目录 /CONTENTS

01

优化设计

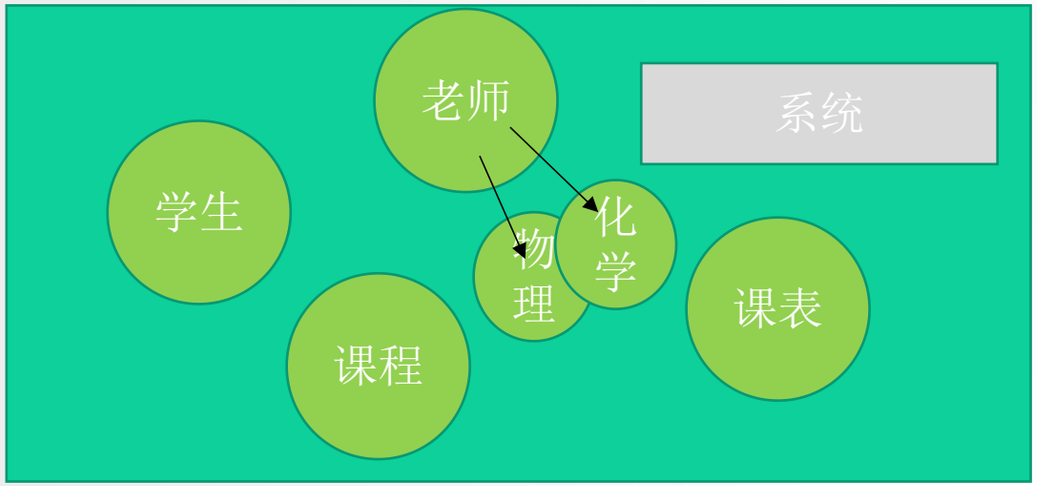
BASIC CONCEPTS

数据库设计

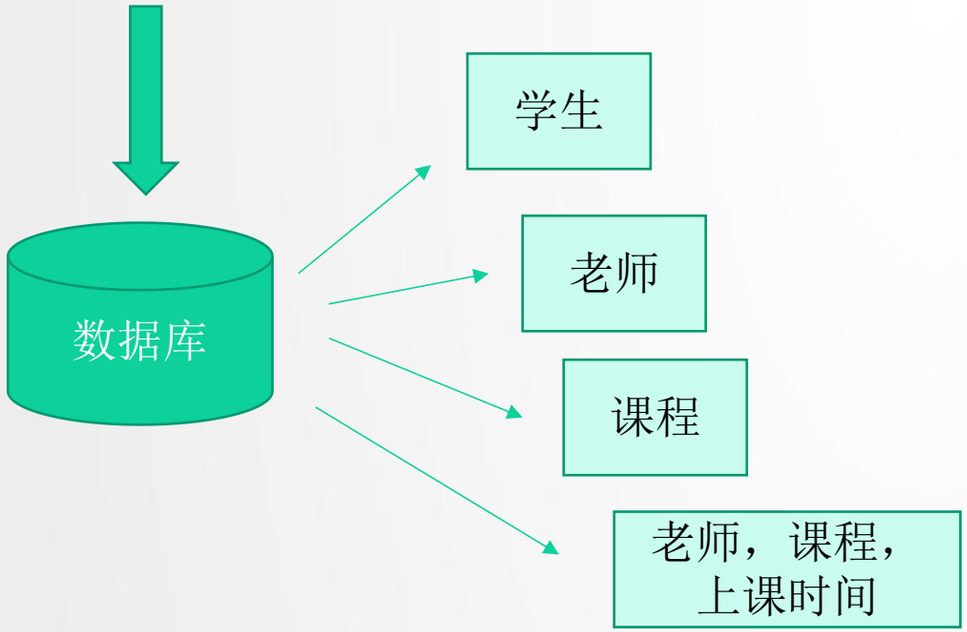
数据库设计，一个软件项目成功的基石。数据库设计也是门学问。

在项目早期由开发者进行数据库设计（后期调优需要DBA）。一个精通OOP和ORM的开发者，设计的数据库往往更为合理，更能适应需求的变化。因为数据库的规范化，与OO的部分思想雷同（如内聚）。而DBA，设计的数据库的优势是能将DBMS的能力发挥到极致，能够使用SQL和DBMS实现很多程序实现的逻辑，与开发者相比，DBA优化过的数据库更为高效和稳定。

