计算机科学1992 Vol.19 No.5

关系数据库语言SQL的标准化

丁宝康　　（复旦大学计算机科学机系，上海200433）

摘　要

本文介绍关系数据库语言SQL的标准化历程，SQL86、SQL89、SQL2、SQL3及各个标准间的区别、联系和扩充。

一、引　言

关系数据库语言SQL(Structured Query Language)是目前广泛流行的一种功能很强、适合于各种机型的数据库语言。早在七十年代中期在IBM的SYSTEM R系统实现了SQL的原型SEQUEL语言。1986年美国国家标准化组织(ANSI)通过了SQL的美国标准，并公布了其正式文本，现称之为"SQL 86"。随后在1987年，国际标准化组织(ISO)也通过了这个标准。后经修订，1989年ISO公布了新的SQL标准，现称为"SQL89"。制订新的SQL标准还在继续，预计在1992年ISO要公布SQL2标准，以后还要公布SQL3标准。本文第二部分介绍SQL86，第三部分介绍SQL89，第四部分介绍SQL2，第五部分简略提及SQL3。

二、SQL86

1. SQL86的组成

SQL86包括四个组成部分：

(1) 模式定义语言：描述关系数据库的表、视图的结构和授权规则。

(2) 数据操纵语言(DML)：用于关系数据库的查询和更新。

(3) 模块语言：用于说明数据库和用宿主语言编写的应用程序间的调用界面。

(4) 嵌入式句法：应用程序中SQL语句的使用规则。

2. 基本概念

一个SQL数据库是表(table)和视图(view)的汇集，它用一个或若干个SQL模式定义。

一个SQL表由行集构成，一行(row)是列(column)的序列，每列对应一个数据项。

一个表或者是一个基本表(base table)，或者是一个视图表(view)。基本表是实际存贮在数据库中的表，而视图是若干个基本表或视图构成内容的子集。

3. 模式描述

一个SQL模式是表、视图和授权的静态定义。例如：

```
CREATE SCHEMA AUTHORIZATION
    STUDY
CREATE TABLE S
    (SNO CHARACTER(5) UNIQUE,
    SNAME CHAR(10),
    AGE INTEGER,
    SEX CHAR(1))
CREATE TABLE SC
    (SNO CHAR(5),
    CNO CHAR(5),
    GRADE FIXED(4, 1))
CREATE TABLE C
    (CNO CHAR(5) UNIQUE,
    CNAME CHAR(10),
    TEACHER CHAR (10) )
CREATE VIEW S4
```

```
    AS SELECT *
       FROM S
       WHERE SEX='M'
CREATE VIEW S6
    AS SELECT S.SNO, SNAME, CNO,
      GRADE
      FROM S, SC
      WHERE S.SNO=SC.SNO
GRANT ALL ON S TO 'WANG'
GRANT SELECT ON SC TO PUBLIC
```

4. 数据操纵语句

SQL的DML语句有下列九个：INSERT、DELETE、UPDATE、SELECT、OPEN、FETCH、CLOSE、COMMIT、ROLLBACK。

(1) **INSERT语句** INSERT语句把一行加入到一个表中。例如：

```
INSERT INTO S(SNO, SNAME, AGE)
    VALUES('S12', 'LTU', 20)
INSERT INTO SC
    VALUES('S12', 'C4', 86.5)
```

(2) **SELECT语句** 这个语句根据条件从给定表中检索一行或者若干行。例如：

```
SELECT SNAME, AGE
FROM S
WHERE SEX='M'
```

SELECT语句也可以对多个表做连接(JOIN)操作：

```
SELECT S.SNO, SNAME, CNO, GRADE
FROM S, SC
WHERE S.SNO=SC.SNO AND GRADE
    >=80
```

SELECT语句既可以单独用，也可以作为其它语句中的一个子表达式，还可以用在视图的定义中。在SELECT语句中可以利用聚合操作(例如求平均值AVG、最大值MAX、最小值MIN等)对数据进行操作：

```
SELECT SNO, MAX(GRADE)
FROM SC
WHERE CNO<>'C5'
GROUP BY SNO
HAVING AVG(GRADE)<80
```

