

# 数据库管理

**NSD DBA 基础**

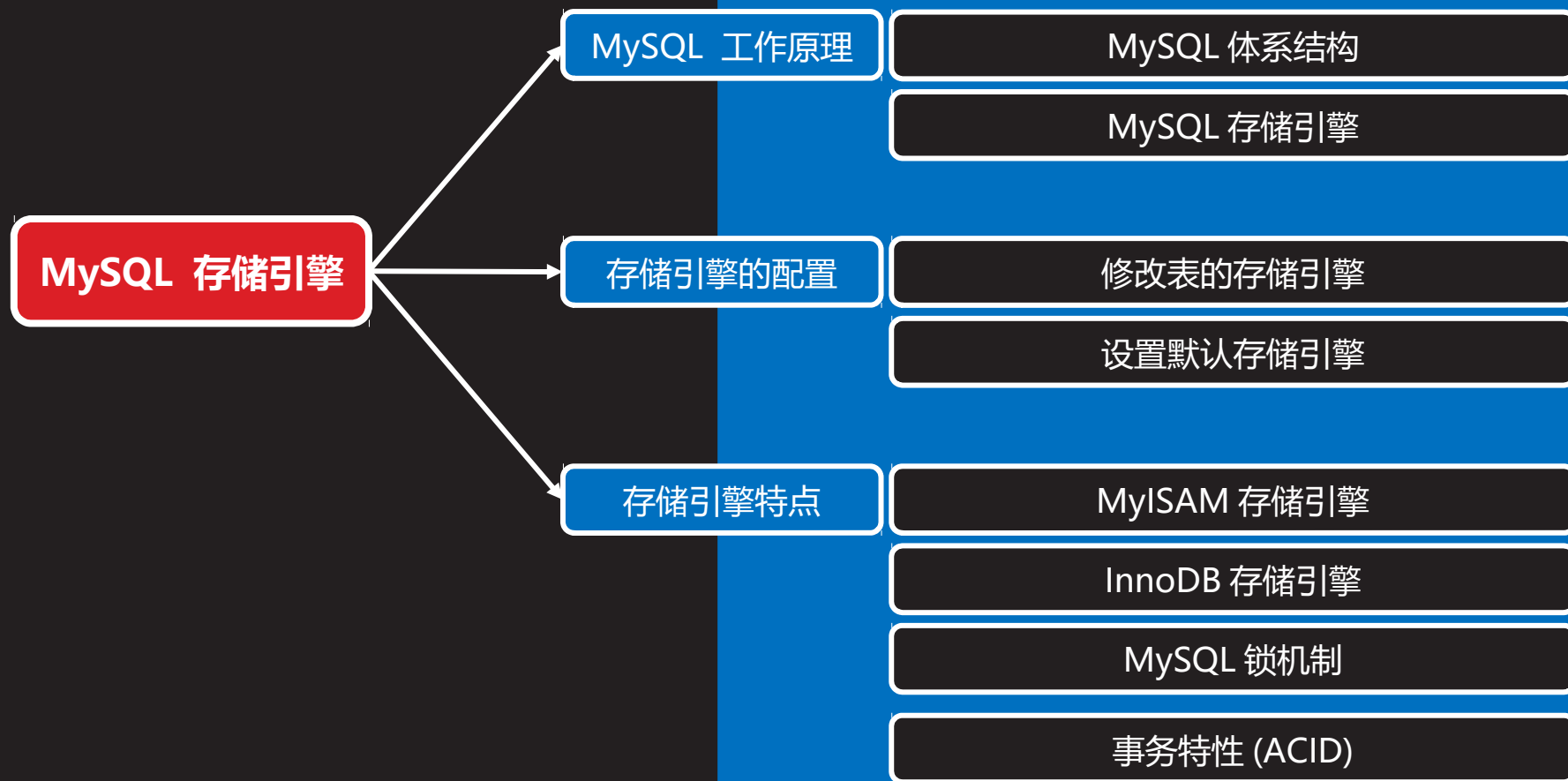
**DAY03**

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	MySQL 存储引擎
	10:30 ~ 11:20	
	11:30 ~ 12:00	数据导入导出
下午	14:00 ~ 14:50	管理表记录
	15:00 ~ 15:50	匹配条件
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



