1. **SQL语言是（ ）的语言，易学习。**
2. **过程化**
3. **非过程化**
4. **格式化**
5. **导航式**
6. **SQL语言具有（ ）的功能。**
7. **关系规范化、数据操纵、数据控制**
8. **数据定义、数据操纵、数据控制**
9. **数据定义、关系规范化、数据控制**
10. **数据定义、关系规范化、数据操纵**
11. **下列SQL语句中，修改表结构的是（ ）。**
12. **ALTER**
13. **CREATE**
14. **UPDATE**
15. **INSERT**
16. **检索所有比“王华”年龄大的学生姓名、年龄和性别。正确的SELECT语句是（ ）。**
17. **SELECT SN,AGE,SEX FROM S**

**WHERE AGE>(SELECT AGE FROM S**

**WHERE SN="王华")**

1. **SELECT SN,AGE,SEX FROM S**

**WHERE SN="王华")**

1. **SELECT SN,AGE,SEX FROM S**

**WHERE AGE>(SELECT AGE**

**WHERE SN="王华")**

1. **SELECT SN,AGE,SEX FROM S**

**WHERE AGE>王华.AGE)**

1. **"假定学生关系是S(S#,SNAME,SEX,AGE),课程关系是C(C#,CNAME,TEACHER),学生选课关系是SC(S#,C#,GRADE)。**

**要查找选修“COMPUTER”课程的“女”学生姓名，将涉及到关系（ ）。"**

1. **S**
2. **SC,C**
3. **S,SC**
4. **S,C,SC**
5. **"设有职工关系Emp（Eno，Ename，Esex，EDno）和部门关系Dept（Dno，Dname， Daddr），创建这两个关系的SQL语句如下：**

**CREATE TABLE Emp(**

**Eno CHAR(4) PRIMARY KEY (Eno),**

**Ename CHAR(8),**

**Esex CHAR(1) CHECK(Esex IN ('M', 'F')),**

**EDno CHAR(4) ,**

**FOREIGN KEY (EDno) REFERENCES Dept(Dno)**

**);**

**CREATE TABLE Dept(**

**Dno CHAR(4) NOT NULL UNIQUE,**

**Dname CHAR(20),**

**Daddr CHAR(30)**

**);**

**直接运行该语句，DBMS会报错，原因是 （1） 。若经过修改，上述两个表创建完毕之后（尚无数据），则下述语句中能被执行的是 （2） 。（ ）。"**

1. **"(1) 创建表 Emp 时，被参照表 Dept 尚未创建;**

**(2) INSERT INTO Emp VALUES('e001', '王', 'M', NULL);"**

1. **"(1) 创建表 Dept 时没有指定主码**

**(2) INSERT INTO Emp VALUES('e001', '王', 'M ', 'd1');"**

1. **"(1) 表 Emp 的外码 EDno 与被参照表 Dept 的主码 Dno 不同名**

**(2) INSERT INTO Emp VALUES('e001', '王', 'X', 'd1');"**

1. **"(1) 创建表 Dept 时没有指定外码**

**(2) INSERT INTO Emp VALUES(NULL, '王', 'M', 'd1');**

1. **不能提高查询性能的措施是（ ）。**
2. **建立相关视图**
3. **建立查询表的聚簇索引**
4. **尽量使用不相关子查询**
5. **根据查询条件建立索引 错误**
6. **"若用如下的 SQL 语句创建了一个表 SC ：**

**CREATE TABLE SC （**

**S# CHAR （ 6 ） NOT NULL ，**

**C# CHAR （ 3 ） NOT NULL ，**

**SCORE INTEGER ，**

**NOTE CHAR （ 20 ））；**

**向 SC 表插入如下行时，（ ）行可以被插入。**

1. **（'201009', '111', 60, ‘必修’）**
2. **（'201009', '111', 60, 必修）**
3. **（NULL, '103', 80, '选修'）**
4. **（'201132', NULL, 86, ' '）**
5. **WHERE子句的表达式“sdept in('IS', 'CS')”等价于（ ）。**
6. **"sdept='IS' or sdept='CS' "**
7. **sdept=All('IS', 'CS')**
8. **sdept=Any('IS', 'CS')**
9. **sdept='IS' and sdept='CS'**
10. **SQL 语言集数据查询、数据操纵、数据定义和数据控制功能于一体，其中， CREATE 、 DROP 、 ALTER 语句是实现（ ）功能。**
11. **数据定义**
12. **数据查询**
13. **数据操纵**
14. **数据控制**
15. **"企业职工和部门的关系模式如下所示，其中部门负责人也是一个职工。**

