1、什么时候使用关系，什么时候使用表？

- 关系同表的区别

  在数学上，关系可以是无限的；

  表的属性没有次序，数学上关系列是有次序的

- 对关系进行限制

  令关系元组数目有限

  赋予关系的列属性名，取消属性之间的次序

对关系进行限制后：关系=表

- 一般情况下，“关系”指限制后的关系，即表

2、关系代数中，集合的并、交、叉需要关系满足什么要求？

要求关系相容，即

- 同元
- 对应列同域
- 不要求对应列同名

3、Πpid,name(S-T) 与 Πpid,name(S) - Πpid,name(T) 是否等价？

不等价，投影和差是不可以分配的

4、自然连接确定选择条件的原则是什么？

- 以属性名是否相同为依据
- 不以语义是否相同为依据
- 要注意参与自然连接的表中是否有不希望做选择条件的同名属性

5、对于关系模式 s（R），c（T），属性组 R∩T= Φ 时，自然连接的结果是什么？

s⋈ c = s X c

6、Null 和 null 是否相等阿？

Null 是不等于 null 的，但是在投影运算时，对于多个为空的值只显示一次

7、 From r1,r2… 表示的是笛卡儿积还是自然连接？

表示的是笛卡儿积，属性之间的连接需要在 where 子句描述

8、 From r1,r2… 中同名属性如何引用？

碰到同名的属性需要引用的时候，要指明所属关系名，如 table.column

9、 ‘济南市山大路’ like ‘济南市%’和‘济南市%’ like ‘济南市山大路’ 表达的结果一样吗？

不一样，Like 具有单向性，前者为真，后者为假

10、 如何匹配以’%’开头的字符串？

可以使用转义符 escape ，可以如下书写 like ‘\%%’ escape ‘\’

11、 分组聚集时，能返回哪些属性？

Select 子句中，可以出现分组属性、聚集函数， 不能出现非聚集的非分组 属性

12、 分组聚集时，having 表示的条件和 where 子句的条件有何不同？

Where 是在分组聚集 G 之前进行的σ运算

having 是在分组聚集 G 之外进行的σ运算

13、 sql 语句中各子句的执行顺序是什么？

如下所示，前边的标号表示执行顺序：

⑤select s.sno,sname,avg(score)

① from s,sc

② where s.sno=sc.sno

and score>=60

③ group by s.sno,sname

④ having count(*)>=3

⑥ order by s.sno;

14、 “全部” 的概念在 sql 中的有哪些书写方法?

- 超集 superset:

  not exists (B except A)

- 关系代数: ÷

  not in(not in)

- 关系演算: " ∀

  not exists(not exists)

15、 集合比较中, =some(…)是什么含义, <>some(…)呢?

= some(…)表示 in, <>some(…) 没有含义

16、 集合比较中, =all(…)是什么含义, <>all(…)呢?

=all(…)没有含义, <>all(…)表示 not in

17、 在 sql 语句中, Unique 的含义是什么

unique 判定是否有重复元组, 不是判定是否只有一个元组, unique(select distinct…)的表现始终为真

18、 视图(view) vs 表(table) ?

- 视图和表都是关系, 都可在 sql 中直接应用
- DB 中存储表的模式定义和数据
- DB 中只存储视图的定义, 不存视图的数据
- 视图数据在使用视图时临时计算
- 物化视图是提高计算的一种手段, 结果等价于临时计算

