# 1 数据库分类DBMS

层次模型

网状模型

关系模型

E-R实体关系模型

对象关系模型

结构化关系模型

半结构化关系模型

# 2 假设使用文件作为数据库的弊端

1. 数据冗余和不一致性
2. 数据访问困难
3. 数据孤立
4. 完整性问题
5. 事物的原子性问题（一个事务要么完全提交 要么完全回滚，不会介于2者之间。）
6. 并发访问异常问题
7. 安全性问题

# 3 约束类型

1. 域约束 数据类型约束（int）
2. 外键约束，引用完整性约束，一个表中的值必须在另一个相关联的表中存在
3. 主键约束，某个字段能唯一表示此字段的行（例如身份证号），**不能为空**
4. 唯一性约束，类似于主键，**但可以为空**
5. 检查性约束，如年龄必须在0-150之间。

# 4 RDB对象

## **·**库****

## ·事务：

详情请看***34-01和34-02***

**ACID测试**

原子性Atomicity：要么同时完成，要么同时完不成

一致性Consistency：状态和一致

隔离性Isolation：两个事务之间相互隔离，存在**隔离级别**，当两个事务都用 一张表时，可能会产生冲突，MySQL会提供许多策略进行事务调度，尽量规 避冲突。

READ-UNCOMMITTED：读未提交，事务1改动，事务2立刻能看见

READ-COMMITTED： 读提交，事务1提交后，事务2才能看见

REPATABLE-READ：可重读，到事务1提交前，第一次看见什么样，第二 次也是这样

SERIABLITABLE：串行

**SHOW GLOBAL VARIABLES LIKE ‘tx\_isolation’;**查看当前隔离级别

持久性Durability：事务完成后，即使服务器宕机，数据仍不会改变。

**事务的状态**

部分提交的

提交的

active活动的

失败的

终止的

## ·表

## ·索引，

所有的键都是索引。

## ·视图

首先视图可以将不同表中的数据通过关联实现多表数据输出，比如说，有一个类型表 Type，有字段id, Name，有个产品Product，有字段id, Name, Id，此时你可以通过建立一张视图Product将产品表和产品类型一起输出，**视图就是一个select语句**，第一次使用后将SELECT查询结果缓存下来，然后被当作一个特殊的表来对待，有时称之为**虚表**，而其所依赖的表被称为**基表**。

## ·账号

包括用户认证和授权（GRANT）***详情请看34-03用户管理及授权***

在MySQL中，账号仅仅用来认证，

**用户**：虚拟用户，仅仅用来验证身份，格式：username@host

用户名：一般为16个字符以内

主机名：www.example.com

172.168.1.2

172.168.0.0/255.255.0.0

**密码**：是通过password（）函数进行设置，

用户的权限则是通过另一套权限设置机制完成的，这套机制为grant-table，在 MySQL中用户的权限一般由以下几个表(都位于mysql库中)设置。

user：用户账号与全局权限的配置

db：库级别的权限

host：与user合并，已废弃，

tables\_priv：表级别的权限

columns\_priv：列级别的权限

procs\_priv：存储过程和存储函数相关的权限

proxies\_priv：代理用户权限

## ·存储函数、

## ·触发器、

## ·事件调度器

**event：**有点像linux中的crontab，可以精确到秒钟。

# 5 数据库查询和存储

存储管理器：

权限及完整性管理器

事务管理器

文件管理器

缓冲区管理器

查询管理器：

DML解释器

DDL解释器

查询执行引擎

# 6 数据库与文件的关系

数据库

表示层 ：RDB对象

逻辑层 ：存储引擎

物理层 : 文件

文件

表示层 ：文件

逻辑层 ：文件系统

物理层 ：block

# 7 关系运算

投影：只输出指定属性，也就是列

选择：只输出符合条件的行

自然连接：两个表之间通过某个字段取值相同进行连接

笛卡尔积：（a+b）\*(c+d)=ac+ad+bc+bd

并： 集合运算

# 8 表空间：table space

用于实现多个表的数据存放在一个文件的管理组件

在InooDB中每表一个表空间

# 9 数据字典：Data Dictionary 类似元数据

关系名称、字段名称、字段类型长度、视图、约束。。。。。。

# 10 SQL模型

SQL模型，The **sql\_mode** server variable can be set to change the default behavior of data validation and allowed SQL syntax，**它是一个服务器变量**。可以设置全局（global）也可以设置会话（per session）

**ANSI QUOTES** ：双引号和反引号作用相同，字符串只能被单引号引用

**IGNORF\_SPACE**：忽略多余的空白符

**STRICT\_ALL\_TABLES**：STRICT\_ALL\_TABLES—When this sql\_mode is not set, invalid data values are allowed, usually converted to valid data by using zero values or truncating the value. When this sql\_mode is set, invalid data values are rejected, and an error is thrown.

**STRICT\_TRANS\_TABLES**：invalid data values in a transactional table are rejected, and an error is thrown

**TRANITIONAL**：Syntax and behavior closely match traditional expectations

# 11 并发控制

**锁**：最简单的并发控制机制，根据锁类型分为读锁和写锁，根据锁粒度分为表锁/页锁 /行锁。MySQL服务器仅仅支持表锁，某些存储引擎支持行锁（如InnoDB）

**LOCK TABLES tb\_name lock\_type;**

**UNLOCK TABLES;**解除所有文件的锁。

**MVCC**：多版本并发控制，数据的快照。

12 MySQL服务器变量

按作用域可分为

全局变量（global）：和用户无关，全局有效

会话变量（per session）：和用户关联，只作用于当前用户。

按生效时间分两类

动态：立即生效

若调整变量为全局，则只对新建会话生效

若调整对象为会话，则只对当前会话生效。

静态：非立即生效

1 改变配置文件

2 通过参数传递给MySQL

查看全局变量：

1 **SHOW GLOBAL VARIABLES LIKE ’sql\_mode’;**

2 **SELECT @@global.sql\_mode;**

3 **SELECT Variable\_name, Variable\_value FROM INFORMATION\_SCHEMA.GLOBAL\_VARIABLES WHERE Variable\_name=’sql\_mode’;**

设置变量：

**SET user\_var\_name = expr**

**| [GLOBAL | SESSION] system\_var\_name = expr**

**| [@@global. | @@session. | @@]system\_var\_name = expr**