数据库设计与程序设计的差异



面向对象设计思路
封装
多态
IOC
AOP…



面向数据（信息）设计思路
关注存储
关注效率
二维表关系
关注完整性

数据库设计早期优化

不要把它仅仅当成一个存储的功能

1、关系明确

2、节省空间

3、提高效率

02

设计原则

SCOPE

数据库种类

- 层级数据库（注册表）
- 关系型数据库
- 时序数据库
- 图数据库
- Key-value数据库
- 对象数据库
- BigTable数据库
- …

效率与空间

文件系统和数据库系统之间的区别。
(1)文件系统用文件将数据长期保存在外存上，数据库系统用数据库统一存储数据;
(2)文件系统中的程序和数据有一定的联系，数据库系统中的程序和数据分离;
(3)文件系统用操作系统中的存取方法对数据进行管理，数据库系统用DBMS统一管理和控制数据;
(4)文件系统实现以文件为单位的数据共享，数据库系统实现以记录和字段为单位的数据共享。

文件系统和数据库系统之间的联系：
(1)均为数据组织的管理技术;
(2)均由数据管理软件管理数据，程序与数据之间用存取方法进行转换;
(3)数据库系统是在文件系统的基础上发展而来的。

精通数据类型

类型	空间	格式
Tinying	1字节	-128~127
Smallint	2字节	-32768~32767
Mediumint	3字节	
Int	4字节	-2147483648~2147483647
bigint	8字节	-9223372036854775808~9233372036854775807
Float	4字节	+-1.175494351E-38
Double	8字节	+-2.2250738585072014E-308
Decimal	8，2	8+2
		Unsigned指无符号数

类型	空间	格式
Varchar		0~65535
Char		0~255
Text		2^16-1字节
Blob		2^16-1字节
Year	1字节	
Date	3字节	YYYY-MM-DD
Time	3字节	HH:MM：ss
datetime		YYYY-MM-DD HH:MM：ss
timestamp	4字节	秒数，最大到2037年

优化设计第二步

范式1NF，2NF，3NF。。。

1NF：**列不可分**。每一列都是不可分割的基本数据项

2NF：1NF的基础上面，非主属性完全依赖于主关键字

3NF：属性不依赖于其它非主属性，消除传递依赖

BCNF：符合3NF，每个表中只有一个候选键

4NF：没有多值依赖

1NF：某列：姓名=（Robert,罗伯特）

2NF：ID，学号，姓名，帐号，年级

3NF：课程表：学员ID，学号，年级

优化设计第三步

要有一些逼数

1、选择小的数据类型

2、单独设计主键，并考虑分布式扩展

3、外键设计

4、索引设计

5、关联关系表设计，多对一，多对多

6、读写频繁的信息，与不频繁的信息分开

7、配置表，日志表，定时任务表等

8、汇总表设计

优化设计第四步

要有一些套路

1、通用型设计
例：人员，部门，角色

2、特别设计
附件，日志，配置，监控等

3、存储设计
类型划分便于分区

4、一些附加字段
创建日期，修改日期，排序

5、流水表
类似于日志，但由业务处理结果组成，帐户变动或业务处理的中间值

6、。。。

Codd的RDBMS12法则

Edgar Frank Codd（埃德加·弗兰克·科德）被誉为“关系数据库之父”，并因为在数据库管理系统的理论和实践方面的杰出贡献于1981年获图灵奖。在1985年，Codd 博士发布了12条规则，这些规则简明的定义出一个关系型数据库的理念，它们被作为所有关系数据库系统的设计指导性方针。

1.信息法则 关系数据库中的所有信息都用唯一的一种方式表示——表中的值。

2.保证访问法则 依靠表名、主键值和列名的组合，保证能访问每个数据项。

3.空值的系统化处理 支持空值（NULL），以系统化的方式处理空值，空值不依赖于数据类型。

4.基于关系模型的动态联机目录 数据库的描述应该是自描述的，在逻辑级别上和普通数据采用同样的表示方式，即数据库必须含有描述该数据库结构的系统表或者数据库描述信息应该包含在用户可以访问的表中。

