Neuedu

文件编号：D000-RIPH206

MySQL数据库开发技术

版本：1.0.0-0.0.0

2017-9-15

文件修改控制

修改编号	版本	修改条款及内容	修改日期
1	1.0.0-0.0.0	创建	2017-9-15

Neuedu

MySQL数据库开发技术

—— 数据库基础

V1.0

课程目标

- 通过此课程的学习使学员能够独立地安装及创建数据库，掌握常用的函数及SQL语句，包括表结构的设计及数据的增删改查操作、掌握视图和索引的创建，了解MySQL数据库的体系结构、存储引擎、权限管理主从复制及分区管理等。

课程目录

Neuedu

内容	参考课时（H）
第一章　数据库基础	3
第二章　数据库和数据表管理	6
第三章　数据操作与事务控制	2
第四章　简单查询	3
第五章　常用函数	4
第六章　多表连接查询	3
第七章　高级查询	4
第八章　视图和索引	4
第九章　用户和权限管理	2
第十章　分区管理	2
合计	33

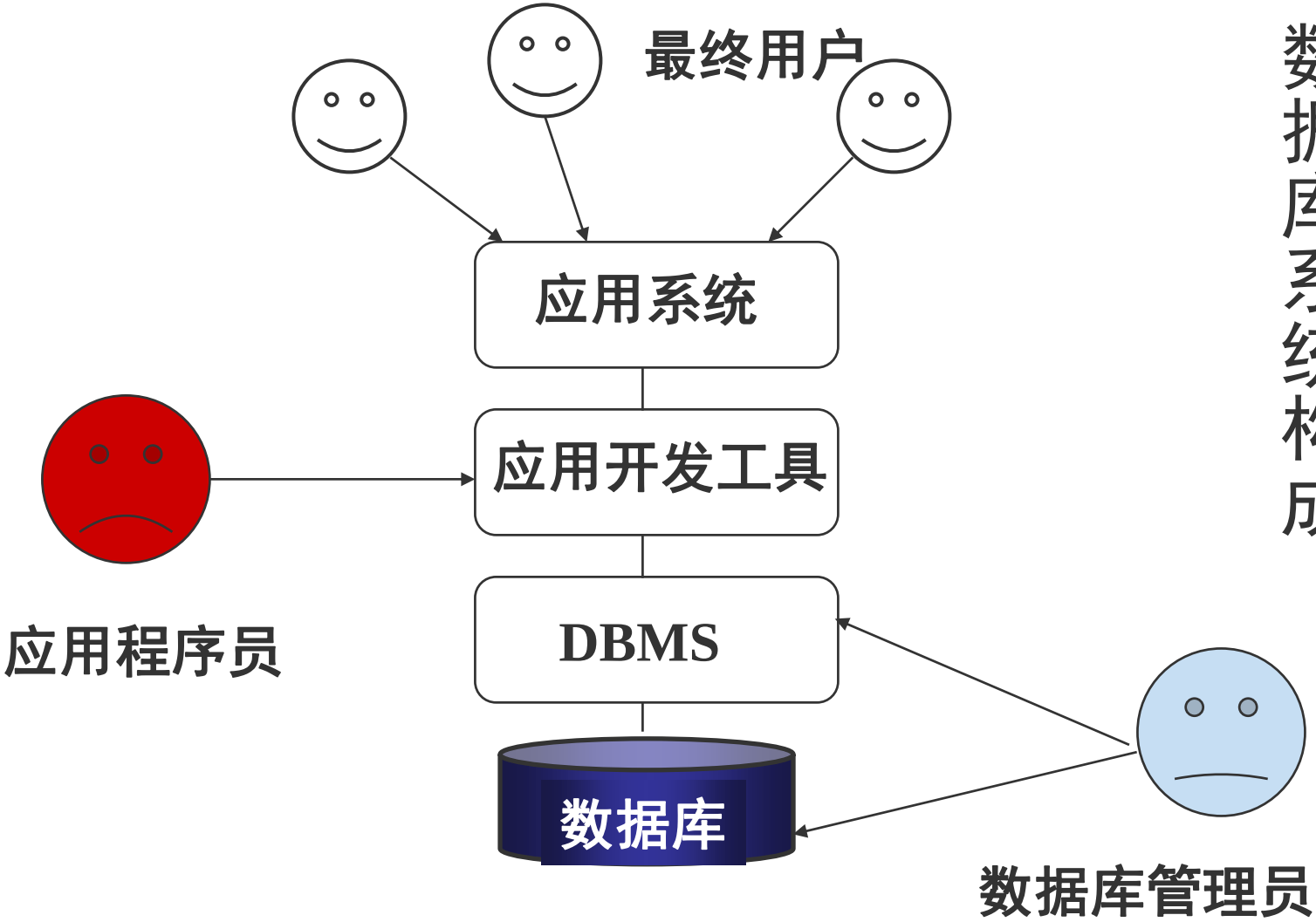
本章内容

节	知识点	掌握程度	难易程度
数据库基础概述	数据库和数据库管理系统概念	了解	
	数据库的发展阶段	了解	
	数据库的类型	了解	
	关系数据库的数据结构	了解	
MySQL数据库介绍	MySQL数据库简介	了解	
MySQL数据库安装	MySQL数据库Linux系统下安装	掌握	
	MySQL数据库Windows系统下安装	掌握	
MySQL数据库操作	MySQL数据库操作	掌握	
MySQL数据库体系结构	MySQL数据库体系结构	理解	难
存储引擎	存储引擎	理解	难
MySQL字符集	MySQL字符集	掌握	

数据库基础概述

- 数据库管理系统概述
  - 数据库（DB）是一种专门存储信息和维护信息的容器，严格地说数据库是“按照数据结构来组织、存储和管理信息的仓库”。
  - 数据库管理系统（Database Management System—DBMS）管理数据库的软件。具有对数据存储、安全、一致性、并发操作、恢复和访问等功能。
  - 数据词典（系统目录），也是一种数据，只不过这些数据记录的是数据库中存放的各种对象的定义信息和其他一些辅助管理信息，包括名字、结构、位置、类型等。这些数据被称为元数据（metadata）。

