 409','www.xxx.edu.cn');
INSERT INTO department VALUES('02','软件学院',null,null);
INSERT INTO student VALUES('170101','宁灿',  '女',19,'01');
INSERT INTO course VALUES('030101','高等数学',null,2,'03');
INSERT INTO sc VALUES ('170101','030101',91,to_date('2016- 01- 08','yyyy- mm- dd'));
```

```
INSERT INTO sc VALUES ('170104','030101',null,null);
```

- 插入子查询结果

```
CREATE TABLE s_score(
sNo CHAR(6),
sName VARCHAR(20)
avg decimal(5,2),
);
INSERT INTO s_score(sNo,sName,avg)
(select s.sNo,s.sName,v.avgscore
from student s,(select sNo,avg(score) from sc group by sNo)
as v(sNo,avgscore)
where s.sNo=v.sNo);
```

(2)更新数据

```
update student
set sex='男'
where sno='001';
```

(3)删除数据

```
delete from student
where sno='001';
```

4. 视图创建与应用

(1) 视图的概念

视图是从一个或几个基本表导出的表。视图本身不独立存储在数据库中，是一个虚表。即数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的基本表中。(课本第 5 版 P121)

视图是一个虚表，它仅仅是把我们需要访问的某些数据以一种特殊的类似于表的形式展现出来。(老师)

⑵ 通过视图访问数据库中数据的优缺点

优点

⑴ 简化查询

⑵ 安全性

优点【课后答案】

视图能够简化用户的操作；

视图使用户能以多种角度看待同一数据；

视图对重构数据库提供了一定程度的逻辑独立性；

视图能够对机密数据提供安全保护。

缺点

⑴ 视图不会帮助我们提高查询性能

⑵ 视图的更新是受到限制的

- 理论上可更新，实际上也可更新
- 理论上可更新，实际上不可更新（聚集函数）
- 理论上不可更新，实际上不可更新

```
create view is_student
as
(select sno,sname,sex,age,dno
from student
where dno=(select dno
        from department
        where dname='信息学院'));

select * from is_student;
drop view is_student;

create view is_student
as
(select sno,sname,sex,age,dno
from student
where dno=(select dno
        from department
        where dname='信息学院'))
with check option;

select * from studnet;
select * from is_student;
delete from student where sno='190101';

insert into is_student
values('190101','佳润',null,null,'02');

drop view s_score;
create view s_score as
(select s.sno,s.sname,v.avgscore
        from student s,(select sno,avg(score)
                from sc group by sno) as v(sno,avgscore)
        where s.sno=v.sno);
select * from s_score;

update s_score
set avgscore = avgscore+10
where sno='XXX';

update s_score
set sname='杨何'
where sno='170104';
```

四、数据库应用

1. JDBC 连接数据库过程步骤

- PPT

(1)加载 JDBC driver 驱动

(2)和数据源连接

(3)连接成功后,执行 SQL 语句

- 网 1

(1)加载 JDBC driver 驱动

(2)连接数据库

(3)创建 Statement 对象

(4)执行 sql 语句

(5)打印出结果

- 网 2

(1)在项目中导入 java.sql 包

(2)加载数据库驱动程序

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

(3)定义数据库的链接地址

```
String  url="jdbc:mysql://localhost/studentserverTimezone=GMT%2B8&useSSL=false";
String databasename="root";
String pass="123456";
```

(4)得到与数据库的连接对象

