SQL 的组成	数据查询	
	数据定义语言（DDL）	**CREATE**：创建数据库或数据库对象
		ALTER：对数据库或数据库对象进行修改
		DROP：删除数据库或数据库对象
	数据操纵语言（DML）	**SELECT**：从表或视图中检索数据
		INSERT：将数据插入到表或视图中
		UPDATE：修改表或视图中的数据
		DELETE：从表或视图中删除数据
	数据控制语言（DCL）	**GRANT**：授予权限
		REVOKE：收回权限

SQL 主要包含	**数据查询**	
	数据定义语言（DDL）	CREATE、ALTER、DROP
	数据操纵语言（DML）	SELECT、INSERT、UPDATE、DELETE
	数据控制语言（DCL）	GRANT（授予）、REVOKE(收回)
	嵌入式和动态 SQL 规则	规定了 SQL 语句在**高级程序设计语言**中使用的**规范方法**，以便适应较为复杂的应用。
	SQL 调用和会话规则	SQL 调用:**包括 SQL 例程和调用规则**,以便提高 SQL 的灵活性、有效性、共享性以及使 SQL 具有更多的高级语言的特征
		SQL 会话规则:可使应用程序连接到**多个 SQL 服务器中的某一个**，并与之交互。

SQL 调用和会话规则	**SQL 调用**	**包括 SQL 例程和调用规则**，以便提高 SQL 的灵活性、有效性、共享性以及使 SQL 具有更多的高级语言的特征
	SQL 会话规则	可使应用程序连接到多个 SQL 服务器中的某一个，并与之交互。

表子查询	子查询返回的结果集是一个表
行子查询	子查询返回的结果集是带有一个或多个值的一行数据
列子查询	子查询返回的结果集是一列数据 该列可以有一行或多行，但每行只有一个值
标量子查询	子查询返回的结果集仅仅是一个值

SELECT 语句中各子句的使用次序及说明	
子句	说明
SELECT	要返回的**列或表达式**
FROM	从中检索数据的**表**
WHERE	指定数据的过滤条件（**行级过滤**）
GROUP BY	**分组**说明
HAVING	**过滤分组（组级过滤）**
ORDER BY	输出**排序**顺序
LIMIT	限制要检索的**行数**

使用视图的优点

1）集中分散数据；
2）简化查询语句；
3）重用 SQL 语句；
4）保护数据安全；
5）共享所需数据；
6）更改数据格式。

使用DROP VIEW语句来删除视图，语法格式如下：
DROP VIEW[IF EXISTS]
view_name[,view_name]…
[RESTRICT|CASCADE]
使用DROP VIEW语句可以一次删除多个视图，但必须在每个视图上拥有DROP权限。同时，为防止因删除不存在的视图而出错，需要在DROP VIEW语句中添加关键字“IF EXISTS”。

存储函数和存储过程的区别		
	存储函数	**存储过程**
输出参数	**不能**拥有输出参数	**可以拥有**输出参数
调用	可以**直接**对存储函数进行调用	需要使用**CALL 语句**对存储过程调用
RETURN 语句	**必须包含**一条 RETURN 语句	**不允许包含** RETURN 语句

在MySQL中，可以使用 CREATE FUNCTION 语句创建存储函数，其常用的语法格式是：

```
CREATEFUNCTION
sp_name([func_parameter[，…]])
RETURNS type
routine_body
```

其中，语法项“func_parameter”的语法格式是：

```
param_name type
```

在此语法格式中：

（1）语法项“sp_name”用于指定存储函数的名称，需注意，存储函数不能与存储过程具有相同的名字。

（2）语法项“func_parameter”用于指定存储函数的参数，这里的参数只有名称和类型，不能指定关键字“IN”“OUT”和“INOUT”。

（3）RETURNS子句用于声明存储函数返回值的数据类型，其中type用于指定返回值的数据类型。

（4）语法项“routine_body”用于指定存储函数的主体部分，也称为存储函数体。所有在存储过程中使用的SQL语句在存储函数中同样也适用，包括前面所介绍的局部变量、SET语句，流程控制语句、游标等。但是，存储函数体中还必须包含一个RETURN value语句，其中value用于指定存储函数的返回值。

数据库的安全保护	**完整性控制**	完整性指数据库中数据的**正确性**和**相容性**。
	安全性控制	安全性指保护数据库以**防止**不合法的使用而造成**数据泄露、更改或破坏**。
	并发控制	**事务**就是为保证数据的一**致性**而产生的一个概念和基本手段。
	数据库的备份与恢复	保证数据库中**数据的可靠性和完整性**。

完整性约束条件	**列**级约束	对数据类型的约束（包括数据类型、长度、精度等）
		对数据格式的约束
		对取值范围的约束
		对空值的约束
	元组约束	指元组中各个字段之间的相互约束。
	表级约束	指若干元组之间、关系之间的联系的约束。

关系模型的完整性约束	**实体**完整性	主键约束	PRIMARY KEY
		候选键约束	UNIQUE
	参照完整性	外键声明	FOREIGN KEY
	用户定义的完整性	非空约束	NOT NULL
		CHECK 约束	CHECK
		触发器	TRIGGER

