SQL进阶知识笔记

一、CASE 表达式

1. 在 GROUP BY 子句里使用 CASE 表达式，可以灵活地选择作为聚合的单位编号或等级，在非定制化统计时能发挥巨大威力。
2. 在聚合函数中使用 CASE 表达式，可以轻松地将行结构的数据转换成列结构的数据。
3. 聚合函数也可以嵌套进 CASE 表达式里使用。
4. 相比依赖于具体数据库的函数，CASE 表达式有更强大的表达能力和更好的可移植性。
5. CASE 表达式是一种表达式而不是语句，有诸多优点。

二、自连接用法（self join）

1. 在需要获取列的组合时，常需要用到“非等值自连接”。
2. 自连接和 GROUP BY 结合使用可以生成递归集合。
3. 可以将自连接看作是不同表之间的连接。
4. 应把表看作行的集合，用面向集合的方式思考。
5. 自连接的开销（性能）更大，应尽量给用于连接的列建立索引。

三、三值逻辑和 NULL

1. 三值逻辑的真值表

- NOT

x	NOT x
t	f
u	u
f	t

- AND

AND	t	u	f
t	t	u	f
u	u	u	f

AND	t	u	f
f	f	f	f

- OR

OR	t	u	f
t	t	t	t
u	t	t	t
f	t	u	f

2. 优先级顺序

- AND：false > unknow > true
- OR：true > unlnow > false

3. NULL 不是值，不能对其使用谓词。
4. 对 NULL 使用谓词后的结果是 unknow。
5. 按步骤追踪 SQL 的执行过程能有效应对逻辑运算情况。
6. 如果 （ NOT ） IN 子查询中有用到表里被选择的列中存在 NULL，SQL语句整体的查询结果永远是空，但（ NOT ）EXISTS 谓词永远不会返回 unknown，只会返回 true 或者 false。
7. 极值函数在统计时会把为 NULL 的数据排除掉，但在输入为空表（空集）时会返回 NULL。
8. 除了 COUNT 以外的聚合函数同上。
9. 对 NULL 进行四则运算的结果都是 NULL。

四、HAVING 子句

1. HAVING 子句是可以单独使用的，此时 SELECT 子句不能引用原表里的列，只能使用常量或者聚合函数。
2. 表不是文件，记录也没有顺序，SQL 不进行排序。
3. SQL 不是面向过程语言，没有循环、条件分支、赋值操作。
4. SQL 通过不断生成子集来求得目标集合。
5. GROUP BY 子句可以用来生成子集。
6. WHERE 子句用来调查集合元素的性质，HAVING 子句用来调查集合本身的性质。

五、外连接的用法（outer join）

1. 在 SELECT 子句中使用标量子查询或者关联子查询，性能开销相当大。
2. 生成固定的表侧栏需要用到外连接。
3. 生成嵌套表侧栏时，事先按照需要的格式准备好主表并 cross join 生成笛卡尔积。
4. 当连接操作的双方是一对一或者一对多关系时，结果的行数并不会增加。

5. 全外连接相当于求集合的和。
6. SQL 不是用来生成报表的语言，不建议用来进行格式转换，必要时考虑用外连接或者 CASE 表达式来解决问题，建议还是交给宿主语言或者应用程序来完成。
7. 异或集（非交集）的开销相当大

- (A UNION B) EXCEPT (A INTERSECT B)
- (A EXCEPT B) UNION (B EXCEPT A)
- FULL OUTER JOIN 结合 WHERE 条件

六、用关联子查询比较行与行（使用窗口函数会更方便）

1. 关联子查询的缺点是代码可读性和性能不好。
2. 作为面向集合语言的 SQL 在比较多行数据时，不进行排序和循环。
3. （开始日期1，结束日期1）OVERLAPS（开始日期2，结束日期2）可以用来查询重叠的时间区间。

七、用 SQL 进行集合运算

1. SQL 能操作具有重复行的集合，可以通过可选项 ALL 来支持，使用 ALL 后不进行排序，性能会有提升。
2. 集合运算符有优先级，必须用括号明确地指定运算顺序。
3. 除法运算没有标准定义。
4. 同一个集合无论加多少次结果都相同，S UNION S UNION S ……UNION S 具有幂等性。
5. A UNION B = A INTERSECT B 等价于 A = B 等价于（A UNION B）EXCEPT（A INTERSECT B）的结果集是空集。
6. 两张表相等时返回“相等”，否则返回“不相等”

```
SELECT CASE WHEN COUNT(*)=0
            THEN "相等"
            ELSE "不相等" END AS result
       FROM (( SELECT * FROM tb_A
               UNION
               SELECT * FROM tb_B)
               EXCEPT
              ( SELECT * FROM tb_A
               INTERSECT
               SELECT * FROM tb_B)) AS tmp;
```

7. 用于比较表与表的 diff

```
(SELECT * FROM tb_A
EXCEPT
SELECT * FROM tb_B)
UNION ALL
(SELECT * FROM tb_B
```

```
EXCEPT
SELECT * FROM tb_A);
```

8. 用差集实现关系除法运算

- 嵌套使用 NOT EXISTS
- 使用 HAVING 子句转换成一对一关系
- 把除法变成减法（这是最好的方法）

9. （A 包含于 B）且（B 包含于A）等价于 A = B。
10. PgSQL 中实现行ID的名字是 oid，须事先在 CREATE TABLE 的时候指定可选项 WITH OIDS。