(3) **DELETE语句** DELETE语句可以根据条件从表中删除一行或多行。例如：

```
DELETE FROM S
WHERE AGE IN(17, 18)
DELETE FROM S
WHERE NOT EXISTS SELECT*
    FROM SC
    WHERE SC.SNO=S.SNO
```

(4) **UPDATE语句** UPDATE语句据条件修改所选择的行。例如：

```
UPDATE S
SET AGE=AGE+1
UPDATE SC
SET GRADE=GRADE+10
WHERE SC.CNO IN
    (SELECT CNO
    FROM C
    WHERE CNAME='Maths')
```

(5) **游标和OPEN、FETCH、CLOSE语句** 一般，一个SELECT语句执行结果可得到一个行集，但在SQL和宿主语言的界面上，每次只允许传输一行。因而SQL提供游标(cursor)机制来说明一个行集。对游标有定义(DECLARE)、打开(OPEN)、推进(FETCH)、关闭(CLOSE)等操作。例如：

```
DECLARE CURSOR GRADESCAN
    FOR SELECT S.SNO, SNAME, CNO,
GRADE
      FROM S, SC
      WHERE S.SNO=SC.SNO
        AND S.SEX=:X
        AND CNO=:Y
OPEN GRADESCAN
oop:
    FETCH GRADESCAN
      INTO (:S-NO, :S-NAME, :C-NO,
            :GRADE)
end-loop;
CLOSE GRADESCAN
```

(6) **COMMIT、ROLLBACK语句** COMMIT语句标志一个事务成功结束。ROLLBACK语句标志一个事务不成功结束，此

时需做事务的恢复工作。

5. SQL86和典型SQL产品的差别

为使SQL标准能为计算机厂商和广大用户所接受，因而初始的SQL86标准局限于大多数系统已实现的特色，比较突出的有以下几点：

（1）SQL86缺乏某些公共的SLQ特色：①可执行的DDL特色。SQL86对表、视图、授权的描述是静态的数据描述，而不是可执行的CREATE语句和GRANT语句。SQL86也没有修改性的DDL语句，例如ALTER(增加、修改列的定义)和DROP(撤消)语句。②动态的SQL语句：PREPARE、EXECUTE。③描述数据库结构本身的目录表(catalog tables)。

（2）SQL86不包括CREATE INDEX语句及存贮管理特色方面的子句(例如"SEGMENTS"或"TABLESPACES")

（3）SQL86增加了某些数据类型以便与主语言匹配。例如：NUMERIC、REAL、DOUBLE PRICISION。

三、SQL89

在SQL86的基础上，SQL89在CREATE TABLE语句中加入完整性描述的三个子句：DEFAULT、CHECK及基本的引用完整性子句。

1. DEFAULT子句(缺省子句)

在用INSERT语句往一个表中加入一行时，有些列的值没有输入(即缺省)，这时系统就自动认为这些列的值是空值。但空值是一个相当复杂和易混淆的概念。SQL89在CREATE语句中加入DEFAULT子句解决此问题。当未输入值时，系统能自动默认为某个预定的值。例如：

```
CREATE TABLE SC
    (SNO CHAR (5),
    CNO CHAR (5),
    GRADE FIXED (4, 1)  DEFAULT(-1))
```

2. CHECK子句(检查子句)

在用CREATE语句描述表的列时，还可以加入一个CHECK子句来限制值的范围。例如：

```
CREATE TABLE SC
    (SNO CHAR(5),
    CNO CHAR(5),
    GRADE FIXED(4, 1) DEFAULT(-1)
      CHECK(GRADE=-1  OR  GRADE>
        =0))
```

3. 引用完整性(referential integrity)

关系数据库的"引用完整性规则"是指"不允许引用不存在的实体"。也就是记录之间是相互牵制的。这可用REFERENCES子句实现。例如：

```
CREATE TABLE S
    (SNO CHAR(5) PRIMARY KEY,
    SNAME CHAR(10),
    AGE INTEGER,
    SEX CHAR(1))
CREATE TABLE SC
    (SNO CHAR(5) REFERENCES S,
    CNO CHAR(5) REFERENCES C,
    GRADE FIXED(4, 1))
```