关系模型的完整性规则是对关系的某种约束条件，关系模型中可以有三类完整性约束，分别是实体完整性、参照完整性和用户定义的完整性。

破题点：本题可从“实体完整性”入手。
实体完整性约束：
（1）**主键约束**：一个表中只能创建一个。PRIMARY KEY索引
（2）**候选键约束**：一个表中可定义若干个。UNIQUE索引。

触发器是用户定义在关系表上的一类由事件驱动的数据库对象，也是一种保证数据完整性的方法。触发器一旦定义，无须用户调用，任何对表的修改操作均为数据库服务器自动激活相应的触发器。
触发器与表的关系十分密切，其主要作用是实现主键和外键不能保证的复杂的参照完整性和数据的一致性，从而有效地保护表中的数据。

解析

在MySQL中，可以使用 CREATE TRIGGER 语句创建触发器，其常用的语法格式是：
CREATE TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_body
（1）语法项“trigger_name”用于指定触发器的名称，触发器在当前数据库中必须具有唯一的名称。如果要在某个特定数据库中创建，名称前面应该加上数据库的名称。故A对。
（2）语法项“trigger_time”用于指定触发器被触发的时刻，它有两个选项，即关键字“BEFORE”和关键字“AFTER”，用于表示触发器是在激活它的语句之前或之后触发。如果希望验证新数据是否满足使用的限制，则使用BEFORE选项；如果希望在激活触发器的语句执行之后完成几个或更多的改变，通常使用AFTER选项。故B对。
（3）语法项“trigger_event”用于指定触发事件，即指定激活触发器的语句的种类，其可以是下述值之一：关键字“INSERT”，表示将新的数据行插入到表时激活触发器；关键字“UPDATE”，表示更改表中某一行数据时激活触发器；关键字“DELETE”，表示从表中删除某一行数据时激活触发器。故C对。
（4）语法项“tbl_name”用于指定与触发器相关联的表名，必须引用永久性表，不能将触发器与临时表或视图关联起来，且同一个表不能拥有两个具有相同触发时刻和事件的触发器。故D错。
（5）关键字“FOR EACH ROW”用于指定对于受触发事件影响的每一行都要激活触发器的动作。
（6）语法项“trigger_body”用于指定触发器动作主体。

下一题

关系模型中常用的关系操作包括2大部分：

关系操作	**查询**（Query）操作	5种基本操作：**选择、投影、并、差、笛卡尔积**
		可用基本操作来定义和导出的操作：**连接、除、交等**
	插入(Insert)、**删除**（Delete）、**修改**（Update）操作	

完整性约束	**实体**完整性约束	指关系的主属性，即主码的组成不能为空，也就是关系的主属性不能是空值 NULL。
	参照完整性约束	定义**外码和主码**之间的引用规则，它是对关系间引用数据的一种限制。 描述定义：若属性 F 是基本关系 R 的外码，它与基本关系 S 的主码 K 相对应，则对于 R 中每个元组在 F 上的值只允许两种可能，即要么取空值，要么等于 S 中某个元组的主码值。
	用户定义的完整性约束	针对某一应用环境的完整性约束条件，它反映了某一具体应用所涉及的数据应满足的要求。

函数依赖	**完全** 函数依赖	设 R 为任一给定关系，X、Y 为其属性集，若 X→Y，且对 **X 中的任何真子集 X'，都有 X'↛Y**，则称 Y 完全函数依赖于 X。
	部分 函数依赖	设 R 为任一给定关系，X、Y 为其属性集，若 X→Y，且 **X 中存在一个真子集 X' 满足 X'→Y**，则称 Y 部分函数依赖于 X。
	传递 函数依赖	设 R 为任一给定关系，X、Y、Z 为其不同属性子集，若 **X→Y，Y↛X，Y→Z**，则有 **X→Z**，称为 Z 传递函数依赖于 X。

第一范式 1NF	设 R 为任一给定关系，如果 R 中每个列与行的交点处的取值都是**不可再分**的基本元素，则 R 为第一范式。
第二范式 2NF	设 R 为任一给定关系，若 R 为 1NF，且其所有非主属性**都完全函数依赖**于候选关键字，则 R 为第二范式。
第三范式 3NF	设 R 为任一给定关系，若 R 为 2NF，且其每一个非主属性**都不传递函数依赖**于候选关键字，则 R 为第三范式。
BCNF	设 R 为任一给定关系，X、Y 为其属性集，F 为其函数依赖集，若 R 为 3NF，且其 F 中所有函数依赖 X→Y(Y 不属于 X)中的 **X 必包含候选关键字**，则 R 为 BCNF。

