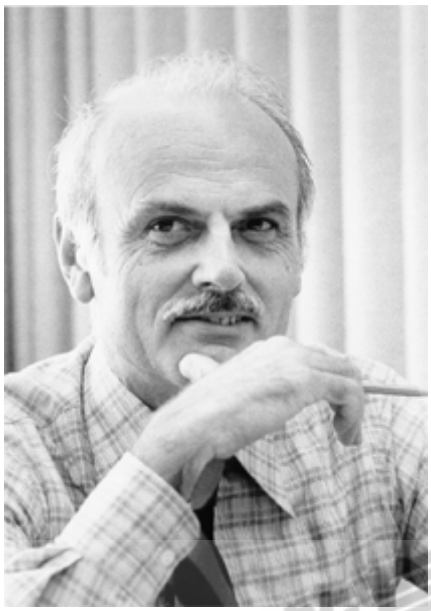
关系型数据库

诞生



1970年6月，IBM的研究员 Edgar Frank Codd 发表了名为“A Relational Model of Data for Large Shared Data Banks”的论文，提出了关系模型的概念，奠定了关系模型的理论基础。

使用**行**、**列**组成的**二维表**来组织数据和关系，表中**行**（**记录**）即可以描述数据**实体**，也可以描述实体间**关系**。
关系模型比网状模型、层次模型更简单，不需要关心数据存储的物理细节，专心于数据的逻辑构建，而且关系模型有论文的严格的数学理论基础支撑。

1976年，IBM实验室System R项目，通过实现数据结构和操作来证明关系模型实用性，并直接产生了**结构化查询语言SQL**。1987年，SQL被ISO组织标准化。

关系模型，有严格的数学基础，抽象级别较高，简单清晰，便于理解和使用。
经过几十年的发展，关系数据库百花齐放，技术日臻成熟和完善。
基于关系模型构建的数据库系统称为RDBMS(Relational DataBase System)。
IBM DB2、Oracle的Oracle和Mysql、微软的MS SQL。以前的Infomix、Sybase等。

Mysql

发展

1985年几个瑞典人为大型零售商的项目设计了一种利用索引顺序存取数据的软件，这就是MyISAM的前身。1996年，MySQL 1.0发布，随后发布了3.11.1版本，并开始往其它平台移植。2000年MySQL采用GPL协议开源。MySQL 4.0开始支持MyISAM、InnoDB引擎。2005年10月，MySQL 5.0成为里程碑版本。

2008年1月被Sun公司收购。

2009年1月，在Oracle收购MySQL之前，Monty Widenius担心收购，就从MySQL Server 5.5开始一条新的GPL分支，起名**MariaDB**。

MySQL的引擎是插件化的，可以支持很多种引擎：
MyISASM，不支持事务，插入、查询速度快。
InnoDB，支持事务，行级锁，MySQL 5.5起的默认引擎

关系模型

为了介绍关系模型，以MySQL数据库为例。

安装

官方下载 https://dev.mysql.com/downloads/ 或安装MariaDB

```
mysql-community-libs-5.7.33-1.el7.x86_64.rpm
mysql-community-client-5.7.33-1.el7.x86_64.rpm
mysql-community-common-5.7.33-1.el7.x86_64.rpm
mysql-community-server-5.7.33-1.el7.x86_64.rpm
```

```
# systemctl start mysqld
# ss -tanl
State       Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN      0      50                 *:3306                        *:*

开机启动
# systemctl enable mysqld
```

mysql 5.7初始使用了随机密码，查看 /var/log/mysqld.log，搜索password就能看到root的初始密码。

```
数据库密码登录
# mysql -u root -p

mysql> alter user root@localhost identified by 'Wayne123@magedu';

mysql> show variables like '%pass%';
mysql> set global validate_password_policy=0;
0就是LOW，要求只要8个字符即可。1是MEDIUM，要求长度至少8，大小写字母、数字、特殊字符

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

# 创建并授权用户
mysql> grant all on *.* to 'wayne'@'%' identified by 'wayne123';
mysql> flush privileges;
```

导入测试脚本

```
# mysql -u root -p < test.sql
```

也可以使用source导入

```
mysql> source /root/test.sql
```

授权

```
GRANT ALL ON employees.* TO 'wayne'@'%' IDENTIFIED by 'wayne123';

REVOKE ALL ON *.* FROM wayne;
```

* 为通配符，指代任意库或者任意表。`*.*`所有库的所有表；`employees.*`表示employees库下所有的表
% 为通配符，它是SQL语句的通配符，匹配任意长度字符串

