数据库个人理解

目录

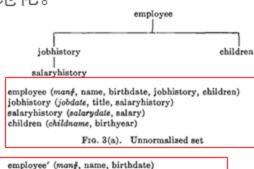
- 1.数据库设计
- 2.事务处理
- 3.锁
- 4. InnoDB

books

- 《深入浅出 MySQL》
 《MySQL 技术内幕: InnoDB存储引擎》
 《高性能 MySQL》
- 《数据库原理及应用(mysql)》

数据库设计

- 0.数据库设计三范式的来源(为什么要有范式):
- Some more complicated data structure in necessary for a relation with one or more nonsimple domains. For this
 reason the possibility of eliminating nonsimple domains appears worth investigating.
 - 与一个或多个非单域的关系需要一些更复杂的数据结构。出于这个原因、消除非简单域的可能性似乎值得研究。
- There is in fact, a very simple elimination procedure, which we shall call normalization.
 - 事实上,有一个非常简单的消除步骤,我们称之为规范化。
- 右图有个规范化的例子,可以看一下 →
- Ps: 斜体的是主键



employee' (manf, name, birthdate)
jobhistory' (manf, jobdate, title)
salaryhistory' (manf, jobdate, salarydate, salary)
children' (manf, childname, birthyear)

Fig. 3(b). Normalized set

有关于本文献的启示—非关系型数据库(nosql)

- Nonsimple domains
- 一行中的值不一定是一个像数字或字符串一样的原始数据类型,也可以是一个嵌套的关系(表),因此可以把一个任意嵌套的树结构作为一个值,这很像30年后添加到SQL中的JSON或XML支持。同时也是未来json存储非关系数据库的基石
- If the user's relational model is set up in normal form, names of items of data in the data bank can take a simpler form than would otherwise be the case. A general name would take a form such as R(g).r.d
- where R is a relational name; g is a generation identifier(optional); r is a role name(optional); d is a
 domain name.

E. F. Codd. 1970. A relational model of data for large shared data banks. Commun. ACM 13, 6 (June 1970), 377 - 387. https://doi.org/10.1145/362384.362685

数据库设计

- 1.数据库设计三范式
- · 数据库表的设计依据, 教我们怎么进行数据库表的设计。
- 设计数据库表的时候,按照以上的范式进行,可以避免表中数据的冗余,空间的浪费。
- 翻译一下就是: 怎么能够创建一张每个字段都不怎么浪费的表。

数据库设计三范式内容

第一范式 (1NF):

列1 列2 列3 列4 列5

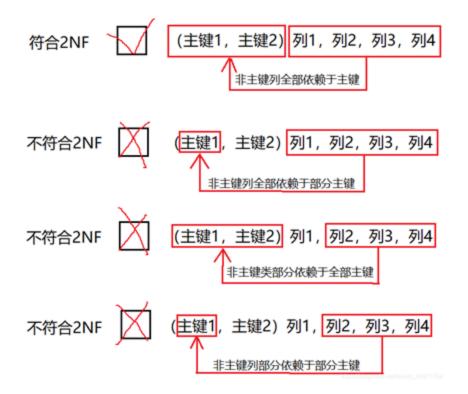
列1唯一确定列2, 列3, 列4, ..., 即列2, 列3, 列4, ...不能再分裂出其它列。

例子:

学号 (主键)	成绩	绩点
0	100	5
1	100	4
2	90	3

CREATE TABLE `test`.`XXX`(`学号`INT NOT NULL auto_increment, `成绩`INT NOT NULL, `绩点`INT NOT NULL, `绩点`INT NOT NULL, PRIMARY KEY(`学号`))
ENGINE=InnoDB DEFAULT CHARSET=utf8;

第二范式 (2NF) :



满足2NF的前提是必须满足1NF。

此外,关系模式需要包含两部分内容。

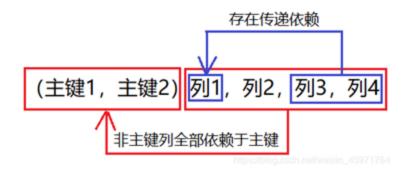
- 一是必须有一个(及以上)主键;
- 二是没有包含在主键中的列必须全部依赖 于全部主键,而不能只依赖于主键的一部 分而不依赖全部主键。

学号(主键)	成绩	绩点
0	100	5
1	100	4
2	90	3

第三范式 (3NF):

满足3NF的前提是必须满足2NF。

另外关系模式的非主键列必须直接依赖于主键,不能 存在传递依赖。



学号 (主键)	成绩	学习科目数量
0	100	5
1	100	4
2	90	3

但是…

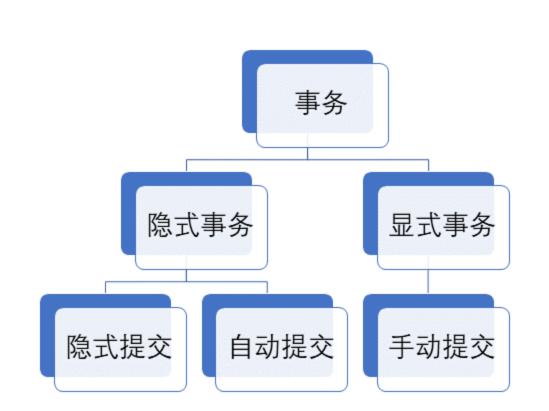
- 请注意: 这里所说的所有有关于"数据库范式"的内容均为一种"规范", 或者说是"通法", 而不是一板一眼的死板学识。
- 请设计数据库时遵循业务逻辑! 不要为了遵循范式而强行遵循。
- 业务逻辑: 老板让你建立一个表记录的内容类型(以老板的话为准)

