week1-intro

文件处理系统缺点：

数据冗余和不一致(dataredundancyandinconsistency)

数据隔离(dataisolation)

访问数据困难(difficulty in accessing data）

完整性问题(integrity problem)

安全性问题(security problem)

并发访问异常(concurrent-access anomalies)

原子性问题（atomicity problems）

文件处理系统→数据库系统

选用 DBMS 的场景：需要管理的数据“highly valuable”，“relatively large”，“accessed by multiple users and applications”

数据类型（data model）：a notation for describing data or information

常见数据类型：

关系模型（relational model）：以 table 为 relation

实体-联系模型(entity-relationship model) 用于数据库设计

半结构化数据模型 用于互联网和大数据场景

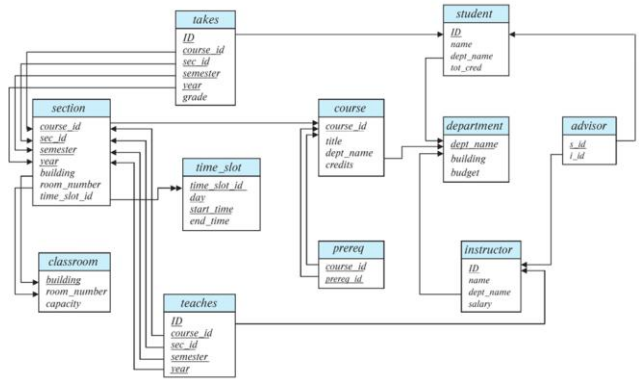
基于对象的数据模型

...

模式（schema）：数据库的整体设计

e.g university 数据库 schema

classroom(<u>*building*</u>, <u>*room_number*</u>, *capacity*)
department(<u>*dept_name*</u>, *building*, *budget*)
course(<u>*course_id*</u>, *title*, *dept_name*, *credits*)
instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)
section(<u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *building*, *room_number*, *time_slot_id*)
teaches(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)
student(<u>*ID*</u>, *name*, *dept_name*, *tot_cred*)
takes(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *grade*)
advisor(<u>*s_ID*</u>, *i_ID*)
time_slot(<u>*time_slot_id*</u>, <u>*day*</u>, <u>*start_time*</u>, *end_time*)
prereq(<u>*course_id*</u>, <u>*prereq_id*</u>)



数据库语言

DDL，data-definition language，如 CREATETABLE（定义）

DML，data-manipulation language，如 SELECT-FROM-WHERE（查询 query）

SQL（Structured Query Language） 声明式语言，关注 What 而不是 How

week2-relational-model

关系数据库（relational database）

结构：数据库-Excel 对应关系

数据库	Excel
关系 (relation)	表 (table)
元组 (tuple)	行 (row)
属性 (attribute)	列 (column)

X：tuple（row）

Y：attribute（column）

X×Y：relation（table）

schema：例如 instructor(ID, name, dept_name, salary)

码（key）：区分 relation 中不同的 tuple

超码（super key）：使得可以唯一标识一个 tuple 的若干属性的组合

候选码（candidate key）：最小超码，具有唯一性

主码（primary key）：被 DB 设计者选中的候选码

外码（foreign key）

例如：

参照关系 instructor(ID, name,dept_name, salary)

被参照关系 department(dept_name,building,budget)

此时，dept_name 是外码（instructor referencing department）

关系代数（relation algebra）：定义在关系、元组和属性上的操作，SQL 的理论基础

select；project

关系运算的组合：笛卡尔积(Cartesian product) A×B；join；natural join；

union；intersect；difference（相容性前提）

链接

week3-sql

SQL：DDL

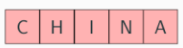
基本数据类型：

数值

- int：整数类型（与机器相关，等价于 integer）
- smallint：小整数整形（int 的子集）
- numeric(p, d)：定点数，（最多）有 p 为数字，小数点右边有 d 位（在 PG 中等价于 decimal）
- float(n)：精度至少为 n (binary) 位的浮点数
- real, double precision：浮点数与双精度浮点数（与机器相关）

字符串

- char(n)：固定长度的字符串，长度为 n（等价于 charater）
- varchar(n)：可变长度的字符串，最大长度为 n （等价于 character varying）
- text：非 SQL 标准，表示任意长度的字符串



null：可能存在但未知（unknown），可能不存在

基本模式定义：
CREATE TABLE department
(dept_name varchar(20) PRIMARY KEY,
building varchar(15),
budget numeric(12,2));