5.统一的数据子语言法则 一个关系数据库系统可以支持几种语言和多种终端使用方式，但必须至少有一种语言，它的语句能够一某种定义良好的语法表示为字符串，并能全面地支持以下所有规则：数据定义、视图定义、数据操作、约束、授权以及事务。（这种语言就是SQL)

6.视图更新法则 所有理论上可以更新的视图也可以由系统更新。

7.高级的插入、更新和删除操作 把一个基础关系或派生关系作为单个操作对象处理的能力不仅适应于数据的检索，还适用于数据的插入、修改个删除，即在插入、修改和删除操作中数据行被视作集合。

8.数据的物理独立性 不管数据库的数据在存储表示或访问方式上怎么变化，应用程序和终端活动都保持着逻辑上的不变性。

9.数据的逻辑独立性 当对表做了理论上不会损害信息的改变时，应用程序和终端活动都会保持逻辑上的不变性。

10.数据完整性的独立性 专用于某个关系型数据库的完整性约束必须可以用关系数据库子语言定义，而且可以存储在数据目录中，而非程序中。

11.分布独立性 不管数据在物理是否分布式存储，或者任何时候改变分布策略，RDBMS的数据操纵子语言必须能使应用程序和终端活动保持逻辑上的不变性。

12.非破坏性法则 如果一个关系数据库系统支持某种低级（一次处理单个记录）语言，那么这个低级语言不能违反或绕过更高级语言（一次处理多个记录）规定的完整性法则或约束，即用户不能以任何方式违反数据库的约束。

落实这些原则

（一）降低对数据库功能的依赖

（二）定义实体关系的原则

牵涉到的实体 识别出关系所涉及的所有实体。

所有权 考虑一个实体“拥有”另一个实体的情况。

基数 考量一个实体的实例和另一个实体实例关联的数量。

（三）列意味着唯一的值

如果表示坐标（0,0），应该使用两列表示，而不是将“0,0”放在1个列中。

（四）列的顺序，可读性问题

（五）定义主键和外键

数据表必须定义主键和外键（如果有外键）。

（六）选择键

（七）是否允许NULL

任何值和NULL拼接后都为NULL。

所有与NULL进行的数学操作都返回NULL。

引入NULL后，逻辑不易处理。

落实这些原则

（九）规范化——范式

1NF

包含分隔符类字符的字符串数据。

名字尾端有数字的属性。

没有定义键或键定义不好的表。

2NF

多个属性有同样的前缀。

重复的数据组。

汇总的数据，所引用的数据在一个完全不同的实体中。

BCNF- “每个键必须唯一标识实体，每个非键熟悉必须描述实体。”