数据库基础概述



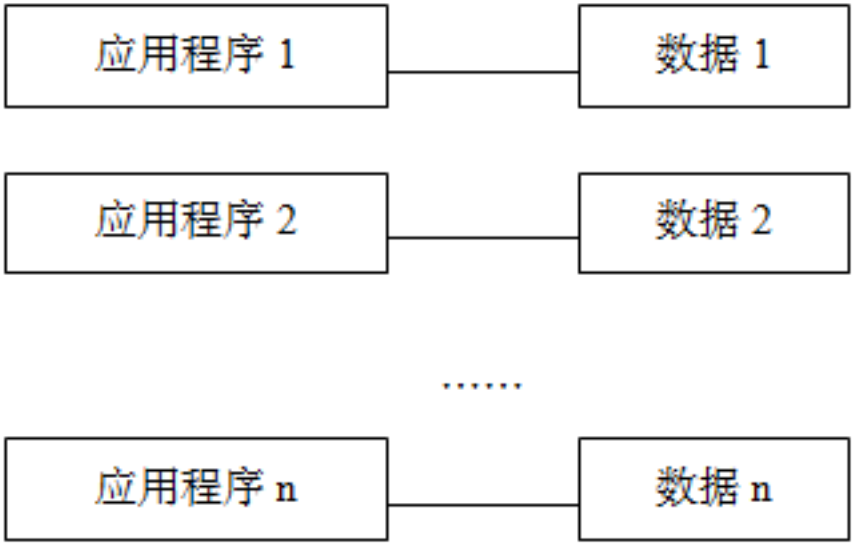
数据库系统构成

数据库基础概述

- 数据管理主要经历过程：
  - **手工管理阶段** ：应用程序管理数据、数据不保存、不共享、不具有独立性。
  - **文件管理阶段** ：文件系统管理数据、数据可长期保存、但共享性差、冗余度大、独立性差。
  - **数据管理阶段**：数据库系统管理数据、数据结构复杂、冗余小、易扩充、较高的独立性、统一数据控制。

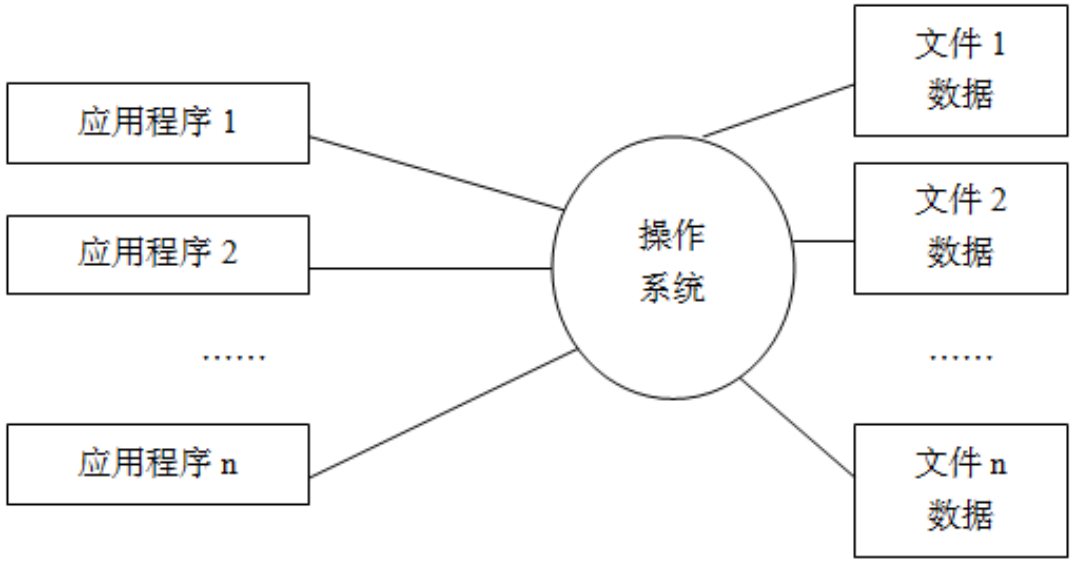
数据库基础概述

- 人工管理阶段：
  - 早期的数据处理都是通过手工进行的，那时的计算机多用于科学计算。每个应用程序根据需求组织数据，数据与程序一一对应，一个程序的数据一般不能被其他程序使用。如图所示。
  - 此阶段没有专门的数据管理软件，程序员既要考虑数据的逻辑结构，还要设计存储数据的物理结构及存取方法等。



数据库基础概述

- 文件系统阶段：
  - 随着操作系统的诞生，文件系统也作为操作系统的一个子系统应运而生了。应用程序可以通过文件系统将的数据组织成一个文件。文件系统提供对文件的访问和管理接口。文件系统阶段程和数据的关系如图所示。这种方式多用于早期的单机信息管理系统。

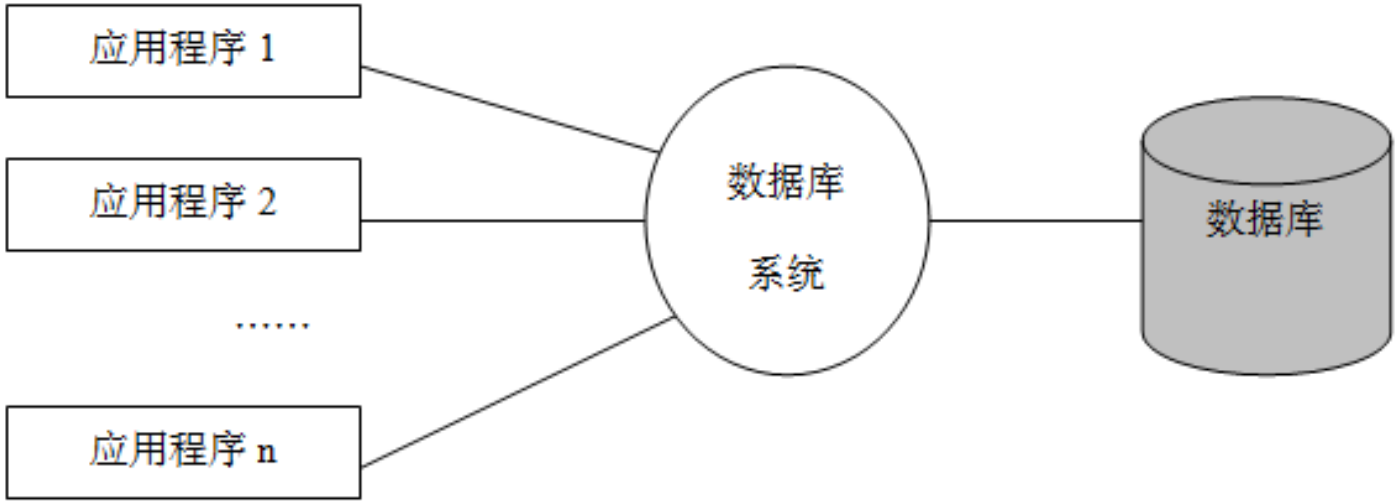


数据库基础概述

- 文件系统不足：
  - 文件系统虽然提供了统一的存取方法来操作数据，但保存数据的格式和结构却由应用程序自定义。从文件中读取数据后，需要自行解析数据。
  - 数据量比较大时检索数据的效率通常很低。
  - 数据冗余度大，相同的数据集合中不同应用程序中使用，经常需要重复定义、重复存储。例如，人事部的档案管理系统和财务部的工资管理系统用到的很多数据是重复的。它们各自使用自己的文件来存储数据
  - 数据不一致性，由于数据重复存储、单独管理，给数据维护带来难度，容易造成数据不一致。