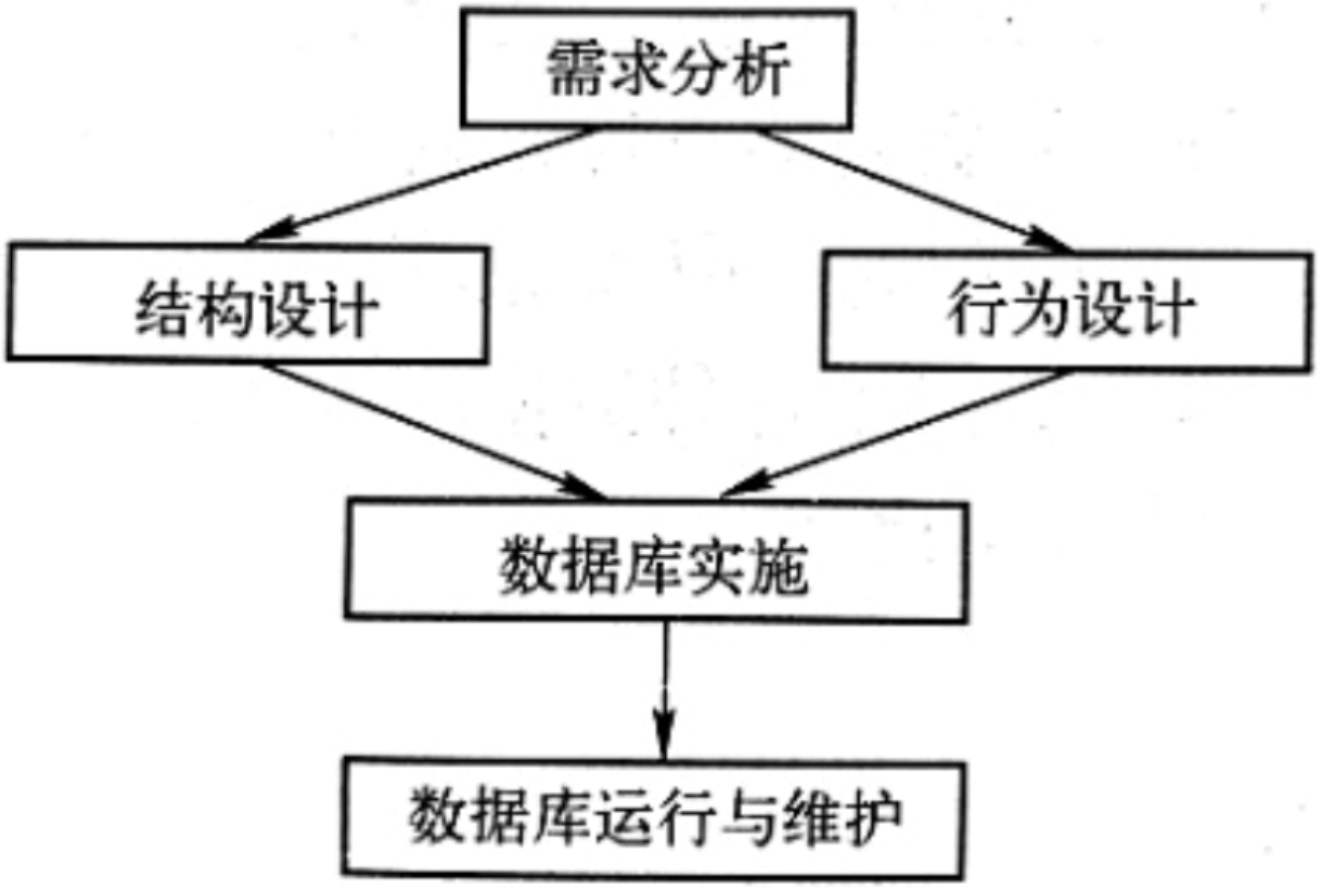


图 3.1　数据库设计的过程

收集与分析数据	**静态结构**	数据分类表	用于数据的总体描述。
		数据元素表	指通常意义下的数据项或属性。
	动态结构	**任务分类表**	根据对数据流程图的分析，可将业务处理过程划分成**不同任务**。
		数据操作特征表	用以描述**任务**和**数据**之间的关系，它包括不同任务对数据执行不同操作的频率。
	数据约束	数据的安全保密性	
		数据的完整性	
		响应时间	
		数据恢复	

数据库的生命周期	数据库**分析与设计**阶段	需求分析
		概念设计
		逻辑设计
		物理设计
	数据库**实现与操作**阶段	数据库的实现
		操作与监督
		修改与调整

数据库设计方法	描述	
直观设计法	是一类**最原始**的数据库设计方法。	
规范设计法	是一类较为**普遍、常用**的数据库设计方法。	**新奥尔良**设计方法
		基于**E-R模型**的数据库设计方法
		基于**第三范式**的设计方法
计算机辅助设计法	通常通过**人机交互**的方式来完成。	

收集与分析数据	**静态结构**	数据分类表
		数据元素表
	动态结构	任务分类表
		数据操作特征表
	数据约束	数据的安全保密性
		数据的完整性
		响应时间
		数据恢复

需求分析报告的内容	**1. 数据库的应用功能目标**	要求明确标明数据库的应用范围及应达到的应用处理功能。
	2. 标明不同用户视图范围	根据机构与职能关系图和数据流程图，并参考任何分类表等，确定不同部门或功能的局部视图范围。
	3. 应用处理过程需求说明	数据流程图；任务分类表；数据操作特征表；操作过程说明书。
	4. 数据字典	数据库系统中**存储三级结构**定义的数据库，通常指的是数据库系统中**各类数据详细描述的集合**。
	5. 数据量	根据数据分类表中的静态数据量和操作特征表中的动态数据量，进行统计计算，求出数据总量。
	6. 数据约束	数据的安全保密性、数据的完整性、响应时间和数据恢复。

