Structured Query Language (SQL)-1

陆伟

Database Systems

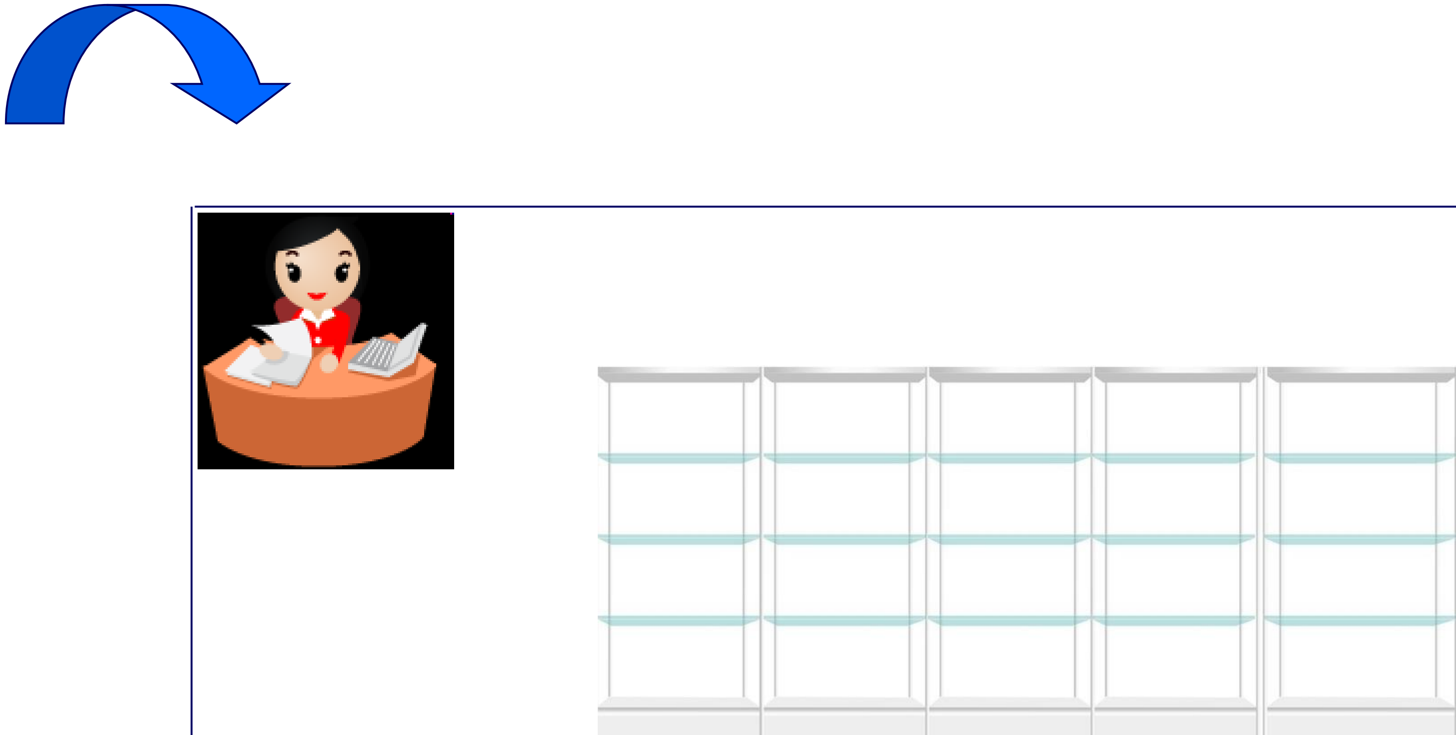
March 26, 2021

Outline

- SQL introduction
- Basic SQL

SQL Introduction

- Analyze deposit



SQL Introduction

- What is SQL
  - SQL is abbreviation of Structured Query Language.
  - It is the way we access the database.
- Objectives of SQL
  - Allow a user to
    - Create the database and relation structure.
    - Perform basic data management tasks. (Insertion, modification, and deletion of data from relation)
    - Perform both simple and complex queries
  - Perform these tasks with minimal user effort

