Class CS211 Number 202115030121 Name 杨昊 Data

**Experiment topic：**

Experiment 1 Creating and managing the database

Experiment 2 Creating table and entering data

1. **Experimental objectives**

Be familiar with the window of SQL Server Management Studio.

Master the method of creating database.

Master the methods of managing databases.

Be familiar with the operation of creating data tables.

Master the operation of creating data tables.

Master the operation of data input and modification.

1. **Experimental tasks**
   1. Experimental contents

Experiment 1:

1）Open the ‘SQL Server Management Studio’ window, expand the server in the

‘Object Explorer’, then right-click the ‘Database’ node, click the ‘New

Database’ command, finally, the ‘New Database’ dialog box will appear.

2）After entering the database ‘XSCJ’ in the ‘Database Name’ box of the ‘New

Database’ dialog box, you can create a database with the default size by clicking

the ‘OK’ button.

3）Right-click the ‘XSCJ’ database, and select the ‘Properties’ command in the

pop-up shortcut menu, the ‘Database Properties’ dialog box will appear.

4）You can add or delete database files by click the ‘File’ tab, and click the ‘OK’

button to complete the database modification.

Experiment 2:

1）Open ‘SQL server Management Studio’ window，click on the ‘New Query’ button in the ‘standard’ toolbar, and the interface as shown in the figure will appear.

2）In the ‘SQL Editor’ toolbar, click the drop-down button on the right side of ‘Available Database’ to switch the current database to ‘XSCJ’ database.

3）In the query window, input the query statements of creating tables.

4）In the ‘Object Explorer’, unfold the database ‘XSCJ’, right-click the ‘table’ node, and click the ‘refresh’ command in the pop-up shortcut menu, you can see the 3tables that have been created above.

5）Right-click each table and chose the ‘create the first 200 rows’ command in the drop-down menu, and input data for each table, note the input order of the three tables

2.2 Experimental preparation

Review the SQL main clauses that will be used in this experiment, and write down the right way to use the clauses in the examples.

* CREATE TABLE

The basic syntax of the CREATE TABLE statement:

CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

column3 datatype constraint,

...

);

Among them, table\_ Name is the name of the table to be created, column1, column2, and column3 are the column names of the table, and datatype is the data type of each column. You can also define constraints for each column.

Here is an example of how to create a user table named users using the CREATE TABLE statement:

CREATE TABLE users (

id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE,

age INT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

* Constraint

1. Primary key constraint: The primary key constraint is used to uniquely identify each row of data in the table. It ensures that the values in the primary key column are unique and not allowed to be empty. A primary key constraint can consist of one or more columns.

CREATE TABLE table\_name (

Id INT PRIMARY KEY

);

CREATE TABLE table\_name (

Id INT,

Name char(10),

PRIMARY KEY (Id, Name)

);

1. Foreign key constraints: Foreign key constraints are used to establish relationships between tables. It ensures that the values of foreign key columns in one table must exist in the primary key columns of another table. Foreign key constraints help maintain referential integrity between tables.

CREATE TABLE foreign\_table (

Id INT PRIMARY KEY,

Name CHAR(10), UNIQUE

);

CREATE TABLE tab\_name (

Id INT REFERENCES foreign\_table(Id)

)

CREATE TABLE tab\_name (

Id INT,

FOREIGN KEY (id) REFERENCES foreign\_table(Id)

);

1. Unique constraint: A unique constraint ensures that the values in a column or column combination are unique, meaning duplicate values are not allowed. Unlike primary key constraints, unique constraints allow null values.

CREATE TABLE table\_name (

ID INT UNIQUE

);

1. Non null constraint: Non null constraint is used to ensure that a column is not allowed to be null. This means that when inserting or updating records, the column must contain valid values.

CREATE TABLE table\_name (

ID INT NOT NULL

);

1. Default constraint: The default constraint specifies a default value for the column. If a value for the column is not provided when inserting a record, the default value will be used.

CREATE TABLE table\_name (

ID INT DEFAULT 0

);

1. Check constraints: Check constraints define the conditions for validating data in a table. Only records that meet the conditions can be inserted or updated into the table.

CREATE TABLE table\_name (

Gender CHAR(1) CHECK( Gender IN (‘0’, ‘1’);

);

* Insert Grammar

The syntax for commonly used insert operations in SQL:

INSERT INTO table\_name (column1, column2, ..., columnN) VALUES (value1, value2, ..., valueN);

INSERT INTO employees (id, name, age) VALUES (1, 'John Doe', 25);

If you want to insert multiple records simultaneously, you can use the following syntax:

INSERT INTO table\_name (column1, column2, ..., columnN)

VALUES

(value1, value2, ..., valueN),

(value1, value2, ..., valueN),

...

(value1, value2, ..., valueN);

1. **Task solutions**

**Analyzes the tasks in the task book, and gives the SQL statements of all the tasks.**

CREATE TABLE XSQK (

stu\_id CHAR(6) NOT NULL PRIMARY KEY,

sname CHAR(8) NOT NULL UNIQUE,

gender CHAR(1) NOT NULL DEFAULT('1') CHECK(gender IN ('0', '1') ),

birthday DATETIME NOT NULL,

department CHAR(10) NOT NULL,

major CHAR(10) NOT NULL,

phone CHAR(11) CHECK( phone >= '10000000000'),

total\_credits TINYINT CHECK( total\_credits >= 0 AND total\_credits <= 200),

remarks TEXT

);

CREATE TABLE XSQK (

stu\_id CHAR(6) NOT NULL PRIMARY KEY,

sname CHAR(8) NOT NULL UNIQUE,

gender CHAR(1) NOT NULL DEFAULT('1') CHECK(gender IN ('0', '1') ),

birthday DATETIME NOT NULL,

department CHAR(10) NOT NULL,

major CHAR(10) NOT NULL,

phone CHAR(11) CHECK( phone >= '10000000000'),

total\_credits TINYINT CHECK( total\_credits >= 0 AND total\_credits <= 200),

remarks TEXT

);

CREATE TABLE XS\_KC (

stu\_id CHAR(6) NOT NULL REFERENCES XSQK(stu\_id),

course\_id CHAR(3) NOT NULL REFERENCES KC(course\_id),

score TINYINT CHECK( score >= 0 AND score < 101),

credit TINYINT,

PRIMARY KEY (stu\_id, course\_id)

);