表示表SC中SNO值一定要在表S中存在。REFERENCES子句也可采用线外编码方式书写：

```
CREATE TABLE SC
    (SNO CHAR(5),
    FOREIGN KEY(SNO)REFERENCES
        S(SNO),
    CNO CHAR(5),
    FOREIGN KEY(CNO) REFERENCES
        C(CNO),
    GRADE FIXED(4, 1))
```

四、SQL2

SQL2尚未正式公布，现介绍其对SQL89扩充的部分。

1. SQL2的级别

与SQL86和SQL89比较，SQL2相当庞大，因此分成两个级别的子集和一个完全的级别。(1)初级SQL2 (Entry SQL2)，在SQL89基础上，增加了某些功能。(2)中级

SQL2（Intermediate SQL2），是SQL2最有价值的一部分。(3) 完全SQL2（Full SQL2），完整的SQL2标准。

2. 初级SQL2

下面分5小点叙述初级SQL2对SQL89的扩充。

(1) **SQLSTATE反馈变量** 在SQL89中，所有SQL语句会返回一个值给程序中的变量，指出SQL语句执行是否成功。这个值就通常所称的"状态码"。在SQL86中，反馈变量称为"SQLCODE"，它的值能表明三类状态：成功执行、没有找到行、错误。这里"错误"是指系统实现时都已定义过的情况。由于SQLCODE的值是系统实现时定义的，因此就难于编写程序对不同错误条件采取不同的动作。为了与SQL89相容，SQL2另用一个反馈变量(称为SQLSTATE)来说明那些错误条件，例如"数据例外"、"违反约束"等。最终，SQL2还是用SQLSTATE来包括SQLCODE的功能。

(2) **AS子句** 在SELECT语句中选取的数据项表中，有些是简单的列名，有些是算术表达式。为了给算术表达式起个名字，可用AS子句。AS子句也可对列名起个新的名字。例如：

```
SELECT SNO, AVG(GRADE)
FROM SC
GROUP BY SNO
```

可写成下列形式：

```
SELECT SNO AS STUDENT-NO,
    AVG(GRADE) AS AVG-GRADE
FROM SC
GROUP BY SNO
```

(3) **分隔符** SQL2中专用名词很多，为了使用户也能使用这些专用名词作为列名或表名，可采用加分隔符(双引号")的办法。例如：

```
CREATE TABLE "TEACHER"
    ("ID OF THE TEACHER"CHAR(5),
    "NAME" CHAR(10),
    "REFERENCES" CHAR(10),
    "FOREIGN" CHAR(10))
```

(4) **支持Ada和C语言作为宿主语言** SQL86和SQL89支持的宿主语言是COBOL、FORTRAN、Pascal、PL/I。SQL2还支持Ada和C语言作为宿主语言。

(5) **对SQL89的修正和澄清**

3. 中级SQL2

下面分26小点叙述中级SQL2对初级SQL2的扩充。

(1) **时间数据类型及其操作** SQL2中包含新的数据类型DATE(日期)、TIME(时间)和TIMESTAMP(时间戳)，并可进行算术操作和算术比较操作。

(2) **关于引用完整性的删除操作的连锁反应** 在SQL89中只说明了引用约束，即在INSERT、DELETE、UPDATE语句违反引用完整性的操作时都要被拒绝执行。SQL2扩充为执行这些操作时要保持引用完整性。中级SQL2就有执行DELETE操作时的说明。例如：

```
CREATE TABLE SC
    (SNO CHAR(5),
    CNO CHAR(5)
      REFERENCES C(CNO)
      ON DELETE CASCADE,
    GRADE FIXED(4, 1))
```

这里REFERENCES子句表示表SC中CNO值一定要在表C中出现。而"ON DELETE CASCADE"子句表示：在表C中删除某个CNO值的行时，也要在表格SC中删除所有这个CNO值所在的行。如果ON子句用另一种型式"ON DELETE SET NULL"，那么表示在C中删除某个CNO值的行时，在表SC中所有这个CNO值都要置换成空值，表示"未知"或"没有值"。