删除：
drop table； 删除表
delete from r；清空表数据，保留表本身

SQL：DML
select, from, where:
先 from（从哪里 select），再 where（select 满足什么条件），最后 select

$\Pi_{name}(instructor)$

```
select name from instructor;
```

去除重复（duplicate）

```
select distinct dept_name from instructor;
```

$$\Pi_{name}(\sigma_{dept\_name=\text{“}Physics\text{”}\wedge salary>70000}(instructor))$$

```
select name from instructor
where dept_name = 'Physics' and salary > 70000;
```

不等于：!= or <>
between a and b: [a,b]
更名：old_name as new_name
*表示所有的属性 例如 select * from instructor
排序：order by （default：升序）
指定排序 order by salary desc（降序）/asc（升序）

week4-lab
学习 windows 系统 PostgreSQL 环境搭建（新建数据库、连接数据库、导入数据、编写 query）
字符串操作：
字符串在 SQL 标准中区分大小写
boolean：true，false，unknown（SQL 中由 null 表示）
模糊查询 链接
转义字符 \
值的拼接 ||
集合操作：并（UNION），交（INTERSECT），差（EXCEPT）

case 表达式：

```
SELECT ID,
    CASE
        WHEN score < 60 THEN 'C'
        WHEN score >= 80 THEN 'A'
        ELSE 'B'
    END
FROM marks;
```

week5-null-aggregate（空值和聚合查询）

布尔类型（boolean）：

true

false

unknown(SQL 中使用 null 表示)

(false 和 unknown 的结果不会出现在 SQL 的查询结果中)

- AND
  - true and unknown is unknown
  - false and unknown is false
  - unknown and unknown is unknown
- OR
  - true or unknown is true
  - false or unknown is unknown
  - unknown or unknown is unknown
- NOT
  - not unknown is unknown

聚集函数：max，min，avg，count，sum

除了 count(*)，聚集函数在计算时均忽略 null

仅 count 能与*连用，count(distinct *)invalid

聚集不能直接出现在 where（where 子句中的谓词在聚集之前起作用，筛选了哪些行被 select 进入聚集函数计算）

group by:按一个或多个属性进行分组

出现在 select 子句中但没有被聚集的属性只能是出现在 groupby 子句中的那些属性

对分组限定条件：不使用 where，使用 having

（select …where, group by… having）

理解以下语句的执行次序：

SELECT…

FROM…

WHERE…

GROUP BY…

HAVING…

通过 from…where 得到一个满足条件的关系，group by 在这个关系的基础上分组。如果有 having，将 having 的条件应用到每个分组。对剩下的分组进行 select，得到查询结果。

嵌套子查询（Nested sub-queries）：select-from-where 嵌套在另一个查询中

PG 要求对每个子查询结果都给一个名字（AS <name>），即使不引用关系。

测试元组是否在集合中：WHERE <name> IN/NOT IN <set>

至少比某一个大：>some（即大于最小值）
=some 等价于 in
比所有都大：>all
<> all 等价于 not in

标量子查询（scalarsub-query）只返回包含单个属性的单个元组
exists 测试子查询返回的关系是否为空（子查询结果较多时建议使用，不用 in）
WITH 子句定义临时关系，用于当前查询

week6-change
删：DELETE FROM…
　　WHERE…;
（注意区别 drop 和 delete）
增：
INSERT INTO course (course_id, title, dept_name, credits)
VALUES ('CS-205', 'Database Systems', 'Comp. Sci.', 4);
PG 中使用 COPY 命令批量从文件导入数据，速度远快于 insert
改：
UPDATE…
SET…
(WHERE…)
update+case

修改关系（alter table）
新增属性：ALTER TABLE products ADD COLUMN description text;
删除属性：ALTER TABLE products DROP COLUMN description;
（如果属性被其他关系的外码引用，那么不能直接删除）
其他：

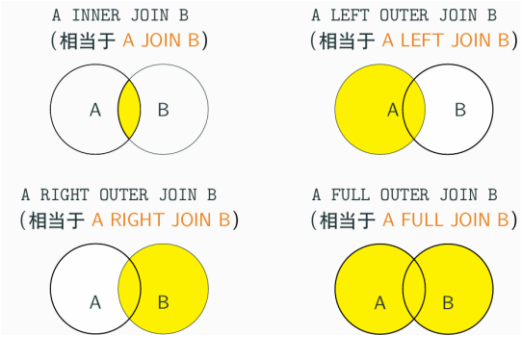
```
-- 修改属性的数据类型
ALTER TABLE products
ALTER COLUMN price TYPE numeric(10,2);

-- 修改属性名
ALTER TABLE products
RENAME COLUMN product_no TO product_number;

-- 修改关系名
ALTER TABLE products RENAME TO items;
```

