珠联璧合—PowerBuilder与数据库配合开发技术

- PowerBuilder与数据库配合开发技术
- 数据库实体关系模型设计方法
- 如何用S-Designor设计数据库

北京市晓通网络数据库研究所

珠联璧合－PowerBuilder与数据库配合开发技术

作者 侯志平 徐常胜

北京晓通网络数据库研究所

前 言

用 PowerBuilder 开发应用系统确实是一件令人振奋的事情，她让开发人员体会到了美与和协的含义，使应用系统的生产率得以成倍的提高。目前，PowerBuilder 以其优良的性能和普及率领导着数据库应用技术的发展潮流，每个应用开发人员都应掌握她。

第一册《PowerBuilder 使用方法与实例 4.0/5.0》讲述 PowerBuilder 的安装步骤，开发方法，以及各种开发工具的使用方法和技巧。

本册资料以一个精心设计的实例贯穿始终。在讲完 PowerBuilder 开发环境的每一部分后，都配有完成此实例的操作步骤，并详细讲解实例的设计思路和设计方法.

应用实例以当前流行的MDI风格作为窗口界面，包含多媒体动画、OLE对图象的处理、动态查询、树状浏览、远程拨号、中国式报表制作等许多用户感兴趣的问题；涉及到了数据管道、用户对象、用户事件、继承等开发技术，为广大读者提供了一套非常有价值的应用范例，只要适当修改就可以变成自己的应用.

本册资料还配有完成实例的源代码盘。源代码盘是按照应用开发的实际步骤设计的，从最初开发到完善应用的每一阶段都有相应的源代码。读者可以跟着书中的实例学习，然后再与相应阶段的源代码比较。初学 PowerBuilder 的读者可以跟着实例快速地掌握 PowerBuilder 的开发方法，并可模仿这个实例开发出有一定水平的应用系统。

第二册（含 4.0 和 5.0 上下两册）《PowerBuilder 语言、事件、函数和属性4.0/5.0》用大量的例子讲解了 PowerScript 语言、函数、对象属性和事件。特别是对在 PowerBuilder 中起重要作用的函数和事件投入了大量的笔墨。本书的特点是例子丰富实用，有很多在实际开发中常用到的程序段，读者可以借鉴使用。

第三册《PowerBuilder 高级应用技术 4.0/5.0》讲述了开发高级应用的方法。本册资料以专题的形式讲解了多文档界面（MDI），对象连接与嵌入（OLE），动态数据交换(DDE)，动态连接库（DDL），动态数据窗口，动态统计图，数据窗口的列校验，子数据窗口，树状浏览，数据转换技术，微软邮件系统（MAPI），面向对象的开发技术，中国式报表，基础类库，安装

盘制作工具，开发工具包等的使用方法。对于那些有一定 PowerBuilder 使用经验的读者来说，可以通过本册资料掌握深层次的开发方法，学会用更巧更快的方法开发出高水平的应用。

第四册《珠联璧合- PowerBuilder与数据库配合开发技术》强调数据库设计的重要性，讲解了数据库实体关系模型设计方法；讲述了数据库规则、存储过程、触发器和视图在开发中的作用；讨论了并发控制和事务处理方法；给出了 PowerBuilder 应用在不同数据库间移植的方法。本册资料通过大量实例表明：数据库设计是一只看不见的手，在很大程度上决定着信息系统的成败；而良好的设计会使 PowerBuilder 的开发更简捷、更有效，是应用成功的基石。因此，对追求高品味和一流质量的开发人员来说，这是一本难得的好书。

目　录

第一章 优秀的数据库设计是应用成功的基石

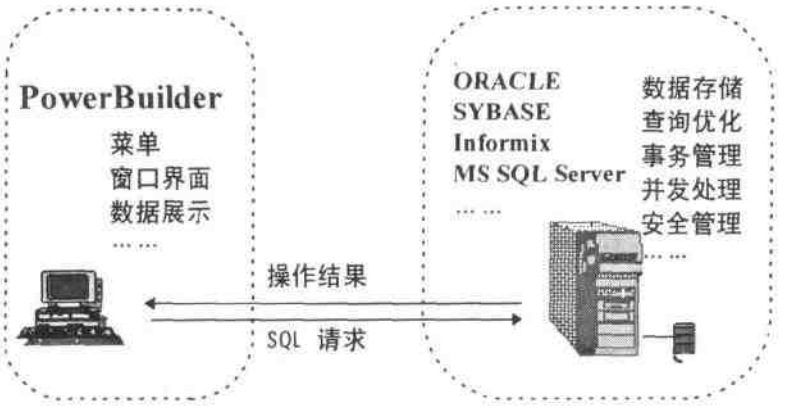
万丈高楼平地起，数据库设计如同高楼的基石，是开发高品质应用的前提。优秀的设计会给后续开发人员提供无限的发展空间，使之感到游刃有余和轻松自如。如何充分发挥数据库的功能，使 PowerBuilder 在信息系统中恰如其分地扮演她的角色是本书的要解决的问题。