数据库基础概述

- 文件系统阶段：
  - 数据库系统是由计算机软件和硬件资源组成的系统，它实现了有组织地、动态地存储大量关联数据，便于多用户访问。数据库系统与文件系统的重要区别是数据的充分共享、交叉访问，应用程序的高度独立性。文件系统阶段程和数据的关系如图所示。
  - 数据库对数据的存储是按照同一结构进行的，不同应用程序都可以直接操作这些数据。数据库系统对数据的完整性、唯一性和安全性都提供有效的管理手段。数据库系统还提供管理和控制数据的简单操作命令。



数据库基础概述

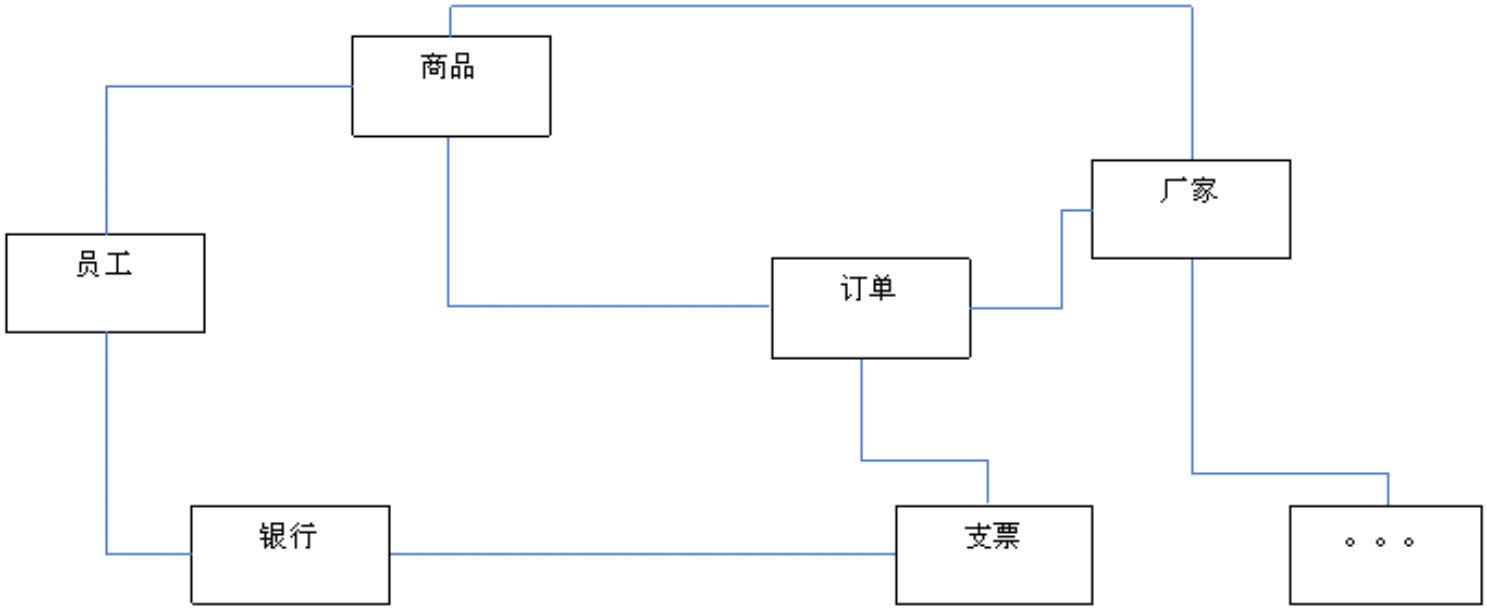
- 数据库的特征：
  - 数据结构化
  - 实现数据共享
  - 减少数据冗余
  - 数据独立性

