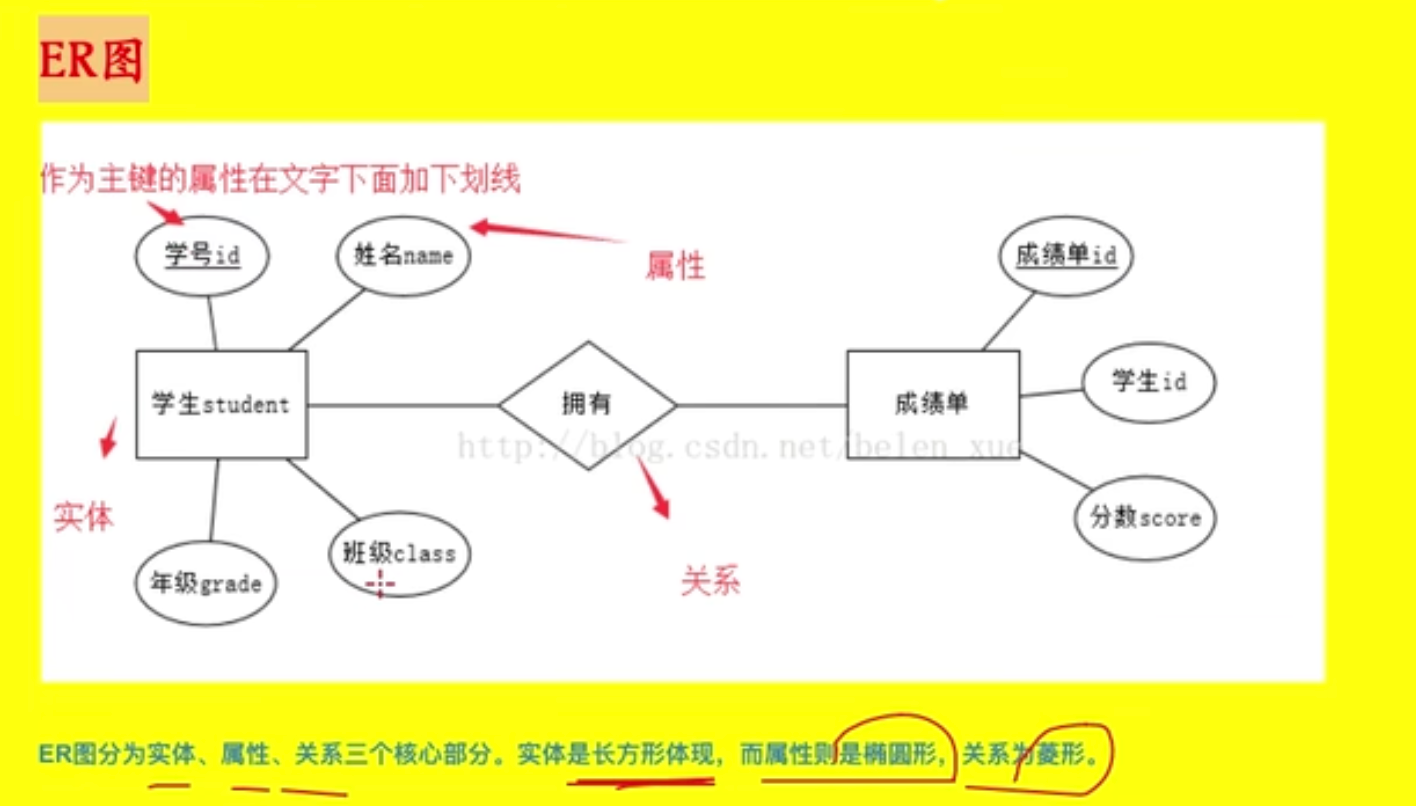
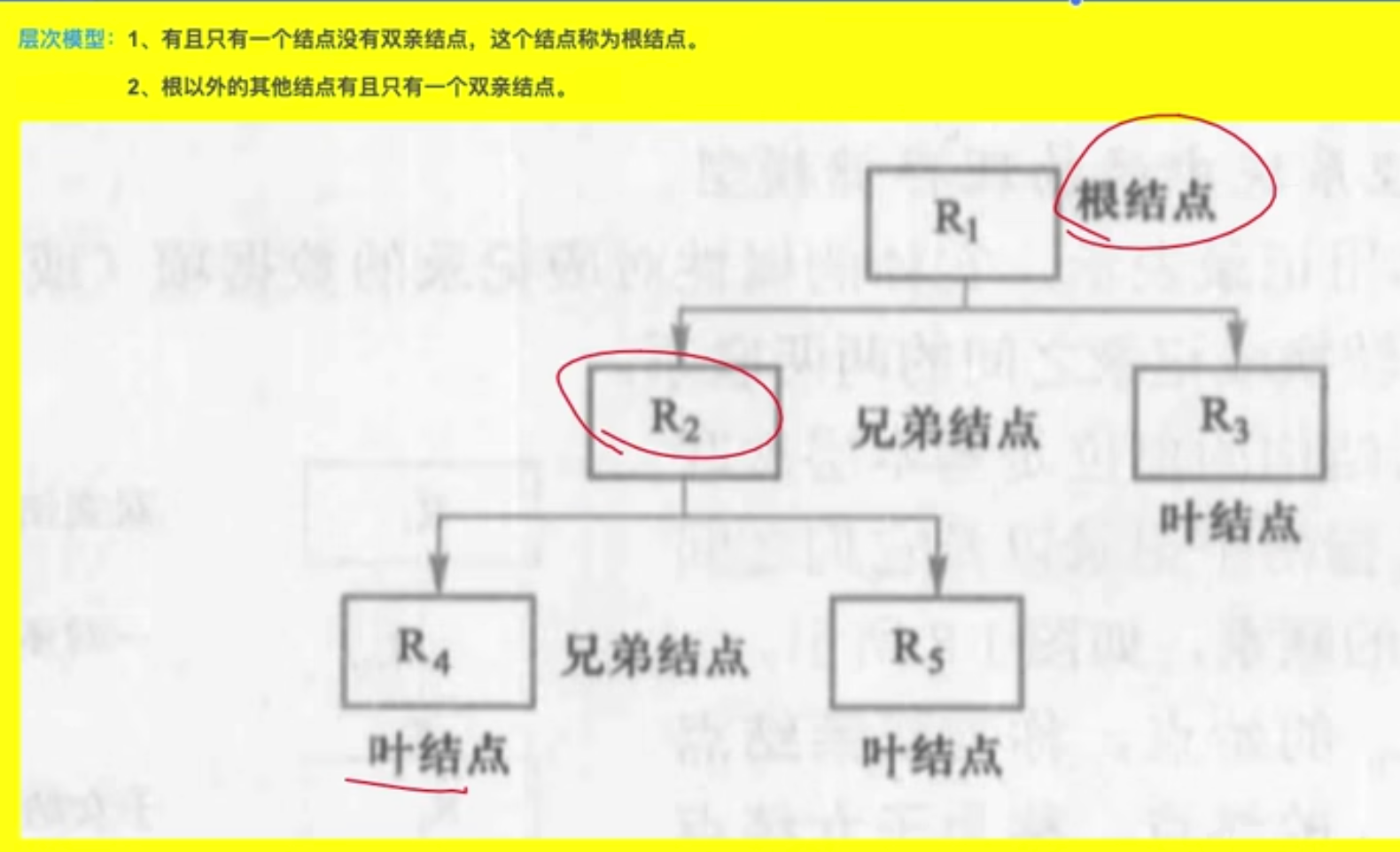
**数据库系统原理**