1.1 PowerBuilder 与数据库是“配合”与“协调”的关系

PowerBuilder 是客户/服务器体系结构下客户端的开发工具，用于开发客户应用程序。这个程序首先建立一个与数据库的通信渠道，然后将用户的需求以某种方式传送给数据库服务器。在应用程序接收到数据库服务器返回的数据后，它分析返回的数据并呈现给用户。因此我们说，客户应用程序只完成请求和表现数据的工作，是用户操作计算机的人机界面，大多数数据处理是由服务器完成的。

数据库服务器是一个存取数据和管理数据的软件，它针对客户的请求为客户提供数据服务。这些服务包括数据插入、修改和查询等。客户对数据库服务器提出请求用的语言是 SQL(Strucrured Query Languege)。SQL 是大多数数据库服务器使用的查询语言。

因此我们说，PowerBuilder 与数据库的关系是“配合”与“协调”的关系。PowerBuilder 完成数据请求、数据表现、菜单、界面等表象方面的工作，而数据库服务器完成数据库数据的存储管理、安全管理、并发控制、事务管理、完整性维护、查询优化等工作。下图表示了 PowerBuilder 与数据库的配合工作方式。



我们看到，PowerBuilder 作为数据库服务器客户应用的开发工具，完成的主要是表示逻辑方面的工作，而数据库服务器完成的是事务逻辑管理和数据存取方面的工作。

1.2 PowerBuilder 与数据库各自的作用

1.2.1 PowerBuilder 可以完成的工作

PowerBuilder 是面向对象的快速的开发工具，可以非常好地完成下述的工作：

1. 设计窗口和定义窗口中的控制。窗口中的控制主要包括命令、图象按钮、滚动条、列表框、下拉列表框、下拉图象列表框、收音机按钮、打钩选择框、数据窗口、单行及多行编辑器等等。
2. 生成菜单。
3. 生成智能的操纵数据库的数据窗口对象，该对象可查询和更新数据库而无需用 SQL 编程。数据窗口操纵的数据源可以是表、视图、存储过程（如果数据库支持的话）以及外部数据等几种类型。在同一个窗口上，几个不同的数据窗口还可以同时对不同的数据库系统进行操作。
4. 可以在程序（script）中直接书写 SQL 语句操纵数据库。
5. 可以极方便地生成和维护数据库（即使开发人员根本不懂 SQL 语句）。
6. 在图形方式下生成数据库查询。用户即使不会 SQL 语句也可以在此环境中生成复杂的查询，并将其当作一类独特对象保留。
7. 从一个数据库往另外一个数据库拷贝数据及其结构，实现数据库之间的数据转换。

1.2.2 数据库可以完成的工作

数据库是应用系统的核心，从 PowerBuilder 的编程角度讲，它的作用有以下几个方面。

1. 接收 SQL 指令，执行 SQL 指令，把指令的执行结果返回给 PowerBuilder。
2. 查询优化。对于从 PowerBuilder 发来的 SQL 指令，数据库要先对其进行语法和句法分析，然后进行查询优化。查询优化分为两类，一类是基于语法的优化，这类优化的根据是 SQL 语句的写法，相同的查询，不同的写法会导致完全不同的查询效率；另一类是基于成本的优化，这类优化与 SQL 语句的写法无关，仅与要完成的工作有关。基于成本的优化技术极大地降低了对开发人员的要求，可以使开发人员更专心地解决应用中的问题。
3. 事务处理。数据库的特点就是数据的集中管理和共享。在通常情况下总是有若干个事务并发地运行，这些并行的事务可能并发地存取相同的数据。因此数据库管理系统的一个重要任务就是要有一种机制去保证这种并发的存取和修改不破坏数据的完整性，确保这些事务能正确地运行并取得正确的结果。
4. 并发处理。事务并发执行时若不加控制的话将导致不正确的结果和数据库的不一致状态。为保证数据库数据正确地反映所有事务的更新，以及在一事务修改数据库的数据时，其它事务不同时修改这些数据，数据库必须用锁来控制对数据的并发存取。