在MySQL数据库中，数据库系统对数据的安全管理是使用_____1_____、数据库用户权限确认等访问控制措施。

第1空

1 ✗

试题解析

参考答案

身份验证

可以使用RENAME USER语句修改一个或多个已经存在的MySQL用户账号，其使用的语法格式是：

RENAME USER old_user TO new_user[,old_user TO new_user]…

在RENAME USER语句的使用中，需要注意以下几点。

（1） RENAME USER语句用于对原有MySQL账户进行重命名。

（2）要使用RENAME USER语句，必须拥有MySQL中的mysql数据库的UPDATE权限或全局CREATE USER权限。

（3）倘若系统中旧账户不存在或者新账户已存在，则语句执行会出现错误。

为了删除一个或多个用户账号以及相关的权限，可以使用DROP USER语句，在DROP USER语句的使用中，需要注意以下几点：

（1） DROP USER语句可用于删除一个或多个MySQL账户，并消除其权限。

（2）要使用DROP USER语句，必须拥有MySQL中mysql数据库的DELETE权限或全局CREATE USER权限。

（3）在DROP USER语句的使用中，如果没有明确地给出账户的主机名，则该主机名会默认为是%。

（4）用户的删除不会影响到他们之前所创建的表、索引或其他数据库对象，这是因为MySQL并没有记录是谁创建了这些对象。

可以使用 CREATE USER 语句来创建一个或多个 MySQL 账户，并设置相应的口令，其常用的语法格式是：

CREATE USER user[IDENTIFIED BY [PASSWORD]‘password’]

语法格式介绍如下：

（1）语法项“user”指定创建用户账号，其格式为'user_name'@'host name'。其中，user_name表示用户名，host_name表示主机名，即用户连接MySQL时所在主机的名字。如果在创建的过程中，只给出了账户中的用户名，而没指定主机名，则主机名会默认为是“%”，其表示一组主机。

（2）语法项“IDENTIFIED BY子句”是可选项，用于指定用户账号对应的口令，若该用户账号无口令，则可省略此子句。

（3）关键字“PASSWORD”是可选项，用于指定散列口令，即若使用明文设置口令时，而忽略PASSWORD关键字；如果不想以明文设置口令，且知道PASSWORD()函数返回给密码的散列值，则可以在此口令设置语句中指定此散列值，但需要加上关键字PASSWORD。

（4）语法项“password”指定用户账号的口令，其在IDENTIFIED BY关键字或PASSWORD关键字之后。设定的口令值可以是只有字母和数字组成的明文，也可以是通过PASSWORD()函数得到散列值。

在CREATE USER语句的使用中，需要注意以下几点。

（1）要使用CREATE USER语句，必须拥有MySQL中mysql数据库的INSERT权限或全局CREATE USER权限。

（2）使用CREATE USER语句创建一个用户账户后，会在系统自身的mysql数据库的user表中添加一条新记录。如果创建的账户已经存在，则语句执行会出现错误。

（3）如果两个用户具有相同的用户和不同的主机名，MySQL会将他们视为不同的用户，并允许为这两个用户分配不同的权限集合。

（4）如果在CREATE USER语句的使用中，没有为用户指定口令，那么MySQL允许该用户可以不使用口令登录系统，然而从安全的角度而言，不推荐这种做法。

（5）新创建的用户拥有的权限很少。他们可以登录到MySQL，只允许进行不需要权限的操作，比如使用SHOW语句查询所有存储引擎和字符集的列表等，不能使用USE语句来让其他用户已经创建了的任何数据库成为当前数据库，因而无法访问那些数据库的表。

用户账号管理	创建用户账号	CREATE USER
	删除用户	DROP USER
	修改用户账号	RENAME USER
	修改用户口令	SET PASSWORD

可以使用 CREATE USER 语句来创建一个或多个 MySQL 账户，并设置相应的口令，其常用的语法格式是：

CREATE USER user[IDENTIFIED BY [PASSWORD]‘password’]

关键字“PASSWORD”是可选项，用于指定散列口令，即若使用明文设置口令时，需忽略PASSWORD关键字；如果不想以明文设置口令，且知道PASSWORD()函数返回给密码的散列值，则可以在此口令设置语句中指定此散列值，但需要加上关键字PASSWORD。

MySQL 的用户账号	**数据库名**	**mysql**
	表名	**user**
	列名	**user**
	拥有 MySQL 的**所有权限**的用户	**root**

新建的MySQL用户必须被授权，可以使用GRANT语句来实现，常用的语法格式是：

```
GRANT
Priv_type[(column_list)]
[,priv_type[(column_list)]]…
ON [object_type]priv_level
TO user_specification[,user_specification]…
[WITH GRANT OPTION]
```

