

ch1

database system

包括相关的数据和程序以及方便的环境
管理数据：

1. 高价值
2. 相当大
3. 经常被许多用户和应用同时访问。

问题：

1. 数据冗余和不一致
2. 难以存储数据
3. 数据孤岛 大量文件和格式
4. 约束

目标

数据更新

多个用户同时访问

安全问题

DATA_MODELS

描述工具集合：

数据

数据关系

数据语义

数据约束

抽象程度：

物理层面：描述数据如何存储

逻辑层面：描述存储的数据以及关系

视图：隐藏数据细节的应用

physical data independence

可在不修改逻辑模式的情况下修改物理模式

Data Definition Language

```
create table instructor
(
    ID char(5),
    name varchar(20)
)
'''注意 用圆括号而不是花括号'''
```

SQL Query Language

```
select name
from instructor
where dept_name= 'cccc'
```

为了便于实现计算复杂的功能，sql通常会进行内嵌处理
对语言进行拓展 嵌入sql 进而支持sql
RDBMS提供开发API，支持调用SQL

Database Design:

逻辑设计 物理设计

Query processor

ddl interpreter dml compiler query elvation engine

Transaction Management

一个事务是一系列操作的集合。

ch2 Intro to Relational Model

Structure of Relational Databases

attributes 属性 relation schema 关系表

relation instance r defined over R is $r(R)$ r是实例

每个属性的允许值集被称为域属性

属性值需要是原子的，不可分割的。

特殊值null是每个域的元素 指示这个值是unkown

Relations are Unordered

元组的顺序无关紧要

Database Schema

Database Schema 是数据库的逻辑结构

Keys

K 属于 R

如果K是能够识别R唯一元组的码，则为超码

如果K比较小，则为候选码 candidate key

候选码中被选定的码位主码 primary key

外码 是在其他关系中出现的码

Referencing relation 引用表

Referenced relation 被引用表

外码必须是被引用表的候选码

否则不是外码而是引用关系

SQL 的六个基本操作（关系代数角度）

您提到的是从**关系代数**角度定义的六个基本操作，这与我之前回答的SQL语句角度不同。从关系代数理论的角度，六个基本操作确实是：

1. 选择/筛选 (Selection, σ)

- 用于从关系中选择满足特定条件的元组（行）
- 相当于SQL中的 `WHERE` 子句
- 关系代数表示： $\sigma_{\text{条件}}(\text{关系})$
- SQL示例：

```
SELECT * FROM Students WHERE age > 20;
```

2. 投影 (Projection, π)

- 用于从关系中选择特定的属性（列）
- 相当于SQL中SELECT子句中的列列表
- 关系代数表示： $\pi_{\text{属性列表}}(\text{关系})$
- SQL示例：

```
SELECT name, age FROM Students;
```

3. 并集 (Union, \cup)

- 合并两个关系的元组，去除重复

- 要求两个关系的属性数量和类型相同
- 关系代数表示：关系1 \cup 关系2
- SQL示例：

```
SELECT * FROM Students1 UNION SELECT * FROM Students2;
```

4. 差集 (Set Difference, -)

- 从第一个关系中删除在第二个关系中出现的元组
- 关系代数表示：关系1 - 关系2
- SQL示例：

```
SELECT * FROM Students1 EXCEPT SELECT * FROM Students2;
```

5. 笛卡尔积 (Cartesian Product, \times)

- 将两个关系的所有可能组合配对
- 关系代数表示：关系1 \times 关系2
- SQL示例：

```
SELECT * FROM Students, Courses;
```

6. 重命名 (Rename, ρ)

- 为关系或属性提供新名称
- 关系代数表示： $\rho_{\text{新名称}}(\text{关系})$
- SQL示例：

```
SELECT S.name FROM Students AS S;
```

关系代数与SQL对应表

关系代数操作	符号	SQL对应
选择/筛选	σ	WHERE 子句
投影	π	SELECT 列表
并集	\cup	UNION
差集	-	EXCEPT 或 MINUS

关系代数操作	符号	SQL对应
笛卡尔积	\times	逗号分隔表或 CROSS JOIN
重命名	ρ	AS

Select Operation

1. 筛选 σ
 - i. $=, >, <, \leq, \geq, \neq$
 - ii. \wedge / \neg
2. 投影 π
 - i. 用于从关系中选择特定的属性（列）
3. 笛卡尔积
 - i. $m \times n = m * n$ 个元组
4. 并集
 - i. $r \cup s$ 两者应该有相同的属性数量
5. 差
6. 重命名

ch3 Introduction to SQL

SQL

domain types in sql

`char` , `varchar`, `int` , `smallint`, `numeric`, `real` `double precision` `float`
 定长字符串 变长字符串 整数 小整数 定点数 浮点数和双精度浮点数