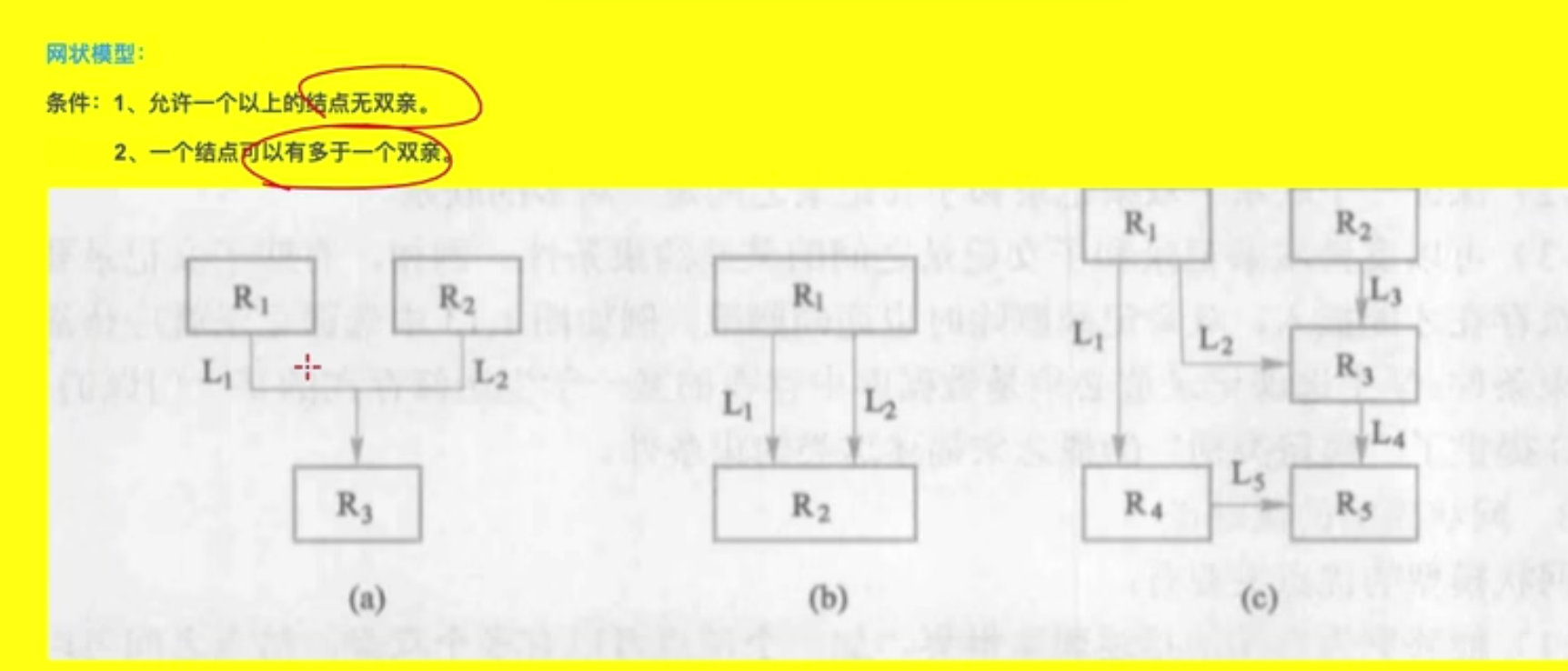
【【拯救者】数据库系统概论（原理）速成(期末+考研/复试+专升本均可) 新版，补充范式等内容】https://www.bilibili.com/video/BV1jf4y147jz?vd\_source=ff8a320929da3e2f46c73f4514b93500