在此语法中：

（1）语法项“priv_type”用于指定权限的名称。例如 SELECT、UPDATE、DELETE等数据库操作。

（2）可选语法项“column_list”用于指定权限要授予给表中哪些具体的列。

（3）语法项“ON 子句”用于指定权限授予的对象和级别。

（4）可选项“object_type”用于指定权限授予的对象类型，包括表、函数和存储过程，分别用关键字“TABLE”“FUNCTION”和“PROCEDURE”标识。

（5）语法项“priv_level”:用于指定权限的级别，其可以授予的权限有这样几个：列权限、表权限、数据库权限和用户权限。

（6）语法项“TO子句”用来设定用户的口令，以及指定被授予权限的用户user。若在TO 子句中给系统中存在的用户指定口令，则新密码会将原密码覆盖；如果权限被授予给一个不存在的用户，MySQL会自行执行一条CREATE USER语句来创建这个用户，但同时必须为该用户指定口令。由此可见，GRANT语句亦可以用于创建用户账号。

（7）语法项“user_specification”是TO子句中的具体描述部分，其常用的语法格式是：

```
user[IDENTIFIED BY[PASSWORD]’password’]
```

（8）语法项“WITH子句”为可选项，用于实现权限的转移或限制。

权限的转移可以通过GRANT语句中使用WITH子句来实现。如果将WITH子句指定为关键字“WITH GRANT OPTION”，则表示TO子句中所指定的所有用户都具有把自己所拥有的权限授予给其他用户的权利，而无论那些其他用户是否拥有该权限

权限的级别	**列**权限	SELECT、INSERT 和 UPDATE
	表权限	SELECT、INSERT、DELETE、UPDATE、REFERENCES、CREATE、ALTER、INDEX、DROP、ALL
	数据库权限	除了可以指定为授予表权限时的所有值之外，还可以是下面这些值：CREATE TEMPORARY TABLES、CREATE VIEW、SHOW VIEW、CREATE ROUTINE、ALTER ROUTINE、EXECUTE ROUTINE、LOOK TABLES。
	用户权限（**最有效率**）	除了可以指定为授予数据库权限时的所有值之外，还可以是下面这些值：CREATE USER、SHOW DATABASES。

注意：“不包括”，故本题选B。

解析

本题考查权限的转移。

账户权限管理	查看用户授权表	SHOW GRANTS FOR
	授予	GRANT
	转移	在 GRANT 中添加子句：WITH GRANT OPTION
	撤销	REVOKE

GRANT用于指定权限的名称，即查询。
ON用于指定要授予权限的数据库名或表名，即关系S。
TO用来设定用户的口令，以及指定被授予权限的用户，即WANG。
WITH GRANT OPTION表示TO子句中所指定的所有用户都具有把自己所拥有的权限授予给其他用户的权利，而无论那些其他用户是否拥有该权限。
故本题选D。

解析

新建的MySQL用户必须被授权，可以使用GRANT语句来实现，常用的语法格式是：

```
GRANT
   priv_type[(column_list)]
[,priv_type[(column_list)]]…
   ON [object_type]priv_level
   TO user_specification[,user_specification]…
   [WITH GRANT OPTION]
```

在此语法中：

（1）语法项“**priv_type**”用于指定**权限的名称**。

（2）可选语法项“column_list”用于指定权限要授予给表中哪些具体的列。

（3）语法项“**ON 子句**”用于指定权限授予的对象和级别。

（4）可选项“**object_type**”用于指定权限**授予的对象类型**。

（5）语法项“**priv_level**”:用于指定**权限的级别**。

故本题选C。

破题点：本题可从“授权”入手。

账户**权限**管理	查看用户授权表	SHOW GRANTS FOR
	授予	GRANT
	转移	在 GRANT 中添加子句：WITH GRANT OPTION
	撤销	REVOKE

解析

显式定义事务的语句	标记事务的**开始**	BEGIN TRANSACTION
	标记事务的**结束**	COMMIT：提交（正常结束）
		ROLLBACK：回滚（发生故障）

故不包括B。

参考答案

事务与程序很相似，但它们是两个彼此相联而又不同的概念：程序是静止的，事务是动态的，是程序的执行而不是程序本身；同一程序的多个独立执行可以同时进行，每一步执行则是一个不同的事务。

解析

为了保证数据的一致性和正确性，数据库系统必须保证事务具有四个特征，即原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持续性（Durability），这四个特征也简称为事务的ACID特征。

事务的特征	描述
原子性	事务是**不可分割**的最小工作单位，所包含的这些操作是一个整体。
一致性	事务必须满足数据库的**完整性约束**，且事务执行完毕后将数据库由一个一致性状态转变到另一个一致性状态。
隔离性	事务是彼此**独立的、隔离的**，即一个事务的执行**不能被**其他事务所**干扰**。
持续性	也称为永久性，是指一个事务一旦提交，它对数据库中数据的改变就应该是**永久性**的，且接下来的其他操作或故障对其执行结果**无影响**。