   并发处理是数据库最重要的问题之一，解决这一问题的办法是加锁。大多数据库都有自动加锁功能。当数据库认为必要时，会在相关的对象上加上一个适当的锁。但是，自动加的锁不一定恰当，需要我们编程干预。
5. 存取权限管理。用户对数据库对象的操作权限是用授权的办法来管理的。例如，对于表 auths，如果仅想使某用户具有查询的权限（不具有修改的权限），则可以仅授此用户 Select 权。
6. 数据库规则。规则是加在数据库表列上的约束条件，是在数据库中设定的。它可以限定列的取值范围和是否可空等等。如果用户给此列输入的值不满足约束条件，则数据库会报错，数据库只接受满足条件的数据。因此，规则可以保证输入的数据都满足要求的条件。
7. 主键和外部键。主键和外部键是维持表之间特定关系的一种方法。是实体之间一对多关系的体现。例如 auths 表的主键是作者代码 auths.author_code，对应的外部键是 books 表的作者代码 books.author_code。books 表中的作者代码值（如 A00001）在作者表 auths 中必须存在。如果我们为 books 表录入的作者编码值（如 ABCDEF）在 auths 表中不存在，则主键与外部键之间的约定关系就不满足，数据库不会接受这样的数据。这种主键值与相应的外部键值必须完全匹配的性质叫数据库的“相关完整性”。

8. 存储过程。存储过程是编译好且存储在数据库中的 SQL 语句和控制流语言的集合。它有以下的功能:

   - 可以极大地增强了 SQL 语言的功能、效率和灵活性。
   - 存储过程不同于普通的 SQL 语句或批处理的 SQL 语句，因为它们是被预编译和优化过的。这种编译过的过程可以极大地改善 SQL 语句的处理的性能。

9. 触发器。触发器是一种特殊类型的存储过程，它在插入、删除或修改特定表中数据时起作用。触发器通过维持表间的数据的一致性，保持数据的相关完整性。触发器的主要优点是在不管什么原因造成数据库数据变化时（无论是录入人员输入数据还是应用程序的影响）都能够自动响应。

   触发器有以下的用途:

   - 触发器可以通过数据库中相关的表进行连环更新。例如， auths 表上的删除触发器可相应地删除 books 表中的与之匹配的行。
   - 触发器能够“回退”那些破坏相关完整性的变化。例如，可以在 books 表上生成一个插入触发器，如果插入的值不满足特定的条件，则插入被回退。
   - 触发器可产生比规则更为复杂的限制。与规则不同，触发器可以引用列或数据库的对象，从而可以写出比规则更为复杂的数据限定。
   - 通过触发器还可以在一定程度上实现数据库的复制功能。

10. 视图。视图是查看一个或多个表中数据的另外一种方式。可将视图看成是一个移动的窗口，通过它可以看到感兴趣的数据。视图数据是从一个或多个实际表中获得的，这些数据被物理地存放在数据库中。那些用于产生视图的表叫做该视图的基表。一个视图也可从另一视图中产生。视图看上去非常象其它的数据库表，对它的显示和操作同其它表一样。使用视图有以下的优点:
    - 看到的就是需要的。视图允许用户集中在他们感兴趣的特定数据上或他们所负责的数据上进行工作。那些对特定用户或对特定工作无关的数据可被排除在视图之外。
    - 简化数据操作。视图不仅可以简化用户对数据的理解，也可以简化他们的操作。那些被经常使用的查询可被定义为视图，从而使得用户不必为以后的操作每次指定全部的查询条件。
    - 视图允许不同的用户以不同的方式查看同样的数据。这一优点对许多兴趣不同的用户共享同一数据库时是特别重要的。
    - 安全性。通过视图，用户只能查询和修改他们所能见到的数据。如果一个视图和它所引用的表和视图都属于同一用户所有，该所有者便可授与其它用户

使用该视图的权限，而同时又保留使用该视图的基表或基视图的权限。这是一种简单有效的安全机制。通过定义不同的视图并且有选择性地授与它们权限，用户可被限制在使用数据的不同子集上。