4NF

三元关系（实体:实体:实体）。

潜伏的多值属性。（如多个手机号。）

临时数据或历史值。（需要将历史数据的主体提出，否则将存在大量冗余。）

（十）选择数据类型

（十一）优化并行

设计DB时就应该考虑到对并行进行优化，比如，timestamp类型。

命名规则

表名规则
1、要用前缀，但不要用无意义的前缀
2、下划线分隔
3、全小写

列名规则
1、一般不用前缀
2、下划线分隔
3、全小写

03

设计案例

CASE

键设计

物理主键，好建索引，消除传递依赖

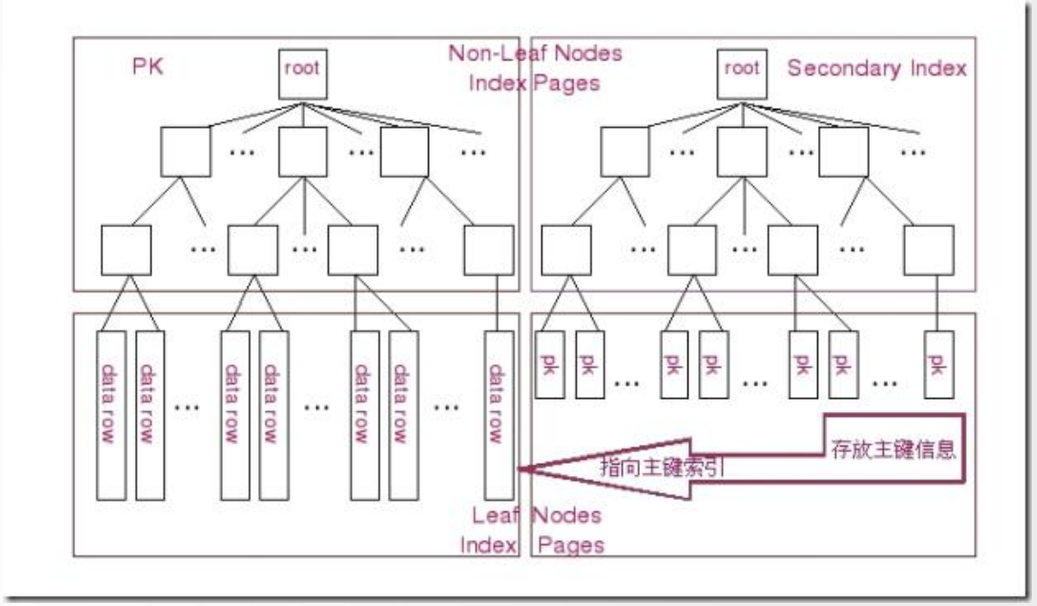
主键类型，普通系统是int或bigint，效率问题

Uuid，容量问题，防碰撞

取消所有的联合主键（课程表中：年级+课程名）

索引设计

B-Tree 索引是 MySQL 数据库中使用最为频繁的索引类型，除了 Archive 存储引擎之外的其他所有的存储引擎都支持 B-Tree 索引。不仅仅在 MySQL 中是如此，实际上在其他的很多数据库管理系统中B-Tree 索引也同样是作为最主要的索引类型，这主要是因为 B-Tree 索引的存储结构在数据库的数据检
索中有非常优异的表现。



Hash 索引结构的特殊性，其检索效率非常高，索引的检索可以一次定位，不像B-Tree 索引需要从根节点到枝节点，最后才能访问到页节点这样多次的IO访问，所以 Hash 索引的查询效率要远高于 B-Tree 索引。

Hash值：1111，记录1，记录2，记录3
。。。
根据字段计算hash值，放在一个格子里
。。。
Hash值：2222，记录4，记录5，记录6

索引设计

普通索引：最基本的索引，没有任何限制
唯一索引：与"普通索引"类似，不同的就是：索引列的值必须唯一，但允许有空值。
主键索引：它是一种特殊的唯一索引，不允许有空值。
全文索引：仅可用于 MyISAM 表，针对较大的数据，生成全文索引很耗时好空间。
组合索引：为了更多的提高mysql效率可建立组合索引，遵循”最左前缀“原则。

覆盖索引(Covering Indexes)

索引设计

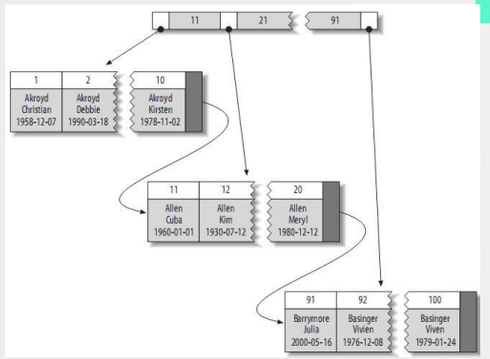
聚簇索引(Clustered Indexes)

聚簇索引保证关键字的值相近的元组存储的物理位置也相同（所以字符串类型不宜建立聚簇索引，特别是随机字符串，会使得系统进行大量的移动操作），且一个表只能有一个聚簇索引。因为由存储引擎实现索引，所以，并不是所有的引擎都支持聚簇索引。目前，只有solidDB和InnoDB支持。

非聚簇索引

二级索引叶子节点保存的不是指行的物理位置的指针，而是行的主键值。这意味着通过二级索引查找行。

InnoDB对主键建立聚簇索引。如果你不指定主键，InnoDB会用一个具有唯一且非空值的索引来代替。如果不存在这样的索引，InnoDB会定义一个隐藏的主键，然后对其建立聚簇索引。一般来说，DBMS都会以聚簇索引的形式来存储实际的数据，它是其它二级索引的基础。



表附加字段设计

常用：
创建时间：create_at
修改时间：update_at
排序：sn

有时用：
说明：desc，remark
备选字段：opts_1，opts_2….