SQL Introduction

- History of SQL
  - E.F.Codd – ‘A relational model of data for large shared data banks’ (1970)
  - 1974, D.Chamberlin defined the Structured English Query Language (SEQUEL)
  - 1976, a revised version SEQUEL/2 – SQL
  - IBM, System R based on SEQUEL/2

SQL Introduction

- Late 1970s, the first commercial implementation of relational DBMS based on SQL – ORACLE
- IBM, 1982, SQL/DS; 1983, DB2
- 1982, the American National Standards Institute began work on a Relational Database Language (RDL) based on a concept paper from IBM.
- SQL-86, SQL-89, SQL-92(SQL2), SQL-99(SQL3), SQL-2003, SQL-2008, SQL-2011

SQL Introduction

- Features of SQL
  - Non-procedual - based on relational algebra and relational calculus
  - Easy to learn

SQLfunctions	Verb
Data query	SELECT
Data define	CREATE, DROP, ALTER
Data manipulation	INSERT, UPDATE, DELETE
Data control	GRANT, REVOKE

SQL Introduction

- SQL can be used in two ways
  - Use SQL interactively by entering the statements at a terminal
  - Embed SQL statements in a procedural language (programmatic SQL)

Basic SQL

- An SQL statement consists of reserved words and user-defined words.
- Most components of an SQL statement are case insensitive.
- Usually a statement terminator (;) is used to end each SQL statement. (Although the standard does not require it)

Basic SQL

- Although SQL is free-format, an SQL statement or set of statements is more readable if indentation and lineation are used.
  - Each clause in a statement should begain on a new line;
  - The beginning of each clause should line up with the beginning of other clause;
  - If a clause has several parts, they should each appear on a separate line and be indented.
  - …

Data Definition Language (DDL)

- SQL Identifiers
  - Used to identify objects in the database, such as table names, view names, and columns.
  - Character set
    - A…Z, a…z
    - 0…9
    - Underscore(_)
  - Restrictions
    - An identifier can be no longer than 128 characters
    - An identifier must start with a letter
    - An identifier cannot contain spaces

Data Definition Language (DDL)

- SQL Scalar Data Types

Data type	Declarations
boolean	BOOLEAN
character	CHAR VARCHAR
bit	BIT BIT VARYING
exact numeric	NUMERIC DECIMAL INTEGER SMALLINT
approximate numeric	FLOAT REAL DOUBLE PRECISION
datetime	DATE TIME TIMESTAMP
interval	INTERVAL
large objects	CHARACTER LARGE OBJECT BINARY LARGE OBJECT

Data Definition Language (DDL)

- Scalar operators
  - +,-,*,/
  - LOWER,UPPER,TRIM,SUBSTRING
  - CURRENT_USER, SESSION_USER
  - …

Data Definition

- Creating a Database
  - The ISO standard does not specify how database are created, and each dialect generally has a different approach.

Data Definition

- Creating a table (CREATE TABLE)

```
CREATE TABLE TableName
{(columnName dataType [NOT NULL][UNIQUE]
 [DEFAULT defaultOption][CHECK(searchCondition)][,…]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),][,…]}
{[FOREIGN KEY (listOfForeignKeyColumns)
    REFERENCES ParentTableName[(listOfCandidateKeyColumns)],
      [MATCH {PARTIAL|FULL}
      [ON UPDATE referentialAction]
      [ON DELETE referentialAction]][,…]}
{[CHECK(searchCondition)][,…]})
```

Integrity Enhancement Feature

- Entity Integrity
  - The primary key of a table must contain a unique, non-null value for each row.
  - The PRIMARY KEY clause can be specified only once per table.
  - Alternate keys can be ensured uniqueness using the keyword UNIQUE.
  - Every column that appears in a UNIQUE clause must be declared as NOT NULL.