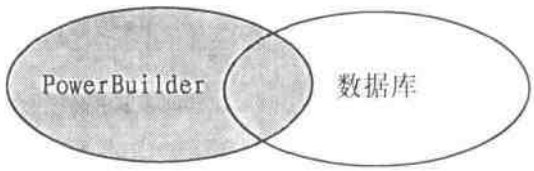
- 逻辑数据独立性。视图可帮助用户屏蔽真实表结构变化带来的影响。例如，如果 PowerBuilder 数据窗口建立在表 auths 上，假如表 auths 发生了变化，则数据窗口应当相应地修改。如果我们在被修改的表上定义一个视图，通过这个视图掩盖掉表的修改，则此时数据窗口不用变化，从而使应用程序与数据库库结构相互独立，互不影响。

11. 序号发生器。数据库可以自动生成连续的序号，如定单号和发票号等。数据库这么作的目的是避免多个用户（成百上千个用户）同时申请下一个序号造成应用的瓶颈。

1.3 PowerBuilder 与数据库配合开发时要考虑的问题

上节讲述了 PowerBuilder 与数据库各自的功能。PowerBuilder 与数据库配合开发时存在以下两个问题：

1. PowerBuilder 完成的主要是应用前台的问题，数据库完成的主要是应用后台的问题。然而，由于数据库和 PowerBuilder 的功能都在增强，PowerBuilder 的功能在向数据库方向延伸，数据库的功能在向开发工具方向延申，一些功能是相互交叉的（如 PowerBuilder 的 Query 画笔与数据库视图，PowerBuilder 列的扩展属性和数据库规则等）。所以，有些问题用 PowerBuilder 或数据库都能解决。对于应用中的某个特定的问题，究竟用数据库还是用 PowerBuilder 来解决呢？用数据库解决问题的好处是什么，用 PowerBuilder 解决的好处是什么？



2. 对于非数据库类属的工作（如并发处理和事务管理），PowerBuilder 的表现特征是什么样的呢，PowerBuilder 是如何针对它们编程的呢？

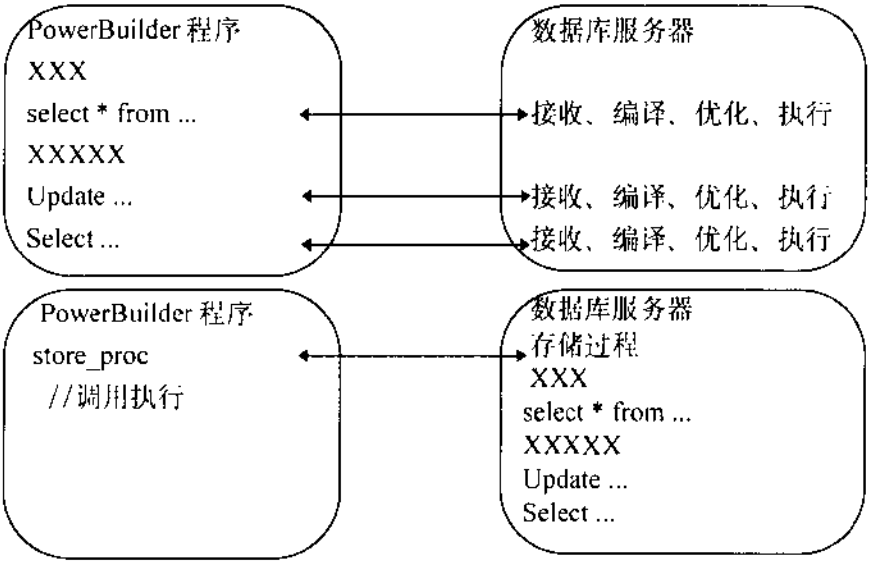
下面我们列出应用中会遇到的问题，供大家思考和讨论。

1. 维护体现企业规则的表与表之间关系即可以用 PowerBuilder 编程完成，又可以用数据库触发器完成。例如，如果作者的数据被删除，则该作者的著作数据也相

应地删除，这件事情即可以用窗口上的事件处理程序完成，又可以用数据库触发器完成（参见本套丛书第一册“使用方法与实例”中“数据窗口”一章）。这种表间的关系是随着企业规章制度的变化而变化的。如果用 PowerBuilder 编程解决这一问题，则程序的维护将很困难。原则上这样的问题应该用数据库触发器来解决。用触发器解决这一问题的优点是：