```
con = DriverManager.getConnection(url,databasename,pass);
```

(5)声明 sql 语句

```
select * from rj1602 where Sno='201616040212';
```

(6)得到语句对象

```
Statement sql
```

(7)执行 sql 语句

```
sql = con.CreateStatement();
sql.executeQuery("select * from rj1602 where Sno='201616040212'");
```

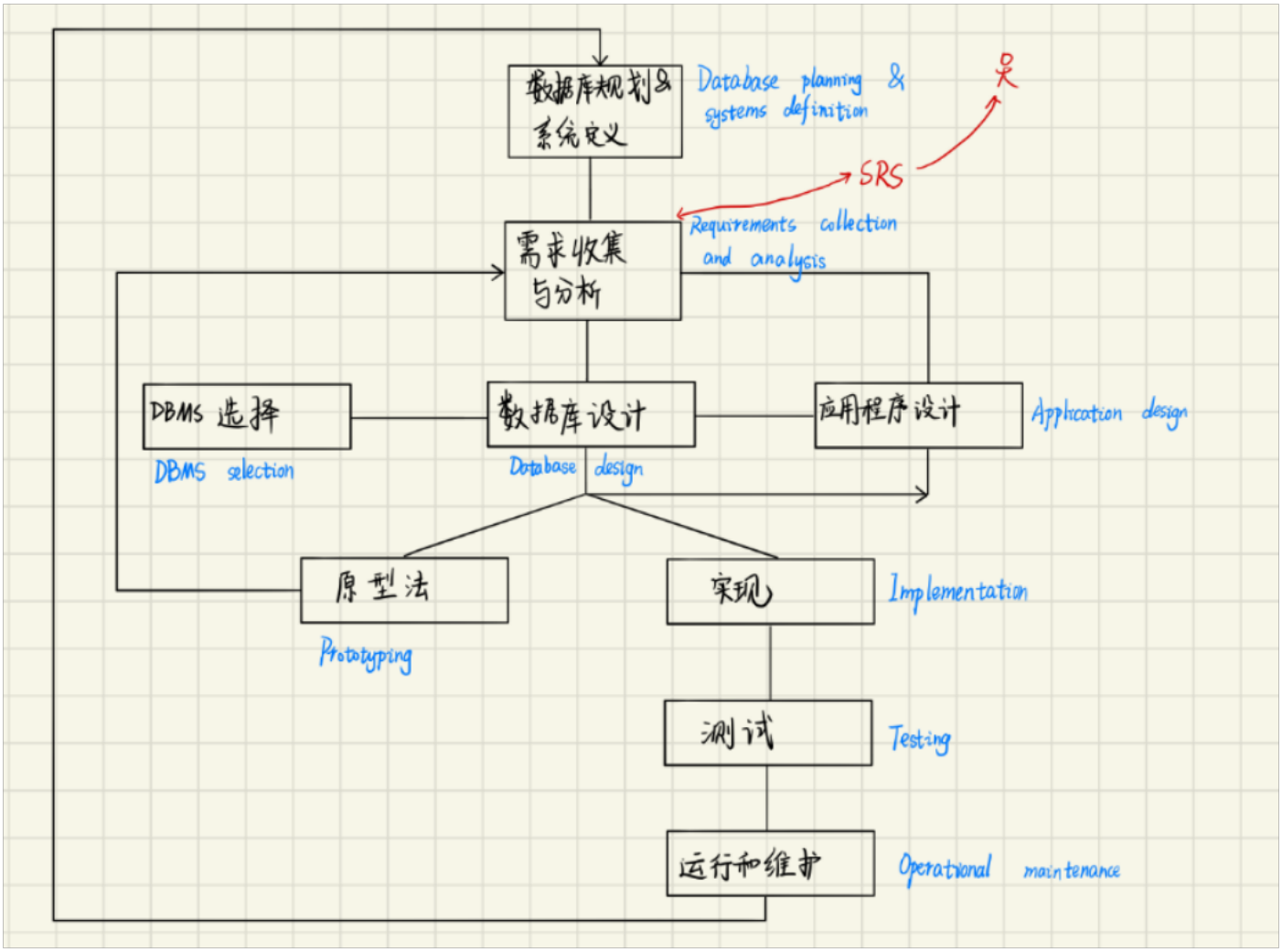
(8)处理 sql 语句的返回结果

(9)关闭对象

2. JAVA 数据库应用基本方法

五、数据库设计

1. 数据库系统生存周期模型及各阶段主要任务与目标



(1)数据库规划和系统定义

① 数据库规划——定义任务和目标

② 系统定义——定义系统边界、确定用户视图

(2)数据收集与分析

收集和分析有关数据库应用程序要支持的组织部分信息的过程,并使用该信息来识别用户对新系

统的需求。

(结果:需求规格说明书)

(3- 1)数据库设计

把需求转化为将来所采用的 DBMS 能够支持的模型。

或:

根据将来所采用的DBMS能够支持的模型,把需求收集与分析获取的结果转化为将来DBMS能够支持的模型。

(数据库设计的三个阶段: ①概念设计 ②逻辑设计 ③物理设计)

(3- 2)DBMS 选择(逻辑设计阶段)

(3- 3)应用程序设计

(原型法)

(4)实现

(5)测试

(6)运行和维护

2. 概念设计任务与目标,概念设计技术—ER 建模

(1) 概念设计任务与目标

概念设计对需求负责。

以另外一种形式对用户的需求规格说明书加以重新表述,目的是消除以自然语言表述的需求规格说明书中的二义性、挖掘隐含需求等问题。

结果:概念模型。

- 任务——把用户以自然语言为主的需求规格说明书用另外一种形式加以重新表述。

(要求:无二义性、非技术性。)

- 目标－－

  ①消除用户以自然语言为主的需求规格说明书中的二义性、隐含需求等问题。

  ②重新表述的方式能够使得设计人员、程序员和用户能够看得懂,并且进行交流和讨论。

(2) 概念设计技术－ER 建模

- 实体　实体型
- 联系　联系型
- 递归联系　二元联系　多元联系
- 简单属性　复合属性(如:地址)　单值属性(如学号)　多值属性(如电话号码)
  导出属性(属性值可以通过其他的属性计算得到,如:年龄(可以由出生年月计算得到))
- 候选码　主码
- 强实体　弱实体
- (1:1)　(1:*)　(*:*)
- 强制参与　可选参与

3. 逻辑设计任务与目标

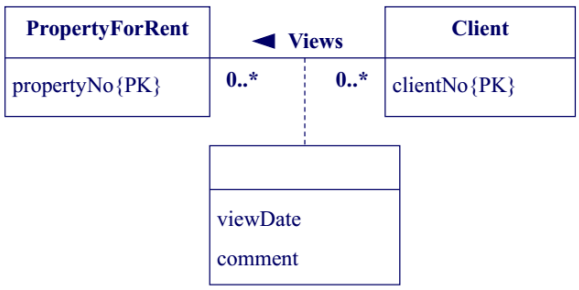
- 任务:把概念设计的结果向将来所采用的 DBMS 能够支持的模型去转化 (把 E- R 模型向关系模型转化)。
- 目标: 将 E- R 模型转化为将来 DBMS 能够支持的模型。
- (结果: 逻辑模型)

4. ER 模型向关系模型转化过程

4.1 一对多联系 (映射为外键)

把一方实体型作为父实体，多方实体作为子实体，把父实体的主关键字副本放到子实体中作为外键来映射。

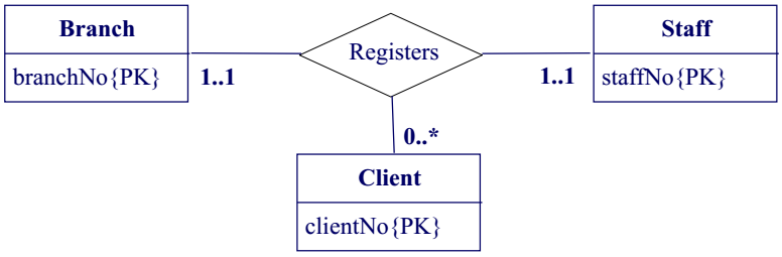
⑴ 多对多的二元联系



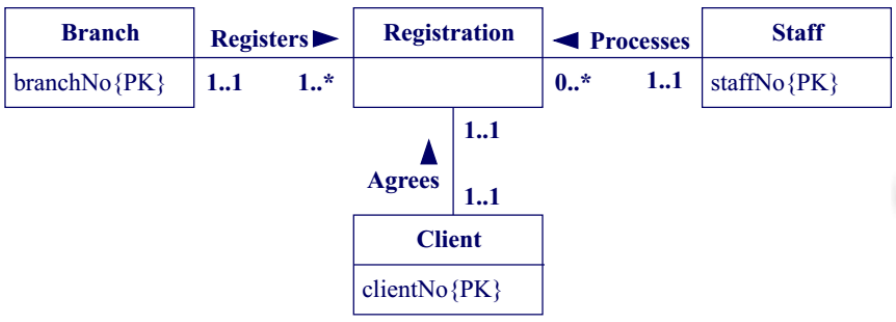