(3) **域(Domain)** 在SQL2中可用CREATE DOMAIN语句来指出值域，然后再用列名与域名对应，进而定义列的类型、长度。例如：

```
CREATE DOMAINSCORE IS DECIMAL(4,1)
```

```
    DEFAULT(-1)
    CHECK(VALUE=-1 OR VALUE>=0)
    NOT NULL
CREATE TABLE SC
    (SNO CHAR(5),
    CNO CHAR(5),
    GRADE SCORE)
```

(4) **多字符集(包括双字节)** SQL89只提供单字符集的数据类型。SQL2扩充为可定义新的字符集，以表示非拉丁字母(例如日文、中文等)。

(5) **多校核序列** 在EDCDIC码中，所有小写字母排在大写字母之后，而在ASCII码中，所有小写字母排在大写字母之前。这就对姓名等处理不方便。

在SQL2中，把所有校核序列都认为实现时已定义了的。而在SQL3中，在实现时还可重新定义，譬如用FOLDED名词表示大写字母和小写字母等价。

(6) **外连接(outer join)** 在用SELECT语句做连接操作时，遇到不匹配情况。例如：

```
SELECT*
FROM S, SC
WHERE S.SNO=SC.SNO
```

如果S中有一个SNO值在SC中不出现，那么这个SNO值在SELECT操作的结果表中也不会出现，这是因为连接操作未采用空值。为弥补这点，SQL2中增加外连接操作。上例用外连接实现的语句如下：

```
    S LEFT JOIN SC ON (S.SNO=SC.SNO)
```

这里"LEFT JOIN"表示在S中出现而在SC中不出现的SNO值在结果表中也要出现，出现时CNO、GRADE的值为空值。

还有两个类似的操作是"RIGHT JOIN"和"FULL JOIN"。

(7) **自然连接(natural join)** 自然连接是前面提到的连接操作的缩写。例如"S LEFT JOIN SC ON(S.SNO=SC.SNO)"可写成："S NATURAL LEFT JOIN SC"。

还有两个类似的操作是"NATURAL RIGHT JOIN"和"NATURAL FULL JOIN"。

外连接在连接的列上不一定要求有相同的列名，但自然连接在连接的列上要求有相同的列名。

(8) **交(intersect)和差(except)操作**

例如交操作：

```
(SELECT * FROM S WHERE SEX='M')
    INTERSECT
(SELECT * FROM S WHERE='WANG%')
```

例如差操作：

```
SELECT * FROM S WHERE SEX='F')
    EXCEPT
(SELECT * FROM S WHERE  AGE>19)
```

(9) **卷游标(scroll cursors)** 在SQL89中，游标打开后，推进的顺序是从找到的行集的第一行到最后一行，不能改变这个顺序，既不能返回也不能跳跃式。SQL2提供的FETCH操作可以任意改变顺序。例如已知一个命名为STUDENT-CURSOR的游标被打开，那么可使用下列FETCH语句：

```
FETCH NEXT FROM STUDENT-CURSOR
FETCH PRIOR FROM STUDENT-CURSOR
FETCH RELATIVE +3 FROM STUDENT-
    CURSOR
FETCH RELATIVE -3 FROM STUDENT-
    CURSOR
FETCHABSOLUTE 5 FROM STUDENT-CU-
    RSOR
FETCH ABSOLUTE -7 FROM STUDENT-
    CURSOR
FETCH FIRST FROM STUDENT-CURSOR
FETCH LAST FROM STUDENT-CURSOR
```

这里，NEXT(PRIOR)表示把游标推进(返回)一行，RELATIVE+3(-3)表示把游标推进(返回)3行，ABSOLUTE +5(-7)表示把游标指向找到行集的第5行(倒数第7行)，FIRST(LAST)表示把游标指向找到行集的第1行(最后一行)。

(10) **行表达式** 用"行表达式"形式可把若干算术比较表达式合写成一行的形式。

例如行表达式

(S1.SNO, S1.AGE)>(S2.SNO, S2.AGE)

是下列表达式的缩写：

S1.SNO>S2.SNO OR (S1.SNO=S2.SNO AND S1.AGE=S2.AGE)