**1**

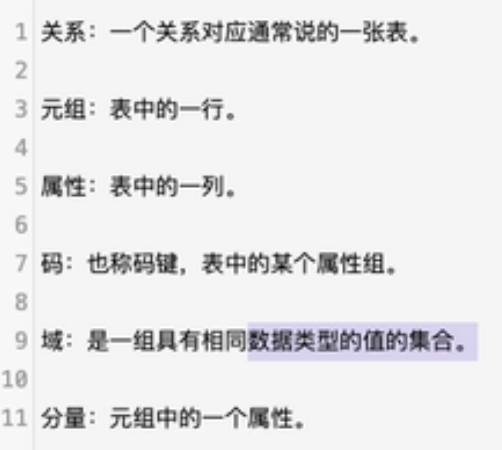












**SQL基础语言**

**[该类型的内容暂不支持下载]**

表：创建、清除数据、删除

数据：增删查改、去重过滤运算排序

**CREATE TABLE 创建**

|  |
| --- |
| C++ CREATE TABLE 表名 ( 列1名 int, 列2名 varchar[255], 列3名 varchar[255] ); |

**格式：**

|  |
| --- |
| C++ **CREATE TABLE** 表名 **(** 列名 类型， 列名 类型[最大长度]， …… 列名 类型[最大长度] **);** |

