人工智能与大数据学院教师1、人工智能与大数据学院教师2

岗位试讲内容

注意事项：

1．每位考生试讲时间为10分钟；

2．试讲内容：统一指定1个内容并根据高职学生的特点进行试讲；

3．采用板书教学，可自带教具，不能使用多媒体辅助教学；

4．考生报到时需提交教材打印件和授课教案各5份。

一、教学内容：

教材：数据库原理与应用技术；航空工业出版社；主编：张云、张杏芝、杨永霞

内容：5.1.1 单表查询的基本语法格式 + 5.1.2 无条件查询和条件查询

二、教材封面：



三、试讲章节中的案例使用到的数据库信息（若自备案例则无需关注

此内容）

（1）db_student_course 数据库包含了 3 张数据表，分别是 t_student、t_course、t_sc。

（2）t_student 数据表信息如下：

字　段	数据类型	完整性约束
student_id（学号）	VARCHAR(20)	主键
student_name（姓名）	VARCHAR(50)	非空
gender（性别）	CHAR(2)	默认值为男
birth_date（出生日期）	DATE	非空
phone（手机号码）	VARCHAR(20)	无约束

t_student 数据表内容如下：

student_id	student_name	gender	birth_date	phone
S20260101	刘某	男	2007-08-06	132****1234
S20260102	王某	女	2008-04-01	136****5691

（3）t_course 数据表信息如下：

字　段	数据类型	完整性约束
course_id（课程编号）	VARCHAR(20)	主键
course_name（课程名）	VARCHAR(100)	非空
credit（学分）	INT	默认值为 1

t_course 数据表内容如下：

course_id	course_name	credit
C001	数据库原理	3
C002	Python 编程	3

（4）t_sc 数据表信息如下：

字　段	数据类型	完整性约束
student_id（学号）	VARCHAR(20)	与 course_id 组合为主键，非空，外键
course_id（课程编号）	VARCHAR(20)	与 student_id 组合为主键，非空，外键
score（成绩）	INT	非空，值域为 0～100

t_sc 数据表内容如下：

student_id	course_id	score
S20260101	C001	85
S20260101	C002	90
S20260102	C001	78

四、试讲内容：



项目 5

数据查询

项目导读

数据表创建成功并插入数据后，便可以对其进行查询操作。数据查询是用户对数据表执行的最频繁的操作，旨在从数据表中筛选、提取并整合出满足特定业务需求的数据。

本项目主要讲解数据查询中的单表查询和多表查询，涵盖无条件查询、条件查询、聚合查询、分组查询等单表查询，以及内/外连接查询、子查询、合并查询等多表查询。

项目目标

知识目标

- 识记 SELECT 语句中各子句的语法和用途。
- 掌握单表查询相关知识。
- 掌握多表查询相关知识。

能力目标

- 能够使用单表查询对单张数据表进行条件查询、分组查询、排序查询等。
- 能够使用连接查询（内连接和外连接）对多张数据表进行数据关联查询。
- 能够使用子查询解决依赖中间结果集的复杂查询问题。

素养目标

- 培养数据查询与分析的能力，养成用数据说话的职业习惯。
- 领悟“数据即资产”的核心内涵，培养从海量数据中提炼信息，赋能业务决策的意识。

项目准备

全班学生以 3～5 人为一组进行分组，各组选出组长。组长组织组员扫码观看“认识查询语句”视频，讨论并回答以下问题。

问题 1：什么是查询语句？



问题 2：查询语句的用途有哪些？

认识查询语句

5.1 单表查询

单表查询是指从单张数据表中查询所需要的数据。

5.1.1 单表查询的基本语法格式

在数据库中查询数据的基本语句是 SELECT 语句。使用 SELECT 语句进行单表查询的基本语法格式如下。

```
-- DISTINCT 表示对查询结果去重，*表示所有字段
SELECT [DISTINCT] 字段 1[, 字段 2, ...] | *
FROM 表名                          -- 指定查询的表
[WHERE 条件]                       -- 可选，查询满足条件的记录
-- 可选，按指定字段（分组字段）对查询结果进行分组，并对分组后的结果进行条件筛选
[GROUP BY 分组字段 [HAVING 条件]]
-- 可选，按指定字段（排序字段）对查询结果进行排序（ASC 表示升序，DESC 表示降序，默认为 ASC）
[ORDER BY 排序字段 [ASC|DESC]]
[LIMIT [起始位置,] 条数];          -- 可选，限制查询结果返回的记录条数
```