1. `create table` 用圆括号
2. `create` 中要 标注 主码 外码以及reference

Update to tables

sql不区分大小写

`insert` 插入一行数据

`delete`

`drop table`

`alter` 增加属性A 数据类型为D 但不支持从表中删除属性

```
alter table r add A D
```

The select Clause

select列出了所有想要的属性结果。

允许在关系和查询结果中重复

删除重复元素 使用 distinct

```
select distinct dept_name
```

关键词all指定不删除重复项

```
select all dept_name
```

```
select * 查询所有的属性
```

select 语句中可以包括运算 +-*/

The where Clause

where 子句指定了结果必须满足的条件

使用 and or not

```
<, <=, >, >=, =, <>
```

the from clause

the from clause 列出了查询的涉及到的关系

可以结合笛卡尔积

The Rename Operation

```
old-name as new-name
```

KEYWORD *as* as 可以省略

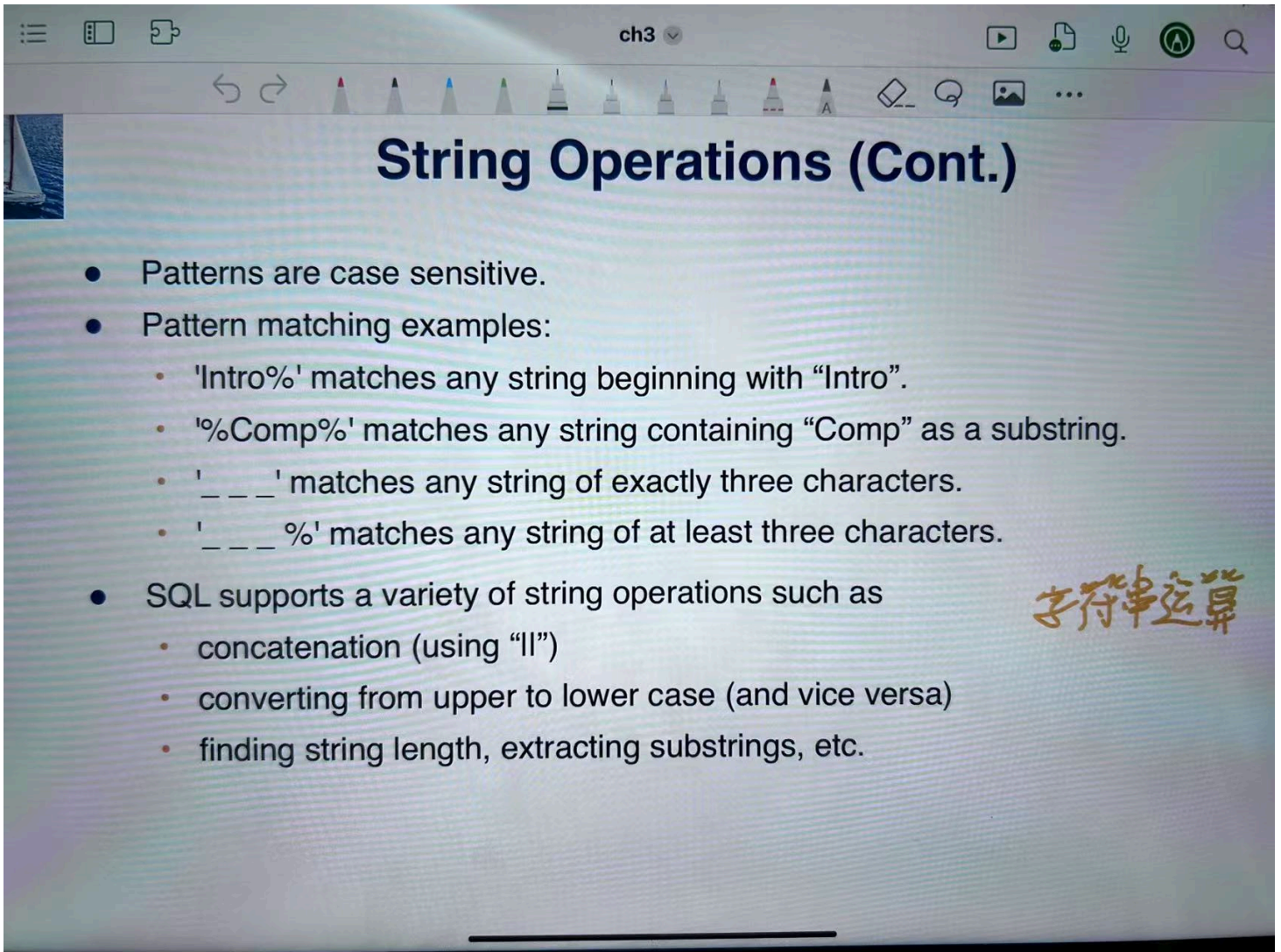
self join example

大概是自身的笛卡尔积

string operation 字符串操作

- % 匹配任何子字符串
- _ 匹配任何单个字符
- like 用于字符串匹配

```
select name
from instructor
where name like '%dar%'
=====
like '100 \%' escape '\'
```



The screenshot shows a presentation slide titled "String Operations (Cont.)". The slide contains a list of bullet points about string patterns and SQL string operations. A handwritten note in Chinese, "字符串运算" (String Operations), is written in the bottom right corner of the slide content area.

- Patterns are case sensitive.
- Pattern matching examples:
 - 'Intro%' matches any string beginning with "Intro".
 - '%Comp%' matches any string containing "Comp" as a substring.
 - '___' matches any string of exactly three characters.
 - '___ %' matches any string of at least three characters.
- SQL supports a variety of string operations such as
 - concatenation (using "||")
 - converting from upper to lower case (and vice versa)
 - finding string length, extracting substrings, etc.

字符串运算

Ordering the Display of Tuples

排序 默认升序

desc 降序 asc 升序

order by dept_name ,name 先按照 dept_name 排 后按照name排序

where clause predicates

between

where salary between 90000 and 100000

where (instructor.ID.dept_name)=(teaches.ID,'Biology')

Set Operations