- 企业规则可以在数据库中集中控制。
- 如果企业规则发生了变化只需要相应修改数据库触发器的内容，无需修改 PowerBuilder 应用程序，系统容易维护。
- PowerBuilder 编程变得简单。
- 系统运行效率会提高。但是，过多过滥地使用触发器会反而使系统的效率降低。

2. 体现企业规则的数据运算即可以用 PowerBuilder 编程完成，又可以用数据库存储过程完成。例如，职工奖金的计算，银行利息计算等。这种计算也是随着企业规章制度和国家政策的变化而变化的。如果用 PowerBuilder 编程解决这一问题，程序的维护也很困难。象这样的问题应该用数据存储过程来解决。用存储过程解决这一问题的优点是：
   - 企业规则可以在数据库中集中控制。
   - 如果企业规则发生了变化只需要相应修改存储过程的内容，无需修改 PowerBuilder 应用程序，系统容易维护。
   - PowerBuilder 编程变得简单。
   - 系统运行效率会提高。因为存储过程是存放在数据库服务器中的，已经编译和优化好，如果用 PowerBuilder 完成则需较大的网络通信量，数据库也要对接收到的 SQL 语句进行编译和优化，从而会使系统的效率降低。



3. 事务是由数据库管理的最小的逻辑工作单元，在一个事务中所有对数据库的更新操作要么全成功要么全失败。事务的划分（起点和终点）是 PowerBuilder 的任务，（它提供了事务处理语句 Commit 和 Rollback 等），但在窗口环境中如何划分事务，如何编写程序管理事务是一个较难的问题。
4. 并发处理是多个人同时更新数据库中相同数据时的处理方式，数据库可以自动加锁，但这种自动锁不一定适合实际的需求。例如，用户 A 修改数据库表 auths，相应的记录被加锁。此时其他用户不能修改被加锁的记录（Oracle 的锁加在行上，Sybase 的锁加在页上），如果此时用户 A 的工作被中断（接电话或外出），其他用户必须等待，直到用户 A 返回后释放他锁住的记录为止。在实际的并发的条件下会出现很多难以预料的复杂情况，“恰当”地加锁是 PowerBuilder 的任务，也是一个较难的问题。
5. 序号生成。显然，这个工作应该由数据库来完成（如果数据库支持序号自动生成的话）。序号也可以在 PowerBuilder 中生成，但这样做会使几十个乃至上百个用户同时抢一个序列号，形成应用的瓶颈，这在大系统中是不可取的。
6. 视图。视图是数据库原始数据的某种变换。在有的时候，这种变换可以用 PowerBuilder 的 Query 工具或数据窗口直接完成。但是，前面讲过的视图的许多优良特征是 Query 和数据窗口本身不具备的。哪些工作应留给 PowerBuilder 完成，哪些应留给视图完成是数据库设计人员和开发人员认真研究的问题。例如：
   - 视图建在多表上，数据窗口建在视图上，则数据窗口不能修改对应的基表数据，这一特征是视图带来的；如果数据窗口直接建在多表上，则可以通过一定的技术修改对应表的数据。
   - 视图可以建在视图上，可以对数据做非常复杂的变换，但 Query 不能建在 Query 上（可以建在视图上），从而 Query 的复杂度会受一定的限制。
   - 视图是数据库对象，可以对用户进行适当的授权，但 Query 是 PowerBuilder 对象，没有授权机制。
7. 数据库表列的约束既可以在数据库中完成，又可以在 PowerBuilder 中完成。在这一点上它们的功能是完全重叠的，但它们的特性差异较大。在数据库中定义的约束条件只有把数据送到数据库时才起作用，屏幕上录入数据正确与否数据库无法在录入时知道。而在 PowerBuilder 中定义的列约束只在屏幕录入时才起作用，数据库不管它的对与错。因此，这两种约束是同等重要的。哪些约束用数据库完成，哪些约束用 PowerBuilder 完成，哪些约束两者都要做也是数据库设计人员和开发人员需要认真研究的问题。例如：
   - 对于非常重要的列，两种约束都应存在。因为仅仅在 PowerBuilder 中的列约束不能防止用户使用其它手段破坏数据（用 Sybase 的 isql 工具录入数据会绕过 PowerBuilder 的约束）。
   - 对于在录入时需要立即响应对与错的约束应该用 PowerBuilder 完成，因为数据库无法做到这一点。

