数据定义：表约束与索引

准备工作:

- 创建stu_info数据库和grade表

```
CREATE DATABASE stu_grade;
USE stu_grade;
CREATE TABLE grade(
    id int,
    name varchar(20),
    grade float);
```

- 创建purchase数据库和product表

```
CREATE DATABASE purchase;
USE purchase;
CREATE TABLE Product (Product_ID CHAR(10) COMMENT '商品编号',
                      Product_Name VARCHAR(100) COMMENT '商品名称',
                      Product_Code VARCHAR(10) COMMENT '商品编码',
                      Product_Place CHAR(50) COMMENT '产地',
                      Product_Date DATE COMMENT '生产日期',
                      Price FLOAT COMMENT '价格',
                      Unit VARCHAR(20) COMMENT '单位',
                      Detail VARCHAR(20) COMMENT '规格',
                      SubSort_ID VARCHAR(10) COMMENT '子类别id',
                      Sort_ID VARCHAR(10) COMMENT '根类别id');
```

一、表约束

从对象级别来看，表的约束分为字段级别约束和表级别约束。常见的字段级别约束包括：

- 非空约束(`NULL/NOT NULL`)
- 唯一约束(`UNIQUE`)
- 主键约束(`PRIMARY KEY`)
- 默认值(`DEFAULT`)
- 外键约束(`REFERENCES`)

其中，主键约束应同时满足**非空约束**和**唯一性约束**。

1.主键 `PRIMARY KEY`

- 单字段主键

单字段主键即在1个字段上构建的主键约束。

```
字段 数据类型[(宽度)] PRIMARY KEY
-- 或者
PRIMARY KEY (字段)
```

示例1: 将数据表grade中id字段设置为主键。

```
USE stu_info;
ALTER TABLE grade MODIFY id int PRIMARY KEY;
DESC grade;
```

- 多字段主键

即在2个或多个字段上构建的主键约束。

```
PRIMARY KEY (字段1, 字段2, ....)
```

示例2: 新建数据表example，并创建id字段，属性为INT(10)，并设置为主键。

```
CREATE TABLE example (id int(10) PRIMARY KEY);
CREATE TABLE example2 (id int(10),
                       PRIMARY KEY (id));
DESC example;
```

示例3: 新建数据表course，创建stu_id、course_id和grade三个字段，其中stu_id和course_id设置为多字段主键。

```
CREATE TABLE course (stu_id int,
                     course_id int,
                     grade float,
                     primary key (stu_id, course_id));
DESC course;
```

2.非空 NOT NULL

设置非空约束的字段必须有值。

示例4: 将grade表的username字段改为非空

```
ALTER TABLE grade MODIFY username VARCHAR(20) NOT NULL;
DESC grade;
```

3.唯一性 UNIQ

设置唯一性约束的字段的值不重复，即任意两行的该字段值不相同。

> 创建唯一性索引。

示例5: 将数据表grade中username字段设置唯一性约束。

```
ALTER TABLE grade MODIFY username VARCHAR(20) UNIQUE;
DESC grade;
```

4.默认值 DEFAULT

即如果没有该字段未设定值，则取默认值。

示例6：将数据表grade中grade字段设置默认约束值为0。

```
ALTER TABLE grade MODIFY grade FLOAT DEFAULT 0;
DESC grade;
```

5.自增 AUTO_INCREMENT

> - 只有设置为主键或者唯一性约束的列才能继续设置 AUTO_INCREMENT 。
> - 在插入行时，若未在设定 AUTO_INCREMENT 的属性列上设置值，则该属性列的值相对上一行对应属性列值加上1。

示例7：在数据表grade中id字段，设置为字段值自动增加。

```
ALTER TABLE grade MODIFY id int AUTO_INCREMENT;
DESC grade;
```

课堂练习1

创建以下数据表的约束

字段	数据类型	约束	说明
Product_ID	char(10)	主键	商品编号
Product_Name	varchar(100)	唯一性	商品名称
Product_Code	varchar(10)	非空	商品编码
Product_Place	char(10)		商品产地
Place_Date	date		生产日期
Price	float	默认值为0	商品价格
Unit	varchar(20)		单位
Detail	varchar(20)		规格
SubSort_ID	varchar(10)	非空	子类别ID
Sort_ID	varchar(10)	非空	类别ID

