

**SQL 特点** 综合统一 高度非过程化 面向集合的操作方式 以同一种语法结构提供两种使用方式 语言简捷, 易学易用

**Common Queries** **Point query:** Find all rectangles containing a given point **Range query:** Find all objects within a query rectangle **Nearest neighbor:** Find the point closest to a query point **Intersection query:** Find all the rectangles intersecting a query rectangle 磁盘存储速度顺序扫描远大于随机读取 **I/O cost:** Number of disk sectors retrieved from secondary storage **CPU cost:** Number of CPU

十 数据库安全性与完整性

触发器 **Trigger** SQL 标准写法 Create Trigger name Before | After | Instead Of events  
[referencing-variables] [For Each Row] When (condition) action **Create Trigger Trigger1 After Delete On**  
**S Referencing Old Row As O For Each Row [No condition] Delete From R where A = O.B**



存在的问题：不可重复读。即在一次事务之间，进行了两次读取，但是结果不一样，可能第一次 id

- R3(C, D, A)和R4(C, D, B)
- R3的函数依赖 $CD \rightarrow A$ ，且码为CD，R3属于BCNF
- R4的函数依赖 $BC \rightarrow D$ ，且码为BC，R4属于BCNF

范式分解结果可能不同

```
%sql
CREATE OR REPLACE FUNCTION auditlogfunc() RETURNS TRIGGER AS $example_table$
BEGIN
    Insert INTO track(carID, time, position, username)
    SELECT distinct new.carID, CURRENT_TIMESTAMP(0), new.position, track.username
    FROM track
    WHERE new.carID = track.carID;
    RETURN NEW;
END;

$example_table$ LANGUAGE plpgsql;
Create Trigger Trigger1
INSTEAD OF Insert On CurrentTrack
For Each Row
EXECUTE PROCEDURE auditlogfunc();
```