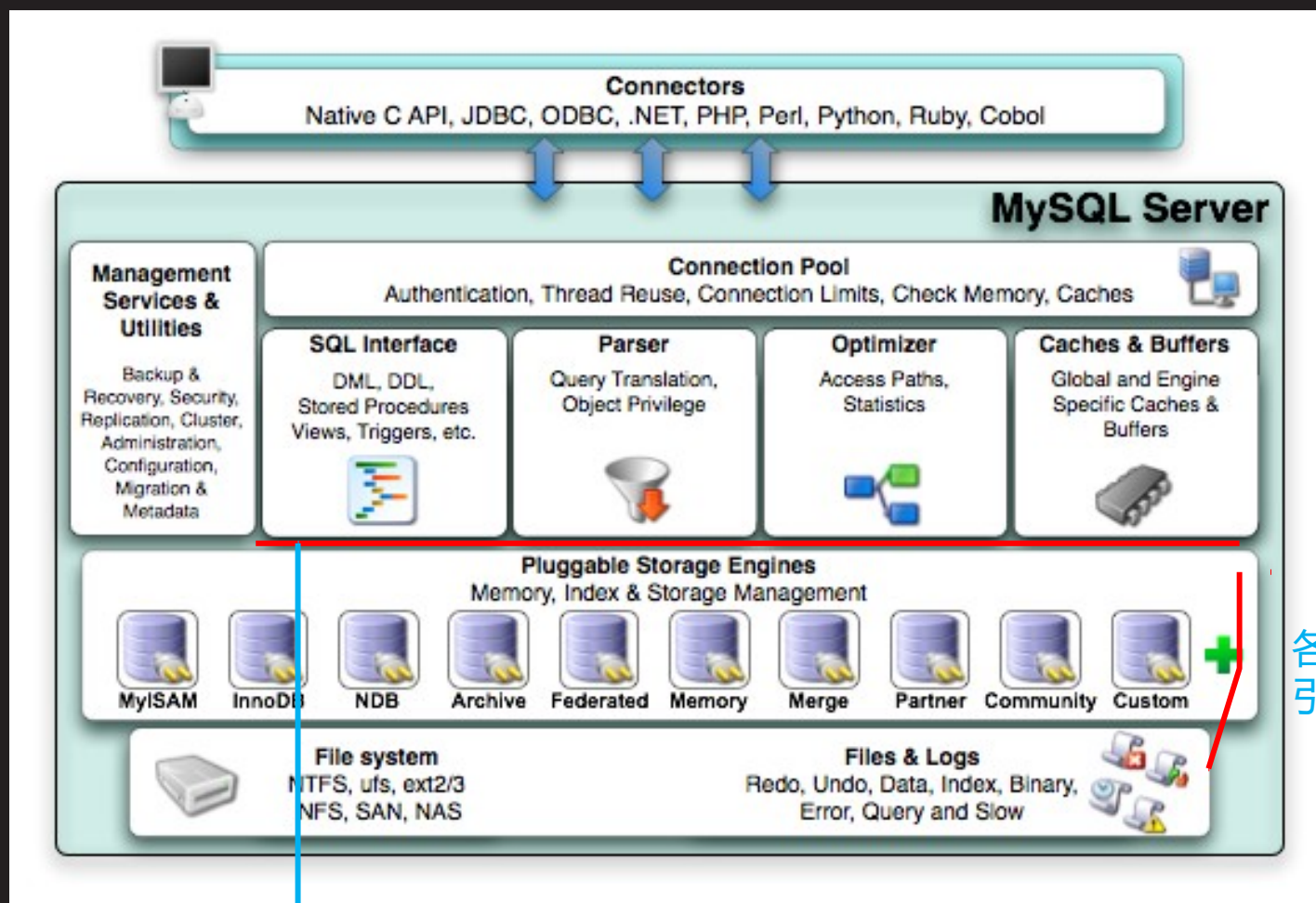
# MySQL 存储引擎



# MySQL 工作原理



# MySQL 体系结构



各种存储  
引擎组件

SQL 接口 / 解析器 / 优化器 / 缓存

# MySQL 存储引擎

- 作为可插拔式的组件提供
  - MySQL 服务软件自带的功能程序，处理表的处理器
  - 不同的存储引擎有不同的功能和数据存储方式
- 默认的存储引擎
  - MySQL 5.0/5.1 ---> MyISAM
  - MySQL 5.5/5.6 ---> InnoDB



# MySQL 存储引擎 (续 1)

- 列出可用的存储引擎类型
  - \_ SHOW ENGINES; 或 SHOW ENGINES\G

```
mysql> SHOW ENGINES;
```

Engine	Support	Comment
PERFORMANCE_SCHEMA	YES	Performance Schema
CSV	YES	CSV storage engine
MRG_MYISAM	YES	Collection of identical MyISAM tables
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)
MyISAM	YES	MyISAM storage engine
MEMORY	YES	Hash based, stored in memory, useful for temporary tables
ARCHIVE	YES	Archive storage engine
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys
FEDERATED	NO	Federated MySQL storage engine

9 rows in set (0.00 sec)



# 存储引擎的配置

---



# 修改表的存储引擎

- 建表时手动指定
  - 未指定时，使用默认存储引擎
  - SHOW CREATE TABLE xxx\G ; 可确认

```
mysql> USE test
Database changed
mysql> CREATE TABLE intab(
    -> id int(4)
    -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW CREATE TABLE intab\G
***** 1. row *****
      Table: intab
Create Table: CREATE TABLE `intab` (
  `id` int(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```



# 设置默认存储引擎

- 修改 /etc/my.cnf 配置文件