新增加一个弱实体型，让两个强实体型分别与新增加的弱实体型发生两个一对多联系。

⑵ 多对多的递归联系

⑶ 复杂联系类型（多元）



把多元联系映射为实体型（弱实体），让三个实体型分别与新增加的弱实体型发生联系。



⑷ 多值属性

把实体型的多值属性分离出去，组成一个新的实体型（弱实体型），让原来的实体型与新增加的弱实体型发生一对多的二元联系。

4.2 映射

⑴ 强实体

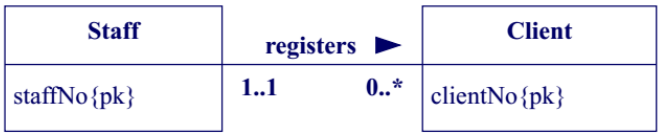
把强实体映射为关系，强实体的属性映射为关系的属性，强实体的候选码作为映射关系的候选码。

⑵ 弱实体

把弱实体映射为关系，弱实体的属性映射为关系的属性。

⑶ 一对多的二元联系

把一方实体型作为父实体，多方实体作为子实体，把父实体的主关键字副本放到子实体中作为外键来映射。



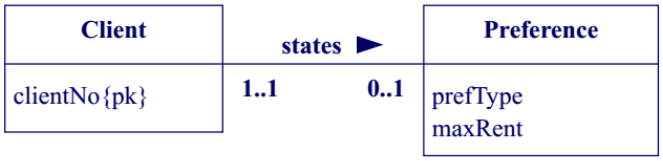
Staff (staffNo, fName, lName, position, sex, DOB)
Primary Key： staffNo
Client(clientNo,fName,lName,telNo,staffNo)
Primary key: clientNo
Alternate key: telNo
Foreign key: staffNo references staff(staffNo)

⑷ 一对一的联系

- 双方都是全部参与－－将两个实体型的属性放到一个关系中

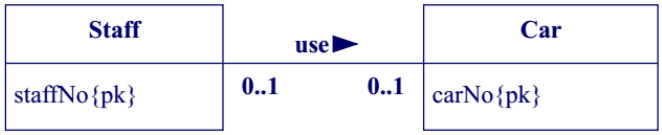
Client(clientNo,fName,lName,telNo,prefType,maxRent,staffNo)
Primary key: clientNo
Alternate key: telNo
Foreign key: staffNo references staff(staffNo)

- 一方是全部参与，一方是部分参与－－把部分参与一方的实体型作为父实体，把全部参与一方的实体型作为子实体，把父实体的主关键字的副本放到子实体中作为外键。



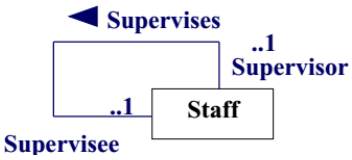
Client(clientNo,fName,lName,telNo,staffNo)
Primary key: clientNo
Alternate key: telNo
Foreign key: staffNo references staff(staffNo)
Preference(clientNo,prefType,maxRent)
Primary key: clientNo
Foreign key: clientNo references Client(clientNo)

- 双方都是部分参与－－把任意一方作为父实体，另一方作为子实体，把父实体的主关键字的副本放到子实体中作为外键。



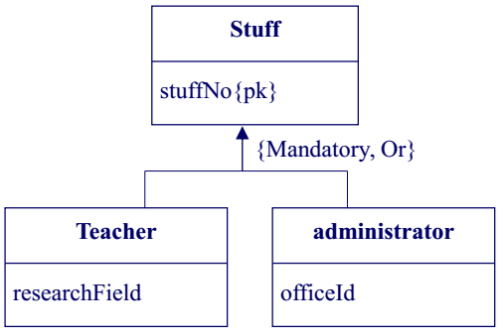
⑸ 一对一的递归联系

- 如果联系双方都是全部参与，只要把实体型的候选码复制一份，放到它后面即可。



- 如果联系双方一方是全部参与，一方是部分参与
  - ✧ 把主关键字副本复制一份，保留一个关系即可
  - ✧ 再分离出一个新的关系
- 如果双方都是部分参与，则创建一个新的关系。

⑹ 超类和子类（可选）



4.3 总结

实体/联系	映射
强实体	创建包含所有简单属性的的关系