集合运算

`union` 并 `intersect` 交 `except` 差 会自动排除所有的同类相
如果要保留在最后加上 `all`

Null Value

标示未知或者不存在

任何涉及空的算术表达式结果都是空

谓词为空可以用于检查空值

以及对于 `and` 和 `or` 比较

`true` - `unknown` - `false`

`avg` 平均 `min` 最小 `max` 最大 `sum` 求和 `count` 计数

Aggregate Functions

`group by`

聚合函数之外的选择自居中的属性必须出现在 分组里表中

`having`

作为一种过滤条件 会在分组之后在过滤，

比较相反 `where` 是分组之前过滤

Nested Subqueries（嵌套子查询）

在 `select` `from` `where` 中都可以嵌套

Set Membership

集合成员

`in` `not in`

set Comparison

`some` clause

某一个 实例可以使得 `comp` 成立

不过 `<>` `some` `!=` `not in`

all clause

所有

exists clause

存在

correlation name 相关名

correlation subquery 相关子查询

not exists clause

不存在

test for absence of duplicate tuples

检测是否存在重复元组

unique

如果不存在重复元组，就返回true

Subqueries in the From Clause

Subqueries in the From Clause

允许在from中进行子查询

with clause

提供了定义临时关系的方式

```
with max_budget(value) as
(
    select max(budget)
    from department
)
```

Scalar Subquery

标量子查询

不过如果子查询返回多个结果，运行错误。

Modification of the Database

Deletion

`delete from instructor` 删除所有

`where dept_name='finance'` 添加限制条件

另外删除的时，如果有聚集函数可能会在调用过程中影响结果，最好使用with

Insertion

添加新的元组

`insert into course values()`

可以将查询结果插入到表中

Update

```
update instructor
set salary =salary *1.05
```

case statement for Conditional Updates

可以实现条件更新

```
update instructor
set salary = case
    when salary <= 1000 then salary * 1.05
    else salary *1.03
end
```

ch4 Intermediate SQL

Join Expression

Joined Relations

连接操作 对两个关系进行操作 并返回另一个关系的结果

- Natural join
- Inner join
- Outer join

Natural join

自然匹配连接所有有着相同value的属性，并且只保留每个通用列的一个副本
同名属性只出现一次

```
select name ,course_id
from student natural join takes
```

```
select A1,A2,A3      An
from r1 natural join r2 natural join r3 ,, natural join rn
where P
```

Dangerous in Natural join

不相关的属性有着相同的名字的时候，会造成困扰

with Using clause

替代on 简化逻辑

```
select name,time
from (student natural join takes) join course using (course_id)
```

