第四章 多表查询

课前回顾

1. 求字符串的字符数

```
SELECT CHAR_LENGTH('ABC');
```

2. 将字符串"超用心"和"在线教育"拼接成新的字符串

```
SELECT CONCAT("超用心", "在线教育");
```

3. 求"2019-05-04"到现在一共有多少天

```
SELECT TIMESTAMPDIFF(DAY, '2019-05-04', NOW());
```

4. 如果字段score的值大于90，则展示为优秀，否则展示为良好

```
SELECT IF(score>90, '优秀', '良好');
```

章节内容

- 表间关系 **重点**
- 约束 **重点**
- 索引 **重点**
- 连接查询 **重点**

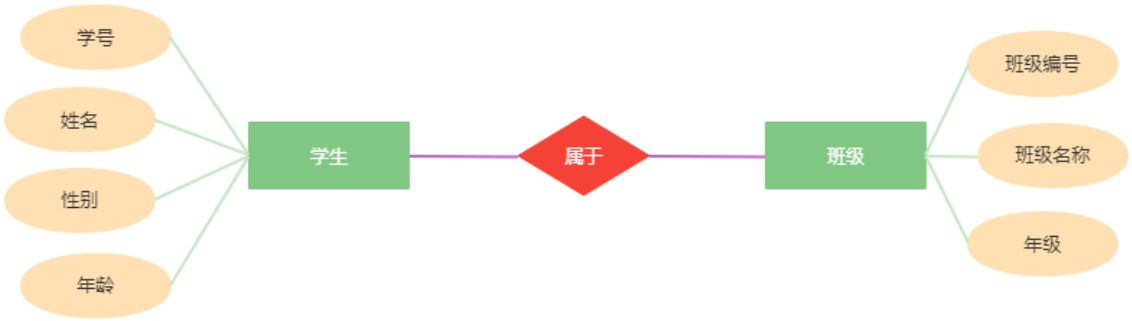
章节目标

- 掌握表间关系的建立
- 掌握常用约束
- 掌握索引的创建与使用
- 掌握多表关联查询

第一节 表与表之间的关系

1. 表与表之间的关系

数据表是用来描述实体信息的，比如可以使用数据表来描述学生信息，也可以用数据表来描述班级信息，这样就会存在学生表和班级表。而学生和班级显然存在着一种关系：



这种关系在数据库中体现就称之为表与表之间的关系。数据库通过主外键关联关系来体现表与表之间的关联关系。

2. 主外键关联关系

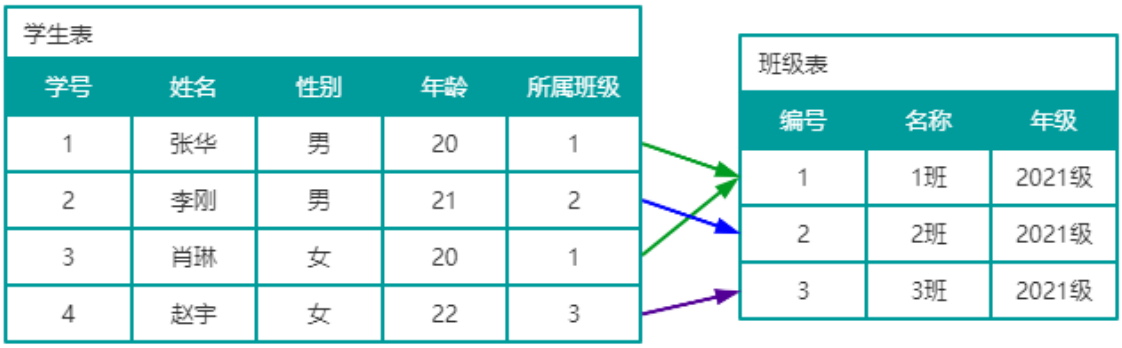
学生表

学号	姓名	性别	年龄
1	张华	男	20
2	李刚	男	21
3	肖琳	女	20
4	赵宇	女	22

班级表

编号	名称	年级
1	1班	2021级
2	2班	2021级
3	3班	2021级

如图所示，此时学生表和班级表并没有任何关系，然而实际上学生和班级是存在归属关系。可以在学生表中添加一个字段，表名该学生所属班级，该字段值使用的是班级表中的主键，在学生表中称之为外键。这样学生表中的所属班级（外键）与班级表中的编号（主键）就产生关联关系，这种关联关系称为主外键关联关系。