  - \_ default-storage-engine=xxxx

```
[root@dbsvr1 ~]# vim /etc/my.cnf
```

```
[mysqld]
```

```
....
```

```
default-storage-engine=InnoDB
```

```
[root@dbsvr1 ~]# service mysql restart
```

```
Shutting down MySQL....
```

[ 确定 ]

```
Starting MySQL.....
```

[ 确定 ]



# 存储引擎特点

---

# Myisam 存储引擎

- 主要特点
  - 支持表级锁
  - 不支持事务、事务回滚、外键
- 相关的表文件
  - 表名 .frm 、
  - 表名 .MYI
  - 表名 .MYD



# InnoDB 存储引擎

- 主要特点
  - 支持行级锁定
  - 支持事务、事务回滚、支持外键
- 相关的表文件
  - xxx.frm 、 xxx.ibd
  - ibdata1
  - ib\_logfile0
  - ib\_logfile1



# MySQL 锁机制

- 锁粒度
  - 表级锁：一次直接对整张表进行加锁。
  - 行级锁：只锁定某一行。
  - 页级锁：对整个页面（MySQL 管理数据的基本存储单位）进行加锁。
- 锁类型
  - 读锁（共享锁）：支持并发读。
  - 写锁（互斥锁、排它锁）：是独占锁，上锁期间其他线程不能读表或写表。



# MySQL 锁机制（续 1）

- 查看当前的锁状态
  - 检查 Table\_lock 开头的变量，% 作通配符

```
mysql> SHOW STATUS LIKE 'Table_lock%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Table_locks_immediate | 70 |
| Table_locks_waited | 0 |
+-----+-----+
2 rows in set (0.00 sec)
```



# 事务特性 (ACID)

- Atomic : 原子性
  - 事务的整个操作是一个整体, 不可分割, 要么全部成功, 要么全部失败。
- Consistency : 一致性
  - 事务操作的前后, 表中的记录没有变化。
- Isolation : 隔离性
  - 事务操作是相互隔离不受影响的。
- Durability : 持久性
  - 数据一旦提交, 不可改变, 永久改变表数据





# 事务特性 (ACID) (续 1)

## • 示例

mysql> show variables like "autocommit"; // 查看提交状态  
mysql> set autocommit=off; // 关闭自动提交  
mysql> rollback ; // 数据回滚  
mysql> commit; // 提交数据