简化连接条件，是指前者的course_id = 后者的 course_id

Join Condition

on 根据on的条件匹配 join 元组

Outer Join

外连接 -- 可以避免信息的丢失

连接并且将不匹配的元组 连接到结果元组中

使用null

left outer join - right outer join - full outer join

left

把左边元组中不匹配的元组加到结果元组中 对应的值为空

right

同理

Full Outer Join

全外连接 将左右的都合并到者的结果中

Join types
inner join
left outer join
right outer join
full outer join

view 视图

在一些情况中，不希望所有用户都能够看见完整逻辑
视图提供了像某些用户隐藏某些数据的机制
任何不是概念模型但是可见的关系被称为视图和虚拟关系
视图不储存关系

```
create view v as <query expression>
```

view definition and use

query expression
查询语句

views defined using other views

- 1. 直接依赖
- 2. 间接依赖
- 3. 递归依赖 自身

view expansion

把用view拓展开
循环执行至没有view，只要不是是递归的循环就会终止

Materialized views

物化视图
某些数据库允许物理存储 视图关系
定义视图时创建的实体副本 被称为实体化视图
如果查询过程中使用的关系被更新， 实体化视图的结果过时

实践上不要通过视图更新数据

some update cannot be translated uniquely

某些更新操作可能无法正确的映射到底层

And some not at all

某些操作完全不适用

在一个限制了 dept_name = 'History' 的视图中，不能插入 dept_name = 'Biology' 的元组。这是因为视图的定义中包含了一个约束条件，插入的元组必须满足该条件。

View Updates in Sql

在简单的视图中可以更新

from 中只有一个relation

select 只有这个attribute names of the relation 并且不含有任何表达式聚合函数或者distinct等等

select 中不存在的元素会被设为null

query 中没有group 和 having

Transactions 事务

有一系列的查询或更新组成

结束语 commit work 提交工作 永久性改变

rollback work 回滚事务，撤销事务中的所有更改，将数据库恢复到事务开始之前的状态。

Integrity Constraints 完整性约束

避免偶然性的损失

Constraints on a Single Relation 单个关系上的约束

- not null
- primary key
- unique
- check (p), where P is a predicate

not null

宣称属性不能为空

unique

unique(A1,A2,A3)

```
CREATE TABLE instructor (  
    ID CHAR(5),  
    email VARCHAR(50) UNIQUE  
);
```

候选码可以为空 主码不可为空

the check clause

约束条件

Referential Integrity 引用完整性

外键是一个表中的列或列的组合，用于引用另一个表的主键或候选键。
外键值必须在被引用表中存在，或者为空（NULL）。

Cascading Actions in Referential Integrity

引用完整性中的级联操作

```
CREATE TABLE child_table (  
    child_id INT PRIMARY KEY,  
    parent_id INT,  
    FOREIGN KEY (parent_id) REFERENCES parent_table(parent_id)  
    ON DELETE CASCADE 删除级联  
    ON UPDATE CASCADE 更新级联  
    同步更新数据  
);
```

set default 设为默认值

set null 设为空值

integrity Constraint Violation During Transactions

完整性违规

如何在插入的时候避免造成完整性违规

1. 先插入外码 后插入 新值
2. 先设为空 插入新值 后更新
3. 推迟完整性检查

Complex Check Condition

Assertions

```
create assertion <assertion-name> check (<predicate>)
```

是 SQL 中的一种机制，用于定义数据库的全局约束（Assertions）

数据库会在每次插入、更新或删除操作后，动态检查断言是否被满足。

Large-Object Types

大对象：

- blob
- clob

User-Defined Types

用户定义的数据类型

```
create type Dollars as numeric(12,2) final
```

Domains 域

```
CREATE DOMAIN domain_name AS data_type  
[DEFAULT default_value]  
[CONSTRAINT constraint_name CHECK (condition)];
```

domain_name：域的名称。

data_type：域的数据类型（如 INT、VARCHAR）。

DEFAULT：为域指定默认值。

CHECK：定义域的约束条件。

Index Creation

创建索引

```
create index <name> on <relation-name>(attribute)
```

索引 是数据库中的一种数据结构，用于加速查询和数据检索操作。通过为表中的一个或多个列创建索引，可以显著提高查询性能，尤其是在处理大规模数据时。

Authorization

- read
- insert
- update

- delete
grant 授予权限
grant on to
grant select on department to Amit ,sa
| insert | update | delete | all privileges |

revoke on from

如果权限授予两次，只撤销一次，仍有权限

roles

create a role

grant to

```
grant select on department to Amit with grant option
-- 给a权限，并且还可以给别人
revoke select on department from Amit,Satoshi cascade
-- 权限撤销权限 应向所有用户
revoke select on department from Amit ,Satoshi restrict
-- 限制撤销权限 存在依赖的时候操作会失败
```

ch5 Advanced SQL

JDBC

JDBC（Java Database Connectivity）是 Java 提供了一种用于与数据库交互的 API（应用程序编程接口）。它允许 Java 应用程序通过标准化的方式连接到各种关系型数据库（如 MySQL、PostgreSQL、Oracle 等），执行 SQL 查询和更新操作。