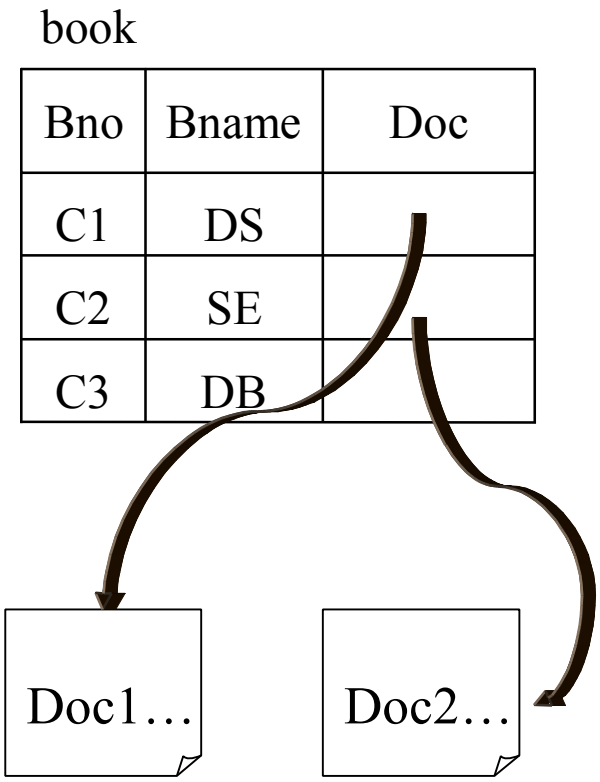
19、 数据库中如何存储大对象值？

- LOB
  - Large OBject
  - 用于存储大空间的值
- LOB 存储实现
  - 指针 ＋ 文件
- LOB 访问
  - 一般使用专用语句访问
  - Oracle:

    SelectBlob doc into …

    from book

    where bno=‘c1’;



20、 对参照关系操作需进行的参照完整性检查有哪些？

例：s 通过外码 dno 参照 dept(dno)

- delete (from s)

不会破坏参照完整性，不进行检查；

- insert (into s)

需检查保证新插入的元组：1)外码为空，

或 2)等于被参照关系某个元组主码的值；

- update (s.dno)

外码 oldvalue 不检查；

外码 newvalue 检查同 insert；

21、 对被参照关系操作需进行的参照完整性检查有哪些？

- insert (into dept)

不会破坏参照完整性，不进行检查；

- delete
  - 当参照关系中，无元组外码等于要删除元组主码值时：
    没有问题，如：删除(D3,体)，(s3,丙)
  - 当参照关系中，有元组外码等于要删除元组主码值时：
    DBMS 拒绝删除(默认) 或 级联删除 或外码置空:
  - 选择何种处理方式，是模式定义的一部分，一般建议选择默认方式(拒绝)
- update 被参照关系，不需要对 newvalue 检查
  - 当参照关系中，无元组外码等于 oldvalue 时
    没有问题，
  - 当参照关系中，有元组外码等于 oldvalue 时：
    DBMS 拒绝删除(默认) 或 级联删除 或外码置空:
  - 选择何种处理方式，是模式定义的一部分，一般建议选择默认方式(拒绝)

22、 用户和角色的关系是什么？

- 连接使用数据库，必须以一个特定的用户身份
- 不能使用 role 身份直接连接使用数据库
- 一个用户可同时具备多个角色
- 一个角色可以为多个用户拥有
- 角色可以拥有角色
- 角色拥有关系可以传递
- 权限可以授予用户或者角色
- 用户拥有授给本用户的所有权利，以及本用户具有的角色的所有权利

23、 多候选码时，主码的选择有什么指导原则？

- 选极少变化的属性、选习惯的属性；
- 主码一旦选择，整个组织有效

- 实体参与不同联系时，应使用相同主码

24、 E-R 图中，全部参与 vs 部分参与？

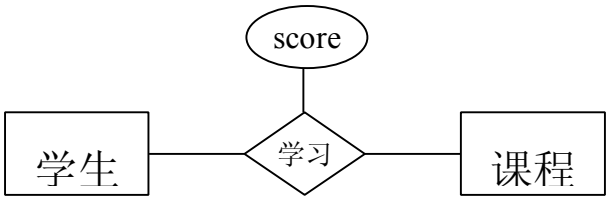
- 如果每个实体，至少参与某联系集的一各联系中，称全部参与
- 否则，称作部分参与
- 全部参与是联系的一种约束
- 部分参与不是联系的约束
- 全部参与使用双线段表示