解析

并发操作问题	**丢失更新**	设有两个事务 T1 和 T2,当它们同时读入同一数据并加以修改时，**事务 T2 的提交结果会破坏事务 T1 提交的结果**，由此导致事务 T1 的修改被丢失。
	不可重复读	事务 T1 读取某一数据后，事务 T2 执行更新操作，当事务 T1 **再次读取数据时，得到与前一次不同的值**。
	读“脏”数据	事务 T1 **修改**某一数据，并将其写回磁盘，事务 T2 读取同一数据后，事务 T1 由于某种原因**被撤销**，这时事务 T1 已修改过的数据恢复原值，事务 T2 读到的数据就与数据库中的数据**不一致**，则事务 T2 读到的数据就为“脏”数据，即不正确的数据。

所以在DB技术中，“脏数据”是指未提交随后又被撤消的数据。故选D。

解析

以粒度来描述封锁的数据单元的大小。DBMS可以决定不同粒度的锁。由最底层的数据元素到最高层的整个数据库，粒度越细，并发性就越大，但软件复杂性和系统开销也就越大。锁住整个数据库，DBMS的管理和控制最简单，只需设置和测试一个锁，故系统开销也最小。选择粒度过大或过小，系统的总体性能都会下降，所以，大多数高性能系统都选择折中的锁粒度，至于哪一层最合适，则与应用环境下事务量、数据量及数据的易变性特征等都紧密相关。

解析

破题点：本题可从“防止不可重读”入手。

封锁的级别	描述
0级封锁	指封锁的事务不重写其他非0级封锁事务的未提交的更新数据。该状态实用价值不大。
1级封锁	指被封锁的事务不允许重写未提交的更新数据。这**防止丢失更新**的发生。
2级封锁	指被封锁的事务既不能重写也不读未提交的更新数据。这除了1级封锁的效果外还**防止了读脏数据**。
3级封锁	指被封锁的事务不读未提交的更新数据，不写任何（包括读操作的）未提交数据，**防止了不可重读**的问题。这是严格的封锁，它保证了多个事务并发执行的“可串行化”。

故本题选D。

解析

封锁带来的一个重要问题是可能引起**“活锁”**与**“死锁”**。	在并发事务处理过程中，由于锁会使一事务处于等待状态而调度其他事务处理，因而该事务可能会因优先级低而永远等待下去，这种现象称为**“活锁”**。
	两个以上事务循环等待被同组中另一事务锁住的数据单元的情形，称为**“死锁”**。

解析

基本的封锁类型	**排他锁** （Exclusive Lock，**X 锁**）	写操作
	共享锁 （Shared Lock，**S 锁**）	读操作

故本题选B。

解析

封锁的工作原理	(1) 若事务 T 对数据 D 加了 **X 锁**，则**所有别的事务**对数据 D 的锁请求**都必须等待**直到事务 T 释放锁。
	(2) 若事务 T 对数据 D 加了 **S 锁**，则别的事务**还可**对数据 D **请求 S 锁**，而对数据 D 的 **X 锁请求必须等待**直到事务 T 释放锁。
	(3) 事务执行数据库操作时都要先请求相应的锁，即对**读**请求 **S 锁**，对**更新**（插入、删除、修改）请求 **X 锁**。
	(4) 事务一直**占有**获得的锁直到结束（COMMIT 或 ROLLBACK）时释放。

排他锁即X锁，故所有别的事务的锁请求都必须等待，即事务T2不能再给数据A加任何锁，所以选B。

解析

破题点：本题可从“活锁”入手。

问题的解决	**活锁**	一种最简单的办法是“**先来先服务**”
	死锁	一次性锁请求 锁请求排序 序列化处理 资源剥夺

故本题选B。

解析

DBMS需要提供死锁预防、死锁检测和死锁发生后的处理技术与方法。
预防死锁的办法在操作系统中已普遍讨论，其中主要有如下几种：
（1）一次性锁请求
（2）锁请求排序
（3）序列化处理
（4）资源剥夺
死锁检测可以用图论的方法实现，并以正在执行的事务为结点。

解析

问题的解决	**活锁**	一种最简单的办法是“先来先服务”
	预防死锁	**一次性**锁请求 锁请求**排序** **序列化**处理 **资源剥夺**

解析

在MySQL中使用SQL语句备份与恢复数据库中表数据的方法，即使用SELECT INTO…OUTFILE语句备份数据，使用LOAD DATA…INFILE语句恢复数据的方法中，有一点不足，就是只能导出或导入数据的内容，而不包括表的结构，如果表的结构文件损坏，则必须先设法恢复原来表的结构。

备份数据： SELECT INTO… OUTFILE	FIELDS 子句	TERMINATED BY	指定字段值之间的符号
		[OPTIONALLY] ENCLOSED BY	指定包裹文件中字符值的符号
		ESCAPED BY	指定转义字符
	LINES 子句	TERMINATED BY	指定一个数据行结束的标志
	DUMPFILE，而非“OUTFILE”时	导出的备份文件里面所有的数据行都会彼此紧挨着放置，即**值和行之间没有任何标记**。	