**类型**

**整型**

|  |
| --- |
| C++ integer(size) int(size) smallint(size) tinyint(size) |

**小数**

|  |
| --- |
| C++ decimal(size.d) numeric(size.d) |

**字符**

|  |
| --- |
| C++ char(size) varchar(size) |

**日期**

|  |
| --- |
| C++ date(yyyymmdd) |

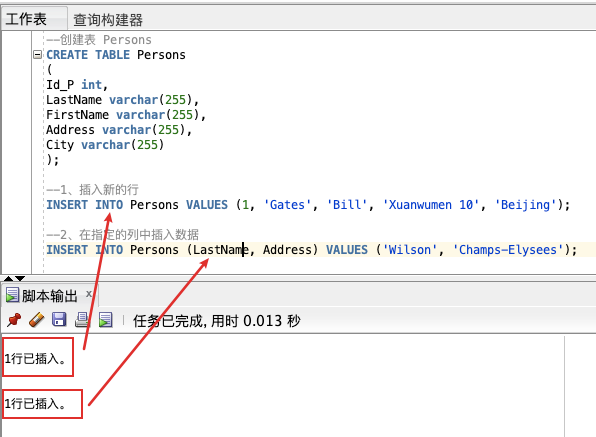
**TRUNCATE TABLE 清除表数据**

|  |
| --- |
| C++ TRUNCATE TABLE 表名; |

**DROP TABLE 删除表**

|  |
| --- |
| C++ DROP TABLE 表名; |

**INSERT 增加（插入）**



**全插入**

|  |
| --- |
| C++ INSERT INTO 表名 VALUES (值1，值2....); |

**部分插入**

|  |
| --- |
| C++ INSERT INTO 表名 (列1，列2....) VALUES (值1，值2....); |

**DELETE 删除数据**

**全部删除**

|  |
| --- |
| C++ DELETE FROM 表名; |

**单个删除**

|  |
| --- |
| C++ DELETE FROM 表名 WHERE 列名 = 值; |

**UPDATE 修改（更新数据）**

**多列修改**

|  |
| --- |
| C++ UPDATE 表名 SET 列名1 = 新值1，列名2 = 新值2 WHERE 列名 = 某值; |

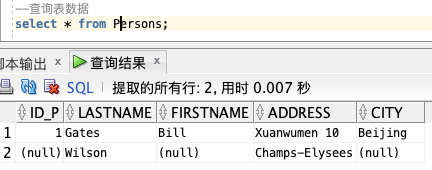
**单列修改**

|  |
| --- |
| C++ UPDATE 表名 SET 列名 = 新值 WHERE 列名 = 某值; |

**SELECT 查找**

**全部查询**

|  |
| --- |
| C++ SELECT \* FROM 表名； |



**指定查询**

|  |
| --- |
| C++ SELECT 列名 FROM 表名； |

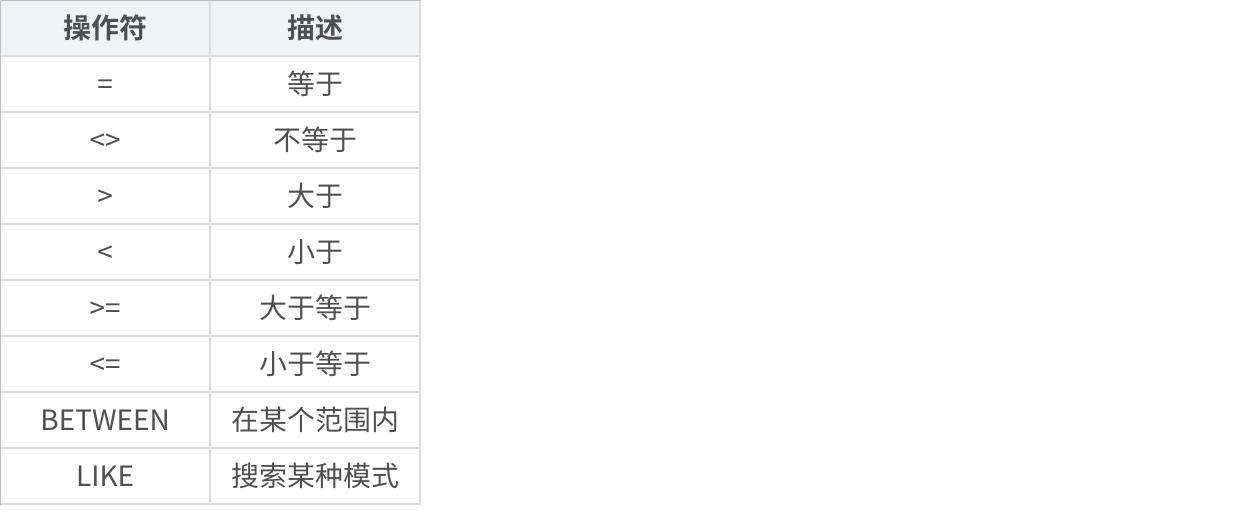


**DISTINCT 去重（使。。显著）**

|  |
| --- |
| C++ SELECT DISTINCT 列名 FROM 表名; |

**WHERE 条件过滤**

|  |
| --- |
| C++ SELECT 列名 FROM 表名 WHERE 列 运算符 值; |



**点击图片可查看完整电子表格**

**BETWEEN的使用：查询列的值在值1和值2范围内的记录：**

|  |
| --- |
| C++ SELECT \* FROM 表名 WHERE 列名 BETWEEN 值1 AND 值2; |

**LIKE的使用（针对字符类型的，是否包含对应子串）**

A：% 包含零个或多个字符的任意字符串：

1、like’Mc%’ 将搜索以字母 Mc 开头的所有字符串（如 McBadden）。

2、like’%inger’ 将搜索以字母 inger 结尾的所有字符串（如 Ringer、Stringer）。

3、like’%en%’ 将搜索在任何位置包含字母 en 的所有字符串（如 Bennet、Green、McBadden）。

B：\_（下划线） 任何单个字符：

like’\_heryl’ 将搜索以字母 heryl 结尾的所有六个字母的名称（如 Cheryl、Sheryl）。

C：[ ] 指定范围 ([a-f]) 或集合 ([abcdef]) 中的任何单个字符：

1，like’[CK]ars[eo]n’ 将搜索下列字符串：Carsen、Karsen、Carson 和 Karson（如 Carson）。

2、like’[M-Z]inger’ 将搜索以字符串 inger 结尾、以从 M 到 Z 的任何单个字母开头的所有名称（如 Ringer）。

D：[^] 不属于指定范围 ([a-f]) 或集合 ([abcdef]) 的任何单个字符：