(11) **子串和并接操作** SQL2提供求字符串的子串和并接操作。例如：

```
SUBSTRING('ABCDEFG', 3, 2)
'STUDENT NAME:' || S.SNAME
```

(12) **CASE表达式** SQL2提供类似于程序设计语言的条件表达式形式：

```
CASE(SEX)
    WHEN 'F' THEN 'FEMALE'
    WHEN 'M' THEN 'MALE'
END
CASE
    WHEN SEX='F' THEN 'FEMALE'
    WHEN SEX='M' THEN 'MALE'
END
CASE(:X)
    WHEN 'FEMALE' THEN 'F'
    WHEN 'MALE' THEN 'M'
END
CASE
    WHEN N=0 THEN 0 ELSE A/N
END
```

(13) **COALESCE和NULLIF函数** 这两个函数是CASE表达式的缩写。例如：COALESCE(S.AGE, "AGE IS NULL")，表示CASE WHEN S.AGE IS NULL THEN "AGE IS NULL" ELSE S.AGE END。

NULLIF(SC.GRADE,-1)表示CASE SC.GRADE=-1 THEN NULL ELSE SC.GRADE END。

(14) **CAST函数** CAST函数用于数据类型的转换。例如：CAST X AS DECIMAL(5, 2)把变量X的值转换成DECIMAL(5, 2)类型的数。

(15) **变长字符串** 在SQL89中，字符串类型的列的长度必须是定长，例如CHAR(20)。SQL2中加入"CHARACTER VARYING"变长字符串数据类型，只要指出最大长度就行。例如：

```
CREATE TABLE S (SNO CHAR(5),
    SNAME CHARACTER VERYING (20))
```

(16) **LENGTH函数** 求字符串长度可用LENGTH函数。例如：

CHARACTER-LENGTH (S.SNAME)

求变量的逻辑长度。

OCTET-LENGTH(S.SNAME) 求变量在机中实际存贮长度。

(17) **视图中的查询表达式** SQL89对视图的操作有各种限制，例如不能做并操作。SQL2取消了各种限制。

(18) **值表达式中的子查询** 在值表达式的子查询中可出现语句中任何地方的变量值。

(19) **GET DIAGNOSTICS语句(取诊语句)** 这个语句可得到由SQLSTATE提供的有关错误的信息。例如，如果SQLSTATE指出一个语句违反了引用完整性，那么这个语句指出应满足的引用完整性规则。

(20) **SET AUTHORIZATION语句** 这语句用于改变当前的授权标识。

(21) **动态SQL** SQL的语句可以写在应用程序中，与应用程序源程序一起被"编译"或"处理"。而"动态SQL"语句直至执行时才提供。例如，一个SELECT语句可以从终端上输入，而不是在应用程序中直接写出。EXECUTE IMMEDIATE语句把包含一个SQL语句的字符串看成一个操作符，然后执行这个语句。PREPARE语句把包含一个SQL语句的字符串看成一个操作符，编译这个语句以便反复执行这个语句。EXECUTE语句执行经过这样预处理的语句。SQL89中没有动态SQL语句。

(22) **模式操纵语言** 模式操纵语言涉及到各种形式的ALTER和DROP语句。ALTER语句可以在表中加入新的列、加入或修改DEFAULT子句、加入或删除CHECK子句

及引用完整性约束。DROP语句是撤消基本表或视图。

(23) **目录表(catalog tables)** SQL2的目录表是一个固有表集，用于描述表、视图、列的性质。任何一个应用程序都可查阅目录。

(24) **约束名** 在每个约束说明中，可含有一个约束名子句。例如：

```
CREATE TABLE S
    (SNO CHAR(5) CONSTRAINT S-KEY
        PRIMARY KEY, ……)
```

表示S-KEY是列SNO的主键特性的命名。在GET DIAGNOSTICS语句执行时发现违反这个约束时，能把这个约束名S-KEY显示出来。可以用ALTER语句把约束名子句删去。