```
mysql>
mysql> select * from t1;
+-----+
| name |
+-----+
| bob  |
+-----+
1 row in set (0.00 sec)

mysql> delete from t1;
Query OK, 1 row affected (0.05 sec)

mysql> rollback;
Query OK, 0 rows affected (0.06 sec)

mysql> select * from t1;
+-----+
| name |
+-----+
| bob  |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select * from t1;
+-----+
| name |
+-----+
| bob  |
+-----+
1 row in set (0.00 sec)

mysql> delete from t1;
Query OK, 1 row affected (0.02 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t1;
Empty set (0.00 sec)
```



# 案例 1：MySQL 存储引擎的配置

1. 可用的存储引擎类型
2. 查看默认存储类型
3. 更改表的存储引擎



# 数据导入导出

---

数据导入导出

数据导入导出

设置搜索路径

数据导入

数据导出

# 数据导入导出

---

# 设置搜索路径

- 查看默认使用目录及目录是否存在

```
mysql> show variables like "secure_file_priv";
```

Variable_name	Value
secure_file_priv	/var/lib/mysql-files/

```
[root@localhost ~]# ls -ld /var/lib/mysql-files/
drwxr-x---. 2 mysql mysql 31 4月 19 14:15 /var/lib/mysql-files/
```



# 设置搜索路径（续 1）

- 修改目录及查看修改结果

```
[root@localhost ~]# mkdir /myload ; chown mysql /myload
[root@localhost ~]# vim /etc/my.cnf
[mysqld]
secure_file_priv="/myload"
[root@localhost ~]# systemctl restart mysqld
```

```
mysql> show variables like "secure_file_priv";
```

Variable_name	Value
secure_file_priv	/myload/



# 数据导入

- 基本用法

```
– LOAD DATA INFILE “目录名 / 文件名”  
INTO TABLE 表名  
FIELDS TERMINATED BY “分隔符”  
LINES TERMINATED BY “\n” ;
```

- 注意事项

- 字段分隔符要与文件内的一致
- 指定导入文件的绝对路径
- 导入数据的表字段类型要与文件字段匹配
- 禁用 SELinux



# 数据导入 (续 1)

- 案例需求
  - 将本地用户的记录导入 userdb 库的 user 表里
  - 为每条用户记录添加记录编号

```
mysql> create database userdb;
mysql> create table userdb.user(
    -> name char(50),
    -> password char(1),
    -> uid int(2),
    -> gid int(2),
    -> comment varchar(100),
    -> homedir char(60),
    -> shell char(50),
    -> index(name)
    -> );
```

表结构参考 /etc/passwd 文件





## 数据导入 (续 2)

```
mysql> load data infile "/myload/user.txt" into table userdb.user fields
terminated by ":" lines terminated by "\n";
```

```
mysql> alter table userdb.user add id int(2) zerofill primary key
auto_increment first;
```

```
mysql> select id ,name,uid from userdb.user limit 3;
```

id	name	uid
01	root	0
02	bin	1
03	daemon	2

```
3 rows in set (0.00 sec)
```



# 数据导出

- 基本用法
  - \_ SELECT 查询 .. ..
  - INTO OUTFILE “ 目录名 / 文件名”
  - FIELDS TERMINATED BY “ 分隔符”
  - LINES TERMINATED BY “\n” ;
- 注意事项
  - \_ 导出的内容由 SQL 查询语句决定
  - \_ 禁用 SELinux



# 数据导出 (续 1)

- 案例需求
  - 导出 userdb 库 user 表中 uid 小于 100 的用户记录
  - 导出 mysql 库 user 表的前 10 条记录, 只需包括 user、host 两个字段的信息

```
mysql> select * from userdb.user where uid<100 into outfile "/myload/user2.txt";
Query OK, 18 rows affected (0.00 sec)
```

