数据库

2019年7月11日 20:31

数据库对象 编辑 讨论

有表，索引，视图，图表，缺省值，规则，触发器，用户，函数等。

中文名	数据库对象	性　质	计算机
外文名	A database object	类　别	编程

目录

数据库的四个基本概念：

- 数据（data）：
  - 存储的基本对象，描述事物的符号记录称为数据。
  - 数据的含义称为数据的语义，数据与其语义不可分。
- 数据库（DB）：
  - 长期存储在计算机内、有组织的、可共享大量数据的集合。
  - 概括来讲，数据库数据具有永久存储、有组织、可共享三个基本特点。
- 数据库管理系统（DBMS）：
  - 位于用户与操作系统之间的一层数据管理软件。
  - 数据定义语言DDL
  - 数据操纵语言DML
  - 数据控制语言DCL
- 数据库系统（DBS）：
  - 数据库系统是由数据库、数据库滚利系统（及其应用开发工具）、应用程序和数据库管理员组成的存储、管理、处理和维护数据的系统。

物理独立性：用户的应用程序与数据库中数据的物理存储是互相独立的。
逻辑独立性：用户的应用程序与数据库中数据的逻辑结构是相互独立的。

基本的关系操作:
查询操作可分为：选择、投影、连接、除、并、差、交、笛卡尔积;
其中选择、投影、并、差、笛卡尔积是5种基本操作。

关系完整性
实体完整性：主码不为空;
参照完整性：外码为空或者等于被参照的某个值;
用户定义的完整性。

关系代数
传统集合运算：并、差（属于R但不属于S）、交；笛卡尔积（n+m列的元组集合）
专门的关系运算：选择

SQL

功能	动词
数据查询	SELECT
数据定义	CREATE / DROP / ALTER
数据操纵	INSERT / UPDATE / DELETE
数据控制	GRANT / REVOKE

操作对象	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

模式、表、视图和索引的关系：
一个数据库可以创建多个模式，一个模式中包括多个表、视图和索引等数据库对象。

- 模式的相关操作
  - 定义模式：**CREATE SCHEMA 模式名 AUTHORIZATION 用户名;(分号表示这段代码结束)**。必须有管理员权限或者获得了数据库管理员给的权限才行。
  - 分号前可以加入创建表、视图等数据库对象的操作。
  - 删除模式：DROP SCHEMA 模式名 <CASCADE|RESTRICT>，CASCADE（级联）表示把该模式下所有数据库对象全部删除；RESTRICT（限制），表示该模式如果有数据库对象了，就拒绝删除。
- 基本表的操作
  - 定义基本表：CREATE TABLE 表名（<列名> <数据类型> <列级完整性约束条件>，
    <列名> <数据类型> <列级完整性约束条件>，
    ……，
    <表级完整性约束>
    ）；
  - 数据类型：
  - 完整性约束条件
    - 对约束条件进行命名：CONSTRAINT <条件名><约束条件>
    - 实体完整性：
      - 主键约束：primary key
        创建表时直接添加在值后加约束

        单一约束：create table 表名(列名1 列值1 primary key...)；

        联合约束：create table 表名(列名1 列值1, 列名2 列值2, ... 列名n 列值n, primary key(列名1,列名2);)

        创建表后，补充约束alter table 表名 add primary key(列名1);

        删除约束：alter table 表名 drop primary key;
      - 唯一约束：unique，数据唯一，可以为Null

        添加约束：create table 表名(列名1 列值1 约束条件,列名2 列值2 unique, ...)