数据库基础概述

- 数据库类型
  - 按数据模型特点分：
    - 网状型数据库
    - 层次型数据库
    - 关系型数据库

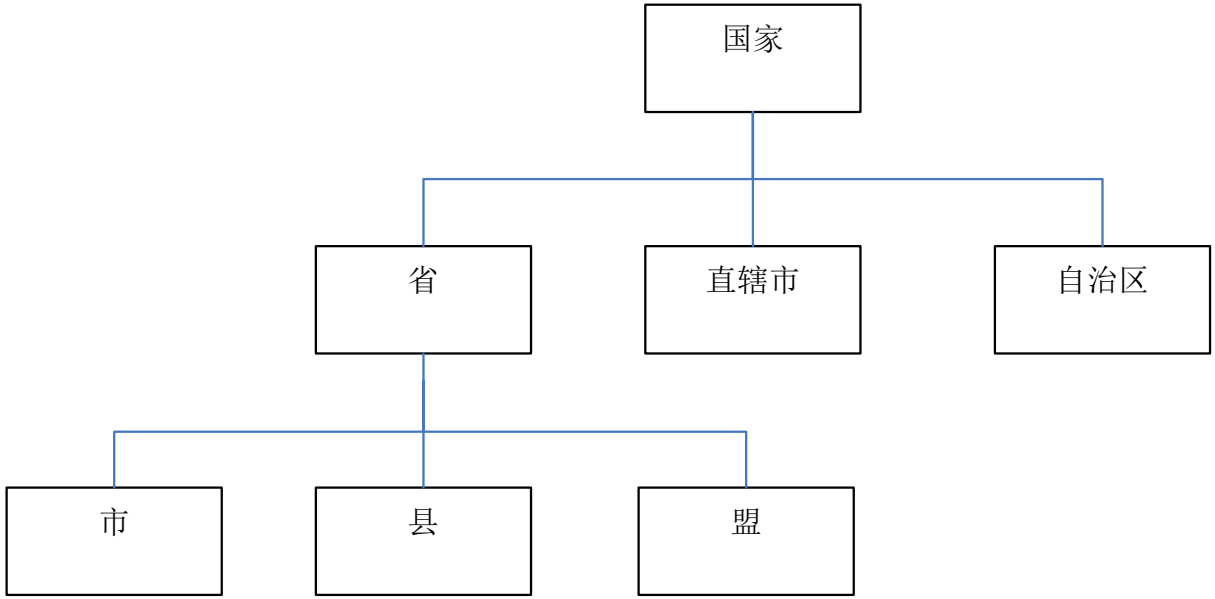
数据库基础概述

- 网状数据库：采用记录类型为节点的网状数据模型



数据库基础概述

- 层次数据库：采用层次模型模拟现实世界中按层次组织起来的事物



数据库基础概述

- 关系型数据库：采用二维表结构组织和管理数据，并规定了表内和表间数据的依赖关系

学号	姓名	性别	电话
Stu10001	张力	男	83620089
Stu10002	王南	女	86368888
Stu10004	李那	女	23910001

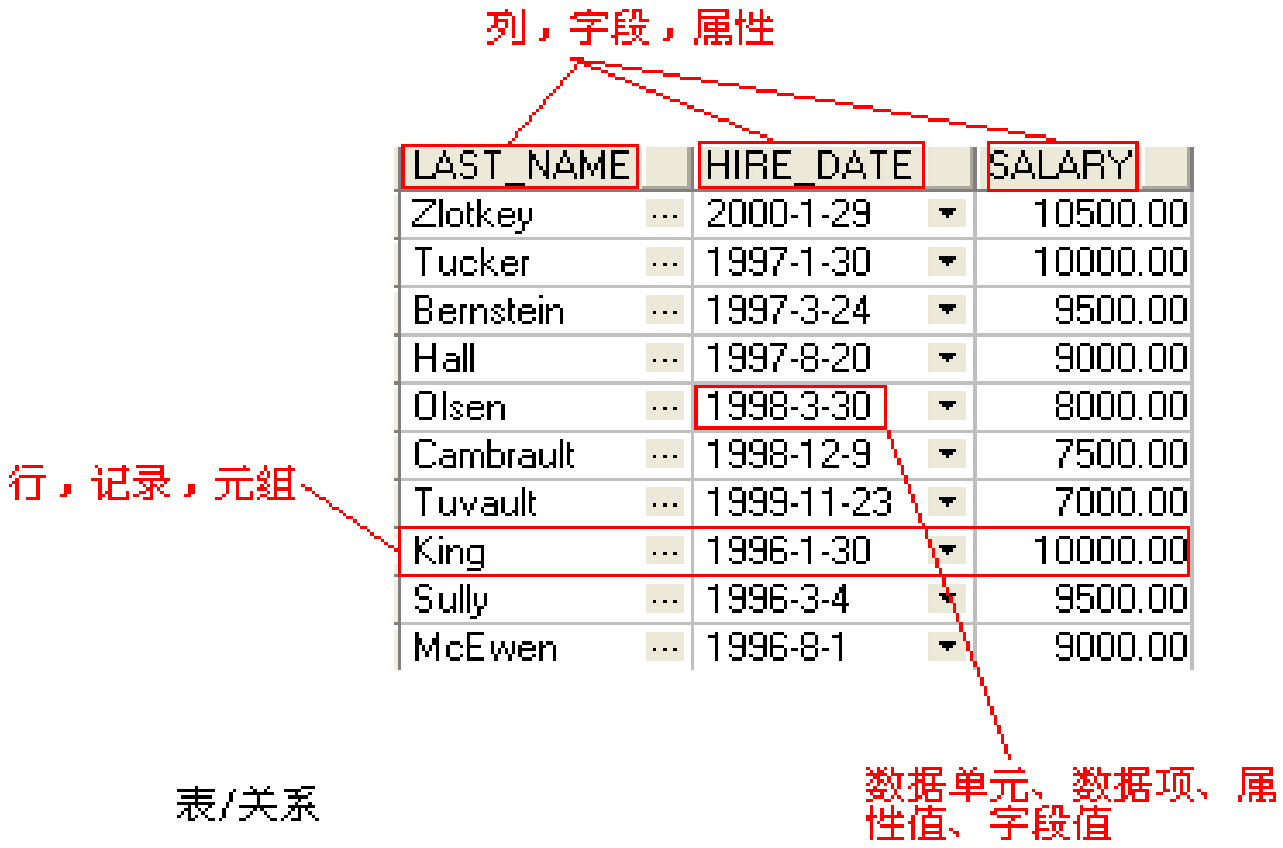
学号	科目	成绩
Stu10001	英语	80
Stu10001	高数	90
Stu10002	物理	60

数据库基础概述

- **关系数据库**是指一些相关的表和其他数据库对象的集合。对于关系数据库来说，关系就是表的同义词。
- 表是由行和列组成（类似二维数组的结构）。
  - 列包含一组命名的属性（也称字段）。
  - 行包含一组记录，每行包含一条记录。
  - 行和列的交集称为数据项，指出了某列对应的属性在某行上的值，也称为字段值。
  - 列需定义数据类型，比如整数或者字符型的数据。

数据库基础概述

- 关系数据库的数据结构图示：



数据库基础概述

- 关系型数据库管理系统（RDBMS）
  - 数据库管理系统（DBMS）是用来管理数据的计算机软件，它能使用户方便地定义和操纵数据、维护数据的安全性和完整性，以及进行多用户下的并发控制和恢复数据库。
  - 关系型数据库管理系统（RDBMS）是应用最广泛的一种数据库管理系统，关系型数据库管理系统以表、字段和记录等结构来组织数据。表用来保存数据，每个表由一组字段来定义其结构，记录则是表中的一条数据。本章介绍的MySQL就是一款常用的关系型数据库管理系统。

数据库基础概述

- 常见关系数据库
  - Oracle
  - DB2
  - Sybase
  - Microsoft SQL Server
  - MySQL
  - MySQL就是当前Web开发中尤其是PHP开发中使用最为广泛的数据库。

MySQL数据库系统简介

- MySQL是由瑞典 MySQL AB公司开的一种开放源代码的关系型数据库管理系统（RDBMS），目前属于 Oracle 旗下产品。MySQL数据库系统使用最常用的数据库管理语言——结构化查询语言（SQL）进行数据库管理。由于MySQL是开放源代码的，因此任何人都可以在General Public License的许可下下载并根据个性化的需要对其进行修改。MySQL因为其速度、可靠性和适应性而备受关注。大多数人都认为在不需要事务化处理的情况下，MySQL是管理内容最好的选择。

MySQL数据库系统简介

- MySQL关系型数据库于1998年1月发行第一个版本。它使用系统核心提供的多线程机制提供完全的多线程运行模式，提供了面向C、C++、Eiffel、Java、Perl、PHP、Python等编程语言的编程接口，支持多种字段类型并且提供了完整的操作符。

MySQL数据库系统简介

- 2001年MySQL4.0版本发布。在这个版本中提供了新的特性：新的表定义文件格式、高性能的数据复制功能、更加强大的全文搜索功能等。目前，MySQL已经发展到MySQL5.7，功能和效率方面都得到了更大的提升。

MySQL数据库系统简介

- 三款MySQL分支产品
  - Percona Server
    - 由领先的MySQL咨询公司Percona发布，是一款独立的数据库产品，其可以完全与MySQL兼容，可以在不更改代码的情况了下将存储引擎更换成XtraDB 。
  - MariaDB
    - 由MySQL的创始人麦克尔·维德纽斯主导开发，可以将它视为MySQL的扩展集，它不仅提供MySQL提供的所有功能，还提供其他功能。MariaDB还声称自己是MySQL的替代，因此从MySQL切换到MariaDB时，无需更改任何基本代码即可安装它。
  - Drizzle
    - Drizzle与MySQL有很大差别，它不是MySQL的替代产品，它与MySQL不兼容。

MySQL数据库系统简介

- 三款MySQL分支产品

产品	价格	目标	主要功能	是否可投入生产?
Percona Server	免费	提供 XtraDB 存储引擎的包装器和其他分析工具	XtraDB	是
MariaDB	免费	扩展 MySQL 以包含 XtraDB 和其他性能改进	XtraDB	是
Drizzle	免费	提供比 MySQL 更强大的可扩展性和性能改进	高可用性	是

MySQL的Linux系统下安装

- 下载MySQL的安装包
  - 在MySQL官网上下载最新版的Ubuntu Linux专用的MySQL
  - https://dev.mysql.com/downloads/mysql/
  - mysql-server_5.7.17-1ubuntu16.04_amd64.deb-bundle.tar
- 解压文件,命令为：
  - root@ubuntu:/fly/mysql# tar -xvf mysql-server_5.7.17-1ubuntu16.04_amd64.deb-bundle.tar
  - 解压开来后，一共有11个deb包
- 用sudo dpkg -i [包名]命令逐个安装
  - 因为包与包中间存在依赖关系，这里安装有个先后顺序。

MySQL的Linux系统下安装

- 安装的顺序是：
  1. mysql-common_5.7.17-1ubuntu16.04_amd64.deb
  2. libmysqlclient20_5.7.17-1ubuntu16.04_amd64.deb
  3. libmysqlclient-dev_5.7.17-1ubuntu16.04_amd64.deb
  4. libmysqld-dev_5.7.17-1ubuntu16.04_amd64.deb
  5. mysql-community-client_5.7.17-1ubuntu16.04_amd64.deb
  6. mysql-client_5.7.17-1ubuntu16.04_amd64.deb
  7. mysql-community-source_5.7.17-1ubuntu16.04_amd64.deb
- 这里需要再安装一个依赖包叫**libmecab2**，安装好后，继续安装最后一个：
  8. mysql-community-server_5.7.17-1ubuntu16.04_amd64.deb

  安装过程中需要设置数据库密码。
- 到这里，所有的已经安装完毕。输入Mysql -u root -p可以登陆数据库了。

MySQL的Linux系统下安装

- 文件默认位置
  - /usr/bin　　客户端程序和脚本
  - /usr/sbin mysqld　　服务器
  - /var/lib/mysql　　日志文件，数据库
  - /usr/share/doc　　文档
  - /usr/include/mysql　　包含（头）文件
  - /usr/lib/mysql　　库
  - /usr/share/mysql　　错误消息和字符集文件

MySQL的Linux系统下安装

- 安装MySQL后，默认情况下，服务器是启动的
- 停止服务

  /etc/init.d/mysql stop

```
root@neuubuntu:~# /etc/init.d/mysql stop
[ ok ] Stopping mysql (via systemctl): mysql.service.
```

- 启动服务

```
root@neuubuntu:~# /etc/init.d/mysql start
[ ok ] Starting mysql (via systemctl): mysql.service.
root@neuubuntu:~#
```

MySQL的Linux系统下安装

- 输入如下格式的命令登录数据库
  mysql -u user -p
  - -h后面MySQL数据库服务器
  - -u后面用户名
  - -p用于指定用户名对应的密码
  - 如果-p后面没有输入密码，系统会提示用户输入

```
root@neuubuntu:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

MySQL数据库操作

- 查看版本
  - MySQL>select version();
  - SELECT是从数据库中查询数据的标准SQL语句，VERSION()是MySQL函数，用于返回MySQL数据库的版本信息。每行命令后面需要输入分号（;）后才能被执行。输入完成后，按下回车键，执行结果如下：

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.17    |
+-----------+
1 row in set (0.00 sec)

mysql>
```

MySQL数据库操作

- MySQL服务器在安装后默认创建了information_schema、mysql、performance_schema、sakila、sys、world六个数据库。
- 使用show databases命令可以查看数据库
  mysql>show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

MySQL数据库操作

- 使用USE命令切换SQL语句作用的数据库，语法如下：
  USE 数据库名;
  mysql>use mysql;
- 执行USE语句后，在执行其他SQL语句默认都作用于指定的数据库。

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

MySQL数据库操作

- 用于显示数据库中的表信息，其语法结构如下：
  mysql>show tables;

```
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| engine_cost               |
| event                     |
| func                      |
| general_log               |
| gtid_executed             |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| innodb_index_stats        |
| innodb_table_stats        |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| proxies_priv              |
| server_cost               |
| servers                   |
| slave_master_info         |
| slave_relay_log_info      |
| slave_worker_info         |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
```

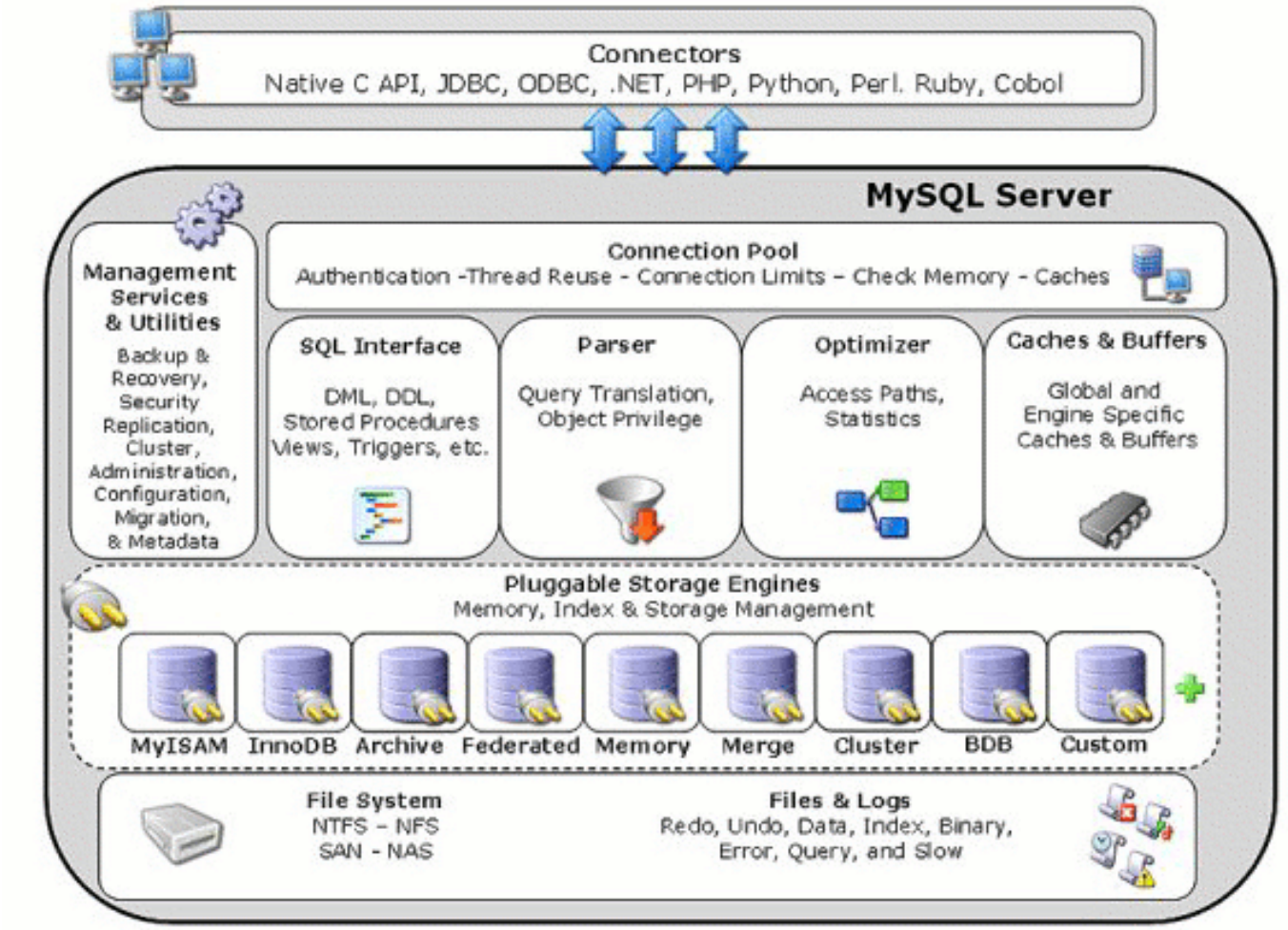
MySQL数据库操作

- 查看具体的表结构，其语法结构如下：
- describe tablename;
  mysql>describe user;显示MySQL数据库中user表的列信息

```
mysql> describe user;
+------------------------+-----------------------------------+------+-----+-----------------------+-------+
| Field                  | Type                              | Null | Key | Default               | Extra |
+------------------------+-----------------------------------+------+-----+-----------------------+-------+
| Host                   | char(60)                          | NO   | PRI |                       |       |
| User                   | char(32)                          | NO   | PRI |                       |       |
| Select_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Insert_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Update_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Delete_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Create_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Drop_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Reload_priv            | enum('N','Y')                     | NO   |     | N                     |       |
| Shutdown_priv          | enum('N','Y')                     | NO   |     | N                     |       |
| Process_priv           | enum('N','Y')                     | NO   |     | N                     |       |
| File_priv              | enum('N','Y')                     | NO   |     | N                     |       |
| Grant_priv             | enum('N','Y')                     | NO   |     | N                     |       |
| References_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Index_priv             | enum('N','Y')                     | NO   |     | N                     |       |
| Alter_priv             | enum('N','Y')                     | NO   |     | N                     |       |
| Show_db_priv           | enum('N','Y')                     | NO   |     | N                     |       |
| Super_priv             | enum('N','Y')                     | NO   |     | N                     |       |
| Create_tmp_table_priv  | enum('N','Y')                     | NO   |     | N                     |       |
| Lock_tables_priv       | enum('N','Y')                     | NO   |     | N                     |       |
| Execute_priv           | enum('N','Y')                     | NO   |     | N                     |       |
| Repl_slave_priv        | enum('N','Y')                     | NO   |     | N                     |       |
| Repl_client_priv       | enum('N','Y')                     | NO   |     | N                     |       |
| Create_view_priv       | enum('N','Y')                     | NO   |     | N                     |       |
| Show_view_priv         | enum('N','Y')                     | NO   |     | N                     |       |
| Create_routine_priv    | enum('N','Y')                     | NO   |     | N                     |       |
| Alter_routine_priv     | enum('N','Y')                     | NO   |     | N                     |       |
| Create user_priv       | enum('N','Y')                     | NO   |     | N                     |       |
```

MySQL体系结构

- 一个由多个子系统构成的层次化系统



MySQL体系结构

- Connectors:用来与客户端应用程序建立连接的数据库接口
- Management Services & Utilities:系统管理和服务控制相关的辅助工具
- Connection Pool：负责处理与用户访问有关的各种用户登录、线程处理、内存和进程缓存需求
- Sql Interface：提供从用户接受命令并把结果返回给用户的机制
- Parser:对SQL语句进行语法分析和解析，构造一个用来执行查询的数据结构
- Optimizer:优化查询语句，以保证数据检索动作的效率达到或者非常接近最最优。使用一种“选取-投影-联结”策略来处理查询，即先根据有关的限制条件进行选取(Select 操作)以减少将要处理的元组个数，再进行投影以减少被选取元组列的属性字段的个数，最后根据连接条件生产最终的查询结果