关系

表

在关系数据库中，关系就是二维表，由行和列组成。

MySQL是行存数据库。数据是一行行存的，列必须固定多少列。
行Row，也称为记录Record，元组。
列Column，也称为字段Field、属性。
字段的取值范围叫做 域Domain。例如gender字段的取值就是1或者2两个值。

```
+--------+------------+------------+-------------+--------+------------+
| emp_no | birth_date | first_name | last_name   | gender | hire_date  |
+--------+------------+------------+-------------+--------+------------+
| 10001  | 1953-09-02 | Georgi     | Facello     |      1 | 1986-06-26 |  <-- row、行、记录、元组
| 10002  | 1964-06-02 | Bezalel    | Simmel      |      2 | 1985-11-21 |
| 10003  | 1959-12-03 | Parto      | Bamford     |      1 | 1986-08-28 |
| 10004  | 1954-05-01 | Chirstian  | Koblick     |      1 | 1986-12-01 |
| 10005  | 1955-01-21 | Kyoichi    | Maliniak    |      1 | 1989-09-12 |
| 10006  | 1953-04-20 | Anneke     | Preusig     |      2 | 1989-06-02 |
| 10007  | 1957-05-23 | Tzvetan    | Zielinski   |      2 | 1989-02-10 |
| 10008  | 1958-02-19 | Saniya     | Kalloufi    |      1 | 1994-09-15 |
| 10009  | 1952-04-19 | Sumant     | Peac        |      2 | 1985-02-18 |
| 10010  | 1963-06-01 | Duangkaew  | Piveteau    |      2 | 1989-08-24 |
+--------+------------+------------+-------------+--------+------------+
列、字段、Field、Column
```

维数：关系的维数指关系中属性的个数
基数：元组的个数

注意在关系中，属性的顺序并不重要。理论上，元组顺序也不重要，但是由于元组顺序与存储相关，会影响查询效率。

候选键

关系中，能唯一标识一条元组的属性或属性集合，称为候选键。
候选键，表中一列或者多列组成唯一的key，通过这一个或者多个列能唯一的标识一条记录。
表中可能有多个候选键。

PRIMARY KEY主键

从候选键中选择出主键。

主键的列不能包含空值null。主键往往设置为整型、长整型，可以为自增AUTO_INCREMENT字段。
表中可以没有主键，但是，一般表设计中，往往都会有主键，以避免记录重复。
InnoDB的表要求使用主键。

Foreign KEY外键

严格来说，当一个关系中的某个属性或属性集合与另一个关系（也可以是自身）的候选键匹配时，就称作这个属性或属性集合是外键。

约束Constraint

为了保证数据的完整正确，数据模型还必须支持完整性约束。

“必须有值”约束
某些列的值必须有值，不许为空NULL。

域约束Domain Constraint

限定了表中字段的取值范围

实体完整性Entity Integrity

PRIMARY KEY约束定义了主键，就定义了**主键约束**。主键不重复且唯一，不能为空。

引用完整性Referential Integrity

外键定义中，可以不是引用另一张表的主键，但是，往往实际只会关注引用主键。

外键：在表B中的列，引用了表A中的主键，表B中的列就是外键。

A表称为主表，B表称为从表。

插入规则
不需要指定。
如果在表B插入一条记录，B的外键列插入了一个值，这个值必须是表A中存在的主键值。

更新规则
定义外键约束时指定该规则。

删除规则
定义外键约束时指定该规则。

外键约束的操作

设定值	说明
CASCADE	级联，从父表删除或更新会自动删除或更新子表中匹配的行
SET NULL	从父表删除或更新行，会设置子表中的外键列为NULL，但必须保证子表列没有指定NOT NULL，也就是说子表的字段可以为NULL才行
RESTRICT	如果从父表删除主键，如果子表引用了，则拒绝对父表的删除或更新操作
NO ACTION	标准SQL的关键字，在MySQL中与RESTRICT相同。拒绝对父表的删除或更新操作

外键约束，是为了保证数据完整性、一致性，杜绝数冗余、数据错误。

实体-联系E-R

数据库建立，需要收集用户需求，设计符合企业要求的数据模型。而构建这种模型需要方法，这种方法需要成为E-R实体-联系建模。也出现了一种建模语言——UML（Unified Modeling Language）统一建模语言。

实体Entity：现实世界中具有相同属性的一组对象，可以是物理存在的事物或抽象的事物。

联系Relationship：实体之间的关联集合。

实体间联系类型

假设有实体部门，实体员工

类型	描述	解决方案
一对多联系 1:n	一个员工属于一个部门，一个部门有多个员工	员工外键；部门主键
多对多联系 m:n	一个员工属于多个部门，一个部门有多个员工	建立第三表
一对一联系 1:1	假设有实体管理者，一个管理者管理一个部门，一个部门只有一个管理者	字段建在哪张表都行

一对一关系用的较少，往往表示表A的一条记录唯一关联表B的一条记录，反之亦然。
它往往是为了将一张表多列分割并产生成了多张表，合起来是完整的信息，或为了方便查询，或为了数据安全隔离一部分字段的数据等等。

视图

视图，也称虚表，看起来像表。它是由查询语句生成的，不存储任何数据。可以通过视图进行有限制的更新操作。

视图的作用

- 简化操作，将复杂查询SQL语句定义为视图，可以简化查询。
- 数据安全，视图可以只显示真实表的部分列，或计算后的结果，从而隐藏真实表的数据