弱实体	创建包含所有简单属性的关系（主关键字等到每个主实体的联系映射后再确定）
1：*二元联系	将一方实体的主关键字处理为表示多方实体关系的外部关键字
1：1 二元联系	
(a)双方强制参与	组合为一个实体
(b)一方强制参与	将“可选”方实体的主关键字处理为表示“强制”方实体关系的外部关键字
(c)双方可选参与	无进一步消息任选
超类/子类联系	参照超类/子类映射表
*：*二元联系、复杂联系	创建一个关系表示该联系，该关系包含该联系的所有属性。参与联系的所有实体的主关键字作为该关系的外部关键字
多值属性	创建一个新关系表示多值属性，并将主实体的主关键字作为该关系的外部关键字

⑴ 在一个 ER 模型中，如果存在一个实体与其它实体之间存在两个或更多的一对多联系，很可能存在扇形陷阱问题，扇形陷阱问题可以通过改变实体间的联系次序，从而重新构建模型解决；

⑵ 在一个 ER 模型中，如果在实体联系的通路上存在一个或者多个参与性约束最小值为零的情况，很可能存在深坑陷阱问题，深坑陷阱问题可以通过重新添加被遗漏的联系解决。

5. 关系模式规范化理论

1. 给定关系模式R(U,F)，U={A, B, C, D, E, F, G}，F={A→BC, D→E, AD→F, E→G}

(1) 确定该关系模式的候选码

(2) 将关系模式R逐步规范化为满足BC范式要求的关系模式

解：

(1) AD→U，因此关系模式候选码为AD.

(2) 消除关系模式中非主属性对码的部分依赖，得到

$R1(\underline{A},B,C)$，$F_1\{A\to BC\}$

$R_2(\underline{D},E,G)$，$F_2\{D\to E, E\to G\}$

$R_3(\underline{A},\underline{D},F)$，$F_3\{AD\to F\}$

关系模式R_1和R_3中决定因素均为码，满足BCNF

进一步消除关系模式R_2中的非主属性对码的传递依赖，得到

$R_{21}(\underline{D},E)$，$F_{21}=\{D\to E\}$

$R_{22}(\underline{E},G)$，$F_{22}=\{E\to G\}$

关系模式R_{21}，R_{22}中决定因素均为码，满足BCNF

综上所述，分解之后得到的关系模式R_1，R_{21}，R_{22}，R_3均满足BCNF

- 存在的问题：①冗余问题　②增删改异常问题
- 数据依赖：①函数依赖　②多值依赖　③连接依赖
- 函数依赖
  - ✧ A:B=(1:1)　A←→B

    A:B=(1:*)　B→A（多方决定一方）

    A:B=(*:1)　A→B（多方决定一方）
  - ✧ R(U,F)

    U— 属性集合

    F— 关系模式 R 中属性之间的函数依赖关系
  - ✧ 阿姆斯特朗公理：
    - Reflexivity: If B is a subset of A, then A→B
    - Augmentation: If A→B, then A,C→B,C
    - Transitivity: If A→B and B→C, then A→C.
    - Self-fetermination: A→ A
    - Decomposition: If A→B,C, then A→B and A→C.
    - Union: If A→B and A→C , then A→B,C
    - Composition: If A→B and C→D , then A,C→B,D