```
mysql> select user,host from mysql.user into outfile "/myload/user3.txt";
Query OK, 3 rows affected (0.00 sec)
```



## 案例 2：数据导入 / 导出

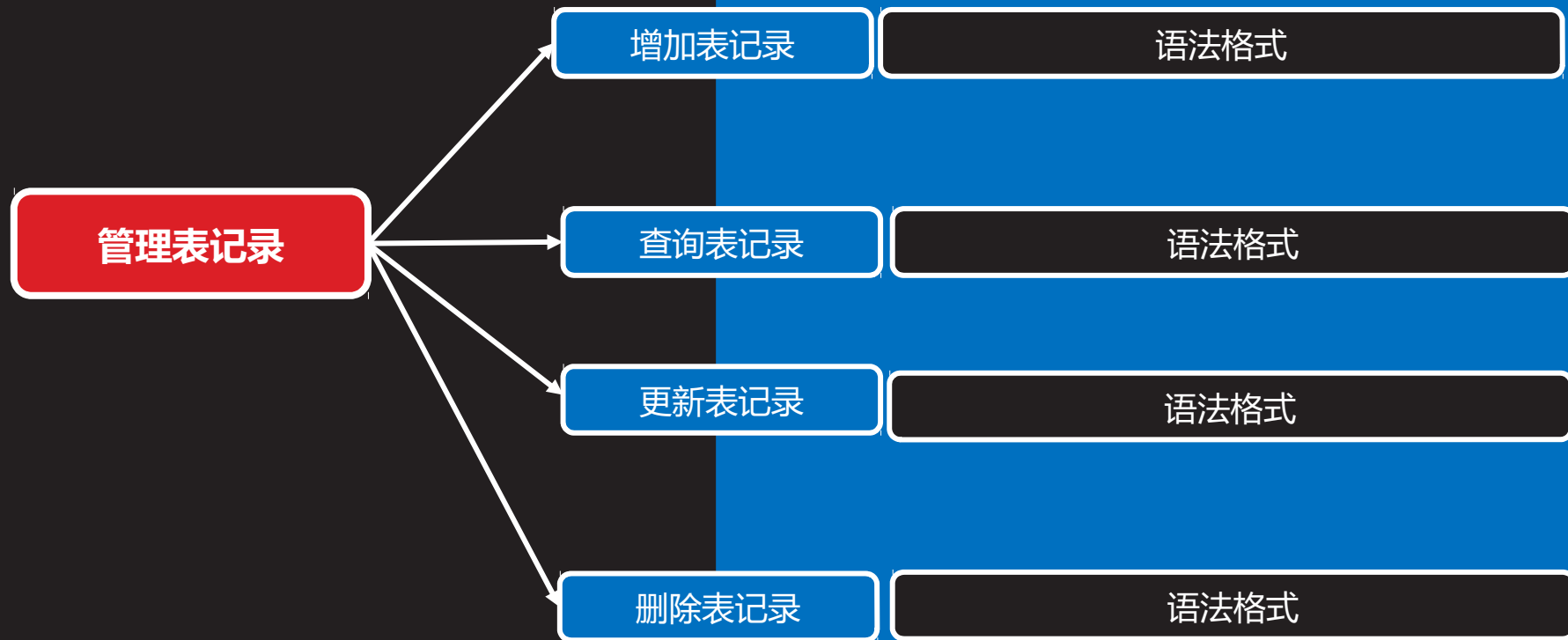
使用 SQL 语句完成下列导出、导入操作：

- 1) 将 /etc/passwd 文件导入 userdb 库 user 表并给每条记录加编号
- 2) 将 userdb 库 user 表中 uid 小于 100 的前 10 条记录导出，存为 /myload/user2.txt 文件



# 管理表记录

---



# 增加表记录

---

# 语法格式

- 格式 1：给所有字段赋值

\_ INSERT INTO 表名

VALUES

( 字段 1 值, ... , 字段 N 值 ),

( 字段 1 值, ... , 字段 N 值 ),

( 字段 1 值, ... , 字段 N 值 ),

... ;

第 1 条表记录

第 2 条表记录

第 3 条表记录



# 语法格式（续 1）

- 格式 2，给指定字段赋值

– INSERT INTO 表名 ( 字段 1,... .., 字段 N)

VALUES

( 字段 1 值, 字段 2 值, 字段 N 值 ),

( 字段 1 值, 字段 2 值, 字段 N 值 ),

( 字段 1 值, 字段 2 值, 字段 N 值 ),

... ;

第 1 条表记录

第 2 条表记录

第 3 条表记录





## 语法格式（续 2）

- 注意事项
  - 字段值要与字段类型相匹配
  - 对于字符类型的字段，要用双或单引号括起来
  - 依次给所有字段赋值时，字段名可以省略
  - 只给一部分字段赋值时，必须明确写出对应的字段名称



# 查询表记录

---

# 语法格式

- 格式 1
  - SELECT 字段 1, ... , 字段 N FROM 表名 ;
- 格式 2
  - SELECT 字段 1, ... , 字段 N FROM 表名  
WHERE 条件表达式 ;
- 注意事项
  - 使用 \* 可匹配所有字段
  - 指定表名时, 可采用 库名 . 表名 的形式



# 更新表记录

---

# 语法格式

- 格式 1，更新表内的所有记录  
    – UPDATE 表名  
      SET  
      字段 1= 字段 1 值，  