Integrity Enhancement Feature

- Referential Integrity
  - Referential integrity is implemented through FOREIGN KEY.
  - A value in the foregin key must refer to an existing valid row in the parent table or be null.
  - Define the foreign key

Integrity Enhancement Feature

- Enterprise Constraints
  - The CREATE ASSERTION statement is an integrity constaint that is not directly linked with a table definition.

```
CREATE ASSERTION StaffNotHandlingTooMuch
  CHECK (NOT EXISTS (SELECT staffNo
                     FROM PropertyForRent
                     GROUP BY staffNo
                     HAVING COUNT(*)>100))
```

Example

department(dNo,dName, officeRoom, homepage)
student(sNo, sName, sex, age, dNo)
course(cNo, cName, cPNo, credit, dNo)
sc(sNo, cNo, score, recordDate)

Create Tables.sql
Insert Values.sql

Data Definition

- Changing a Table Definition (ALTER TABLE)

```
ALTER TABLE TableName
[ADD [COLUMN] columnName dataType [NOT NULL][UNIQUE]
[DEFAULT defaultOption][CHECK(searchCondition)]]
[DROP [COLUMN] columnName [RESTRICT|CASCADE]]
[ADD [CONSTRAINT [constraintName]] tableConstraintDefinition]
[DROP CONSTRAINT constraintName [RESTRICT|CASCADE]]
[ALTER [COLUMN] SET DEFAULT defaultOption]
[ALTER [COLUMN] DROP DEFAULT]
```

Data Definition

- Removing a Table (DROP TABLE)

```
DROP TABLE TableName [RESTRICT|CASCADE]
```

Data Manipulation Language (DML)

- General form

```
SELECT    [DISTINCT|ALL]{*|[columnExpression[AS newName]][,…]}
FROM      TableName[alias][,…]
[WHERE    condition]
[GROUP BY  columnList][HAVING condition]
[ORDER BY columnList]
```

Examples

- Retrieve all columns, all rows

```
SELECT *
FROM Student;
```

- Retrieve specific columns, all rows

```
SELECT sNo,sName
FROM Student;
```

Examples

- Use DISTINCT eliminate the duplicates

```
SELECT DISTINCT sNo
FROM SC;
```

- Calculated fields

```
SELECT sName, 2017-sAge
FROM Student;
```

Examples

- Row selection (WHERE clause)

Condition	**Predicates(谓词)**
comparison	Comparison operators(=,>,<,<>,<=,>=,!=)
range	BETWEEN AND, NOT BETWEEN AND
set membership	IN, NOT IN
pattern match	LIKE, NOT LIKE
null	IS NULL, IS NOT NULL
logical operators	AND, OR,NOT

Examples

- (1) comparison

```
SELECT sName
FROM Student
WHERE sex = '男';
```

- (2)range

```
SELECT sNo,sName,dNo,age
FROM Student
WHERE age NOT BETWEEN 20 AND 23;
```

Examples

- (3)set membership

```
SELECT sName, sex
FROM Student
WHERE dNo NOT IN ('01', '02', '03');
```

- (4)pattern match
  - %--represents any sequence of zero or more characters (wildcard)
  - _--represents any single character
  - Escape character--\,#

Examples

- (4)pattern match

```
SELECT sNo,sName,sex
FROM Student
WHERE sName LIKE ‘张%’;

SELECT *
FROM Course
WHERE cName LIKE ’DB\_%i__’ ESCAPE ’\’;
```

Examples

- (5)null

```
SELECT sNo
FROM SC
WHERE cNo=‘010101’ and score is null;
```

- (6)logical operators

```
SELECT sName
FROM Student
WHERE dNo=‘01' AND Sage<20;
```

Three-State Logic about Null

- Need ways to evaluate boolean expressions and have the result be “unknown” (or T/F)
- Need ways of composing these three-state expressions using AND, OR, NOT:

T AND U = U	T OR U = T	
F AND U = F	F OR U = U	NOT U = U
U AND U = U	U OR U = U	

Examples

- Sorting Results (ORDER BY clause)
  - In general, the rows of an SQL query result table are not arranged in any particular order. We can user ORDER BY clause in the SELECT statement to sort the results.
  - The ORDER BY clause must always be the last clause of the SELECT statement.
  - ASC – ascending order, default order
  - DESC – descending order

Examples

```
SELECT sNo, score
FROM SC
WHERE cNo='010101' ORDER BY score DESC;

SELECT *
FROM Student
ORDER BY dNo, age DESC;

Which is the place for null?
```

Examples

- Using the SQL Aggregate Functions
  - The ISO standard defines five aggregate functions

```
COUNT([DISTINCT|ALL]  *)
COUNT([DISTINCT|ALL] <columnName>)
SUM([DISTINCT|ALL] <columnName>)
AVG([DISTINCT|ALL] < columnName >)
MAX([DISTINCT|ALL] < columnName >)
MIN([DISTINCT|ALL] < columnName >)
```

Examples

- COUNT, MIN, and MAX apply to both numeric and non-numeric fields, but SUM and AVG may be used on numeric field only.
- Apart from COUNT(*), each function eliminates nulls first and operates only on the remaining non-null values.
- Aggregate functions can be used only in the SELECT list and in the HAVING clause.