⑴ 第一范式——属性不可分

sNo	sName	addr	phoneNo
s01	Lucy	No.1 xxx	02988451234 02886654321
s02	Jam	No.2 xxx	02988451234
s03	Susan	No.3 xxx	02988451234 02168654321

- 增加行（通用）

  优点：不受学生电话号码数量的限制

  缺点：sNo 的主码地位丢失；数据冗余

- 增加列

  优点：sNo 的主码地位不受影响；不会造成数据冗余

  缺点：浪费存储空间

- 拆分成两个关系

  优点：消除冗余；不受学生电话号码数量的限制

  缺点：后续查询需要连接，效率会降低

⑵ 第二范式——关系模式中不能存在非主属性（不是候选码的属性）对码的部份依赖

- 问题

  冗余；增删改异常问题

- 改造

  把部分依赖于码的属性连同它依赖的属性分离出去，组成一个新的关系模式。同时，在原来的关系模式中要保留一个外键。

- 如果候选码是单个属性，不会存在非主属性对码的部分依赖，则至少满足第二范式的要求。

⑶ 第三范式——关系模式中不能存在非主属性对码的传递依赖

- 问题

  冗余；增删改异常问题

- 改造

  分解——把传递依赖于码的属性连同它所依赖的属性分离出去，组成一个新的关系模式。同时，在原来的关系中要保留一个外键。

- ✔ 部分依赖是一种特殊的传递依赖。

⑷ BC 范式－－关系模式中不能存在非主属性对码的部分依赖和传递依赖，主属性也不能存在对码的部分依赖和传递依赖（决定因素必含有码）

- 问题

  冗余；增删改异常问题
- 改造

  分解－－把部分依赖或者传递依赖于码的主属性连同它所依赖的属性分离出去组成一个新的关系模式。
- 损失－－某些函数依赖可能会丢失

六、事务管理

- ➤ 并发控制协议－－隔离性
- ➤ 恢复协议－－原子性、持久性
- ➤ 触发机制－－一致性
- 调度的概念

一系列事务它们的操作相互交叉执行的序列称为调度。

- 可串行化调度

能够和某一个串行调度结果相同的调度称为可串行化调度。

- 冲突可串行化调度

若调度 S 可以通过交换某些相继非冲突操作变成某一串行调度,则称调度 S 为冲突可串行化调度。

- 冲突操作（对同一数据）
  - ✧ 读- 写
  - ✧ 写- 读
  - ✧ 写- 写

1. 数据库系统中事务的概念，事务特性

⑴ 事务的概念

事务是完成用户某一特定任务的与数据库的一次或多次交互的逻辑单位。

(对数据库来讲，事务是和数据库交互的一个程序片段。

对于一般意义上的用户来讲，事务是完成一个功能的程序片段或计算机的指令集合。)

事务（Transaction） 是对数据库进行访问或修改的一个或多个操作，这组操作组成一个单位，共同完成一个任务。(参考资料)

(2) 事务特性(ACID)

- 原子性(Atomicity)：执行事务中的操作要么都做，要么都不做。
- 一致性(Consistency)：一致性要求事务维护数据库的完整性约束。
- 隔离性(Isolation)：并发执行的事务之间不能相互影响。
- 持续性(Durability)：事务一旦提交，它一定是永久生效的。

2. 并发事务不加任何控制可能导致的问题

(1) 丢失更新的问题

一个事务数据的更新会把另外一个事务对数据的更新给冲掉。

(2) 未提交读取的问题 (脏读问题)

一个事务读到了另外一个事务尚未提交的数据，导致问题。

(3) 不一致读取的问题 (不可重复读/幻读)

一个事务在执行过程中，对同一个数据读取，结果是不一样的。