其中，SELECT 和 FROM 是必不可少的，用于指明“查询什么数据”和“从哪张表中查询数据”；其他子句都是可选的，用于对查询结果进行进一步筛选、分组、排序和数量控制。

若在一条 SELECT 语句中使用了上述所有子句，则 SQL 语句的实际执行顺序并不是按照其基本的书写顺序执行，而是按照 FROM（确定数据来源表）→WHERE（对原始数据进行条件筛选）→GROUP BY（对筛选后的数据进行分组）→HAVING（对分组结果进行条件筛选）→SELECT（确定最终要查询的字段）→ORDER BY（对查询结果进行排序）→LIMIT（限制返回的数据条数）的顺序执行。

5.1.2 无条件查询和条件查询

根据 SELECT 语句中是否使用 WHERE 子句，可以将数据查询分为无条件查询和条件查询，下面分别介绍。

1. 无条件查询

无条件查询是指在 SELECT 语句中不使用 WHERE 子句进行数据筛选的查询。

在无条件查询中，查询结果不受任何条件限制，系统会从数据表中返回查询字段的全部数据。无条件查询的基本语法格式如下。

```
SELECT 字段1[, 字段2, ...] | *
FROM 表名;
```

无条件查询常用于数据表创建完成后，快速查看表中数据是否正确插入，或对表中数据进行整体浏览。

接下来，通过一个示例练习无条件查询的应用。

【例 5-1】 从 t_course 数据表中查询所有课程的课程名和学分。

```
SELECT course_name, credit
FROM t_course;
```

无条件查询结果如图 5-1 所示。

```
mysql> SELECT course_name, credit
    -> FROM t_course;
+-------------+--------+
| course_name | credit |
+-------------+--------+
| 数据库原理  |      3 |
| Python编程  |      3 |
+-------------+--------+
2 rows in set (0.00 sec)
```

图 5-1 无条件查询结果

提示

请确认 db_student_course 数据库及其中的 t_student、t_course、t_sc 表已创建，且数据表中包含数据。若未创建数据库及表，或未向表中插入数据，可分别参考本书项目 3 的例 3-1、项目 4 的例 4-1 和例 4-4 完成前置准备。

2. 条件查询

条件查询是指在 SELECT 语句中使用 WHERE 子句进行数据筛选的查询，该查询仅返回满足筛选条件的数据。条件查询的基本语法格式如下。

```
SELECT 字段 1[, 字段 2, ...] | *
FROM 表名
WHERE 条件;
```

该语句通过 WHERE 子句对字段值进行比较、范围限定或逻辑判断，实现对查询结果的精确筛选与控制。

根据 WHERE 子句中筛选条件的数量，条件查询可分为单条件查询和多条件查询。当 WHERE 子句中只包含一个条件时，称为单条件查询；当包含多个条件，并通过逻辑运算符（AND、OR 和 NOT）进行组合时，称为多条件查询。多条件查询的基本语法格式如下。

```
SELECT 字段 1[, 字段 2, ...] | *
FROM 表名
WHERE 条件 1 逻辑运算符 条件 2 [逻辑运算符 条件 3 ...];
```

其中，逻辑运算符 AND 表示“并且”，要求所有条件同时满足；OR 表示“或者”，只需满足其中任意条件即可；NOT 表示“非”，用于对条件结果取反。

接下来，通过一个示例练习单条件查询和多条件查询的应用。

【例 5-2】　首先从 t_student 数据表中查询性别为“男”的学生信息，然后从 t_course 数据表中查询学分为 3 且课程名不为空的课程编号。

```
-- 单条件查询
SELECT student_id, student_name, gender
FROM t_student
WHERE gender = '男';
-- 多条件查询
SELECT course_id
FROM t_course
WHERE credit = 3 AND course_name IS NOT NULL;
```