(25) **只读事务和事务一致性级别** SET语句有各种选择子句，为一个应用程序指出它以只读方式可访问数据库的哪些部分，或者指出可与哪些事务并发执行。

(26) **多模块支持** 在应用程序中，有些是使用SQL的被单独编译的例行子程序，"多模块支持"的意思就是为这些应用提供各种新的规则和界限。

4. 完全SQL2

下面分21小点叙述完全SQL2对中级SQL2的扩充。

(1) **时间数据类型及其操作** 除了中级SQL2提供的以外，完全SQL2还提供不同的时间区域和多字段间隔，例如YEAR TO MONTH, YEAR TO DAY, MONTH TO HOUR等等。

(2) **目录表** 在目录表中增加引用完整性约束的信息。

(3) **临时表** 在程序执行时，需要有一些中间数据，可以放在专用的临时表中以供使用。在这些表的说明中可以指出数据是否需要保留到COMMIT结束。

(4) **BIT数据类型** BIT数据类型用于非结构化的二进制数据，如图形数据。例如：

```
CREATE TABLE LAYOUT
    (MAP BIT(5000),
     ……)
```

(5) **并、差、交中的对应选择** 在两个表的列数目、列名、列序不完全相同情况下做并、差、交时，可加入对应选择，使之能实现。例如有表T1和T2：

T1 (SNO, SNAME, AGE, SEX)

T2(SNO, SEX, HIGH, WEIGHT, SNAME)

那么并操作"T1 UNION CORRESPONDING T2"的结果有三列：SNO、SNAME、SEX，列序与并操作的第一个表T1中列序一致。

(6) **并连接(union join)** 假设有表S和SC：

S (SNO, SNAME, AGE, SEX)

SC (SNO, CNO, GRADE)

现在想求每个SNO值的有所信息，在SQL89是难于实现的。并、连接操作也不行。SQL2提供"并连接"操作来实现：

S UNION JOIN SC

操作结果可想象为下列形式：

表S	空列
空列	表SC

"并连接"操作把所要信息合成一张表，然后可用ORDER BY子句分组：

```
SELECT COALESCE (S.SNO, SC.SNO)AS
        S-ID,
   S.*, SC.*
FROM S UNION JOIN SC
ORDER BY S-ID
```

(7) **关于引用完整性的更新操作的连锁反应** 像中级SQL2中有删除操作的连锁反应一样，还有更新操作的连锁反应。例如：

```
CREATE TABLE SC
    (SNO CHAR(5),
   CNO CHAR(5) REFERENCES C(CNO)
       ON UPDATE CASCADE,
   GRADE FIXED (4, 1) )
```

这里"ON UPDATE CASCADE"表示在表C

中某个CNO值修改时，在表SC中相应的CNO值也都改成新值。如果用"ON UPDATE SET NULL"代替，那么表示在表C中某个CNO值修改时，在表SC中相应的CNO值都要置成空值。

(8) **引用完整性中的MATCH选择** 当表中主键不是一列，而是多列组成时，则可用PRIMARY KEY子句形式。例如：

```
CREATE TABLE COURSE
    (CLASS CHAR(5),
    C-NAME CHAR(10),
    FORMAT INTEGER,
    PRIMARY KEY (CLASS, C-NAME) )
```

在其它表的引用完整性约束中可这样描述：

```
CREATE TABLE TEACHER
    (TNO CHAR(5),
    CLASS CHAR(5),
    CNAME CHAR(10),
    FOREIGN KEY(CLASS, CNAME)
        REFERENCES COURSE (CLASS,
        C-NAME))
```

在SQL89中，表TEACHER中某一行的CLASS或CNAME值一个是空值、一个是非空值时，就不进行引用完整性检查了。在完全SQL2中，就要检查那个非空值在表COURSE中是否存在。

(9) **MATCH谓词** 带MATCH的谓词要检查引用完整性约束是否符合。在进行INSERT或UPDATE操作时，要进行"预测试"过程：

```
…WHERE (:CLASS, :C-NAME) MATCH
    SELECT CLASS, C-NAME
    FROM COURSE
```

(10) **大写和小写函数(UPPER,LOWER)** 这两个函数分别把字符串转换成大写和小写两种情况。例如：…WHERE UPPER (S.SNAME) = 'WANG'