(4) 不一致分析问题

3. 两段锁协议，严格、次严格两段锁协议

(1) 两段锁协议

- ✧ 锁的增长阶段

- 锁的收缩阶段

- 两段锁协议只能保证调度中每个事务最终都是提交的情况下，满足冲突可串行化的条件。
- 两段锁协议仅仅是调度冲突可串行化的充分而非必要条件。

⑵ 严格的两段锁协议

把锁的释放时机延迟，延迟到事务提交或者回滚的时刻。

优点：能够保证调度的冲突可串行化，即使最后事务不提交，回滚的话也能保证调度的冲突可串行化。

缺点：降低了效率。

⑶ 次严格的两段锁协议

专用锁提交时再释放，共享锁可以提前释放。

⑷ 死锁

- 概念：两个事务相互等待对方释放锁的僵持局面叫做死锁。
- 检测方法：有向图
- 死锁的避免方法
  - 资源的一次性分配
  - 资源编号依次使用
- 死锁的检测与解除
  - 有向图
  - 中断机制
  - 随机撤销一个事务

4. 事务隔离等级含义，事务边界与隔离等级设计原则

⑴ 事务隔离等级

- READ UNCOMMITTED

  未提交读取（脏读）——有可能读到别的事务尚未提交的数据
- READ COMMIT

提交读取——只能读到别的事务已经提交的数据

- REPEATABLE READ

  可重复读——在同一个事务中，对同一个数据的多次读，值是一样的（取决于第一次读取到的数据），即使在读的过程中别的事务是数据变化了，对你没有影响。

- SERIALIZABLE

  串行化——对同一个数据是串行化执行的

(2) 事务边界与隔离等级设计原则

- 事务的设计应该尽可能短小，尤其不应该把与用户、与设备交互的操作放到事务中。（业务面向用户，事务面向数据库。）
- PostgreSQL 的隔离等级——提交读取、串行化

5. 数据库恢复机制与方法，日志记录原则，恢复过程

(1) 数据库恢复机制与方法

- 更新策略

  Immediate 立刻

  Deferred 延迟

- 刷新策略

  Force 强制

  No- Force 不强制

Failures Happen

Immediate/No-Force ⇒ Undo/Redo

Immediate/Force ⇒ Undo/No-Redo

Deferred/No-Force ⇒ No-Undo/Redo

Deferred/Force ⇒ No-Undo/No-Redo

(2) 日志记录原则

- 登记的次序严格按并发事务执行的时间次序。
- 必须先写日志文件，后写数据库。

⑶ 恢复过程

七、提高查询性能

1. 索引提高查询性能原理

- PPT 翻译

索引是一种辅助文件，可以更高效地搜索数据文件中的记录。它通常定义在数据文件的一个字段上，称为索引字段。

每个索引项都是索引值与指向该值作为记录一部分出现的页的指针的配对。

为了快速搜索，条目按索引字段值排序。

- 网页查询

索引就是通过事先排好序，从而在查找时可以应用二分查找等高效率的算法。

一般的顺序查找，复杂度为 O(n)，而二分查找复杂度为 O(log2n)。当 n 很大时，二者的效率相差及其悬殊。

在一个或者一些字段需要频繁用作查询条件，并且表数据较多的时候，创建索引会明显提高查询速度，因为可由全表扫描改成索引扫描。

2. 索引类型及应用

⑴ 分类 1

- 主索引
- 次级索引/辅助索引
- 聚集索引

⑵ 分类 2

- 密集索引（稀疏索引）
- 稀疏索引（主索引）

⑶ B+ Trees（密集索引）

（顺序文件上的索引、B+树索引、散列(hash)索引、位图索引）

⑷ 创建索引