      字段 2= 字段 2 值，  
      字段 N= 字段 N 值；



## 语法格式（续 1）

- 格式 2，只更新符合条件的部分记录

– UPDATE 表名

SET

字段 1= 字段 1 值，

字段 2= 字段 2 值，

字段 N= 字段 N 值；

WHERE 条件表达式；



## 语法格式（续 2）

- 注意事项
  - 字段值要与字段类型相匹配
  - 对于字符类型的字段，要用双或单引号括起来
  - 若不使用 WHERE 限定条件，会更新所有记录
  - 限定条件时，只更新匹配条件的记录



# 删除表记录

---



# 语法格式

- 格式 1, 仅删除符合条件的记录  
\_ DELETE FROM 表名 WHERE 条件表达式;
- 格式 2, 删除所有的表记录  
\_ DELETE FROM 表名;



## 案例 3：操作表记录

- 练习表记录的操作
  - 表记录的插入
  - 表记录的更新
  - 表记录的查询
  - 表记录的删除



# 匹配条件

## 匹配条件

### 基本查询条件

数值比较

字符比较 / 匹配空 / 非空

逻辑匹配

范围匹配

### 高级查询条件

模糊查询

正则表达式

聚集函数

四则运算

去重查询

### 操作查询结果

查询结果排序

查询结果分组

查询结果过滤

限制显示行数

# 基本查询条件

---

# 数值比较

- 字段类型必须数据数值类型

类 型	用 途
=	等于
> 、 >=	大于、大于或等于
< 、 <=	小于、小于或等于
!=	不等于



# 字符比较 / 匹配空 / 非空

- 字段类型必须

类 型	用 途
=	相等
!=	不相等
IS NULL	匹配空
IS NOT NULL	非空



# 逻辑比较

- 多个判断条件时使用

类 型	用 途
OR	逻辑或
AND	逻辑与
!	逻辑非
()	提高优先级



# 范围内匹配 / 去重显示

- 匹配范围内的任意一个值即可

类 型	用 途
In ( 值列表 )	在...里...
Not in ( 值列表 )	不在...里...
Between 数字 1 and 数字 2	在...之间...
DISTINCT 字段名	去重显示





# 高级查询条件

---

# 模糊匹配

- 基本用法
  - \_ WHERE 字段名 LIKE '通配字符串'
  - \_ 通配符 \_ 匹配单个字符
  - \_ % 匹配 0~N 个字符
- 示例

列出 name 值 "J" 或 "Y" 结尾的记录

