数据库期末考试重点复习指南

本文档针对数据库期末考试的六大核心考点进行整理：

1. **综合大题**：E-R图、范式 (Normalization)
2. **代码大题**：存储过程 (Stored Procedure)、存储函数 (Stored Function)、触发器 (Trigger)、游标 (Cursor)

第一部分：综合大题理论与技巧

1. E-R 图 (Entity-Relationship Diagram)

核心概念：

- **实体 (Entity)**：客观存在的事物（如：学生、课程）。用 **矩形** 表示。
- **属性 (Attribute)**：实体的特征（如：姓名、学号）。用 **椭圆** 表示。
- **关系 (Relationship)**：实体之间的联系（如：选课、授课）。用 **菱形** 表示。

考试重点 - 关系的类型 (基数)：

- **1:1 (一对一)**：例如，系主任与系。
- **1:N (一对多)**：例如，班级与学生（一个班级有多个学生，一个学生只属于一个班级）。
- **M:N (多对多)**：例如，学生与课程（一个学生选多门课，一门课由多个学生选）。**注意：多对多关系在转化为关系模式时，必须单独生成一张新表。**

做题技巧：

1. 先找出所有的“名词”作为实体。
2. 找出“动词”作为关系。
3. 确定关系的类型（1:1, 1:N, M:N）。
4. 不要忘记把属性连在对应的实体或关系上（M:N 的属性通常在关系上，如“成绩”）。

2. 范式 (Normalization)

目的：减少数据冗余，消除插入、删除和更新异常。

三大范式判定口诀：

- **第一范式 (1NF)**：**属性不可分**。确保每一列都是原子的，不能有“地址”列里面包含“省、市、区”。
- **第二范式 (2NF)**：在 1NF 基础上，**消除非主属性对主键的部分函数依赖**。
  - *通俗解释*：主要针对联合主键。如果主键是 (学号, 课程号)，那么“姓名”只依赖于“学号”，不依赖于“课程号”，这就是部分依赖，不符合 2NF。需要拆表。
- **第三范式 (3NF)**：在 2NF 基础上，**消除非主属性对主键的传递函数依赖**。
  - *通俗解释*：A -> B -> C。如果表中存在：学号 -> 学院名 -> 学院地址。学院地址是依赖于学院名的，而不是直接依赖于学号。需要拆分出“学院表”。

第二部分：四大代码题型详解 (由浅入深)

注：以下代码以 MySQL 语法为例，这是高校考试中最常见的标准。

1. 存储过程 (Stored Procedure)

概念：一组为了完成特定功能的 SQL 语句集。类似于编程语言中的 void 函数。
特点：可以有 IN (输入), OUT (输出), INOUT 参数。通过 CALL 调用。
基本模版：

```
CREATE PROCEDURE 过程名(IN 参数名 类型, OUT 参数名 类型)
BEGIN
  -- 逻辑代码 (INSERT, UPDATE, DELETE, SELECT...)
END;
```

例题 1 (浅)：创建一个存储过程，输入学号，删除该学生记录。

```
CREATE PROCEDURE DeleteStudent(IN s_id INT)
BEGIN
  DELETE FROM Student WHERE id = s_id;
END;
```

2. 存储函数 (Stored Function)

概念：类似于存储过程，但必须返回一个值。通常用于计算。
特点：只能有 IN 参数（默认），必须有 RETURNS 类型，必须有 RETURN 语句。可以在 SQL 语句中直接使用（如 SELECT MyFunc(...)）。
基本模版：

```
CREATE FUNCTION 函数名(参数名 类型) RETURNS 返回类型
BEGIN
  DECLARE 变量名 类型;
  -- 逻辑计算
  RETURN 变量名;
END;
```

例题 2 (浅)：创建一个函数，输入两个数，返回它们的和。

```
CREATE FUNCTION AddNum(a INT, b INT) RETURNS INT
BEGIN
  RETURN a + b;
END;
```

3. 触发器 (Trigger)

概念：一种特殊的存储过程，它不是由用户主动调用，而是在对表进行 INSERT, UPDATE, DELETE 操作时自动触发。
关键词：BEFORE / AFTER (时机)，NEW / OLD (引用数据)。

- NEW.column：访问新插入或更新后的值。
- OLD.column：访问被删除或更新前的值。

基本模版：

```
CREATE TRIGGER 触发器名
AFTER/BEFORE INSERT/UPDATE/DELETE ON 表名
FOR EACH ROW
BEGIN
   -- 触发逻辑
END;
```

例题 3 (浅)：当向 Score 表插入成绩时，自动在 Log 表记录当前时间。

```
CREATE TRIGGER LogScoreInsert
AFTER INSERT ON Score
FOR EACH ROW
BEGIN
   INSERT INTO Log(operation_time, message) VALUES (NOW(), 'New score added');
END;
```

4. 游标 (Cursor)

概念：用于在存储过程或函数中，逐行处理查询结果集 (SELECT Result)。SQL 通常是集合操作，游标允许你通过循环一行一行地处理数据。
四步曲：

1. **定义** (DECLARE cursor_name CURSOR FOR select_statement)
2. **打开** (OPEN cursor_name)
3. **取值** (FETCH cursor_name INTO var1, var2...)
4. **关闭** (CLOSE cursor_name)