```
CREATE [UNIQUE] INDEX  index-name
    ON  table (indexing-attribute-order-list) [CLUSTER];
```

3. 查询性能分析方法，查询分析结果理解与问题发现

⑴ 查询优化是程序员和数据库管理员利用访问方法来确保应用程序尽可能有效地访问数据的过程。

优化查询的步骤之一是检查那些查询访问的表的物理存储特征。

查询优化的下一步是分析查询的运行时特征。(查找瓶颈)

⑵ PostgreSQL 分析工具

pg_class

select relname,reltuples,relpages from pg_class where relname='''lot';

```
explain select distinct course_id from course where course_term = 'Fal02';
```

```
NOTICE:  QUERY PLAN:
Unique  (cost=12223.09..12339.76 rows=4667 width=4)
        -> Sort (cost=12223.09..12223.09 rows=46666 width=4)
        ->  Seq Scan on course  (cost=0.00..8279.99 rows=46666 width=4)
```

```
explain analyze select distinct course_id
from course
where course_term = 'Fal02';
```

```
NOTICE:  QUERY PLAN:
Unique  (cost=12223.09..12339.76 rows=4667 width=4)
        (actual time=1643.87..1797.34 rows=41803 loops=1)
 ->  Sort  (cost=12223.09..12223.09 rows=46666 width=4)
        (actual time=1643.86..1706.05 rows=41803 loops=1)
 ->  Seq Scan on course  (cost=0.00..8279.99 rows=46666 width=4)
        (actual time=184.83..1075.11 rows=41803 loops=1)
Total runtime: 1899.24 msec
```

4. 根据实际情况正确运用索引提高查询性能

⑴ 索引是有大量数据的时候才建立的，没有大量数据反而会浪费时间，因为索引是使用二叉树建立.

⑵ 当一个系统查询比较频繁，而新建，修改等操作比较少时，可以创建索引，这样查询的速度会比以前快很多，同时也带来弊端，就是新建或修改等操作时，比没有索引或没有建立覆盖索引时的要慢。

⑶ 索引并不是越多越好，太多索引会占用很多的索引表空间，甚至比存储一条记录更多。

对于需要频繁新增记录的表，最好不要创建索引，没有索引的表，执行 insert、append 都很快，有了

索引以后，会多一个维护索引的操作，一些大表可能导致 insert 速度非常慢。

八、简答题