事务处理

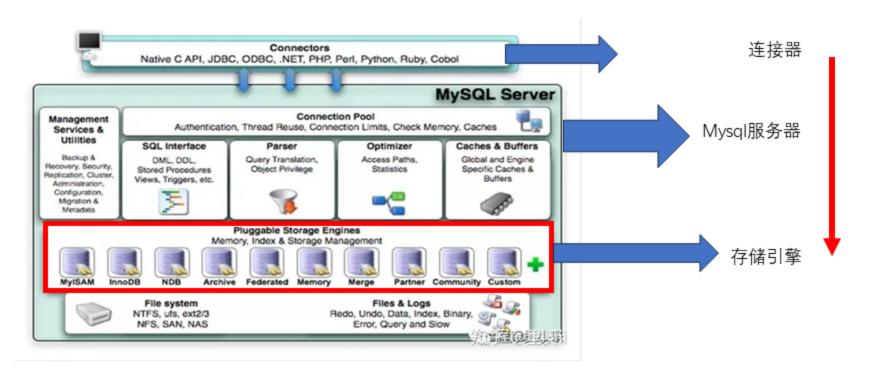
- 事务,是一批对数据库的操作;可以理解为批处理文件(windows 的.bat/linux的.shell)
- 于此同时,事务由于mysql的单进程和多线程,也可以理解为在一个进程中的一个线程。
- 事务处理分为显式事务和隐式事务。

- 事务语法如下:
- BEGIN;
- · 你要执行的sql语句;
- COMMIT;

显式事务:用commit/rollback的 隐式事务:又称自动提交事务,意 思是在你写完一个事务后不 commit/rollback, sql server会自 动帮你提交

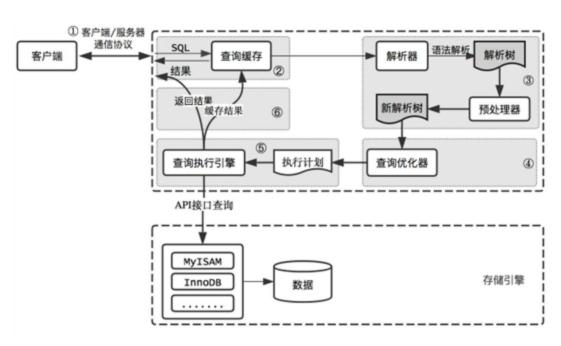


事务处理-知识之mysql运行原理



Mysql服务器---what happening?

• 当向MySQL发送一个请求的时候,MySQL到底做了些什么呢?



TCP/IP

```
in [5]: Import socket
      mysel_port = 3306
      # DOE-TROPHSON
      client_socket = socket.socket(socket.AF_IMET, socket.SOCK_STREAF)
     try:
         client socket.connect((mysel host, mysel port))
         password * 'eeeeee
         database . 'test'
         # FORTHWAY TO JUST #
         login_request = (
             username.encode('utf-8') + b'\x00' +
            5"(x00" * 21 * # /E/E
             password.encode('utf-8') * b'\x00' *
            database.encode('utf-8') * b'\u00'
         client_socket.sendall(login_request)
         # 提收并打印服务等的电应
         response * client_socket.recv(1824)
         version bytes a response(4:4)
         plugin_name_bytes * response[-len(b'caching_shak_password'):]
         version * version bytes.decode("utf-8")
         plugin name * plugin name bytes_decode("utf-8")
         print(f"N(A-V) (version)")
         print(f"hummid HH: (plugin_name)")
      except Exception as or
         print(f"%%%@@; (e)")
      finally:
        # XM082
         cliest_socket.close()
      response_length = int.from_bytes(header, byteorder='little')
      * DEBES
      data_packet = response(4:)
      · DOWNKERBEINGE
      print(f"WEEEEE; (response length)")
      print(f"BBSAW: (data_packet)")
     RETT
     him jii fi fili aching shat password
     paratword'usbb"
```

zyb123nya/How-it-works-MySQL: about MySQL8.0 running. (github.com)

这个东西其实已经由各大 connectors(MySQL:: MySQL Connectors)实现了,这里只是复现一 下原理。

缓存查询

- Mysql8.0已经废除;但是我们可以分析其逻辑:
- 1. 将你的select语句进行hash运算,得到hash值。
- 2.存入cache,等待mysql server响应
- 3.mysql server启动,打开cache,接受select语句,通过计算hash 值进行匹配。

• 但我们可以搞一个简易版本(已经上传至github)

缓存查询与索引

缓存查询与redis

解析器

查询优化器

执行计划

执行计划与事务的区别

- 执行计划: 描述sql语句在数据库的执行过程
- 事务处理: 一组sql操作
- 好比一个是史官, 一个是工人。

存储引擎 (innodb为例)

锁

- 概念:字面意思,对数据加一个锁。
- 如果把表/库/页当成一道门,那么锁就是限制人员(事务提交)进出的一个工具。