        后期追加： alter table 表名 add unique(列名);
      - 自动增长：

        (auto_increment(MySQL)，identity(SQLServer)，sequence(oracle))：数据必须为整型，被删除行不影响后续自增数。

数据查询
select （ALL|DISTINCT） 目标表达式
from ……
(where 条件表达式)
(GROUP BY + 列名 +（HAVING 条件表达式）)
(ORDEE BY + 列名 +（ASC|DESC）);

- GROUP BY字句：将结果按列名的值进行分组，通常会用在需要对每组的数据使用聚集函数的场景；如果带HAVING短语，则只有满足HAVING的组才会被输出。
- ORDER BY子句，结果表按列名的值升序或者降序排序后输出。

- select后面的目标表达式，不止是属性列，也可以是字符串常量或者函数、算数表达式等；还可以制定别名
  - 字符串常量的情况：输出结果的这一列的值都为这个字符串常量
    - 如：select ‘hhahaha’ 输出时，会有一列的名称与值均为hahhah（回去验证）
  - 别名的情况：如：select birthday-1 BIN , Sno ,输出时，birthday-1表达式表示的列的列名为BIN。
  - 目标表达式前面可以加上DISTINCT来消除重复的值（行）。
- where后面的条件表达式：
  - 表 3.6 常用的查询条件

查询条件	谓词
比较	=，>，<，>=，<=，!=，<>，!>，!<；NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

  - 字符匹配操作条件：LIKE + '匹配串' + （ESCAPE） + '换码字符'
    - 匹配串可以是完整字符也可以是由通配符（%与_）构成的
    - %：代表任意长度的字符串；_：代表单个任意字符。
    - 如果要查询的字符串本身含有上面两个通配符，则要使用ESCAPE；如：'DA_dae' ESCAPE '\';这里，转码字符为\，表示斜杆后面的下划线不是通配符。
- ORDER BY 子句
  - 按照某一个列的值升序或者降序 A（升序）、DESC）降序，默认升序。
  - 对于空值，排序的顺序由具体系统来决定!
- 聚集函数
  - 截图
  - 如果在前面标注DISTINCT ,则计算时会取消重复值。
  - 遇到空值时，除了COUNT(*)外，都会调过空值。
  - 聚集函数只能使用在SELECT子句和GROUP BY 中的HAVING子句中!
- GROUP BY 子句
  - HAVING语句与WHERE语句的区别：WHERE子句作用于整个表，而HAVING语句只作用于满足条件的组。即HAVING在选择之前还要进行一次分组。
  - HAVING是分组条件的补充。且用于补充WHERE不能用聚集函数这个缺点。

连接查询

- 嵌套查询：基本思想：由前一个表中数据逐个对后一个表中数据进行遍历查询满足条件的元组。
- 索引会加快扫描?
- 子查询中不能使用ORDER BY ，因为只能对最终结构进行排序。

后期追加： alter table 表名 add unique(列名);

- 自动增长：

  （auto_increment(MySQL)，identity(SQLServer)，sequence(oracle)）：数据必须为整型，被删除行不影响后续自增数。

  这里的约束在修改时，都是使用add。

- **域完整性**
  - **数据类型**
  - **check短语来制定值应该满足的条件：如CKECK(SEX IN (1,2));也可以用来定义元组（表级）的约束。**
  - **非空：not null**

    alter table 表名 modify 字段名 字段类型 not null;要把这个字段类型也写出来
  - **默认值约束：default**

    添加约束：create table 表名(列名1 列值1 约束,列名2 列值2 约束 default 默认值, ...)

    alter table 表名 alter 列名 set default 默认值;
- **引用完整性**
  - **外键约束：foreign key**

    创建外键：建表的同时添加: constraint 外键关系名 foreign key(从表的字段) references 主表名(主表的字段),
    添加到列名后单独一行。

    后期追加：alter table 主表 add primary key(主表列名)

    alter table 从表 add foreign key(从表列名) references 主表(主表列名)