第二章 数据库服务器的特点

好的数据库设计应当充分考虑应用的需求和开发工具的特征，应当充分利用和挖掘数据库的功能。一流的数据库设计会给整个应用系统带来以下的好处：

1. 维护容易。当需求发生变化时，优良的数据库设计会使应用系统容易地适应这种变化。
2. 编程简单。由于数据库设计充分挖掘了数据库的功能，所以应用程序的复杂度可以明显降低。
3. 加快开发速度。因为数据库设计充分地考虑了需求和开发工具的特征，所以开发人员会感到左右逢源，得心应手，从而可以加快开发速度。

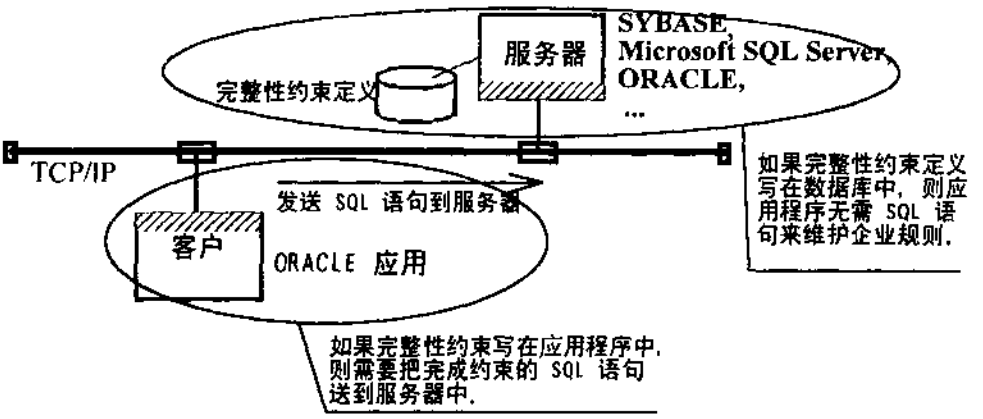
一流的数据库设计来源于对数据库特色的充分理解。本章针对 PowerBuilder 讲述大型数据库（如 Oracle 和 Sybase）的特色。

数据库具有以下特征：

- 完整性约束。完整性约束允许在表上定义某种约束条件，这些条件作为表定义的一部分存在，从而强制表中的数据满足一定的规则。
- 存储过程。存储过程是由流控制语句（if ... else）和 SQL 语句书写的过程，这个过程经过编译和优化后存储在数据库服务器中，使用时只要调用既可。在 Oracle 中还可以把若干个有联系的过程组合在一起成为程序包。
- 数据库触发器。触发器是一种特殊的存储过程，不同的是这种过程不是由程序调用来执行，而是通过数据库数据的更新自动地“触发”执行。
- 事务处理。事务是最小的逻辑工作单元，在这个单元中对数据库所有的更新要么全成功要么全失败。
- 并发处理。无需任何说明，ORACLE 自动提供行级锁，Sybase 自动提供页级锁，允许用户在没有冲突的情况下更新表中不同的行。行级锁和页级锁对联机事务处理非常有用。
- 序号生成器。数据库可以自动生成连续的序号供应用程序使用。
- 遵守工业标准的 SQL。
- 视图（应包含在 SQL 中，因为视图特别重要，所以单独列出）。视图是原始数据库表的变换，可给应用程序带来安全性、简单性和独立性。

2.1 完整性约束

完整性约束是数据库用于维护数据库完整性的一种机制。这种约束是表定义的一部分，是内部的，与在应用程序中维护数据库的完整性不同，它的代价小而且性能高。对于客户服务器体系结构下的应用来说，用应用程序发送 SQL 语句来维护数据库的完整性还会增加网络开销。Oracle 和 Sybase 都支持在 create 语句中书写遵守 ANSI 标准的约束语句。



完整性约束有以下两点作用:

1. 使企业的规则与数据库联系起来。
2. 防止操作员或终端用户输入错误的数据，破坏数据库的完整性。

完整性约束有以下几种:

- NOT NULL 约束
- 缺省值
- UNIQUE 约束
- PRIMARY KEY 约束
- FOREIGN KEY 约束
- CHECK 约束

2.1.1 NOT NULL 约束

NOT NULL 的含义是列中不能有空值。NOT NULL 在创建表时定义: