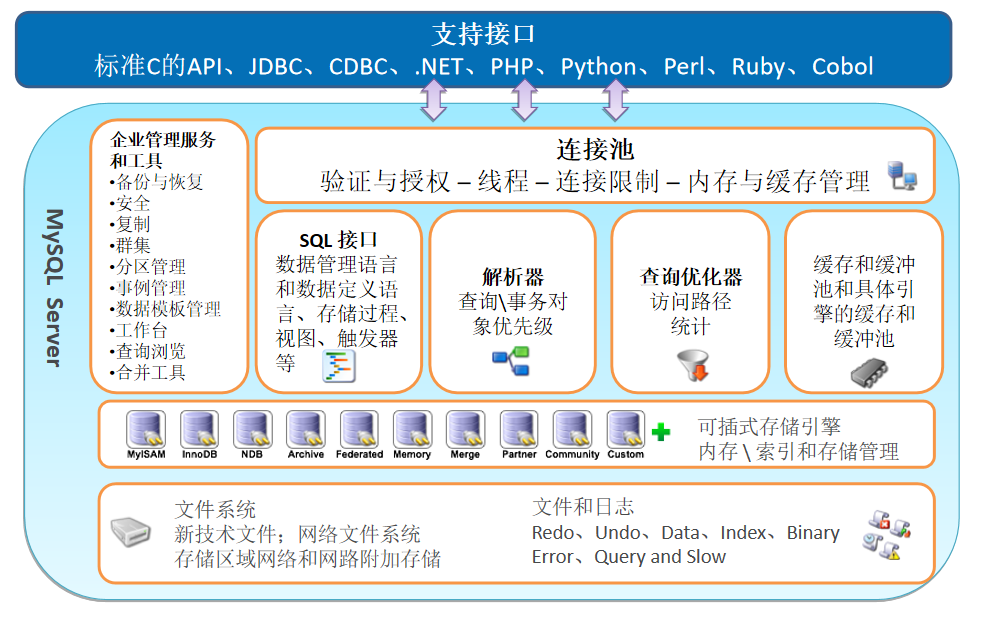
**Mysql篇**

1. **Mysql的概念**

Mysql时一款免费开源的关系型数据库，特点是免费开源、轻量便捷、功能齐全、可用于windows和Linux系统、适用于中小型甚至大型网站应用。

1. **Mysql的运行机制/逻辑架构**



1. connectors

MySQL首先是一个网络程序，其在TCP之上定义了自己的[应用层协议](https://dev.mysql.com/doc/internals/en/client-server-protocol.html)。所以要使用MySQL，我们可以编写代码，跟MySQL Server建立TCP连接，之后按照其定义好的协议进行交互。当然这样比较麻烦，比较方便的办法是调用SDK，比如Native C API、JDBC、PHP等各语言MySQL Connector，或者通过ODBC。但通过SDK来访问MySQL，本质上还是在TCP连接上通过MySQL协议跟MySQL进行交互。

1. Connection Management

每一个基于TCP的网络服务都需要管理客户端链接，MySQL也不例外。MySQL会为每一个连接绑定一个线程，之后这个连接上的所有查询都在这个线程中执行。为了避免频繁创建和销毁线程带来开销，MySQL通常会缓存线程或者使用线程池，从而避免频繁的创建和销毁线程。客户端连接到MySQL后，在使用MySQL的功能之前，需要进行认证，认证基于用户名、主机名、密码。如果用了SSL或者TLS的方式进行连接，还会进行证书认证。

1. SQL Interface

MySQL支持DML（数据操作语言）、DDL（数据定义语言）、存储过程、视图、触发器、自定义函数等多种SQL语言接口。

1. Parser

MySQL会解析SQL查询，并为其创建语法树，并根据数据字典丰富查询语法树，会验证该客户端是否具有执行该查询的权限。创建好语法树后，MySQL还会对SQl查询进行语法上的优化，进行查询重写。

1. Optimizer

语法解析和查询重写之后，MySQL会根据语法树和数据的统计信息对SQL进行优化，包括决定表的读取顺序、选择合适的索引等，最终生成SQL的具体执行步骤。这些具体的执行步骤里真正的数据操作都是通过预先定义好的存储引擎API来进行的，与具体的存储引擎实现无关。

1. Caches & Buffers

MySQL内部维持着一些Cache和Buffer，比如Query Cache用来缓存一条Select语句的执行结果，如果能够在其中找到对应的查询结果，那么就不必再进行查询解析、优化和执行的整个过程了。

1. Pluggable Storage Engine

存储引擎的具体实现，这些存储引擎都实现了MySQl定义好的存储引擎API的部分或者全部。MySQL可以动态安装或移除存储引擎，可以有多种存储引擎同时存在，可以为每个Table设置不同的存储引擎。存储引擎负责在文件系统之上，管理表的数据、索引的实际内容，同时也会管理运行时的Cache、Buffer、事务、Log等数据和功能。

**3 Mysql数据表的存储位置**

MySQL数据表以文件方式存放在磁盘中

包括表文件、数据文件以及数据库的选项文件

位置：MySQL安装目录\data下存放数据表。目录名对应数据库名，该目录下文件名对应数据表

**4 结构化查询语句**



**4.1 DDL**

CREATE DATABASE [IF NOT EXISTS] ——创建数据库

DROP DATABASE [IF EXISTS] ——删除数据库

SHOW DATABASES ——查询数据库

USE [数据库名] ——使用数据库

创建数据库表语句：

CREATE TABLE [ IF NOT EXISTS ] `表名` (

`字段名1` 列类型 [ 属性 ] [ 索引 ] [注释] ,

`字段名2` 列类型 [ 属性 ] [ 索引 ] [注释] ,

… …

`字段名n` 列类型 [ 属性 ] [ 索引 ] [注释]

) [ 表类型 ] [ 表字符集 ] [注释] ;

|  |  |  |
| --- | --- | --- |
| 列类型 | 数值类型 |  |
| 字符串类型 |  |
| 日期、时间 |  |
| NULL值 |  |
| 属性 | UNSIGNED | 无符号，不允许数值为负数 |
| ZEROFILL | 0填充的，不足位数用0填充 |
| AUTO\_INCREMENT | 自增长，可用于设置主键，整数类型，可设置步长和起始值 |
| NULL/NOT NULL | 允许值为空或非空 |
| DEFAULT | 用于设置默认值 |
| 注释 | COMMENT ‘注释名’ |  |
| 数据引擎/表类型 | ENGINE | MyISAM、InnoDB 、HEAP、BOB、CSV等 |
| 索引 | INDEX | 主键索引，外键索引、联合索引、全文索引、普通索引 |

修改表数据

修改表名：ALTER TABLE 旧表名 RENAME AS 新表名

添加字段：ALTER TABLE 表名 ADD 字段名 列类型 [ 属性 ]

修改字段：ALTER TABLE 表名 MODIFY 字段名 列类型 [ 属性 ]

修改字段2：ALTER TABLE 表名 CHANGE 旧字段名 新字段名 列类型 [ 属性 ]

删除字段：ALTER TABLE 表名 DROP 字段名

说明：多个字段操作用括号包起来

删除表数据

DROP TABLE [IF EXISTS] [表名]

**4.2 DML**

1.INSERT语法

INSERT INTO 表名 [ ( 字段1, 字段2, 字段3, … ) ] VALUES ( '值1', '值2', '值3', …)

说明：

字段或值之间用英文逗号隔开

“字段1, 字段2…”该部分可省略，但添加的值务必与表结构数据列顺序相对应,且数量一致

可同时插入多条数据，values 后用英文逗号隔开

1. UPDATE语法

UPDATE 表名 SET column\_name = value [ , column\_name2 = value2, …. ]

[ WHERE condition ];

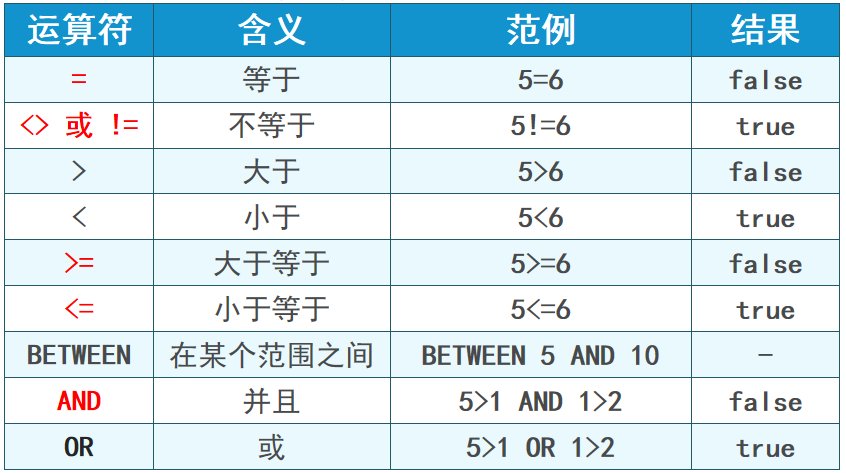
说明：

column\_name 为要更改的数据列

value 为修改后的数据，可以为变量、具体值、表达式或者嵌套的SELECT结果

condition为筛选条件，如不指定则修改该表的所有列数据

where中的运算符（红色标注为推荐使用，带=号的查询效率高）



1. DELETE语法

DELETE FROM 表名 [ WHERE condition ];

说明：

condition为筛选条件，如不指定则删除该表的所有列数据（慎重！！！）

1. TRUNCATE语法

TRUNCATE [TABLE] table\_name

说明：区别于DELETE命令

相同：都能删除数据、不删除表结构，但TRUNCATE 速度更快

不同：使用TRUNCATE TABLE重新设置AUTO\_INCREMENT计数器，使用TRUNCATE TABLE不会对事务有影响

***以下为扩展内容：***

* 1. 物理删除和逻辑删除
     1. 物理删除：使用删除语句删除，叫物理删除
        1. 后果：正常情况下数据无法恢复
        2. 优点：会释放数据空间
     2. 逻辑删除：默认情况下，该数据表当中所有的数据，是不会真正删除的
        1. 为什么要有逻辑删除：IT公司最珍贵的是数据
        2. 使用增量表来实现这一点，使用字段来指定该条数据是否被删除
        3. 好处：数据不会真正丢失
        4. 问题：表当中的数据量会很大，会要求做查询优化
  2. 数据版本号
     1. 如果多个用户，在需要同时修改某个数据的时候，如果不允许（效率不允许）使用排他策略，那么我们需要使用到数据版本号（乐观锁）。
     2. 每次修改数据，版本号都需要递增
     3. 每次修改数据，需要把版本号携带着，如果受影响数据量为0，则代表该次修改失败
  3. 分布式下的数据库集群
     1. 分库分表
     2. 读写分离
     3. 数据缓存

**4.3 DQL（最核心、最重要）**

查询数据库数据，如SELECT语句

简单的单表查询或多表的复杂查询和嵌套查询

数据库语言中最核心、最重要的语句

使用频率最高的语句

SELECT [ALL | DISTINCT]

{ \* | table.\* | [ table.field1 [ as alias1] [, table.field2 [as alias2]][, …]] }

FROM table\_name [ as table\_ alias ]

[ left|out|inner join table\_name2 ] #联合查询

[ WHERE … ] #指定结果需满足的条件

[ GROUP BY …] #指定结果按照哪几个字段来分组

[ HAVING …] #过滤分组的记录必须满足的次要条件

[ ORDER BY… ] #指定查询记录按一个或者多个条件排序

[ LIMIT { [ offset,] row\_count | row\_count OFFSET offset }] ; #指定查询的记录从哪条至哪条

说明：

[] 括号代表可选的；

{} 括号代表必须的；

# MySQL语句中的注释符，也可以用 /\*该处为注释\*/

1. 不推荐使用\*，查询效率低
2. 可以使用as关键字来为表名或表字段起别名，as关键字也可以省略
3. DISTINCT可以去重，注意的是所有字段值都相同，才算是重复
4. inner join 内连接（等值或非等值的连接查询，以及自连接查询）,返回匹配行
5. left join 左连接（显示前表所有记录条数，后表若无数据，则用NULL表示）
6. right join 右连接（显示后表所有记录条数，前表若无数据，则用NULL表示）
7. 自然连接 （笛卡尔积 m\*n）
8. order by 排序 ASC（升序，默认） DESC(降序)，使用多个字段排序时，优先级从左到右
9. limit m,n 分页，m表示起始位置，n表示结束位置，m可以省略

****limit分页公式：curPage是当前第几页；pageSize是一页多少条记录****

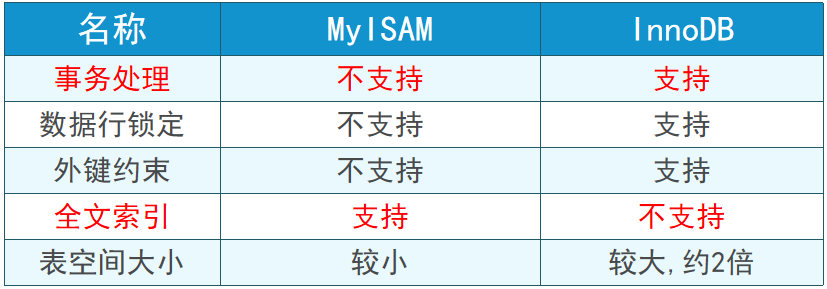
****limit (curPage-1)\*pageSize,pageSize，****

****总页数公式：totalRecord是总记录数；pageSize是一页分多少条记录****

****int totalPageNum = (totalRecord +pageSize - 1) / pageSize;****

1. like模糊匹配：%：0-n个字符 \_：1个任意字符
2. 匹配NULL或NOT NULL要用is ,而不能用等号
3. group by 分组查询，通常与having过滤条件一起使用
4. mysql函数：<https://www.runoob.com/mysql/mysql-functions.html>

**5 MyISAM和InnoDB引擎的区别**



适用场景：

MyISAM：适合做为增量表，节约空间，速度快，查询效率高，

InnoDB：适合做业务数据表，应用于增删改比较频繁的场合，支持事务，安全性好

InnoDB类型数据表只有一个\*.frm文件，以及上一级目录的 ibdata1 文件

MyISAM类型数据表对应三个文件：

\*.frm -- 表结构定义文件

\*.MYD -- 数据文件

\*.MYI -- 索引文件

delete from table时，innodb不会重新简历表，而会一行一行的删除。

myisam使用非聚集索引，索引和数据分开，只缓存索引；innodb使用聚集索引，索引和数据存在一个文件。