(11) **位置函数(POSITION)** 这个函数指出一个子串在某字符串中的起始位置。例如函数POSITION ('WANG HUA', 'HU') 返回值是5。

(12) **FROM子句中嵌套的"表"表达式** SQL89要求FROM子句中提及的表都是简单的表名，而完全SQL2允许提及的表名是一个SELECT表达式或其它值表达式得到的值。这基本上是和视图概念一致的。

(13) **视图实现** SQL89对视图的引用有各种限制。例如，如果视图是用一个SELECT表达式定义、并且包含有GROUP BY子句或SUM一类的聚合操作，那么在包含有GROUP BY子句或聚合操作的SELECT语句中就不能再引用这个视图。完全SQL2取消了这些限制。

(14) **"表"表达式(table expression)** "表"表达式是一个行集。例如：

```
INSERT INTO SC
    (TABLE ('S4', 'C5', 85),
        ('S4', 'C8', 90.5),
        ('S7', 'C5', 95))
```

(15) **SET SCHEMA语句** 这个语句在动态SQL语句中用于改变表名和视图名的模式限定。例如程序要用到模式STUDY中的表S，那么要写成"STUDY.S"。在执行SET SCHEMA "STUDY"语句后，动态SQL语句PREPARE中的表名S都会自动处理成"STUDY.S"。

(16) **CHECK子句中的子查询** SQL89中表描述的CHECK子句里只能引用该子句所在表中的列名。完全SQL2允许CHECK子句中包含子查询，也能引用其它表的列名。例如：

```
CREATE TABLE SC
    (SNO CHAR(5),
    CNO CHAR(5),
    GRADE FIXED(4,1),
    CHECK (CNO<>'C222'
        OR NOT EXISTS
            (SELECT*
            FROM S
            WHERE S.SNO=SC.SNO
```

```
        AND S.SEX='M')))
```

这里CHECK子句表示学习C222这门课的学生都是女同学。

(17) **简单断言** 一个断言是由一个CHECK子句构成的，在CHECK子句中允许出现子查询。例如：

```
CREATE ASSERTION SC-RULE
CHECK (CNO<>'C222'
   OR NOT EXISTS
        (SELECT*
     FROM S
     WHERE S.SNO=SC.SNO
       AND S.SEX='M'))
```

(18) **约束检查的推迟** 完全SQL2提供一种方法在一个事务内可推迟一个或多个完整性约束的检查。例如：

```
SET CONSTRAINTS  OFF
   ……(一个或多个SQL语句)
SET  CONSTRAINTS  ON
```

(19) **在SELECT子句中可出现多个DISTINCT专用词** SQL89规定在SELECT子句中只能出现一个DISTINCT专用词。完全SQL2取消了这个限制。例如：

```
SELECT COUNT(DISTINCT SNO), COUN-
       T(DISTINCT CNO)
FROM SC
```

(20) **对列授于INSERT权限** SQL89定义的INSERT权限是对一个表的所有列起作用。完全SQL2可以把INSERT权限定在某些列上。例如：GRANT INSERT (SNO,CNO) ON SC TO 'WANG'

(21) **自我引用的UPDATE和DELETE语句** SQL89规定UPDATE和DELETE语句中的WHERE子句中不允许引用进行修改和删除工作的表。完全SQL2取消这个限制。例如：

```
UPDATE SC SET GRADE=GRADE*1.1
WHERE GRADE<>-1
   AND SC.GRADE<SELECT AVG(GRA-
       DE)
     FROM SC X
     WHERE X.CNO=SC.CNO
       AND X.SNO<>SC.SNO
DELETE FROM SC
WHERE GRADE<>-1
   AND SC.GRADE<=ALL SELECT X.
GRADE
     FROM SC X
     WHERE X.CNO=SC.CNO
       AND X.SNO<>SC.SNO
```

上面UPDATE语句是把每门课程中低于本课程平均成绩的成绩上浮10%。DELETE语句是把每门课程中最低一个成绩所在的行删去。

五、SQL3

制订SQL3标准的工作还在进行着。主要工作集中在以下几个方面：

(1) 递归并(recursive union)