MySQL体系结构

- Caches & Buffers：保证使用频率最高的数据或结构能够以最有效率的方式被访问，缓存的类型有：表缓存、记录缓存、键缓存、权限缓存、主机名缓存等。

MySQL体系结构

- 最与众不同的特点是**插件式存储引擎**
- 插件式表存储引擎是底层物理结构的实现，负责为数据库执行实际的数据I/O操作，它是基于表而不是数据库的。可以根据实际应用需求为每个表设定不同的选择。
- 插件式存储引擎的核心是文件访问层的一个抽象接口，任何人都可以利用这个API接口去建立新的文件访问机制

存储引擎

- 存储引擎就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等技术的实现方法。因为在关系数据库中数据的存储是以表的形式存储的，所以存储引擎简而言之就是指表的类型。数据库的存储引擎决定了表在计算机中的存储方式。
- 在Oracle和SQL Server等数据库中只有一种存储引擎，所有数据存储管理机制都是一样的。而MySQL数据库提供了多种存储引擎，用户可以根据不同的需求为数据表选择不同的存储引擎，用户也可以根据自己的需要编写自己的存储引擎，MySQL的核心就是存储引擎。

存储引擎

- 使用MySQL命令“show engines;”，即可查看MySQL服务实例支持的存储引擎。

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)
```

存储引擎

- InnoDB存储引擎的特点:
  - 支持外键（Foreign Key）
  - 支持事务（Transaction）：如果某张表主要提供OLTP支持，需要执行大量的增、删、改操作（insert、delete、update语句），出于事务安全方面的考虑，InnoDB存储引擎是更好的选择。
  - 最新版本的MySQL已经开始支持全文检索。

存储引擎

- MyISAM存储引擎的特点：
  - MyISAM具有检查和修复表的大多数工具。
  - MyISAM表可以被压缩
  - MyISAM表最早支持全文索引
  - 但MyISAM表不支持事务
  - 但MyISAM表不支持外键（Foreign Key）。
- 如果需要执行大量的select语句，出于性能方面的考虑，MyISAM存储引擎是更好的选择。

存储引擎

Neuedu

- MEMORY存储引擎：
  - MEMORY存储引擎是MySQL中一类特殊的存储引擎。该存储引擎使用存在于内存中的内容来创建表，每个表实际对应一个磁盘文件，格式为.frm。这类表因为数据在内存中，且默认使用HASH索引，所以访问速度非常快；但一旦服务关闭，表中的数据会丢失。
  - 每个MEMORY表可以放置数据量的大小受max_heap_table_size系统变量的约束，初始值为16MB，可按需求增大。此外，在定义MEMORY表时可通过MAX_ROWS子句定义表的最大行数。
  - 该存储引擎主要用于那些内容稳定的表，或者作为统计操作的中间表。对于该类表需要注意的是，因为数据并没有实际写入磁盘，一旦重启，则会丢失。

存储引擎

Neuedu

- MySQL5.7默认的默认的存储引擎是InnoDB。

  使用MySQL命令

  ```
  set default_storage_engine=MyISAM;
  ```

  可以“临时地”将MySQL“当前会话的”存储引擎设置为MyISAM，使用MySQL命令“show engines;”可以查看当前MySQL服务实例默认的存储引擎。

存储引擎

Neuedu

- **存储引擎的选择**
  - 不同存储引擎都有各自的特点，以适应不同的需求

MySQL存储引擎功能对比

功　　能	InnoDB	MyISAM	Memory
存储限制	64TB	256TB	RAM
支持事务	支持	无	无
空间使用	高	低	低
内存使用	高	低	高
支持数据缓存	支持	无	无
插入数据速度	低	高	高
支持外键	支持	无	无

MySQL字符集

- 字符(Character)是指人类语言最小的表义符号，例如'A'、'B'等。
- 给定一系列字符，对每个字符赋予一个数值，用数值来代表对应的字符，这个数值就是字符的编码(Character Encoding)。
- 给定一系列字符并赋予对应的编码后，所有这些字符和编码对组成的集合就是字符集(Character Set)。MySQL中提供了多种字符集，例如latin1、utf8、gbk等。
- 字符序(Collation)是指在同一字符集内字符之间的比较规则。只有确定字符序后，才能在一个字符集上定义什么是等价的字符，以及字符之间的大小关系。每个字符序唯一对应一种字符集，一个字符集可以对应多种字符序，其中有一个是默认字符序(Default Collation)。

MySQL字符集

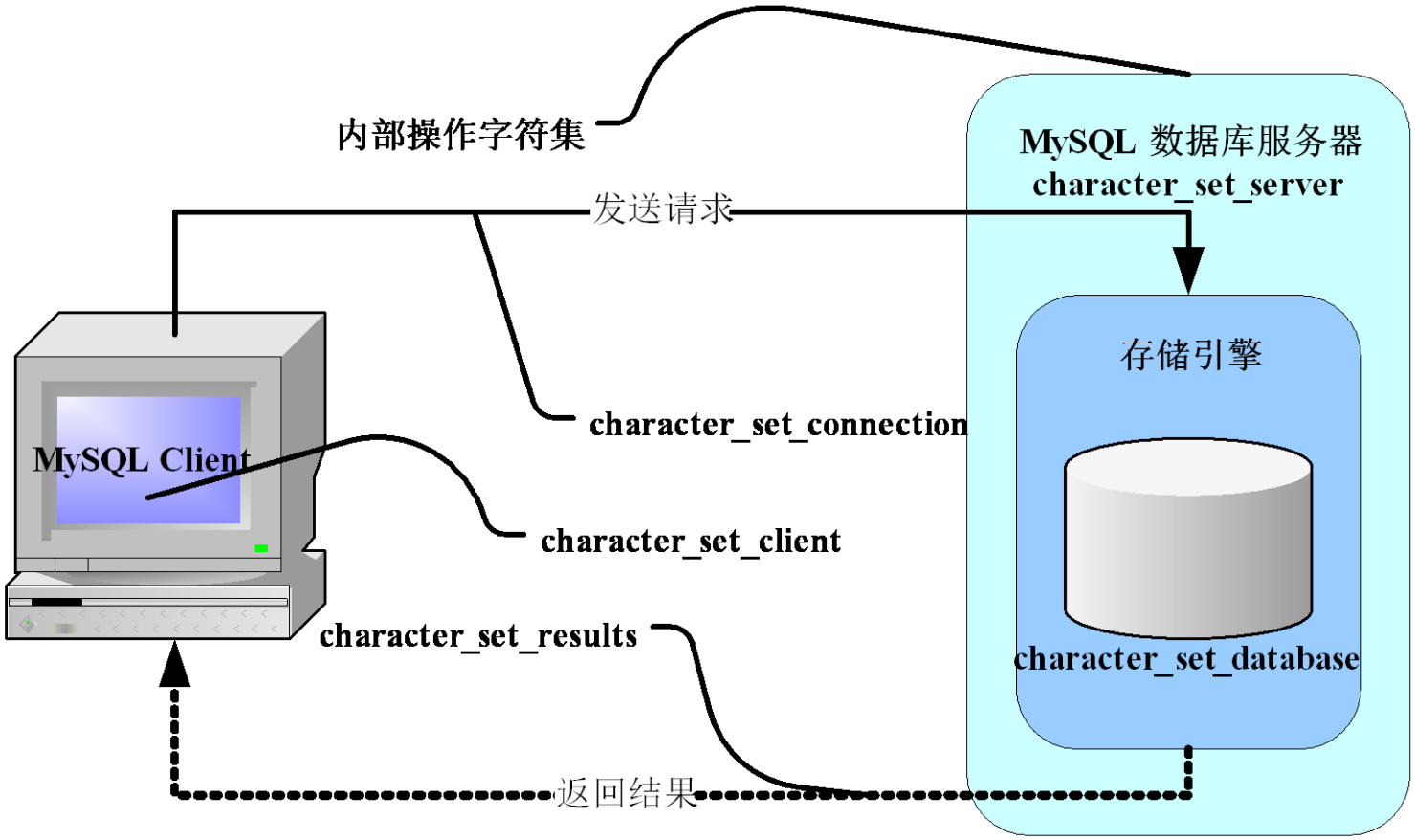
- MySQL中的字符序命名规则：以字符序对应的字符集名称开头，以国家名居中（或以general居中），以ci或cs或bin末尾。例如：latin1字符集对应有latin1_swedish_ci、latin1_spanish_ci、latin1_german1_ci等字符序。以ci结尾的字符序表示大小写不敏感；以cs结尾的字符序表示大小写敏感；以bin结尾的字符序表示按编码值比较。例如在字符序gbk_general_ci规则中，字符'a'和'A'是等价的。

MySQL字符集

- 字符(Character)是指人类语言最小的表义符号，例如‘A’、‘B’等。
- character_set_client：客户端（MySQL命令窗口）的字符集（未加说明的是latin1字符集）
- character_set_connection：连接层字符集
- character_set_database：当前选中数据库的字符集
- character_set_filesystem：MySQL服务器文件系统的字符集，默认的字符集为binary
- character_set_results：结果集的字符集
- character_set_server：MySQL服务器的字符集
- character_set_system：元数据(字段名、表名、数据库名等)字符集，默认的字符集为utf8

MySQL字符集

MySQL中的字符集转换过程



MySQL字符集

- 查看MySQL数据库服务器和数据库MySQL字符集
  mysql> show variables like '%char%';