1. 什么是数据库?

数据库(DataBase，简称 DB): 数据库是长期储存在计算机内的、有组织的、可共享的数据集合。

2. 什么是数据库系统?

数据库系统(DataBase System，简称 DBS): 数据库系统是由数据库、数据库管理系统(及其应用开发工具)、应用程序和数据库管理员组成的存储、管理、处理和维护数据的系统。

3. 什么是数据库管理系统?

数据库管理系统(DataBase Management System，简称 DBMS): 数据库管理系统是位于用户与操作系统之间的一层数据管理软件,用于科学地组织和存储数据、高效地获取和维护数据。

4. 索引的概念

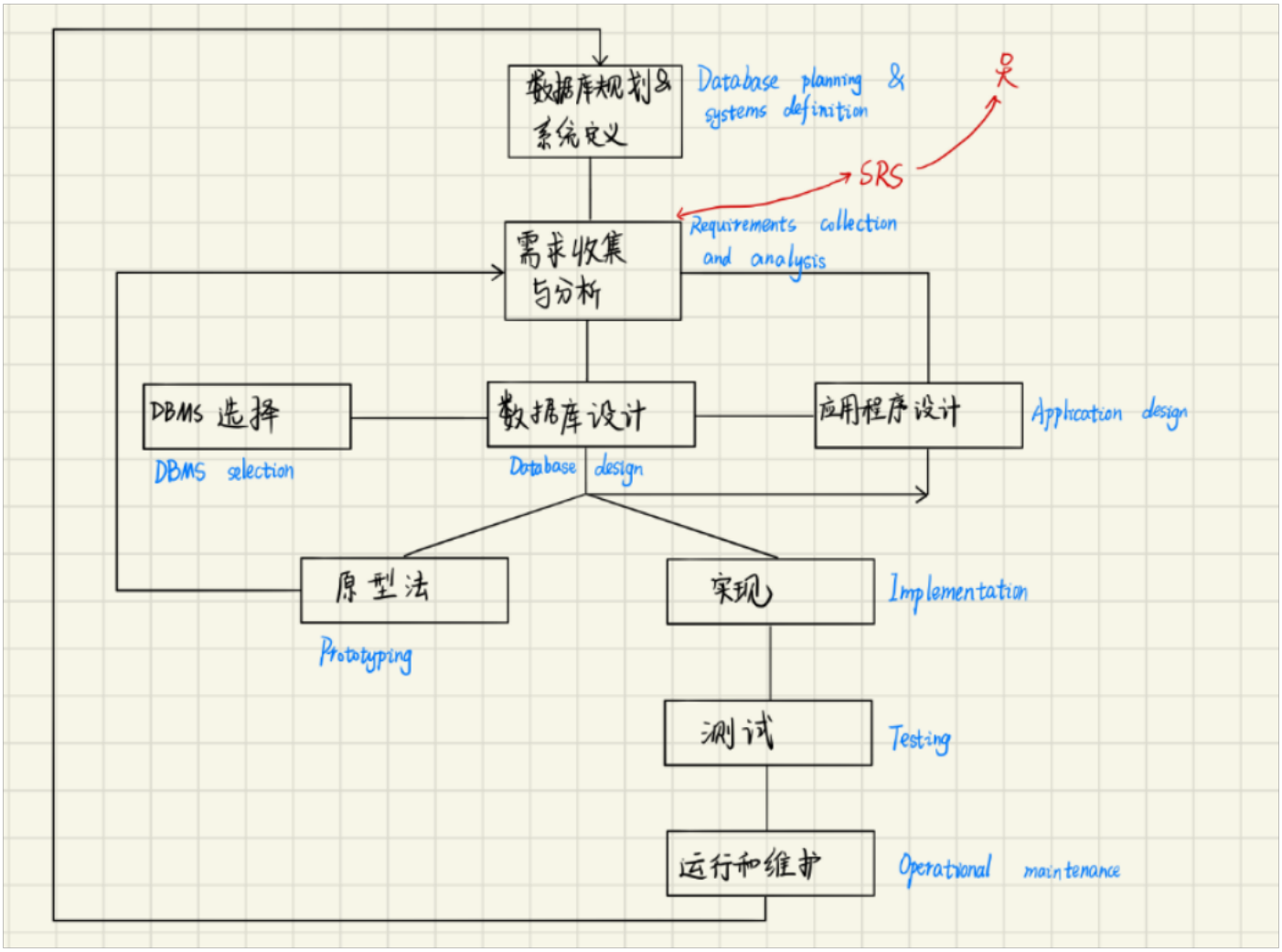
索引是一种帮助我们提高查询性能的方法。(老师)

索引是一种数据结构,它允许数据库高效地找到关系中那些在索引属性上取给定值的元组,而不用扫描关系中的所有元组。(第 6 版课本 P77)

索引是对数据库表中一列或多列的值进行排序的一种结构,使用索引可快速访问数据库表中的特定信息。(百度百科)

索引:是一个数据结构,用来快速访问数据库表格或者视图里的数据,需要存储空间。(CSDN)

5. 数据库设计的主要阶段



(1)数据库规划和系统定义

① 数据库规划——定义任务和目标

② 系统定义——定义系统边界、确定用户视图

(2)数据收集与分析

收集和分析有关数据库应用程序要支持的组织部分信息的过程,并使用该信息来识别用户对新系统的需求。

(结果:需求规格说明书)

(3- 1)数据库设计

把需求转化为将来所采用的 DBMS 能够支持的模型。

或:

根据将来所采用的DBMS能够支持的模型,把需求收集与分析获取的结果转化为将来DBMS能够支持的模型。

(数据库设计的三个阶段: ①概念设计 ②逻辑设计 ③物理设计)

(3- 2)DBMS 选择(逻辑设计阶段)

(3- 3)应用程序设计

(原型法)

(4)实现

(5)测试

(6)运行和维护

6. 关系模型的特点

(1)优点

①关系模型与非关系模型不同,它是建立在严格的数学概念的基础上的。

②关系模型的概念单一,数据结构简单、清晰,用户易懂易用。

③关系模型的存取路径对用户透明,从而具有更高的数据独立性、更好的安全保密性,也简化了程序员的工作和数据库开发建立的工作。

(2)缺点

①对“现实世界”实体的表达能力比较弱(网络答案)

②语义过载(网络答案)

③3 不能很好的支持业务规则(网络答案)

④4 由于存取路径对用户透明,查询效率往往不如非关系数据模型。因此,为了提高性能,必须对用户的查询请求进行优化,增加了开发数据库管理系统的难度。(课本)

7. 事务的 ACDI 特性

(1) 原子性(Atomicity)：执行事务中的操作要么都做，要么都不做。

(2) 一致性(Consistency)：一致性要求事务维护数据库的完整性约束。

(3) 隔离性(Isolation)：并发执行的事务之间不能相互影响。

(4) 持续性(Durability)：事务一旦提交，它一定是永久生效的。

8. JDBC 连接数据库的过程

(1) 加载 JDBC driver 驱动

(2) 链接数据库

(3) 创建 Statement 对象

(4) 执行 sql 语句

(5) 打印出结果