**职工（职工号，姓名，年龄，月薪，部门号，电话，地址）**

**部门（部门号，部门名，电话，负责人代码，任职时间）**

**请将下面的 SQL 语句空缺部分补充完整。**

**CREATE TABLE 部门（**

**部门号 CHAR(4) PRIMARY KEY，**

**部门名 CHAR(20)，**

**电话 CHAR(13)，**

**负责人代码 CHAR(5)，**

**任职时间 DATE，**

**FOREIGN KEY \_\_\_\_\_\_\_\_ ）；**

**查询比软件部所有职工月薪都要少的职工姓名及月薪的 SQL 语句如下：**

**SELECT 姓名, 月薪**

**FROM 职工**

**WHERE 月薪< ( SELECT \_\_\_\_\_\_\_\_**

**FROM 职工, 部门**

**WHERE 部门号 = \_\_\_\_\_\_\_\_ ) ;**

**则下面哪个答案是正确的（ ）。"**

1. **"(负责人代码) REFERENCES 职工(职工号)**

**MIN(月薪)**

**( SELECT 部门号 FROM 部门 WHERE 部门名 = '软件部' )"**

1. **"(部门号) REFERENCES 职工(部门号)**

**MAX(月薪)**

**部门.部门号 AND 部门名 = '软件部'"**

1. **"(部门号) REFERENCES 部门(部门号)**

**ALL(月薪)**

**职工.部门号 AND 部门.部门名 = '软件部'"**

1. **"(电话) REFERENCES 职工(电话）**

**月薪**

**职工.部门号 AND 部门名 = '软件部 。"**

1. **"有关系 S（ S ＃， SNAME ， SAGE ）， C（ C ＃， CNAME ）， SC（ S ＃， C ＃， GRADE ）。其中 S ＃是学生号， SNAME 是学生姓名， SAGE 是学生年龄， C ＃是课程号， CNAME 是课程名称。要查询选修 “ACCESS” 课的年龄不小于 20 的全体学生姓名的 SQL 语句是：**

**SELECT SNAME**

**FROM S ， C ， SC**

**WHERE 子句**

**这里的 WHERE 子句的内容是 （ ）。**

1. **" S.S# = SC.S# and C.C# = SC.C# and SAGE>=20 and CNAME='ACCESS'**
2. **SAGE in>=20 and CNAME in 'ACCESS'**
3. **S.S# = SC.S# and C.C# = SC.C# and SAGE in>=20 and CNAME in 'ACCESS'**
4. **SAGE>=20 and CNAME='ACCESS'**
5. **要实现记录的物理顺序与索引项次序一致，应选择的索引类型是（ ）。**
6. **聚簇索引**
7. **B+树索引**
8. **单一索引**
9. **HASH 索引**
10. **"给定供应关系 SPJ（供应商号，零件号，工程号，数量），查询至少供应了 3 项工程（包含 3 项）的供应商，输出其供应商号和供应零件数量的总和，并按供应商号降序 排列（ ）。**

**SELECT 供应商号，SUM（数量）FROM SPJ**

**（1）**

**（2）**

**（3） ；**

**选择一项：**

1. **(1) GROUP BY 供应商号**

**(2) HAVING COUNT（DISTINCT（工程号））>2**

**(3) ORDER BY 供应商号 DESC "**

1. **"(1) ORDER BY 供应商号**

**(2) WHERE COUNT（工程号）> 2**

**(3) GROUP BY 供应商号 DESC "**

1. **"(1) GROUP BY 供应商号**

**(2) HAVING（DISTINCT（工程号））> 2**

**(3) ORDER BY 供应商号 "**

1. **(1) ORDER BY 供应商号**

**(2) WHERE 工程号> 2**

**(3) GROUP BY 供应商号"**

1. **允许取空值但不允许出现重复值的约束是（ ）。**
2. **UNIQUE**
3. **FOREIGN KEY**
4. **NULL**
5. **PRIMARY KEY**
6. **"在SQL 的查询语句中，允许出现聚集函数的是（ ）。**
7. **" SELECT 子句和 HAVING 短语**
8. **HAVING 短语**
9. **WHERE 子句**
10. **SELECT 子句**
11. **设关系数据库中一个表 S 的结构为： S（ SN ， CN ， grade ），其中 SN 为学生名，CN 为课程名，二者均为字符型；grade 为成绩，数值型，取值范围 0 ~ 100 。若要更正张明的化学成绩为85分，则可用（ ）。**
12. **UPDATE S SET grade＝85 WHERE SN＝'张明' AND CN＝'化学'**
13. **UPDATE S SET grade＝'85' WHERE SN＝'张明' AND CN＝'化学'**
14. **UPDATE grade＝'85' WHERE SN＝'张明' AND CN＝'化学'**
15. **UPDATE grade＝85 WHERE SN＝'张明' AND CN＝'化学'**
16. **在视图上不能完成的操作是（ ）。**
17. **在视图上定义新的表**
18. **更新视图**
19. **查询**
20. **在视图上定义新的视图**
21. **"在数据库系统中，数据完整性约束的建立需要通过数据库管理系统提供的数据（ ）语言来实现。**
22. **定义**
23. **操纵**
24. **控制**
25. **查询**
26. **SQL 语言中的视图 VIEW 是数据库的（ ）。**
27. **外模式**
28. **存储模式**
29. **模式**
30. **内模式**
31. **若要撤销数据库中已经存在的表 S ，可用（ ）。**
32. **DROP TABLE S**
33. **DELETE TABLE S**
34. **DROP S**
35. **DELETE S**
36. **通过重建视图能够实现（ ）。**
37. **数据的逻辑独立性**
38. **程序的逻辑独立性**
39. **数据的物理独立性**
40. **程序的物理独立性**