概论

数据库的历史

- 人工；
- 文件系统；
- 数据库：
  - 1960年代末：层次性、网络；
  - 1970年代：关系模型；
  - 现在：OR （面向对象关系模型）, XML。

Database 数据库: a collection of data.

- very large amounts of data （海量的数据）；
- Structured and interrelated data （结构化的信息以及信息之间的联系）；
- operational data （与运转相关的信息）；
- persistent data （持久保存——**数据库只能建在磁盘上**）.

数据库通过 **DBMS (Database Management System)** 建立和管理（可以看成系统软件）。

通过 applications 应用程序使用数据。

为什么要数据库?

- 数据共享 information sharing；
- 数据独立性 data independence：
  - 一个程序 + 一个数据 = 数据非独立 （P1 + D1, P2 + D2, ...）；
  - 多个程序 + 一个数据库 = 数据独立（于程序） ((P1, P2, ..., Pn) + D）。

DBMS 数据库管理软件

- 需要持久存储海量数据；
- 需要能够高效访问数据。

DBS 数据库系统

- 硬件 hardware；
- 数据库 database；
- 数据库管理系统 DBMS；
- 用户（DBA 数据库管理员, applications programmer 应用程序员, end user 终端用户）

关系数据模型

数据模型：提供了一套概念性的描述框架（工具）。**三要素**：

- 数据结构（数学表示）（例如，关系模型的数据结构即为关系 relation / table）；
- 对数据的操作；

- 对数据的约束。

一些数据模型

- 今天常用的模型：关系模型（relational），对象关系模型（object relational, OR）、半结构化数据模型（semistructured）；其中现在常用关系模型，半结构化模型更加灵活，但是较少使用。
- 其他模型：纯面向对象模型（OO）；
- 历史模型：层次模型、网络模型。

关系模型的特点

- 简单 simple，但是有一定限制 limited，万能的 versatile；
- 提供了有限但是有用的一系列操作；
- 允许利用 SQL 语言进行实现；
    - 使用非常简单：只需要陈述需要什么；
    - 效率稍显低下：但是可以通过一些优化改善。

关系：二维表格。属性 attribute (columns, headers) / 元组 tuple (rows)

> 数学上的关系：笛卡尔积的子集。

关系模式 (relation schema)：关系名与属性集合。

- 写作 *relation_name (attribute_list)*；实质上属性是无序的集合 set，并不是有序的表 list；但是一般使用时看成有序（定义的次序）；
- 可以把每一个属性的类型和其他信息添加进去；
- 一般来说，不随着时间而变化。
- **数据库模式**：数据库中存在许多关系，因此数据库模式即为关系模式的集合。

关系实例 (relation instance)：是元组的集合，可以随着时间改变，是一个无序的集合，且数据库仅维护当前的实例。

- **数据库实例**：关系实例的集合。

定义域 (domain)：属性的原子参数（类型）；

- 可以在关系模式中加入，例如 R(A1: D1, A2: D2)（D1、D2 为定义域）。

键值 (key)：一系列能够区分不同元组的属性（没有两个元组有着相同的键值）；

- 可以在关系模式中表示，例如 R(A1, A2, A3)（A1、A2 位键值）；
- 实质上，键值是一种语义上的约束；
- 有时候，键值需要使用人造信息（artificial keys）；
- 关系模式中kennel有许多种键值的选择方式。

为什么选用关系模型?

- 非常简单的模型；
- 非常高层次的编程语言：SQL，简单但是强大；
- 设计理论：数学上是严谨的。

SQL：标准的数据库语言。

- DDL (data definition language) + DML (data manipulate language)

- **SQL 中的关系**：
  - **stored relation**：利用表 table 来存储；
  - **view**：不存储，按需建立（临时计算）；
  - **temporary table**：通过 SQL 处理器建立。
- SQL 的数据类型：CHAR，BIT，BOOLEAN，……
  - SQL 2016 可以自定义类型。
- SQL 的值
  - 用单引号来表示字符串，字符串内存在单引号时，利用两个单引号表示；
  - 任意值可以为空 （NULL）；
  - 日期与时间：
    - 日期：`DATE 'yyyy-mm-dd'`；例如：`DATE '2011-01-11'`；
    - 时间：`TIME 'hh:mm:ss'`；例如：`TIME '15:30:02.5'`。
- 定义关系模式：

```
CREATE TABLE name (
     column_1     type_1,
      ... ,
     column_n     type_n
);
```

  例：

```
CREATE TABLE Student (
     sno CHAR(10),
     name VARCHAR(20),
     age INTEGER,
     dept VARCHAR(30)
);
```

- 删除关系模式：

```
DROP TABLE relation;
```

- 更改关系模式：

```
ALTER TABLE relation
ADD column type;
ALTER TABLE relation
DROP column;
```

- 缺省值（如果不定义，默认为 NULL）：

```
CREATE TABLE Student (
     sno CHAR(10),
     name VARCHAR(20),
     age INT DEFAULT 18,
     dept VARCHAR(30) DEFAULT ‘CS’
);
```

- 定义键值
  - 单参数键值：利用 PRIMARY KEY；一个模式中，最多仅有一个主要键值。

    ```
    CREATE TABLE Student (
          sno CHAR(10) PRIMARY KEY,
          name VARCHAR(20),
          age INTEGER,
          dept VARCHAR(30)
    );
    ```

  - 多参数键值：将多个属性组成一个 PRIMARY KEY。

    ```
    CREATE TABLE SC (
          sno CHAR(10),
          cno CHAR(5),
          grade INTEGER,
          PRIMARY KEY (sno, cno)
    );
    ```

> - UNIQUE 与 PRIMARY KEY 的差别与联系：
>   - 他们都声明了这个属性的唯一性；
>   - 一个关系模式只有一个 PRIMARY KEY，但可能有许多 UNIQUE 属性；
>   - PRIMARY KEY 不能为空 （NULL），但是 UNIQUE 可以为空 （NULL）。