默认值(default)：null
修改默认值（SET DEFAULT）；清除默认值（DROP DEFAULT）
完整性约束：保证用户对数据库所做的修改不破坏数据的一致性
添加约束：ALTER TABLE products ADD CONSTRAINT some_name UNIQUE (product_no);
行约束/表约束

check 约束：check(P) 关系中的每个元组满足条件 P
（check 需要满足的条件是 not false。因此，允许出现 null， 即 unknown）

SQL 排名：
RANK

week7-join
natural join:所有相同名称属性的值相等 使用 join using (属性) 去除重复属性
使用 on 指定任意条件
引入 on 子句，让 on 成为 join 独有的谓词表达式，增强代码可读性（join…on…）
默认 inner join
outer join: 可在结果中包含 null，不丢失数据



视图 view：虚拟的表（关系）
定义视图：
CREATE VIEW faculty AS
SELECT ID, name, dept_name
FROM instructor;

事务 transaction: a sequence of query and/or update statements.
索引：An index on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation

week8-advanced
高级数据类型
日期和时间类型

- date：日期（4 个数字），用于表示某年某月某日。
- time：一天中的时间，用于表示时分秒；也可以指定秒的小数点位数，time(p)。
- timestamp：date 和 time 的结合。类似的，也有一个变种timestamp(p)。

interval 表示时间间隔：SELECT time/date '' + interval ''
extract()从给定 timestamp 或 interval 中抽取某个字段：EXTRACT(field FROM source)

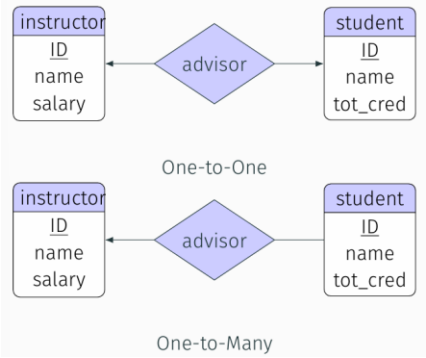
由于浮点数不精确，在对精度有要求的场景，建议不使用 float(n)、 real 或 double precision。而使用 numeric 或者 decimal。
serial

类型转换：
cast (e as t)
PG 中::符号表示类型转换 or to_xxx()方法（如 to_date，to_char）

授权 authorization：存取权限（包括读取数据、插入新数据、更新数据、删除数据等权限）
GRANT <权限列表> ON <关系或视图名> TO <用户或角色 role>;
在 PG 中如何查看所有角色
创建角色 CREATE ROLE

Week9-advanced
定义函数 function→返回结果
定义过程 procedure→执行一些操作
触发器 trigger：a statement that the system executes automatically as a side effect of a modification to the database
使用程序语言（如 python）访问数据库
SQL 注入攻击

Week10-design
E-R 模型 实体（entity）-联系（relationship）模式：将现实世界的需求映射到概念模式
实体集：相同类型的一个实体（可区别所有其他对象的一个对象）集合
联系集：相同类型联系（实体间的相互关联）的集合



实体集：instructor，student
联系集：advisor
属性（简单 vs 复合属性，单值 vs 多值属性，派生属性）
映射：One-to-One，One-to-Many，Many-to-Many，Many-to-One
（one 的一侧有箭头，many 的一侧没有箭头）
实体的全参与：两条线表示
绘制 E-R 图：draw.io
实体集的主码 弱实体集
去除冗余属性
将 E-R 图转化为关系模式
模式的合并

Week11-norm

“好的”关系数据库设计：无损分解

检查关系模式是否符合已知范式，如果否，则将其进行无损分解

1NF：

（对关系模式的基本要求，不满足 1NF 的数据库不是关系数据库）

函数依赖举例

ins_dep (ID,name,salary,dept_name, building, budget)这个关系模式有哪些函数依赖

平凡依赖

如果 $\beta \subseteq \alpha$，那么函数依赖 $\alpha \rightarrow \beta$ 就是平凡的（trivial）。

例：ID→ID，ID，name→name 是平凡依赖

闭包：由 A→B 和 B→C 推断 A→C

函数依赖可以进行推断（运算），如从 $A \rightarrow B$ 和 $B \rightarrow C$ 可以推断 $A \rightarrow C$。因此，函数依赖可以构成闭包（closure）。我们用 F^+ 表示集合 F 的闭包。