```
mysql> SELECT * FROM stu_info
      -> WHERE name LIKE 'J%' OR name LIKE '%y';
```

name	gender	age
Jim	girl	24
Lily	girl	20
Jerry	boy	27



# 正则匹配

- 基本用法

- WHERE 字段名 REGEXP '正则表达式'

- ^ \$ . [ ] \*

- 示例

找出 name 为 "J" 或 "Y" 开头的记录

```
mysql> SELECT * FROM stu_info WHERE name REGEXP '^J|Y$';
```

name	gender	age
Jim	girl	24
Lily	girl	20
Jerry	boy	27



# 四则运算

- 运算操作
  - 字段必须是数值类型

类 型	用 途
+	加法
-	减法
*	乘法
/	除法
%	取余数（求模）



# 聚集函数

- MySQL 内置数据统计函数
  - avg( 字段名 ) : 求平均值
  - sum( 字段名 ) : 求和
  - min( 字段名 ) : 统计最小值
  - max( 字段名 ) : 统计最大值
  - count( 字段名 ) : 统计个数



# 操作查询结果

---

# 查询结果排序

- 基本用法

- SQL 查询

ORDER BY 字段名

[ asc | desc ]



通常是数值类型字段

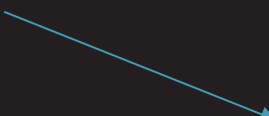


# 查询结果分组

- 基本用法

- SQL 查询

group by 字段名



通常是字符类型字段





# 查询结果过滤

- 基本用法
  - SQL 查询 HAVING 条件表达式;
  - SQL 查询 where 条件 HAVING 条件表达式;
  - SQL 查询 group by 字段名 HAVING 条件表达式;



# 限制查询结果显示行数

- 基本用法
  - SQL 查询 LIMIT N; 显示查询结果前 N 条记录
  - SQL 查询 LIMIT N,M ; 显示指定范围内的查询记录
  - SQL 查询 where 条件查询 LIMIT N ; 显示查询结果前 N 条记录
  - SQL 查询 where 条件查询 LIMIT N , M ; 显示指定范围内的查询记录

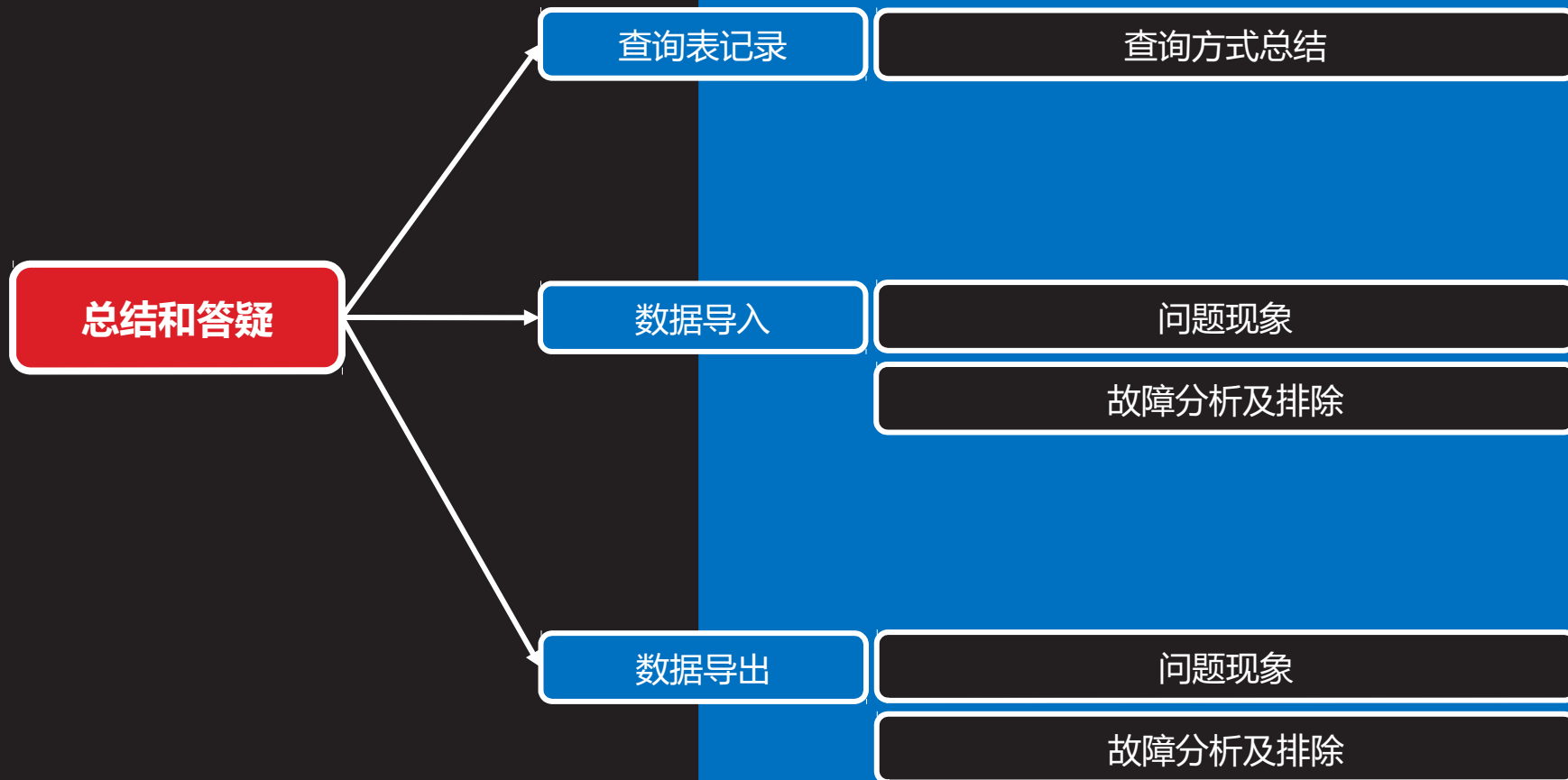


## 案例 4：查询及匹配条件

- 练习常见的 SQL 查询及条件设置
  - 创建 stu\_info 表，并插入数据
  - 练习常见 SQL 查询及条件设置



# 总结和答疑



# 查询表记录

---

# 查询方式总结

查询方式	关键字
查询条件	where
分组 / 排序 / 限制条 目数 / 查询结果过滤	order by/group by/limit/having
单表查询	select ... from ... where ...
嵌套查询	select ... from ... Where ... (select ... )
多表查询	select ... from 表 1, 表 n where ...
左连接查询	left ... Join ... on
右连接查询	right ... join ... on



# 数据导入



# 问题现象

- 数据导入失败
  - 报错 1 : Errcode: 13 - Permission denied: .. ..
  - 报错 2 : Data too long for column .....