Examples

```
SELECT COUNT(*)
FROM Student;
```

```
SELECT COUNT(DISTINCT sNo)
FROM SC;
```

```
SELECT COUNT(*) AS countOf
FROM Course
WHERE credit >= 2;
```

Examples

```
SELECT AVG(score)
FROM SC WHERE cNo='1';

SELECT MAX(score)
FROM SC WHERE cNo='1';
```

Examples

- Grouping Results (GROUP BY clause)
  - All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.
  - When the WHERE clause is used with GROUP BY, the WHERE clause is applied first.
  - The ISO standard considers two nulls to be equal for purposes of the GROUP BY clause.

```
SELECT cNo,COUNT(sNo)
FROM SC
GROUP BY cNo;
```

Examples

- Restricting groupings (HAVING clause)
  - HAVING clause is designed for use with the GROUP BY clause to restrict the groups that appear in the final result table.
  - Similar in syntax, HAVING and WHERE serve different purposes.
  - The ISO standard requires that column names used in the HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.
  - The HAVING clause is not a necessary part of SQL.

Examples

```
SELECT sNo
FROM SC
WHERE score>60
GROUP BY sNo
HAVING COUNT(*)>3;
```

Examples

- Subqueries
  - In SQL statement, a SELECT-FROM-WHERE statement is called a query block.
  - A query block embedded within WHERE or HAVING clause of another query block is called a subquery or nested query.
  - Subqueries may also appear in INSERT, UPDATE, and DELETE statements.

Examples

- There three types of subquery
  - A scalar subquery returns a single column and single row.
  - A row subquery returns multiple columns, but again only a single row.
  - A table subquery returns one or more columns and multiple rows.

Examples

- (1) use of IN predicate

查询与刘晨在同一个系的所有学生姓名

```
SELECT sNo
FROM Student
WHERE sName='张望';
```

Suppose the result is ‘03’

```
SELECT sNo,sName,dNo
FROM Student
WHERE dNo='03';
```

```
SELECT sNo, sName, dNo
FROM Student
WHERE dNo IN
        (SELECT dNo
           FROM Student
           WHERE sName='张望');
```

Examples

查询选修了课程名为‘矩阵论’的所有学生姓名

```
SELECT sNo, sName
FROM Student
WHERE sNo IN
      SELECT sNo
      FROM SC
      WHERE cNo IN
           （SELECT cNo
             FROM Course
             WHERE cName=‘矩阵论’）;
```

Examples

- For all the subqueries we discuss above, each subquery executes only once and the result of the subquery is used by supquery. The condition of subquery is not dependent on the condition of supquery. We call this type of query independent query.

Examples

- (2) use of comparison operators in subquery
  - The subquery must place after the comparison operator.
  - The subquery SELECT list must consist of a single column name or expression, except for subqueries that use the keyword EXISTS.
  - The ORDER BYclause may not be used in a subquery.

Examples

```
SELECT sNo, sName
FROM Student
WHERE sNo IN=
        SELECT sNo
        FROM SC
        WHERE cNo IN=
                SELECT cNo
                FROM Course
                WHERE cName='矩阵论' ;
```

Examples

- (3) use aggregate function in subquery

  查询所有年龄小于平均年龄的学生姓名

```
SELECT sNo,sName
FROM Student
WHERE age < (SELECT AVG(age)
             FROM Student);
```

Examples

- (4)use ANY/ SOME and ALL in subquery
  - >ANY, <ANY, >=ANY, <=ANY, =ANY, <>ANY/!=ANY
  - >ALL, <ALL, >=ALL, <=ALL, =ALL(no meaning), <>ALL/!=ALL