    删除外键关联：alter table 表名 drop foreign key 外键关系名。
  - 建表时建立外键，首先主表相应字段也要添加primary key的约束！外键关系名应该是随便起的。
    - 例子：

```
create table dept
(    dept_id int primary key,
     dept_name nvarchar(100) not null,
     dept_address nvarchar(100)
)

creat table emp
(     emp_id int constraint pk_emp_id_a primary key, --主键约束
     emp_name nvarchar(20) not null,
     emp_sex nchar(1),
dept_id int constraint fk_dept_id_b foreign key references dept(dept_id) --外键约束 )
```

  - 参考：https://blog.csdn.net/cyj142602/article/details/79542580

- 修改表：ALTER TABLE 表名 + ADD / DROP / ALTER

ADD（增加）	(COLUMN) + 新列名 +数据类型 +(完整性约束)
ADD（增加）	表级完整性约束
DROP（删除）	(COLUMN) + 列名 + (CASCADE|RESTRICT)
DROP CONSTRAINT（删除）	完整性约束名 + (CASCADE|RESTRICT)
ALTER COLUMN （修改）	列名 + 数据类型

  - 相关的一些东西回去补充。
- 删除表：DROP TABLE 表名 (CASCADE|RESTRICT) 默认是RESTRICT

- 索引的操作
  - 在关系数据库中，索引是对表中一列或多列的值进行排序的一种存储结构，它是表中一列或多列的值的集合（另一种表），而且其中包含了对应表中记录的引用指针。索引的作用相当于图书的目录，可以根据目录中的页码快速找到所需的内容。
  - 一般会自动在主码上建立索引。
  - 索引也是表的组成部分，建立太多索引会影响更新和插入的速度。
  - 建立了索引后，当查询索引相关的字段时，就会自动选择索引来进行查询访问（？）。
  - mysql的索引分为单列索引（主键索引，唯一索引、普通索引）和组合索引。
    - 单列索引：一个索引只包含一个列，一个表可以有多个单列索引。
    - 组合索引：一个组合索引包含两个或两个以上的列。
  - 建立索引：
    - 单列索引： CREATE INDEX IndexName ON `TableName`(`字段名`(length))
      或者ALTER TABLE TableName ADD INDEX IndexName(`字段名`(length))
      如果是CHAR,VARCHAR,类型,length可以小于字段的实际长度,如果是BLOB和TEXT类型就必须指定长度。
      例子：

      第一种方式：

```
CREATE INDEX account_Index ON `award`(`account`);
```

      第二种方式:

```
ALTER TABLE award ADD INDEX account_Index(`account`)
```

    - 唯一索引：所要求的列中的值唯一，但允许有空值。
      其sql格式是 CREATE UNIQUE INDEX IndexName ON `TableName`(`字段名`(length));
      或者 ALTER TABLE TableName ADD UNIQUE (column_list)
    - 主键索引：不允许有空值。主键索引建立的规则是 int优于varchar,一般在建表的时候创建,最好是与表的其他字段不相关的列或者是业务不相关的列.一般会设为 int 而且是 AUTO_INCREMENT自增类型的
    - 组合索引： CREATE INDEX IndexName On `TableName`(`字段名`(length),`字段名`(length),...);
      - 遵循最左前缀原则：where查询的条件要按照建立索引时候字段的排序方式;
      - 如：
        1、index(‘c1’ , ‘c2’ , ‘c3’) where ‘c2’ = ‘aaa’ 不使用索引,where `c2` = `aaa`

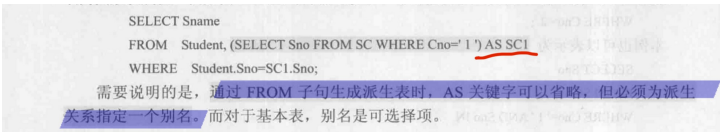
连接查询

- 嵌套查询：基本思想：由前一个表中数据逐个对后一个表中数据进行遍历查询满足条件的元组。
- 索引会加快扫描?
- 子查询中不能使用ORDER BY，因为只能对最终结构进行排序。
- 不相关子查询：子查询的查询条件不依赖父查询；相关子查询相反。
- 带有ANY（SOME）或者ALL的子查询，截一个用例就行；ANY就是某个值，ALL就是要求所有值
- 带有EXISTS的子查询：其中的子查询一般为select *，因为子查询只需要返回真假，列名是没有意义的。即只需要知道子查询的结果是否为空集。