字典表设计

字典表与系统配置表的区别

字典表示例
常用数据的常量化，消费类型，支付类型，物品类型（含层级）

系统配置表示例
各模块功能参数的常量化，key-value

```
CREATE TABLE dict_content
(
  dcc_id bigint NOT NULL,
  dct_id bigint,
  dcc_no character varying(20),
  dcc_name character varying(30),
  extend_no character varying(50),
  remarks character varying(400),
  sort bigint,
  parent_id bigint,
  is_leaf smallint,
  path character varying(400),
  CONSTRAINT dict_content_pkey PRIMARY KEY (dcc_id)
)
```

```
CREATE TABLE sys_config_data
(
  id bigint NOT NULL,
  create_date timestamp without time zone NOT NULL,
  modify_date timestamp without time zone NOT NULL,
  catalog character varying(255),
  content character varying(255),
  description character varying(255),
  sn character varying(255),
  CONSTRAINT sys_config_data_pkey PRIMARY KEY (id)
)
```

附件表设计

存储位置
类型
用途
使用次数
下载次数

#	名字	类型	排序规则	属性	空	默认	注释	额外
1	**fid**	int(11)		UNSIGNED	否	*无*		AUTO_INCREMENT
2	**fuid**	int(11)			是	*NULL*		
3	**ffilename**	varchar(255)	utf8_general_ci		否			
4	**furi**	varchar(255)	utf8_bin		否			
5	**ffilemime**	varchar(255)	utf8_general_ci		否			
6	**ffilesize**	bigint(20)		UNSIGNED	否	0		
7	**fstatus**	tinyint(4)			否	0		
8	**fcreate_time**	datetime			是	*NULL*		
9	**ftype**	varchar(50)	utf8_general_ci		否			
10	**version**	int(11)			是	0		

#	名字	类型	排序规则	属性	空	默认	注释	额外
1	**fid**	int(11)		UNSIGNED	否	*无*		AUTO_INCREMENT
2	**fother_id**	int(11)			是	*NULL*		
3	**fother_table**	varchar(255)	utf8_general_ci		是	*NULL*		
4	**fcategory**	varchar(255)	utf8_general_ci		否			
5	**fusage**	varchar(64)	utf8_general_ci		否			
6	**ffid**	int(11)		UNSIGNED	否	0		
7	**fcount**	int(11)		UNSIGNED	否	0		
8	**fstatus**	int(11)		UNSIGNED	否	0		
9	**fcreate_time**	datetime			是	*NULL*		
10	**version**	int(11)			是	0		

层级结构表设计

```
CREATE TABLE public.ihp_boq_tpl
(
  boq_id bigint NOT NULL, -- 清单id
  boq_no character varying(32), -- 清单编号
  ext_no character varying(32), -- 扩展编号
  boq_name character varying(64), -- 清单名称
  unit_id bigint, -- 计量单位
  boq_kind smallint, -- 清单类型
  boq_mode smallint, -- 清单模式
  boq_rate numeric(18,2), -- 计量率
  formula character varying(32), -- 计算公式
  use_formula smallint, -- 是否启用公式
  parent_id bigint, -- 父节点
  path character varying(128), -- 路径
  level smallint, -- 层次
  end_node smallint, -- 最终节点
  remarks character varying(256), -- 备注
  status smallint, -- 状态
  feature character varying(128),
  frequency character varying(32),
  quota_unit_id bigint,
  quota_tbl_no character varying(8),
  project_id bigint,
  CONSTRAINT pk_ihp_boq_tpl PRIMARY KEY (boq_id)
)
```