基本模版：

```
CREATE PROCEDURE UseCursor()
BEGIN
   DECLARE done INT DEFAULT 0;
   DECLARE v_name VARCHAR(20);
   -- 1. 定义游标
   DECLARE myCursor CURSOR FOR SELECT name FROM Student;
```

```
    -- 定义结束标志 (这是标准写法，记住即可)
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- 2. 打开
    OPEN myCursor;

    -- 循环
    read_loop: LOOP
        -- 3. 取值
        FETCH myCursor INTO v_name;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;

        -- 这里做业务逻辑，比如打印名字
    END LOOP;

    -- 4. 关闭
    CLOSE myCursor;
END;
```

第三部分：综合模拟测试 (合并测试)

场景背景：
做一个简单的电商订单系统。
包含三张表：

1. Products (商品表): pid (主键), pname, stock (库存), price
2. Orders (订单表): oid (主键), pid (外键), quantity (数量), total_price
3. OrderLog (日志表): log_id, oid, action, log_time

题型一：E-R图与范式

题目：如果这就一张大表 AllData(oid, pid, pname, stock, price, quantity, total_price, customer_name, customer_addr)。

1. **分析**：存在什么范式问题？
    - *答案思路*：pname, price 依赖于 pid，不完全依赖主键 oid（如果主键是联合的）或者造成数据冗余。customer_addr 依赖于 customer_name (假设 name 唯一)。这是典型的非3NF。
2. **E-R图**：请画出 用户、商品、订单 的关系。
    - *答案思路*：用户(1) -> (N)订单，订单(N) -> (1)商品 (简化模型)。

题型二：代码综合实战 (核心)

题目 1：存储函数 (计算总价)
创建一个存储函数 CalcTotal(p_id INT, qty INT)，根据商品ID和数量，查询单价并返回总金额。

```
CREATE FUNCTION CalcTotal(p_id INT, qty INT) RETURNS DECIMAL(10,2)
BEGIN
  DECLARE unit_price DECIMAL(10,2);
  -- 查询单价
  SELECT price INTO unit_price FROM Products WHERE pid = p_id;
  -- 返回总价
  RETURN unit_price * qty;
END;
```

题目 2：触发器 (自动扣减库存)
创建一个触发器 AfterOrderPlaced。当在 Orders 表中插入一条新订单时，自动减少 Products 表中对应商品的 stock。

```
CREATE TRIGGER AfterOrderPlaced
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
  -- 更新库存：原库存 - 新订单的数量
  UPDATE Products
  SET stock = stock - NEW.quantity
  WHERE pid = NEW.pid;
END;
```

题目 3：存储过程 (处理订单 - 包含事务逻辑)
创建一个存储过程 PlaceOrder(p_id INT, qty INT)。
逻辑：

1. 检查库存是否充足。
2. 如果充足，插入订单表 (total_price 调用上面的函数)。
3. 如果不充足，抛出错误提示。

```
CREATE PROCEDURE PlaceOrder(IN p_id INT, IN qty INT)
BEGIN
  DECLARE cur_stock INT;
  SELECT stock INTO cur_stock FROM Products WHERE pid = p_id;

  IF cur_stock >= qty THEN
    -- 库存充足，插入订单，利用刚才写的函数计算总价
    INSERT INTO Orders(pid, quantity, total_price)
    VALUES (p_id, qty, CalcTotal(p_id, qty));
  ELSE
    -- 库存不足 (这是 SQL 里的报错方式之一，或者用 SELECT 输出提示)
```

```
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '库存不足';
  END IF;
END;
```

题目 4：游标 (批量盘点)
创建一个存储过程 CheckStockWarning()。
逻辑：使用游标遍历所有商品。如果某商品 stock 小于 10，则在控制台输出（或插入到一个警告表）"商品XX库存预警"。

```
CREATE PROCEDURE CheckStockWarning()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE curr_pid INT;
  DECLARE curr_name VARCHAR(50);
  DECLARE curr_stock INT;

  -- 1. 定义游标：选出所有商品
  DECLARE stockCursor CURSOR FOR SELECT pid, pname, stock FROM Products;
  -- 定义退出处理器
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  -- 2. 打开
  OPEN stockCursor;

  -- 循环
  check_loop: LOOP
    -- 3. 获取一行数据
    FETCH stockCursor INTO curr_pid, curr_name, curr_stock;

    IF done = 1 THEN
      LEAVE check_loop;
    END IF;

    -- 业务逻辑：判断库存
    IF curr_stock < 10 THEN
      -- 这里简单用 SELECT 展示，考试时可能要求插入一张 Warning 表
      SELECT CONCAT('警告: 商品 ', curr_name, ' 库存不足 10 !') AS WarningMsg;
    END IF;
  END LOOP;

  -- 4. 关闭
  CLOSE stockCursor;
END;
```

备考建议

1. **背诵模版**：存储过程、函数、触发器、游标的 CREATE 语法结构必须背熟，尤其是 BEGIN...END，DECLARE 变量的位置。
2. **区分 NEW 和 OLD**：在触发器题中，插入用 NEW，删除用 OLD，更新两个都有。
3. **注意分号**：在 SQL 代码块中，每句结束的分号 ; 很重要。
4. **范式核心**：看到联合主键找部分依赖(2NF)，看到非主键之间的依赖找传递依赖(3NF)。