集合查询

- 并操作（UNION）、交操作（INTERSECT）、查操作(EXCEPT)；这时候会存在两个SELECT但不同于嵌套查询使用括号连接，这时候使用上面三个符号连接。

派生表的查询

- 子查询在FROM中的查询：用例；这时候子查询产生的集合看成一个表，必须要为这个表指定一个别名，可以用AS来指定，也可以直接写别名，即（select .....）+(AS)+别名
- 如：
- 

数据更新：

- 插入数据
  - 插入元组：INSERT INTO<表名>（属性列） VALUES（常量）[，（常量），.....]；
  - 如果INTO子句中没有指明属性列名，则插入时每个属性都要有值。没有赋值的列自动赋空值NULL。
  - 插入子查询结果（多组值）：INSERT INTO <表名> （属性列） 子查询。
- 修改数据
  - UPDATE<表名> SET<列名> = <表达式> 【，<列名> = <表达式>】
    【where <条件>】修改满足where条件的元组值。
- 删除数据
  - DELETE FROM <表名> （where <条件>）

空值处理

- 判断：IS NULL 或者IS NOT NULL
- 空值算术运算（+-*/）为空值，空值与另一个值得比较运算为UNKNOWN；
- UNKNOWN：
  - 与运算：与True时为UNKNOWN;与FALSE时为FALSE.
  - 或运算：与True时为TRUE;与FALSE时为UNKNOWN
  - NOT运算：NOT UNKNOWN为UNKNOWN.
  - 只有条件运算后结果为TRUE的元组才会被输出。

视图：

- 定义视图：CREATE VIEW <视图名> [(<列名>,<列名>.....)] AS <子查询> [WITH CHECK OPTION];
  - WITH CHECK OPTION 表示对视图进行操作时，要满足视图定义中的子查询中的条件（WHERE）；管理系统会自动加上子查询中where的条件。
  - 视图列的命名；（截图）
  - 执行CREATE语句只是把视图定义放入数据字典中，并不执行其中的SELECT语句，当查询视图时才执行。
- 行列子集视图：从单表导出，只是去掉了某些行与列，保留主码；
- 带表达式的视图：用处：为了减少冗余，基本表存基本数据，由基本数据计算而来的各种派生数据是不存放的，由于视图并不实际存储，所以可以用来存放代表式的派生数据。
- 删除视图：DROP VIEW<视图名>[CASCADE]；CASCADE级联删除。如果不加，且视图有子视图会拒绝执行。
- 查询视图：与基本表一样。
  - 视图消解：把定义中的子查询与用户的查询结合起来，转换成等价的对基本表的查询后执行。（注意！有可能因此导致转换后WHERE条件中有聚集函数的存在，会报错）
- 更新视图：语法与更新基本表一样
  - 同样也是先转换成对基本表的更新数据的语句（把子查询中的WHERE放入UPDATE中的where中），同样要注意可能会有不能转换的问题。

and `c3`=`sss` 不能使用索引

2、查询中某个列有范围查询，则其右边的所有列都无法使用查询（多列查询）

Where c1= ‘xxx’ and c2 like = ‘aa%’ and c3=’sss’ 改查询只会使用索引中的前两列,因为like是范围查询

- 组合索引会出现多个索引，如：index(‘c1’，‘c2’，‘c3’) 会有三个索引：C1、（C1、C2）、(C1、C2、C3)

- 全文索引：当文本字段中出现多个一样的字符且需要查找时（及查找条件是： where column lick '%xxxx%'），普通索引会失效，要使用全文索引。

```
ALTER TABLE tablename ADD FULLTEXT(column1,
column2)
```