父子层级，parent_id

表内划分层级

表外划分层级

最快检索设计

ula	parent_id bigint	path character varying(128)	level smallint	end_node smallint	r c
		0,	1	0	
	130163	0,130163,	2	0	
	130165	0,130163,130165,	3	0	
	130167	0,130163,130165,130167,	4	1	
	130167	0,130163,130165,130167,	4	1	
	130163	0,130163,	2	0	
	130173	0,130163,130173,	3	1	
	130173	0,130163,130173,	3	1	
	130173	0,130163,130173,	3	1	
	130173	0,130163,130173,	3	1	
	130163	0,130163,	2	0	
	130183	0,130163,130183,	3	1	
	130183	0,130163,130183,	3	1	
	130183	0,130163,130183,	3	0	
	130189	0,130163,130183,130189,	4	1	
	130183	0,130163,130183,	3	1	
	130163	0,130163,	2	0	
	130195	0,130163,130195,	3	1	
	130195	0,130163,130195,	3	0	
	130199	0,130163,130195,130199,	4	1	

流程表设计

流程主表

任务子表

业务表关联

```
CREATE TABLE process_run
(
  runid bigint NOT NULL,
  subject character varying(256) NOT NULL,
  creator character varying(128),
  userid bigint NOT NULL,
  defid bigint NOT NULL,
  piid character varying(64),
  createtime timestamp without time zone NOT NULL,
  runstatus bigint NOT NULL,
  busdesc character varying(1024),
  entityname character varying(128),
  entityid bigint,
  formdefid bigint,
  CONSTRAINT pk_process_run PRIMARY KEY (runid)
)
```

```
CREATE TABLE process_form
(
  formid bigint NOT NULL,
  runid bigint NOT NULL,
  activityname character varying(256) NOT NULL,
  createtime timestamp without time zone NOT NULL,
  endtime timestamp without time zone,
  durtimes bigint,
  creatorid bigint,
  creatorname character varying(256),
  taskid character varying(64),
  status bigint DEFAULT 0,
  preformid bigint,
  comments character varying(2000),
  entity_id bigint,
  entity_name character varying(128),
  CONSTRAINT pk_process_form PRIMARY KEY (formid)
)
```

04

冗余设计

SERVER PROCESSING METHOD

反范式

适当冗余

1、借鉴覆盖索引的思路，在表内直接放常用的字段
2、提取相关数据，制作汇总表
3、第3NF的违反
4、程序级别的冗余或缓存
5、不要写触发器，不要写存储过程

#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
1	fid	int(11)			否	无		AUTO_INCREMENT	
2	faid	int(11)			否	0			
3	fuid	int(11)			否	0			
4	fuid2	int(11)			否	0			
5	fwallet_qty	double			是	0			
6	faward_qty	double			是	0			
7	frelease_ratio	double			是	0			
8	fstatus	int(11)			是	0			
9	fcreate_time	datetime			是	NULL			
10	version	int(11)			是	0			
11	fuid3	double			是	0			
12	faward_qty_1	double			是	0			
13	faward_qty_2	double			是	0			
14	fentrust_sum_qty	double			是	0			
15	fupdate_time	datetime			是	NULL			
16	fratio_1	double			是	0			
17	fratio_2	double			是	0			
18	faward_total	double			是	0			
19	fratio_0	double			是	0			
20	faward_frozen	double			是	0			

全选 选中项: 浏览 修改 删除 主键 唯一 索

分表分库

•性能：能轻松面对海量数据和高并发的请求处理，好的分布式数据库能做到90%以上的线性增长能力；
•灵活性、弹性：现代的系统的业务和使用场景变化很快，用户的增长也有很多不确定因素。弹性扩容就非常重要。分布式数据库本身有Cloud-Ready的特性，能很容以通过添加设备扩容满足需求，而不需要影响开发；
•多中心、多活：这点在大型应用中很常见，分布式数据库就更容易实现这个功能，当然这里涉及到分布式数据库的同步和一致性的能力，这也是判断分布式数据库好坏的一个重要指标。
•读写分离：主从节点都能发挥作用；例如巨杉SequoiaDB数据库，能在一组三副本的复制组上实现OLTP，NoSQL应用，OLAP多种应用场景同时使用。
•低成本：x86服务器，SATA存储（部分可以用SSD），加上较好的网络带宽就可以了。

1.水平分区：基本对1个或多个键的水平哈希，增强数据的并行处理能力来提高性能；
2.垂直分区：和过去的Partition很像，对数据进行有含义的拆分；
3.混合分区：水平分区和垂直分区共同使用。