(2) 临时视图

(3) 外部过程调用

(4) 用户定义的数据类型

(5) 子表和表的综合

(6) 用户定义的字符集和校核序列

(7) "角色"(roles，更一般的授权结构)

(8) "安全点"(savepoint，嵌套的检索结构)

(9) 触发器和加强型断言

(10) SIMILAR子句(LIKE的扩充)

(11) 枚举型数据类型

(12) 布尔型数据类型

(13) 多空值状态

(14) 引用完整性的PENDANT特色

(15) 异步DML语句

(16) SET CURSOR MODE

(17) 对于查询进行UPDATE操作

参考文献

[1] Phil Shaw, Database language standards, past, present, and future, the International Symposium of IBM Germany on Database Systems of the 90s, 1990.

[2] Database Language SQL, ANSI X3.135-1986, ISO-9075-1987(E),(即通常所称的"S-

计算机科学1992 Vol.19 No.5

数据库查询理论

Ashok K. Chandra

Chan, AK 范明

一、引 言

从Codd引进关系数据库开始，在理解怎样才能从数据库中提取数据方面已做了大量工作。已经提出和开发了许多数据库系统和查询语言，包括SQL，QBE，QUEL，L-DL等。与此同时，数据库的查询理论已成长为一个丰满的技术领域，这一发展把更重实效的开发建立在对问题的全面理解之上。确实，现在逻辑程序设计方面开展的研究就是这种情况。本文的目的不是全面地综述数据库查询理论，而是指出随着该领域的发展，使得该领域成型的一些思想。

二、一阶查询和关系代数

第一个重要的思想来自Codd，他考察了查询关系数据库的两种基本方法：一种是使用一阶逻辑，另一种是使用关系代数。一个一阶查询可以表示为$\{x \mid \varphi(x)\}$，它的输出是使得一阶公式在给定的数据库下为真的所有元组x的集合。公式φ可以使用$\forall$、$\exists$、$\wedge$、$\vee$、$\neg$、$=$和给定的数据库关系。此外，可以允许常数和有限制的算术运算。关系代数由使用诸如投影、选择、连接、差等运算符构造的项组成。

Codd指出可用一阶逻辑表示的任意查询也可以用关系代数表示，并且其逆成立。类似的结果早已被逻辑界认识，但是，在数据库界，Codd的结果具有很大的影响。表示查询的两种不同的方法等价这一事实意味着这是一个相当健壮的概念。此外，一阶逻辑看来是相当有表达能力的，并且术语"完备性"时常被用来表明一种查询语言能够表示所有的一阶查询。事实上，当时的查询语言的设计者们认为，他们的查询语言是"完备的"，被认为具有足够强的表达能力。

然而，这并不意味着表达能力比一阶逻辑弱的语言就不重要。事实上，数据库系统的许多查询常常是简单的，并且可以更有效地回答。此外，一阶查询的子集可以充当数据库系统的内部界面。例如，合取查询（和密切相关的制表查询）对应于选择-投影-连接查询（等价地，$\exists$、$\wedge$查询）。它们可以被优化为唯一最小范式，并且可以充当回答一阶查询或逻辑程序设计的内部表示。一种更小的查询类，选择-投影查询（例如，系统R中的"sargable"查询）有时也被用作存取数据，处理索引、锁和日志等低层次数据库子系统的界面。这种查询对应于在单个关系中挑选满足某种性质的所有元组。例如，"列出计算机科学系年薪在五万至十万之间的所有副教授的姓名"（在这个例子中，数据库有一个关系，包含对应于姓名、职称、系别和年薪的列）。

现在回到一阶查询的表达能力，很早就清楚：尽管对于表示所有的一阶查询，它是一种理想的查询语言，但是仅仅具有这种表达能力是不够的。例如，传递闭包查询常常是有用的，但不能用一阶逻辑表示。传递闭包的一个例子是，"列出直接或间接为Smith

QL86")

[3] Database Language SQL with integrity enhancement, ANSI X3, 135-1989, ISO-9075-1989 (E), （即通常所称的"SQL89"）

[4] Database Language SQL2 and SQL3, ISO working draft.