连接数据库

```
// 现代JDBC连接方式（Java 7及以上）
try (Connection conn = DriverManager.getConnection(
    "jdbc:oracle:thin:@db.yale.edu:2000:univdb", userid, passwd);
    Statement stmt = conn.createStatement()) {
    // 执行数据库操作
}
```

static 静态 非静态必须实例化后使用，静态的话可以用于类名 函数 不用实例化

Prepared Statement

预备语句

```
PreparedStatement pstmt =conn.prepareStatement("insert into instruction values(?,?,?,?)");
```

便于处理

注入式攻击（SQL Injection）是一种常见的网络攻击方式，攻击者通过将恶意的 SQL 代码插入到应用程序的输入字段中，欺骗应用程序执行未预期的 SQL 查询，从而获取、修改或破坏数据库中的数据。

Metadata Features 元数据

1. 描述数据结构：

元数据存储了数据库中表、列、数据类型、约束等信息。例如，表的列名、列的数据类型、主键和外键等都属于元数据。

2. 支持数据管理：

数据库管理系统（DBMS）使用元数据来管理数据的存储、访问和操作。例如，查询优化器会利用元数据来选择最优的查询执行计划。

3. 提供数据的上下文：

元数据为数据提供了语义信息，帮助用户理解数据的含义。例如，列名和注释可以说明列中存储的是什么数据。

4. 增强数据的可用性：

元数据可以帮助用户快速定位和使用数据。例如，通过元数据可以查询某个表有哪些列，或者某列是否有索引。

Finding Primary Key

Transaction Control in JDBC

other JDBC Features

SQLJ embedded SQL in Java

内嵌于Java的sql

遍历赋值输出

ODBC n

Open DataBase Connectivity

Functions and Procedures

Declaring SQL Functions

自定义函数 封装逻辑 提高代码复用性 和可读性

```
create function dept_name(dept_name varchar(20))
returns integer
begin
declare d_count ineger;
select count(*) into d_count
from instructor
where instructor.dept_name= dept_name
return d_count
end
```

Table Functions

table function

可以将表返回为结果 这样的函数被称为 table function

returns 定义函数的返回类型
return 用于指定函数的返回值

SQL Procedures

如果没有返回值 则定义为存储过程

存储过程不返回值，而是执行一系列的操作

```
create procedure dept_count_proc (in dept_name varchar(20),out d_count integer)
begin
-----
end
```

in 用于将值传递给存储过程。 out 用于从存储过程返回值。 inout 既可以作为输入参数，也可以作为输出参数。

在 SQL 中，存储过程（Procedure）的参数可以使用 IN、OUT 和 INOUT 关键字来定义参数的方向。这些关键字用于指定参数在存储过程中的作用，即参数是用于输入、输出还是两者兼有。

call 用于存储过程的调用

过程和函数都可以被调用

多个过程可以有着相同的名字 过程重载 类似于c++的重载

language construct for procedure and function

```
while boolean expression do
  sequence of statement
end while
```

```
repeat
  sequence of statement
until boolean expression
end repeat
```

if-then-else

```
if boolean expression
then statement or compound statement
elseif boolean expression
then statement or compound statement
else statement or compound statement
end if
```

security with external language routines

- 用sandbox techniques 环境
- 单独环境或者单开进程

Triggers 触发器

trigger 是系统自动化执行的statement

- event
 - insert
 - delete
 - update 先删后插
- referencing old row as 操作前的行数
- referencing new row as 操作后的行数

BEFORE UPDATE 和 REFERENCING NEW ROW 的结合

BEFORE UPDATE：触发器在更新操作执行之前触发。

REFERENCING NEW ROW：引用更新操作中即将写入的行数据。

这种组合通常用于在更新操作之前对新数据进行验证、修改或记录日志。

- USE for each statement

- ues referencing old table o referencing new table
- 更加高效

但不推荐使用触发器