like’M[^c]%’ 将搜索以字母 M 开头，并且第二个字母不是 c 的所有名称（如MacFeather）。

E：\* 它同于DOS命令中的通配符，代表多个字符：

c\*c代表cc,cBc,cbc,cabdfec等多个字符。

F：？同于DOS命令中的？通配符，代表单个字符 :

b?b代表brb,bFb等

G：# 大致同上，不同的是代只能代表单个数字。k#k代表k1k,k8k,k0k 。

下面我们来举例说明一下：

1. 例1，查询name字段中包含有“明”字的。

select \* from table1 where name like '%明%'

1. 例2，查询name字段中以“李”字开头。

select \* from table1 where name like '李\*'

1. 例3，查询name字段中含有数字的。

select \* from table1 where name like '%[0-9]%'

1. 例4，查询name字段中含有小写字母的。

select \* from table1 where name like '%[a-z]%'

1. 例5，查询name字段中不含有数字的。

select \* from table1 where name like '%[!0-9]%'

以上例子能列出什么值来显而易见。但在这里，我们着重要说明的是通配符“\*”与“%”的区别。

很多朋友会问，为什么我在以上查询时有个别的表示所有字符的时候用"%"而不用“\*”？先看看下面的例子能分别出现什么结果：

select \* from table1 where name like '\*明\*'

select \* from table1 where name like '%明%'

大家会看到，前一条语句列出来的是所有的记录，而后一条记录列出来的是name字段中含有“明”的记录，所以说，当我们作字符型字段包含一个子串的查询时最好采用“%”而不用“”,用“”的时候只在开头或者只在结尾时，而不能两端全由“\*”代替任意字符的情况下。

原文链接：https://blog.csdn.net/wsx1212123/article/details/106259320

**AND & OR 运算符**

**AND**

|  |
| --- |
| C++ SELECT \* FROM 表名 WHERE 列 运算符 值 AND 列 运算符 值; |

**OR**

|  |
| --- |
| C++ SELECT \* FROM 表名 WHERE 列 运算符 值 OR 列 运算符 值; |

**BETWEEN AND 和 AND 的区别**

**在 SQL 中，BETWEEN AND 和 AND 是两个不同的用法，它们用于不同的查询条件。**

1. BETWEEN AND 用于指定一个范围内的查询条件。例如：

|  |
| --- |
| C++ SELECT \* FROM table\_name WHERE column\_name BETWEEN value1 AND value2; |

这将查询 column\_name 列中在 value1 和 value2 之间的所有记录。

1. AND 是用于连接多个查询条件的逻辑运算符。例如：

|  |
| --- |
| C++ SELECT \* FROM table\_name WHERE condition1 AND condition2; |

这将查询同时满足 condition1 和 condition2 的所有记录。

**在性能方面，BETWEEN AND 通常比 AND 更高效，原因如下：**

* BETWEEN AND 在查询时可以利用索引进行优化，因为它指定了一个连续的范围。数据库可以使用索引来快速定位符合条件的记录。
* AND 对多个条件进行逻辑连接，可能需要进行逐个条件的过滤，对于大数据量或复杂查询条件的情况，性能可能会受到影响。

但是，这并不意味着在所有情况下 BETWEEN AND 都比 AND 更高效。查询的效率受到多种因素的影响，如索引的使用、数据的分布和查询的复杂度等。所以，在实际应用中，最好通过测试和评估来确定哪种查询条件更适合特定的情况。

**ORDER BY 排序**

默认排序为 ASC 升序，DESC 代表降序。

|  |
| --- |
| C++ SELECT \* FROM 表名 ORDER BY 列1,列2 DESC; |