故本题选D，正确描述是：用关键字“DUMPFILE”表示导出的备份文件里面所有的值和行之间没有任何标记。

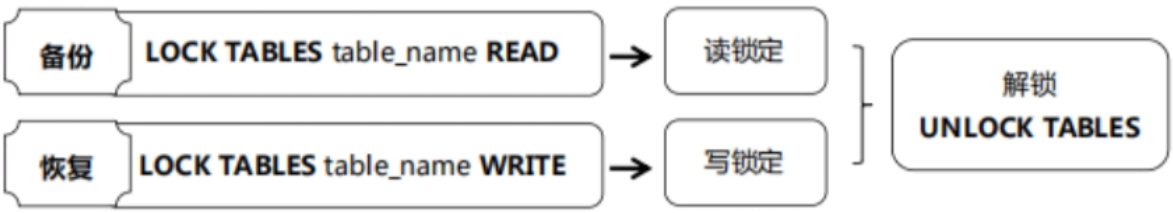
解析

恢复数据： LOAD DATA … INFILE	FIELDS 子句	用于判断字段之间和数据行之间的符号。	
	LINES 子句	TERMINATED BY	指定一行结束的标志
		STARTING BY	指定一个**前缀**，导入数据行时，**忽略**数据行中的**该前缀和前缀之前的内容**。如果某行不包括该前缀，则整个数据行被跳过。

故本题选D。正确描述是：忽略数据行中的该前缀和前缀之前的内容。

解析

破题点：本题可从“解锁”入手。
在多个用户同时使用 MySQL 数据库的情况下，为了数据的一致性，需要在指定的表上使用锁定：



故本题选C。

解析

数据库应用软件的设计与开发过程：

（1）**需求分析**

（2）系统功能与数据库的设计

（3）系统功能与数据库的实现

（3）测试与维护等

可助记为：分析→设计→实现→维护

解析

需要注意的是：在数据库设计时，实体的描述信息可根据实际需求进行增加或删减，如果实体的属性较多，在构建E-R模型时不一定需要把所有的属性都标识在E-R模型上，可以另外用文字说明，这样也使得E-R模型简明清晰，便于分析。

解析

1970年，IBM公司San Jose 研究室的研究员 E.F.Codd 在其发表的论文中提出了数据库的关系模型，开创了数据库关系方法和关系数据理论的研究，为关系数据库技术奠定了理论基础。

20世纪70年代是关系数据库理论研究和原型开发的时代，其主要成果有：

（1）奠定了关系模型的理论基础，给出了人们一致接受的关系模型的规范说明。

（2）研究了关系数据语言，有关系代数、关系演算、SQL语言及QBE等。

（3）研制了大量的RDBMS的原型，攻克了系统实现中查询优化、并发控制、故障恢复等一系列关键技术。

支持数据管理、对象管理和知识管理是第三代数据库系统的基本特征。

技术	数据库系统
与**分布式**处理技术相结合	**分布式**数据库系统
与**并行**处理技术相结合	**并行**数据库系统
与**人工智能**技术相结合	**演绎**数据库、**知识库**和**主动**数据库系统等
与**多媒体**技术相结合	**多媒体**数据库系统
与**模糊**技术相结合	**模糊**数据库系统
与**移动**通信技术相结合	**移动**数据库系统
与**Web**技术相结合	**Web**数据库

助记方式：人演蜘蛛（人-演知主）。只有与人工智能技术相结合出现的数据库系统名称与技术名称无关，其余都与技术名称相关。故本题选B。

数据集市的基本思想是自下而上的数据仓库的开发方法。一般可以将数据集市分为独立的数据集市和从属的数据集市或这两种数据集市的混合。

计算机系统中存在着两类不同的数据处理工作：一类是操作型处理，也称联机事务处理，另一类是分析型处理，也称联机分析处理，一般针对某些主题的历史数据进行分析，支持管理决策，它通常是对海量的历史数据查询和分析。

数据挖掘的功能	描述
概念描述	通过数据挖掘技术，可以**归纳总结**出数据的某些特征。
关联分析	其目的是找出数据库中隐藏的**关联网**。
分类与预测	分类找出一**个类别的概念描述**，它代表了这类数据的整体信息。
聚类	把数据按照**相似性归纳**成若干**类别**。
孤立点检测	指数据中与整体表现行为**不一致**的数据集合。
趋势和演变分析	通过数据挖掘技术，可以描述行为随着时间变化的对象所遵循的**规律或趋势**。

NoSQL系统为了提高存储能力和并发读写能力采用了极其简单的数据模型，支持简单的查询操作，而将负责操作留给应用层实现。该系统对数据进行划分，对各个数据区进行备份，以应对结点可能的失败，提高系统可用性；通过大量结点的并行处理获得高性能，采用的是横向扩展的方式。它弥补了传统数据库由于事务等机制而带来的对海量数据高并发请求处理性能上的欠缺，采用一种非关系的方式来解决大数据存储和管理的问题。

MapReduce技术是Google公司于2004年提出的大规模并行计算解决方案，主要应用于大规模廉价集群上的大数据并行处理。MapReduce以Key/Value的分布式存储系统为基础，通过元数据集中存储、数据以chunk为单位分布存储和数据chunk冗余复制来保证其高可用性。