	=	<>或！ =	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>=MIN
ALL	--	NOT IN	<MIN	<=MIN	>MAX	>=MAX

Examples

```
SELECT sName, age
FROM Student
WHERE age < ANY (SELECT age
                 FROM Student
                 WHERE dNo='01')
  AND dNo <> '01'
ORDER BY age DESC;
```

```
SELECT sName, age
FROM Student
WHERE age <
        (SELECT MAX(age)
         FROM Student
         WHERE dNo='01')
  AND dNo <> '01S'
ORDER BY age DESC;
```

Examples

```
SELECT sName, age
FROM Student
WHERE age<ALL (SELECT age
                FROM Student
                WHERE dNo='01' )
  AND dNo <> '01'
ORDER BY age DESC;
```

```
SELECT sName, age
FROM Student
WHERE age <
       (SELECT MIN(age)
        FROM Student
        WHERE dNo='01')
  AND dNo <> '01'
ORDER BY age DESC;
```

Examples

- Muti-Table Queries
  - To combine columns from several tables into a result table we need to use a join operation.
  - To obtain information from more than one table, the choice is between using a subquery and using a join.
  - If the final result table is to contain columns from different tables, then we must use a join.

Examples

- (1) equijoin and non-equijoin

```
SELECT Student.*, SC.*
FROM Student, SC
WHERE Student.sNo=SC.sNo;

SELECT s.sNo, s.sName, sc.cNo, sc.score
FROM Student s, sc
WHERE s.sNo=sc.sNo;
```

Note: When Cartesian product appear?

Examples

- (2) self-join

```
SELECT FIRST.cNo, SECOND.cPNo
FROM Course FIRST, Course SECOND
WHERE FIRST.cPNo=SECOND.cNo;
```

Examples

- (3) outer join

sNo	sName	sAge
s1	…	…
s2	…	…
s3	…	…

sNo	cNo	score
s1	c1	…
s2	c2	…
s1	c2	…

List all students along with their elective courses even the student have not elect any course.

Examples

```
SELECT s.*, sc.*
FROM Student s, SC sc
WHERE s.sNo = sc.sNo;
```

```
SELECT s.*, sc.*
FROM Student s, SC sc
WHERE s.sNo = sc.sNo(*);
```

```
SELECT s.*, sc.*
FROM Student s LEFT JOIN SC sc
  ON s.sNo = sc.sNo;
```

Examples

- (3) outer join
  - Left outer join
  - Right outer join
  - Full outer join

Examples

- Muti-table join

```
SELECT s.sName, c.cName,sc.score
FROM Student s, Course c, SC sc
WHERE s.sNo = sc.sNo
  and c.cNo=sc.cNo;
```

Examples

- ISO SQL syntax

```
SELECT table1.column,tabel2.column
FROM  talbe1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING(column_name)] |
[JOIN table2
   ON(table1.column_name=table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
   ON(table1.column_name=table2.column_name)];
```

Examples

- EXISTS and NOT EXISTS
  - EXISTS and NOT EXISTS is existential quantifiers. They produce a simple true/false result.
  - Since EXISTS and NOT EXISTS check only for the existence or non-existence of rows in the subquery result table, the subquery can contain any number of columns.
  - Usually we write the subquery as:
    (SELECT * FROM…)

Examples

查询所有没选‘001’课程的学生姓名

```
SELECT sName
FROM Student
WHERE NOT EXISTS
        (SELECT *
          FROM SC
          WHERE sNo=Student.sNo AND cNo='001');
```

Can we use NOT IN to implement this query?