```
mysql> show variables like '%char%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.01 sec)
```

MySQL字符集

- 查看MySQL数据表（table）的MySQL字符集
  mysql> show table status from MySQL like '%slave%';

```
mysql> show table status from mysql like '%slave%';
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+-----------------+----------+
| Name                 | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time | Check_time      | Collation | Checksum | Create_options      | Comment                |
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+-----------------+----------+
| slave_master_info    | InnoDB |      10 | Dynamic    |    0 |              0 |       16384 |               0 |            0 |         0 |           NULL | 2017-03-07 14:35:03 | NULL        |            NULL | utf8_general_ci |     NULL | stats_persistent=0 | Master Information     |
| slave_relay_log_info | InnoDB |      10 | Dynamic    |    0 |              0 |       16384 |               0 |            0 |         0 |           NULL | 2017-03-07 14:35:03 | NULL        |            NULL | utf8_general_ci |     NULL | stats_persistent=0 | Relay Log Information  |
| slave_worker_info    | InnoDB |      10 | Dynamic    |    0 |              0 |       16384 |               0 |            0 |         0 |           NULL | 2017-03-07 14:35:03 | NULL        |            NULL | utf8_general_ci |     NULL | stats_persistent=0 | Worker Information     |
+----------------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+-------------+-----------------+----------+
3 rows in set (0.00 sec)
```

MySQL字符集

- 看MySQL数据列（column）的MySQL字符集
  mysql> show full columns from user;

```
mysql> show full columns from user;
+-----------------------+---------------------------------+-----------------+------+-----+-----------------------+-------+-------------------
------------+---------+
| Field                 | Type                            | Collation       | Null | Key | Default               | Extra | Privileges
           | Comment |
+-----------------------+---------------------------------+-----------------+------+-----+-----------------------+-------+-------------------
------------+---------+
| Host                  | char(60)                        | utf8_bin        | NO   | PRI |                       |       | select,insert,update
,references |         |
| User                  | char(32)                        | utf8_bin        | NO   | PRI |                       |       | select,insert,update
,references |         |
| Select_priv           | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
,references |         |
| Insert_priv           | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
,references |         |
| Update_priv           | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
,references |         |
| Delete_priv           | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
,references |         |
| Create_priv           | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
,references |         |
| Drop_priv             | enum('N','Y')                   | utf8_general_ci | NO   |     | N                     |       | select,insert,update
```

MySQL字符集

- MySQL提供的下列命令可以在不影响其它数据库字符集的基础上临时修改当前的字符集
- SET character_set_client = gbk;
- SET character_set_connection = gbk;
- SET character_set_database = gbk;
- SET character_set_results = gbk;
- SET character_set_server = gbk;
- SET collation_connection = gbk_chinese_ci ;
- SET collation_database = gbk_chinese_ci ;
- SET collation_server = gbk_chinese_ci ;

MySQL字符集

- MySQL中还可以使用MySQL命令： “set names gbk;”一次性地设置character_set_client、character_set_connection和character_set_results的字符集为gbk。

```
mysql> set names gbk;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like '%char%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | gbk                        |
| character_set_connection | gbk                        |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | gbk                        |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

SQL脚本文件

- 在MySQL命令窗口上编辑MySQL命令或SQL语句是很不方便的，更多时候需要将常用的SQL语句封装为一个SQL脚本文件，使用 source 命令运行该SQL脚本文件中的所有SQL语句（SQL脚本文件的扩展名一般为sql）。
- 输入MySQL命令：source init.sql即可执行init.sql脚本文件

小结

- 了解数据库基本概念
- 了解数据库设计思想
- 了解MySQL数据库体系结构
- 掌握MySQL安装及字符集

课后作业

1. 在自己机器上安装MySQL软件，并查看数据库。

Neuedu