3. 主外键关联关系的定义

```
DROP TABLE IF EXISTS cls;
CREATE TABLE cls(
    number INT(11) AUTO_INCREMENT NOT NULL PRIMARY KEY COMMENT '班级编号，主键',
    name VARCHAR(20) NOT NULL COMMENT '班级名称',
    grade VARCHAR(20) NOT NULL COMMENT '年级'
)ENGINE=InnoDB CHARSET=UTF8 COMMENT='班级表';

DROP TABLE IF EXISTS student;
CREATE TABLE student(
    number BIGINT(20) AUTO_INCREMENT NOT NULL COMMENT '学号，主键',
    name VARCHAR(20) NOT NULL COMMENT '姓名',
    sex VARCHAR(2) DEFAULT '男' COMMENT '性别',
    age TINYINT(3) DEFAULT 0 COMMENT '年龄',
    cls_number INT(11) NOT NULL COMMENT '所属班级',
    PRIMARY KEY(number),
    -- 字段cls_number与cls表中的number字段相关联
    FOREIGN KEY(cls_number) REFERENCES cls(number)
)ENGINE=InnoDB CHARSET=UTF8 COMMENT='学生表';
```

4. 约束

4.1 主键约束

```
-- 添加主键约束：保证数据的唯一性
ALTER TABLE 表名 ADD PRIMARY KEY(字段名1,字段名2, ..., 字段名n)
-- 删除主键约束
ALTER TABLE 表名 DROP PRIMARY KEY;
```

4.2 外键约束

```
-- 添加外键约束
ALTER TABLE 表名1 ADD CONSTRAINT 外键名称 FOREIGN KEY(表名1的字段名) REFERENCES 表名2(表名2的字段名);
-- 删除外键约束
ALTER TABLE 表名 DROP FOREIGN KEY 外键名称;
```

4.3 唯一约束

```
-- 为字段添加唯一约束
ALTER TABLE 表名 ADD CONSTRAINT 约束名称 UNIQUE(字段名1, 字段名2, ..., 字段名n);
-- 删除字段的唯一约束
ALTER TABLE 表名 DROP KEY 约束名称;
```

4.4 非空约束

```
-- 为字段添加非空约束
ALTER TABLE 表名 MODIFY 字段名 列类型 NOT NULL;
-- 删除字段非空约束
ALTER TABLE 表名 MODIFY 字段名 列类型 NULL;
```

4.5 默认值约束

```
-- 为字段添加默认值
ALTER TABLE 表名 ALTER 字段名 SET DEFAULT 默认值;
-- 删除字段的默认值
ALTER TABLE 表名 ALTER 字段名 DROP DEFAULT;
```