Examples

- EXISTS and NOT EXISTS
  - Some subqueries using EXISTS or NOT EXISTS can be replace by subqueries with other form (IN etc.), but some can’t.

```
SELECT sNo, sName, sDept
FROM Student S1
WHERE EXISTS
      SELECT *
      FROM Student S2
      WHERE S2.sDept=S1.sDept AND
              S2.sName='刘晨';
```

Examples

- EXISTS and NOT EXISTS
  - There is no direct expression for universal quantifier(全称量词) in SQL. We can use predication calculus to transform a predication using universal quantifiers to a predication using existential quantifiers.

$$(\forall x)P \equiv \neg(\exists x(\neg P))$$

  - There is also no direct expression for implication(全称量词) in SQL. We can also tranform it to expression using existential quantifiers.

$$p \rightarrow q \equiv \neg p \vee q$$

$$(\forall y)p\rightarrow q \equiv \neg(\exists y(\neg(P\rightarrow q)))$$

$$\equiv \neg(\exists y(\neg(\neg P\vee q)) \equiv \neg\exists y(P\wedge\neg q)$$

Examples

List the students who elect all courses.

```
SELECT sName
FROM Student S
WHERE NOT EXISTS
      (SELECT *
        FROM Course C
        WHERE NOT EXISTS
              (SELECT *
                FROM SC
                WHERE sNo=S.Sno AND
                        cNo=C.cNo);
```

Examples

List the students who elect the courses elected by ‘xxx’at least .

```
SELECT DISTINCT sNo
FROM SC SCX
WHERE NOT EXISTS
       (SELECT *
        FROM SC SCY
        WHERE SCY.sNo='xxx' AND
            NOT EXISTS
                 (SELECT *
                   FROM SC SCZ
                   WHERE SCZ.sNo=SCX.sNo
                      AND SCZ.cNo=SCY.cNo);
```

Examples

- Combining Result Tables (UNION, INTERSECT, EXCEPT)
  - In SQL, we can use the normal set operations of Union, Intersection, and Difference to combine the results of two or more queries into a single result table.
  - The three set operators in the ISO standard are called UNION, INTERSECT, and EXCEPT

Operator[ALL][CORRESPONDING [BY{column1[,…]}]]

Examples

```
(SELECT sNo
FROM SC
WHERE
cNo='030101')
UNION
(SELECT sNo
FROM SC
WHERE
cNo='030102');
```

Examples

```
(SELECT sNo
FROM SC
WHERE cNo='010101')
INTERSECT
(SELECT sNo
FROM SC
WHERE cNo='010102');
```

Examples

```
(SELECT sNo
FROM SC
WHERE cNo='030101')
EXCEPT
(SELECT sNo
FROM SC
WHERE cNo='030102');
```

Examples

- …

```
SELECT sv1.sNo,s.sName,sv1.avg_score
From student s, (select sNo,avg(score) as avg_score
              from sc group by sNo)as sv1(sNo,avg_score)
Where s.sNo=sv1.sNo
```

Data Update

- Adding data to the database (INSERT)
  - Adds new rows of data to a table.
  - There are two forms of INSERT statement. The first allows a single row to be inserted into a named talbe and has the following format:

    INSERT INTO TableName [(columnList)]

    VALUES (data ValueList)
  - The second form of the INSERT statement allows multiple rows to be copied from one or more tables to another, and has the following format:

    INSERT INTO TableName [(columnList)]

    SELECT …

Examples

- 插入行

```
INSERT INTO Student
VALUES (‘170120', '陈冬', '男', 18, ‘01' );

INSERT INTO SC(sNo, cNo)
VALUES (‘170120', ‘010101');
```

Examples

- 插入子查询结果

```
CREATE TABLE s_score(
    sNo      CHAR(6),
    sName    VARCHAR(20)
    avg      decimal(5,2),
    );
```

```
INSERT
INTO s_score(sNo,sName,avg)
  (select s.sNo,s.sName,v.avgscore
   from student s,(select sNo,avg(score) from sc group by sNo)
   as v(sNo,avgscore)
   where s.sNo=v.sNo);
```

Data Update

- Modifying data in the database (UPDATE)
  - Modifies existing data in a table.

```
UPDATE TableName
SET columnName1=dataValue1[,columnName2=datavalue2…]
[WHERE searchCondition]
```

Examples

```
UPDATE Student
SET age=22
WHERE sNo=‘170101’;

UPDATE Student
SET age=age+1;
```

Data Update

- Deleting data from the database (DELETE)
  - Removes rows of data from a table

```
DELETE FROM TableName
[WHERE searchCondition]
```

```
DELETE FROM Student
WHERE sNo='070122';

DELETE FROM SC;
```

Data Update

- About consistency of update or delete operation

Data Update

- …