```
mysql> load data infile "/tmp/a.txt" into table user fields
terminated by ":" lines terminated by "\n";
ERROR 29 (HY000): File '/tmp/a.txt' not found (Errcode: 13 -
Permission denied)
```

```
mysql> load data infile "/tmp/a.txt" into table user fields
terminated by ":" lines terminated by "\n";
ERROR 1406 (22001): Data too long for column 'comment' at row
1
```





# 故障分析及排除

- 问题 1 :
  - SELinux 策略阻止访问文件
  - 可执行 `setenforce 0` 禁用 SELinux
- 问题 2 :
  - 导入数据与字段类型不匹配, 需要修改字段类型:  
`alter table 表 modify 字段名 类型 ...`



# 数据导出



# 问题现象

- 导出数据保存到自定义目录失败
  - 报错: (Errcode: 13 - Permission denied) .. ..

```
mysql> select user,host from mysql.user into outfile  
"/datadir/b.txt";  
ERROR 1 (HY000): Can't create/write to file '/datadir/b.txt' (Errcode:  
13 - Permission denied)
```



# 故障分析及排除

- 原因分析
  - 对目录没有 w 权限
- 解决办法
  - 让 mysql 用户对目录有 w 权限

```
[root@dbsvr1 ~]# chown mysql /datadir/  
[root@dbsvr1 ~]# ls -ld /datadir  
drwxr-xr-x. 2 mysql root 4096 5 月 20 23:09 /datadir
```