八、EXISTS 谓词的用法

1. EXISTS 可以将多行数据作为整体来表达高级的条件，而且使用关联子查询时性能非常好。
2. 在 EXISTS 的子查询里，SELECT 子句可以写成：

- SELECT *
- SELECT ‘常量’
- SELECT 列名

3. EXISTS 的输入值是行数据的集合，是二阶谓词。
4. NOT EXISTS 直接具备了差集运算的功能。
5. “肯定” 等价于 “双重否定” 之间的转换：“所有行都……” 等价于 “不……的行—行都不存在”。
6. EXISTS 和 HAVING 都是以集合而不是以个体为单位来操作数据的。
7. EXISTS 主要用于进行“行方向”的量化，ALL 和 ANY 主要用于进行“列方向”的量化。
8. 可以用 IN 谓词代替 ANY，如果遇到 NULL，需要使用 COALESCE 函数。
9. 可以用 ALL 谓词代替 NOT EXISTS。

九、用 SQL 处理数列

1. SQL 处理数列是把数据看成忽略了顺序的集合。
2. SQL 处理数据是把数据看成有序的集合

- 用自连接生成起点和终点的组合
- 在子查询中描述内部的各个元素之间必须满足的关系

3. 要在 SQL 中表达全称量化时，需要将全称量化命题转化成存在量化命题的否定形式，并使用 NOT EXISTS 谓词，因为 SQL 只实现了谓词逻辑中的存在量词。

十、再谈 HAVING 子句

1. HAVING 子句的处理对象是集合而不是记录，WHERE 子句的处理对象是元素（一行数据）。
2. 用于调查集合性质的常用条件机器用途

语句	作用
COUNT(DISTINCT col) = COUNT(col)	col列没有重复的值
COUNT(*) = MAX(col)	col列是连续的编号（起始值是1）
MIN(col) = MAX(col)	col列都是相同值或者都是 NULL
MIN(col) * MAX(col) > 0	col列全是正数或负数
MIN(col) * MAX(col) < 0	col列的最大值是正数，最小值是负数
MIN(ABS(col)) = 0	col列最少有一个是0
MIN(col - 常量) = - MAX(col - 常量)	col列的最大值和最小值与指定常量等距
COUNT(*) = COUNT(col)	col列不存在 NULL

3. 在 SQL 中指定搜索条件时，最重要的是搞清楚搜索的实体是集合还是集合的元素。
4. HAVING 子句可以通过聚合函数（特别是极值函数）针对集合指定各种条件。
5. 如果通过 CASE 表达式生成特征函数，无论多么复杂的条件都可以描述。

十一、性能优化

1. 参数是子查询时，使用（NOT） EXISTS 代替 （NOT） IN ，尤其是在建立过索引的情况下，可提高性能。
2. 参数是子查询时，使用连接代替 IN，可提高性能。
3. 会进行排序的代表性运算（损失性能）

- GROUP BY 子句
- ORDER BY 子句
- 聚合函数（SUM、COUNT、AVG、MAX、MIN）
- DISTINCT
- 集合运算符（UNION、INTERSECT、EXCEPT）
- 窗口函数（RANK、ROW_NUMBER等）

4. 灵活使用集合运算符的 ALL 可选项，可减少排序。
5. 使用 EXISTS 代替 DISTINCT，可减少排序。
6. 在极值函数（MAX / MIN）中使用索引，可减少排序时间。
7. 能写在 WHERE 子句里的条件不要写在 HAVING 子句里，可减少排序时间。
8. 在 GROUP BY 子句和 ORDER BY 子句中使用索引，可减少排序时间。
9. 使用索引时，条件表达式的左侧应该是原始字段。
10. 索引字段是不存在 NULL 的，索引字段不应该使用 IS （NOT） NULL 谓词。

11. 索引字段不能使用否定形式（<>、!=、NOT IN）和 OR 。
12. 联合索引中的第一列必须写在查询条件的开头，而且索引中列的顺序不能颠倒，如果无法保证查询条件里列的顺序与索引一致，可以考虑将联合索引拆分为多个索引。
13. 使用 LIKE 谓词时，只有前方一致的匹配才能用到索引（eg：'a%'）。
14. 在需要类型转换时，显式地进行类型转换（CAST）。
15. 灵活使用 HAVING 子句，减少中间表，可以提升性能。
16. 需要对多个字段使用 IN 谓词时，将它们汇总到一处，可以提升性能。
17. 先进行连接再进行聚合，可以提升性能。
18. 要格外注意避免定义复杂的视图和在视图中进行聚合操作（聚合函数和聚合运算符）。

十二、SQL 编程方法（声明式语言）

1. 养成写注释的好习惯。
2. 使用格式化工具美化格式（缩进、空格、大小写、前置逗号、不使用通配符、ORDER BY 中不使用列编号等）
3. 养成使用标准语法的好习惯。.
4. 写很复杂的 SQL 语句时，可以考虑按照执行顺序从 FROM 子句开始写（自底向上法）。
5. 聚合后不能再引用原表中的列，在对表进行聚合查询时，只能在 SELECT 子句中写入：

- 通过 GROUP BY 子句指定的聚合键
- 聚合函数（SUM、AVG等）
- 常量