有了全文索引，就可以用SELECT查询命令去检索那些包含着一个或多个给定单词的数据记录了。

```
ELECT * FROM tablename
WHERE MATCH(column1, column2) AGAINST('xxx',
'sss', 'ddd')
```

这条命令将把column1和column2字段里有xxx、sss和ddd的数据记录全部查询出来。

- 删除索引：DORP INDEX IndexName ON `TableName`
- length的用处：前缀索引，选择字段数据的前n个字符作为索引，节约索引空间，提高操作速度与效率。
- 覆盖扫描：组合索引包含了所有要查询的字段。所以只需要差索引表就可以了。
- 优点：
  1.可以通过建立唯一索引或者主键索引,保证数据库表中每一行数据的唯一性.
  2.建立索引可以大大提高检索的数据,以及减少表的检索行数
  3.在表连接的连接条件 可以加速表与表直接的相连
  4.在分组和排序字句进行数据检索,可以减少查询时间中 分组 和 排序时所消耗的时间(数据库的记录会重新排序)
  5.建立索引,在查询中使用索引 可以提高性能
- 缺点：
  1.在创建索引和维护索引 会耗费时间,随着数据量的增加而增加
  2.索引文件会占用物理空间,除了数据表需要占用物理空间之外,每一个索引还会占用一定的物理空间
  3.当对表的数据进行 INSERT,UPDATE,DELETE 的时候,索引也要动态的维护,这样就会降低数据的维护速度,(建立索引会占用磁盘空间的索引文件。一般情况这个问题不太严重，但如果你在一个大表上创建了多种组合索引，索引文件的会膨胀很快)。
- 注意点：

  1.在经常需要搜索的列上,可以加快索引的速度

  2.主键列上可以确保列的唯一性

  3.在表与表的而连接条件上加上索引,可以加快连接查询的速度

  4.在经常需要排序(order by),分组(group by)和的distinct 列上加索引 可以加快排序查询的时间,（单独order by 用不了索引，索引考虑加where 或加limit)

  5.在一些where 之后的 < <= > >= BETWEEN IN 以及某个情况下的like 建立字段的索引(B-TREE)

  6.like语句的 如果你对nickname字段建立了一个索引,当查询的时候的语句是 nickname lick '%ABC%' 那么这个索引讲不会起到作用.而nickname lick 'ABC%' 那么将可以用到索引

  7.索引不会包含NULL列,如果列中包含NULL值都将不会被包含在索引中,复合索引中如果有一列含有NULL值那么这个组合索引都将失效,一般需要给默认值0或者 ' '字符串

  8.使用短索引,如果你的一个字段是Char(32)或者int(32),在创建索引的时候指定前缀长度 比如前10个字符 (前提是多数值是唯一的..)那么短索引可以提高查询速度,并且可以减少磁盘的空间,也可以减少I/O操作.

  9.不要在列上进行运算,这样会使得mysql索引失效,也会进行全表扫描

  10.选择越小的数据类型越好,因为通常越小的数据类型通常在磁盘,内存,cpu,缓存中 占用的空间很少,处理起来更快

- 不需要索引的：
  1.查询中很少使用到的列 不应该创建索引,如果建立了索引然而还会降低mysql的性能和增大了空间需求.
  2.很少数据的列也不应该建立索引,比如 一个性别字段 0或者1,在查询中,结果集的数据占了表中数据行的比例比较大,mysql需要扫描的行数很多,增加索引,并不能提高效率
  3.定义为text和image和bit数据类型的列不应该增加索引,
  4.当表的修改(UPDATE,INSERT,DELETE)操作远远大于检索(SELECT)操作时不应该创建索引,这两个操作是互斥的关系

参考：https://www.cnblogs.com/chenshishuo/p/5030029.html
https://blog.csdn.net/qq_36711757/article/details/80642931