25、 E-R 图中，实体 vs 联系？

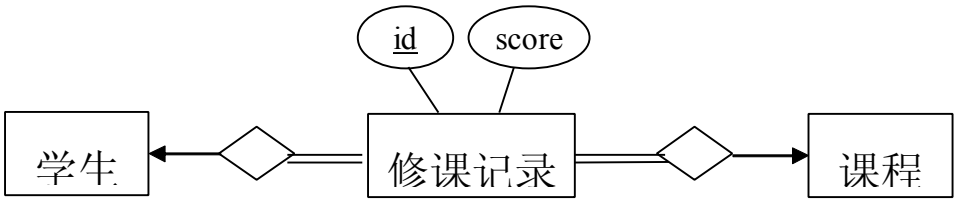
- 使用联系：

  很多情况下，实体和联系都可以接受；此时能用联系不用实体，以图简化 E-R；

  ①使用联系：



  ②使用实体：



- 用联系不足以清晰表述时，要用实体

26、 大模式的缺点有哪些？

- 数据冗余大
- 容易出现数据不一致
- 不能表示某些信息：没有学生的院系
- 插入异常

- 删除异常

之所以出现上述问题，要思考问题的本质，是因为

- 两件事不应使用一个表来表示
- 应该将这个模式分解为两个模式
- 分解必须按照一定的策略，无序分解不可接受

该大则大，该小则小，设计的结果应尽可能真实的反映现实。

27、 模式规范化的基本思路是什么？

- 从泛关系出发
- 通过模式分解得到好的(规范的)模式
- 模式分解：R → {R1,R2,…}

评价模式分解的两个重要指标是：

- 无损连接分解/有损连接分解；
- 是否保持依赖

28、 可以接受的分解必须是无损分解吗？

- 泛关系存在问题，因此进行模式分解；
- 如果分解有损，即分解后的模式不能有效表示泛关系原来含有的全部信息，那么分解没有价值，有损分解是不可实际接受的

29、 与传统的关系模型相比，对象关系模型有哪些扩充？

- 在定义语言上有三个扩充：

  数据类型的扩充（引入复合类型）；

  在类型一级和表一级实现继承性；

  使用“引用类型”。
- 在查询语言方面，用户需记住属性值是单值还是多值。在多值时，需定义新的元组变量。

30、 什么是“饿死”问题？如何解决？

有可能存在一个事务序列，其中每个事务都申请对某数据项加 S 锁，且每个事务在授权加锁后一小段时内释放封锁，此时若另有一个事务 T1 欲在该数据项上加 X 锁，则将永远轮不上封锁的机会。这种现象称为“饿死”（starvation）。

可以用下列方式授权加锁来避免事务饿死。

当事务 T2 中请对数据项 Q 加 S 锁时，授权加锁的条件是：

- 不存在在数据项 Q 上持有 X 锁的其他事务；
- 不存在等待对数据项 Q 加锁且先于 T2 申请加锁的事务。

31、 与集中式 DBMS 比较，DDBMS 环境中在并发控制和恢复方面遇到哪些新问题？

与集中式 DBMS 比较，DDBMS 环境中在并发控制和恢复方面会遇到以下五个问题：

- 数据项的多拷贝之间的一致性问题。
- 在单个场地故障恢复时，局部数据库的数据应和其他场地的同步问题。
- 通信网络的故障处理能力问题。
- 分布式提交的实现问题。
- 分时式死锁的处理问题。

32、 为什么稳定性存储器是不可能实现的？在 DBS 中采用什么方法追求这个目标？

由于存储器受制于本身的故障，会导致信息的丢失，因此稳定性存储器是不可能实现的。但可以通过对非易失性存储器进行技术处理来达到稳定性存储器的目标。主要是通过数据备份和数据银行形式实现。

33、 DBS 中有哪些类型的故障？哪些故障破坏了数据库？哪些故障未破坏数据库，但使其中某些数据变得不正确？

DBS 中 DB 的故障主要有三类：事务故障、系统故障和介质故障。前两类故障未破坏 DB，但使其中某些数据变得不正确，此时只要利用日志撤消或重做事务。介质故障将破坏 DB，此时只能把 DB 备份拷贝到新的磁盘，再利用日志重做事务对 DB 的修改。

34、 死锁的发生是坏事还是好事？试说明理由。如何解除死锁状态？

在 DBS 运行时，死锁状态是我们不希望发生的，因此死锁的发生本身是一件坏事。但是坏事可以转换为好事。如果我们不让死锁发生，让事务任意并发做下去，那么有可能破坏 DB 中数据，或用户读了错误的数据。从这个意义上讲，死锁的发生是一件好事，能防止错误的发生。

在发生死锁后，系统的死锁处理机制和恢复程序就能起作用，抽取某个事务作为牺牲品，把它撤消，做 ROLLBACK 操作，使系统有可能摆脱死锁状态，继续运行下去。

35、 试叙述“串行调度”与“可串行化调度”的区别。

如果多个事务依次执行，则称事务串行调度。

如果利用分时的方法，同时处理多个事务，则称为事务的并发调度。如果一个并发调度的结果与某一串行调度执行结果等价，则称这个并发调度是可串行化调度。