6.外键约束 FOREIGN KEY...REFERENCES

建立外键约束语法:

- （1）通过 `alter table` 语法构建外键约束

```
ALTER TABLE 表名 ADD CONSTRAINT [外键名] FOREIGN KEY(外键字段名)
REFERENCES 外表表名(主键字段名);
```

> 注意：
>
> - 建立外键的表必须是 `InnoDB` 型的表，不能是临时表。因为 `MySQL` 中只有 `InnoDB` 型的表才支持外键；
> - 定义外键名时，不能加引号。如： `constraint 'FK_ID'` 或 `constraint " FK_ID "` 都是错误的；
> - 被参照表必须先于参照表定义，且被参照字段必须为被参照表的主键。

- （2）通过 `CREATE TABLE` 语法构建外键约束

```
CREATE TABLE [数据库名.]表名 (
    字段1 类型[(长度)] 其它列级约束,
    ...
    CONSTRAINT [约束名] FOREIGN KEY (参照字段) REFERENCES 被参照表(被参照表主键);
)
```

例如:

```
USE stu_info;

CREATE TABLE department(
    deptid INT(4) PRIMARY KEY,
    deptname VARCHAR(40) NOT NULL);

CREATE TABLE student(
    sid INT(4) PRIMARY KEY,
    sname VARCHAR(40),
    deptid INT(4) NOT NULL,
    CONSTRAINT fk_deptid FOREIGN KEY (deptid) REFERENCES department(deptid));
```

示例8: 为 `product` 表的 `sort_id` 添加外键约束，被参照表为 `sort`

```
USE purchase;

CREATE TABLE sort (
  sort_id char(2),
  sort_name varchar(50));

ALTER TABLE sort
MODIFY sort_id CHAR(2) PRIMARY KEY; -- 首先，被引用的主表字段应为主键，因此需为sort表的Sort_ID字段添加主键约束，否则建立外键时会提示1215错误

ALTER TABLE product
ADD CONSTRAINT fk_sortid FOREIGN KEY (sort_id) REFERENCES sort(sort_id); -- 然后，为表product的Sort_ID字段添加外键约束

SHOW CREATE TABLE product;  -- 使用show create table 来查看表的构建语句
```

> 建立外键的完整格式:

```
ALTER TABLE 表名 ADD CONSTRAINT [外键名] FOREIGN KEY(外键字段名) REFERENCES 外表表名(主键字段名)
[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT}]
[ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT}]

注意：(1) CASCADE：主表中删除或更新对应的记录时，同时自动地删除或更新从表中匹配的记录。
(2) SET NULL：主表中删除或更新被参照列记录时，同时将从表中的外键列设为空。
(3) NO ACTION：拒绝删除或者更新主表被参照列记录，从表也不进行任何操作。
(4) RESTRICT：拒绝删除或者更新主表被参照列记录。
```

删除外键约束:

```
ALTER TABLE 表名
DROP FOREIGN KEY 约束名称;
```

示例9：删除product表的sort_id上的外键约束

```
ALTER TABLE product
DROP FOREIGN KEY fk_sortid;

SHOW CREATE TABLE product;
```

课堂练习2

`subsort` 表

属性名	类型	约束
subsort_id	char(5)	primary key
subsort_name	varchar(50)	
sort_id	char(2)	

(1) 为 `product` 表添加 `Sort_ID` 的外键约束 `FK_sortid1`

(2) 使用语句为 `subsort` 表的 `Sort_ID` 添加引用自 `sort` 表外键名为 `FK_sortid2` 的外键约束。

(3) 为 `product` 表添加 `Subsot_id` 的引用自 `subsort` 表的外键约束 `FK_subsortid` 。

(4) 删除 `product` 表中的 `Subsort_ID` 的外键约束。

二、创建索引

索引是一种特殊的数据库结构，其作用相当于一本书的目录，以在查询时快速地定位到目标记录行。索引是提高数据查询效率的重要方式之一，提高性能的最常用的工具之一。用户创建的索引指向数据库中具体数据所在位置。当用户通过索引查询数据库中的数据时，不需要遍历所有数据库中的所有数据。