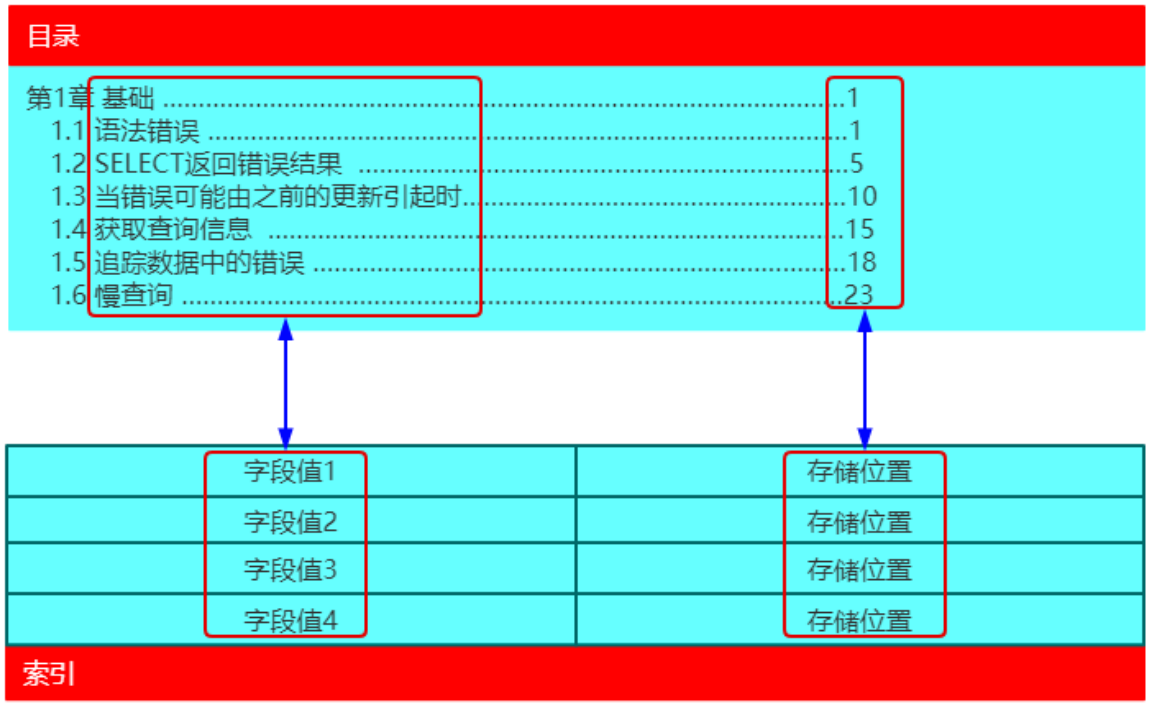
4.6 自增约束

```
-- 为字段添加自增约束
ALTER TABLE 表名 MODIFY 字段名 列类型 AUTO_INCREMENT;
-- 为字段删除自增约束
ALTER TABLE 表名 MODIFY 字段名 列类型;
```

第二节 索引

1 什么是索引

在关系数据库中，索引是一种单独的、物理的对数据库表中一列或多列的值进行排序的一种存储结构，它是表中一列或多列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单。



索引可以对比书籍的目录来理解。

2 索引的作用

- 保证数据的准确性
- 提高检索速度
- 提高系统性能

3 索引的类型

- **唯一索引（UNIQUE）**：不可以出现相同的值，可以有NULL值
- **普通索引（INDEX）**：允许出现相同的索引内容
- **主键索引（PRIMARY KEY）**：不允许出现相同的值
- **全文索引（FULLTEXT INDEX）**：可以针对值中的某个单词，但效率确实不敢恭维
- **组合索引**：实质上是将多个字段建到一个索引里，列值的组合必须唯一

4. 索引的创建、查看、删除

```
-- 创建索引
ALTER TABLE 表名 ADD INDEX 索引名称 (字段名1, 字段名2, ..., 字段名n);
-- 创建全文索引
ALTER TABLE 表名 ADD FULLTEXT 索引名称 (字段名1, 字段名2, ..., 字段名n);
-- 查看索引
SHOW INDEX FROM 表名;
-- 删除索引
ALTER TABLE 表名 DROP INDEX 索引名称;
```

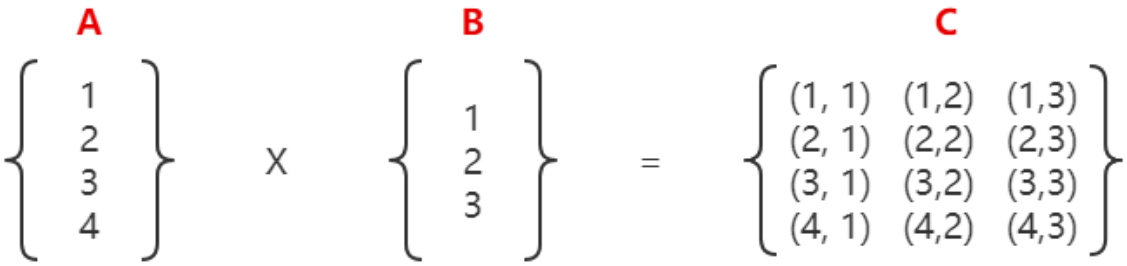
5. 使用索引的注意事项

- 虽然索引大大提高了查询速度，但也会降低更新表的速度，比如对表进行INSERT,UPDATE和DELETE操作，此时，数据库不仅要保存数据，还要保存一下索引文件
- 建立索引会占用磁盘空间的索引文件。如果索引创建过多（尤其是在字段多、数据量大的表上创建索引），就会导致索引文件过大，这样反而会降低数据库性能。因此，索引要建立在经常进行查询操作的字段上
- 不要在列上进行运算（包括函数运算），这会忽略索引的使用
- 不建议使用like操作，如果非使用不可，注意正确的使用方式。like '%查询内容%'不会使用索引，而like '查询内容%'可以使用索引
- 避免使用IS NULL、NOT IN、<>、!=、OR操作，这些操作都会忽略索引而进行全表扫描