NoSQL系统支持的数据存储模型：

（1）**键值存储**：NoSQL数据库采用最多的数据存储方式，数据以Key-Value的形式存储。适合通过主键进行查询或遍历。

（2）**文档存储**：**不需要定义表结构，但可以像定义表结构一样使用**。适合存储系统日志等非结构化数据，可以通过复杂的查询条件来获取数据。

（3）**列存储**：以列为单位来存储数据，擅长以列为单位读入数据，比较适合对某一列进行随机查询处理。

（4）**图存储**：图存储数据库是基于图理论构建的，使用结点、属性和边的概念。

故本题选B。

各局部 E-R 图之间的冲突	**属性**冲突	属性**域**冲突
		属性**取值**单位冲突
	命名冲突	**同名异义**
		异名同义
	结构冲突	**同一对象**在一个局部 E-R 图中作为实体，而在另一个局部 E-R 图中作为属性。
		同一实体在不同的 E-R 图中属性个数和类型不同。
		实体之间的联系在不同的 E-R 图中是不同的类型。

数据库实施需要完成的工作包括：	**加载数据**
	应用程序**设计**
	数据库**试运行**

各局部E-R图之间的冲突主要表现在三个方面：

（1）**属性冲突**：属性域冲突、属性取值单位冲突；

（2）**命名冲突**：同名异义、异名同义；

（3）**结构冲突**：1）同一对象在一个局部E-R图中作为实体，而在另一个局部E-R图中作为属性。2）同一实体在不同的E-R图中属性个数和类型不同。3）实体之间的联系在不同的E-R图中是不同的类型。

重构是指**部分修改数据库的逻辑结构或物理结构**，这往往因应用需求的改变与拓展或发现当初的设计欠妥而引起的，例如增、删、改数据类型，增、 删、改索引与聚集等。

【拓展】

系统维护中最困难的工作是数据库重组与重构。

关系数据模型的优化通常以关系规范化理论为指导，其方法如下：

1、确定各属性间的函数依赖关系。

2、对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

3、判断每个关系模式的范式，根据实际需要确定最合适的范式。

4、按照需求分析阶段得到的处理要求，分析这些模式对于这样的应用环境是否合适，确定是否要对某些模式进行合并或分解。

5、对关系模式进行必要的分解，提高数据操作的效率和存储空间的利用率。

物理设计的任务	**建立索引**（通过 DBMS 提供的有关命令来实现的）	**静态建立索引**	指应用人员**预先**建立索引。 适合于**用户较多**且使用**周期相对较长**的数据。
		动态建立索引	指应用人员在程序内外**临时**建立索引。 适合于**单独用户**或**临时性**使用要求情况。
	建立聚集	聚集：将相关数据**集中存放**的物理存储技术。	

破题点：本题可从“集中存放”入手。
聚集是将相关数据**集中存放**的物理存储技术，借以提高I/O的数据命中率而改善存取速度，其功能由具体的DBMS所提供，如MySQL。故本题选B。
C、D是数据控制语言中包括的2个主要SQL语句的功能。故排除。

物理设计的任务主要是通过对关系建立索引和聚集来实现与应用相关数据的逻辑连接和物理聚集，以改善对数据库的存取效率。

年份	SQL 标准
1989 年	SQL-89
1992 年	SQL-92（或称为 SQL2）
1999	SQL-99（或称为 **SQL3**）

MySQL 函数的基本分类	举例
数学函数	例如 ABS()函数、SORT()函数
聚合函数	例如 COUNT()函数
字符串函数	例如 ASCII()函数、CHAR()函数
日期和时间函数	例如 NOW()函数、YEAR()函数
加密函数	例如 **ENCODE()函数**、ENCRYPT()函数
控制流程函数	例如 IF()函数、IFNULL()函数
格式化函数	例如 FORMAT()函数
类型转换函数	例如 CAST()函数
系统信息函数	例如 USER()函数、VERSION()函数

更新表 (ALTER TABLE) 的相关操作	ADD [COLUMN]	向表中**增加新列**
	CHANGE [COLUMN]	**修改**表中列的**名称或数据类型**
	MODIFY [COLUMN]	只**修改**指定列的**数据类型**，而不会干涉它的列名。还可以通过关键字“FIRST”或“AFTER”修改指定列在表中的**位置**
	ALTER [COLUMN]	**修改或删除**表中指定列的默认值
	DROP [COLUMN]	**删除**表中的列
	RENAME [TO]	为表重命名=**RENAME TABLE**

SQL语句中各个字句的功能如下所示：

（1）SELECT：指定输出的列或表达式（必需）

（2）FROM子句：用于指定数据的来源（必需）

（3）WHERE子句：用于指定数据的选择条件，行级过滤

（4）GROUP BY子句：用于对检索到的记录进行分组

（5）HAVING子句：用于指定组的选择条件，组级过滤

（6）ORDER BY子句：用于对查询的结果进行排序

（7）LIMIT子句：用于限制被SELECT语句返回的行数

在SELECT语句的使用中， 最简单的形式是“SELECT select_expr”。使用这种最简单的SELECT语句，可以进行数据系统所支持的任何运算。