依赖闭包和函数依赖

如果下面至少一个函数依赖属于 F^+，那么 R_1 和 R_2 就构成了 R 的无损分解：

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

BCNF：（不是依赖保持的，BCNF 使得某些依赖丢失）

BCNF

具有函数依赖集 F 的关系模式 R 属于 BCNF 的条件是，对 F^+ 中所有形如 $\alpha \rightarrow \beta$ 的函数依赖，下面至少一个成立：

- $\alpha \rightarrow \beta$ 是平凡的函数依赖
- α 是 R 的一个超码

思考：`ins_dep(ID, name, salary, dept_name, building, budget)` 是否满足 BCNF？

3NF：

3NF

具有函数依赖集 F 的关系模式 R 属于 3NF 的条件是，对 F^+ 中所有形如 $\alpha \rightarrow \beta$ 的函数依赖，下面至少一个成立：

- $\alpha \rightarrow \beta$ 是平凡的函数依赖
- α 是 R 的一个超码
- $\beta - \alpha$ 的每个属性都属于 R 的某个候选码

试分析 `dept_advisor(s_ID, i_ID, dept_name)` 是否满足 3NF？

希望实现：BCNF，无损分解，依赖保持

一般数据库都是在 3NF 和 BCNF 之间选择

检查是否为 BCNF

Algorithm 1: BCNF decomposition

```
result ← {R}
done ← false
while not done do
    if there is a schema R_i that is not in BCNF then
        let α → β be a non-trivial FD that holds on R_i, such
          that α⁺ does not contain R_i
        result ← (result − R_i) ∪ (R_i − β) ∪ (α, β)
    end
    else
        done ← true
    end
end
```

Week12-theory

主流数据库存储基于磁盘，计算时从磁盘读取数据到内存

文件存储：DBMS 将文件组织成一组 pages

Page 的内容：header（描述内容的 meta-data，如 page 大小，校验和，DBMS 版本，压缩信息等）和 data

（组织方式：堆文件 heap file: an unordered collection of pages/records with tuples that are stored in random order

树文件

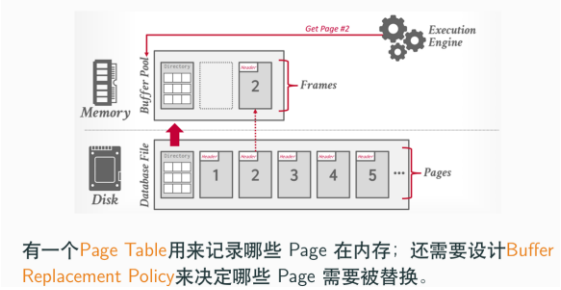
顺序文件

哈希文件）

存储模型：

行存储 row storage：将一个元组的所有属性在 page 内连续存储

列存储 column storage：将所有元组的单个属性在 page 内连续存储



有一个Page Table用来记录哪些 Page 在内存；还需要设计Buffer Replacement Policy来决定哪些 Page 需要被替换。

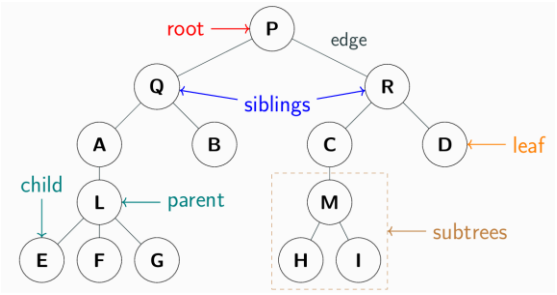
索引 index：a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure

有序索引 ordered index

聚集索引 clustering index：索引的搜索码也能决定包含文件中记录的顺序（顺序文件组织 sequential file organization）vs 非聚集索引：搜索码指定的排序与文件中记录的排序不同

稠密索引 dense index vs 稀疏索引

B+ tree index



哈希索引 hash index

hash function：any function that can be used to map data of arbitrary size to fixed-size values

事务 transaction：a collection of operations that form a single logical unit of work

原子性（atomicity）、一致性（consistency）、隔离性（isolation）、持久性（durability）