第三节 多表查询

1. 笛卡尔积

笛卡尔积又称为笛卡尔乘积，由笛卡尔提出，表示两个集合相乘的结果。



笛卡尔积与多表查询有什么关系呢？每一张表可以看做是一个数据的集合，多表关联串时，这些表中的数据就会形成笛卡尔积。

2. 内连接

内连接相当于在笛卡尔积的基础上加上了连接条件。当没有连接条件时，内连接上升为笛卡尔积。

```
SELECT 字段名1, 字段名2, ..., 字段名n FROM 表1 [INNER] JOIN 表2 [ON 连接条件];

SELECT 字段名1, 字段名2, ..., 字段名n FROM 表1, 表2 [WHERE 关联条件 AND 查询条件];
```

示例：

```
SELECT COUNT(*) FROM stu INNER JOIN score ON stu.id=score.stu_id;

SELECT COUNT(*) FROM stu, score WHERE stu.id=score.stu_id;
```

3.外连接

外连接涉及到两张表：主表和从表，要查询的信息主要来自于哪张表，哪张表就是主表。

外连接查询的结果为主表中所有的记录。如果从表中有和它匹配的，则显示匹配的值，这部分相当于内连接查询出来的结果；如果从表中没有和它匹配的，则显示null。

外连接查询的结果 = 内连接的结果 + 主表中有的而内连接结果中没有的记录

外连接分为左外连接和右外连接两种。左外连接使用LEFT JOIN关键字，LEFT JOIN左边的是主表；右外连接使用RIGHT JOIN关键字，RIGHT JOIN右边的是主表。

3.1 左外连接

```
SELECT 字段名1, 字段名2, ..., 字段名n FROM 主表 LEFT JOIN 从表 [ON 连接条件];
```

示例：

```
SELECT * FROM stu a LEFT JOIN score b ON a.id=b.stu_id WHERE score IS NULL;
```

3.2 右外连接

```
SELECT 字段名1, 字段名2, ..., 字段名n FROM 从表 RIGHT JOIN 主表 [ON 连接条件];
```

示例：

```
SELECT * FROM stu a RIGHT JOIN score b ON a.id=b.stu_id;
```

第四节 子查询

1. 什么是子查询

子查询就是嵌套在其他查询中的查询。因此，子查询出现的位置只有3种情况：在SELECT... FROM 之间、在FROM...WHERE之间、在WHERE之后。

2. SELECT ... FROM之间

执行时机是在查询结果出来之后

查询stu表所有学生信息，并将性别按男、女、其他展示

```
SELECT
    id,
    `name`,
    (SELECT text FROM dict WHERE type='sex' AND value=sex) sex,
    birthday,
    class
FROM
    stu;
```

3. FROM ... WHERE 之间

执行时机是一开始就执行

查询年龄与Java成绩都与汤辰宇的年龄与Java成绩相同的学生信息

```
SELECT c.*, d.* FROM stu c
INNER JOIN
    score d ON c.id=d.stu_id
INNER JOIN
    (SELECT
        TIMESTAMPDIFF(YEAR, a.birthday,NOW()) age,
        b.score
    FROM stu a INNER JOIN score b ON a.id=b.stu_id
    WHERE a.name='汤辰宇'
    AND b.course='Java') e
ON TIMESTAMPDIFF(YEAR, c.birthday,NOW())=e.age AND d.score=e.score
WHERE d.course='Java';
```

4. WHERE 之后

查询Java成绩最高的学生信息

```
SELECT a.*, b.* FROM stu a INNER JOIN score b ON a.id=b.stu_id
WHERE b.score=(SELECT MAX(score) FROM score WHERE course='Java')
AND b.course='Java';
```