单条件查询和多条件查询结果分别如图 5-2 和图 5-3 所示。

```
mysql> SELECT student_id, student_name, gender
    -> FROM t_student
    -> WHERE gender = '男';
+------------+--------------+--------+
| student_id | student_name | gender |
+------------+--------------+--------+
| S20260101  | 刘某         | 男     |
+------------+--------------+--------+
1 row in set (0.00 sec)
```

图 5-2　单条件查询结果

```
mysql> SELECT course_id
    -> FROM t_course
    -> WHERE credit = 3 AND course_name IS NOT NULL;
+-----------+
| course_id |
+-----------+
| C001      |
| C002      |
+-----------+
2 rows in set (0.00 sec)
```

图 5-3　多条件查询结果

提示

IS NOT NULL 用于查询字段值不为空的记录，可使用 IS NULL 查询字段值为空的记录。

另外，根据 WHERE 子句中筛选条件的不同，可将条件查询分为比较运算符查询、范围查询、集合查询和模糊查询，下面分别介绍。

（1）比较运算符查询。

比较运算符查询是指在 WHERE 子句中限定字段值满足某一比较条件的查询，常通过比较运算符=（等于）、<>或！=（不等于）、>（大于）、<（小于）、>=（大于等于）和<=（小于等于）等实现。例如，查询学分值为 3 的课程信息或成绩满足某一数值条件的学生记录。

接下来，通过一个示例练习比较运算符查询的应用。

【例 5-3】 从 t_sc 数据表中查询成绩大于等于 60 的所有记录。

```
SELECT *
FROM t_sc
WHERE score >= 60;
```

（2）范围查询。

范围查询是指在 WHERE 子句中限定字段值必须位于某个指定区间的查询，常通过范围运算符 BETWEEN … AND …实现。例如，查询成绩在某一分数段或出生日期在某一时间范围的学生信息。

接下来，通过一个示例练习范围查询的应用。

【例 5-4】 从 t_sc 数据表中查询成绩在 80～90 的所有记录。

```
SELECT *
FROM t_sc
WHERE score BETWEEN 80 AND 90;
```

查询结果包括区间的开始值和结束值，即 BETWEEN 80 AND 90 会返回成绩等于 80、90，以及介于 80～90 的所有记录。BETWEEN … AND …可搭配 NOT 使用，如 NOT BETWEEN 80 AND 90 表示查询不在 80～90 的记录。

（3）集合查询。

集合查询是指在 WHERE 子句中限定字段值属于某一集合的查询，常通过集合运算符 IN 或 NOT IN 实现。例如，查询选修了指定课程的学生信息或课程编号属于某一集合的记录。

接下来，通过一个示例练习集合查询的应用。

【例 5-5】　从 t_course 数据表中查询课程编号为 C001 或 C002 的所有记录。

```
SELECT *
FROM t_course
WHERE course_id IN ('C001', 'C002');
```

集合中的多个值用英文逗号隔开。

（4）模糊查询。

模糊查询是指在 WHERE 子句中限定字段值包含指定内容的查询，常通过模糊匹配运算符 LIKE 和通配符（%和_）实现。其中，%表示任意长度的字符，_表示单个字符。例如，按姓名关键字查询学生信息或按课程名称中的部分文字查询课程信息。

接下来，通过一个示例练习模糊查询的应用。

【例 5-6】　从 t_student 数据表中查询姓名中包含“王”的所有学生记录。

```
SELECT *
FROM t_student
WHERE student_name LIKE '%王%';
```

提示

通配符可以放在字符前，也可以放在字符后，还可以同时放在字符前后。

5.1.3　聚合查询和分组查询

在数据查询操作中，通常需要对数据表中的数据进行统计。为实现这类数据统计功能，MySQL 提供了多种聚合函数。聚合函数是对一组数据进行计算，并返回单一值的函数。常用的聚合函数如表 5-1 所示。

表 5-1　聚合函数

聚合函数	说　明
COUNT()	统计并返回符合条件的记录条数
SUM()	计算并返回数值字段值的总和
AVG()	计算并返回数值字段值的平均值
MAX()	返回字段值中的最大值
MIN()	返回字段值中的最小值

提示

聚合函数在计算时会自动忽略 NULL 值。

在查询操作中，可借助聚合函数实现聚合查询。此外，将聚合函数与分组查询结合使用，还能进一步实现数据的分组统计。