所有 `MySQL` 列类型都可以被索引，索引有两种存储类型： `B` 型树和 `Hash` 。其中 `B` 型树为 `INNODB` 和 `MYISAM` 存储引擎的默认索引存储类型。

- 索引的优点：
  - 通过创建唯一性索引，可保证数据库表中的每一行数据的唯一性
  - 可以大大加快数据的检索速度
  - 可以加速表与表之间的链接
  - 在使用分组和排序子句进行数据检索时，可显著减少查询中分组和排序的时间
- 索引的缺点：
  - 创建和维护索引需消耗一定系统性能，且随着数据量的增加而增加
  - 索引需要占据额外的物理空间

    - 在索引列进行数据插入、更新和删除时，对应的索引也要动态维护，从而降低了数据的维护速度
- `MySQL` 的索引包括**普通索引、唯一性索引、全文索引、单列索引、多列索引和空间索引**。
    - 普通索引 `index/key`
    - 唯一性索引 `unique index` ：限制该索引对应列的值唯一。主键是一种特殊唯一索引。
    - 全文索引 `fulltext` ：只能创建在 `char` ，`varchar` 和 `text` 类型的字段上。
    - 空间索引 `spatial` ：目前只有 `MYISAM` 存储引擎支持空间索引，而且索引的字段不能为空。

1. 创建表的时候创建索引。

示例10：在 `stu_info` 数据库中创建 `student` 表及相关索引。

```
USE stu_info;
CREATE TABLE student(stu_id int(10),
                     name varchar(20),
                     course varchar(50),
                     score float,
                     DESCription varchar(100),
                     INDEX(stu_id),                          #创建普通索引
                     UNIQUE INDEX unique_id(stu_id ASC),     #创建唯一性索引
                     FULLTEXT INDEX fulltext_name(name),     #创建全文索引
                     INDEX single_course(course(10)),        #创建单列索引
                     INDEX multi(stu_id, name(20))           #创建多列索引
                     );
```

2. 利用 `create index` 在已有表上创建索引。

将索引视为独立的数据库对象。

示例11：创建 `book1` ，然后利用 `create index` 创建索引。

```
CREATE TABLE book1(bookid int NOT NULL,
                   bookname varchar(255) NOT NULL,
                   authors varchar(255) NOT NULL,
                   info varchar(255) NOT NULL,
                   comment varchar(255) NOT NULL,
                   publicyear YEAR NOT NULL
                   );

CREATE INDEX index_id ON book1(bookid);                          #创建普通索引
CREATE UNIQUE INDEX uniqueidx ON book1(bookid);                  #创建唯一性索引
CREATE INDEX singleidx ON book1(comment);                        #创建单列索引
CREATE INDEX mulitidx ON book1(bookname(20), authors(20));       #创建多列索引
CREATE FULLTEXT INDEX fulltextidx ON book1(info);                #创建全文索引

DESC book1;
```

3. 利用 `ALTER TABLE` 在已有表上创建索引。

示例12：使用ALTER TABLE语句在已经存在表上创建索引。

```
-- 先将数据表book1复制为book2表。注意book2只复制了book1的字段及类型。
CREATE TABLE book2 SELECT * FROM book1;

-- 添加索引
ALTER TABLE book2 ADD INDEX index_id(bookid);                    #创建普通索引
ALTER TABLE book2 ADD UNIQUE INDEX uniqueidx(bookid);            #创建唯一性索引
ALTER TABLE book2 ADD INDEX singleidx(comment(50));              #创建单列索引
ALTER TABLE book2 ADD INDEX mulitidx(bookname(20),authors(20)); #创建多列索引
ALTER TABLE book2 ADD FULLTEXT INDEX fulltextidx(info);          #创建全文索引

DROP INDEX uniqueidx ON book2;
```

示例13：删除表book1中名称为fulltextidx的全文索引。

```
ALTER TABLE book1 drop index fulltextidx;
```

示例14：删除表book1中名称为singleidx的单列索引。

```
drop index singleidx on book1;
```

> 查看在某个表上创建的索引
> `show index from 表名;`

课堂练习3

创建数据表 `Product` 的相关索引。

(1) 创建 `Product_Code` 字段的唯一性索引

(2) 创建 `Detail` 字段的普通索引

(3) 创建 `Product_Place` 字段的全文索引