Baan 系统架构

版本	日期	编著	说明
V1R0C	2009/04/25	Andy.Bai	翻译

版权声明

目录

1 介绍

ERP（Enterprise Resource Planning）系统是一个基于分布式计算机平台，包括一个或多个数据库管理系统的 通用的并且十分复杂的商业软件系统。它是由大量的应用程序，函盖了大部分的企业信息需求而组合成的一个全球性的企业信息系统，实现了各种商业对企业运营至关重要的商业流程。这种系统能帮助组织去实现基本的商业功能如：采购、销售、库存管理、财务会计管理、财务控制、人力资源管理，同时它也包括其它的高级的商业功能如：项目管理、生产计划、供应商管理以及销售自动化。

第一代的 ERP 系统现在运行世界最大的公司的后方办公室。ERP 市场在 1998 年增加了 50%达到了 8.6 亿美元，这个市场的领志者 SAP R/3 达到了 22000 个安装点。实际的实施 ERP 所产生的利益非常巨大。

典型的，ERP 系统运行在一个三次的称为 C/S 的结构上。它能够提供对多个数据库实例，作用于底层的数据库模式、用户接口的多个配置和版本（或称为定制），以及与它相关的大量的应用程序的管理。既然 ERP 系统是为多国的公司环境进行设计 ，因此它必须能支持多国语言，多国货币以及特定国家中的商务处理实务。ERP 的这种规模与极其复杂性使得它十分难以部署与维护。无论已在全球成功实施的系统如 SAP R3，BaanERP， 然而它基础的结构，数据模型 ，珀务处理机制，编程技术等对计算机科学家来说还有相当多是未知。

此书的目的是展示作为一个 ERP 的典型代表的 BaanERP,从计算机科学的角度（而不是从业务管理的角度）来展示它的信息技术，以及与之相关建立数据库和分布式系统的概念与技术。一个 BaanERP 批判性的评价会指出其优点与不足点。 本书还会帮助在一般意义上去理解 ERP 的潜在内容，以及 Baan ERP 系统的特别技术，以及关于它的研究与开发工作。

2 概述

2.1 Baan—公司

成立：1978 年
总部：Barneveld (荷兰) 和 Herndon (美国 弗吉亚)
客户：13000 全球 客户（最大的客户为 Boeing）
雇员：大约 4700
年收入：1998 年 736 M USD

2.2 术语

在表 1 中，我们列示出了 Baan 的最重要的与通常意义不同的行业术语：

Baan 术语	意义
公司-company	客户 (有它自己的一组数据表)
进程-session	有与它相对应屏幕画面的应用程序
业系子系统-package	应用程序的一组集合包
'Package VRC'	业务子系统的 版本
'Package Combination'	配置一个特定版本的 Baan ERP 数据模式以及与它相关的(应用程序)和子系统版本（'Baan 环境'）

2.3 Baan Tools 的定制与编程

Baan ERP Tools 包括以下三部分：

- **应用管理**： 用户，客户，数据管理，SQL 查询等
- **用户界面定制**： 版本管理，菜单 ，窗体，报表，进程等；
- **编程**：

一些应用程序管理功能 ，诸如：软件安装，设备管理，审核管理只与 Baan 系统

管理员相关。

2.3.1 应用管理

以下为应用管理的通用功能：

维护公司：

公司（‘company’或称客户）定义了公司代码，名称，币种，以及与它的数据模式相关的业务子系统组合（Package combination）

用户管理

Baan 用户的名称通常与登录系统的名称相同。对每一个用户，开始菜单 ，子系统组合，以及公司号被指定。 这个客户号决定此用户将用哪一个公司里的数据进行工作。见图 1

文本管理

内部的文本必须被内置的文本辑器（已经相当地不流行了）进行编辑，且它们是被 一行一行地进行存储（可以按语言），可以在交付条件，物料描术，发票页脚等地方被使用。

作业管理

一项作业由周期性执行的可配置顺序的打印和程序处理（例如每两个小时）。

数据库管理

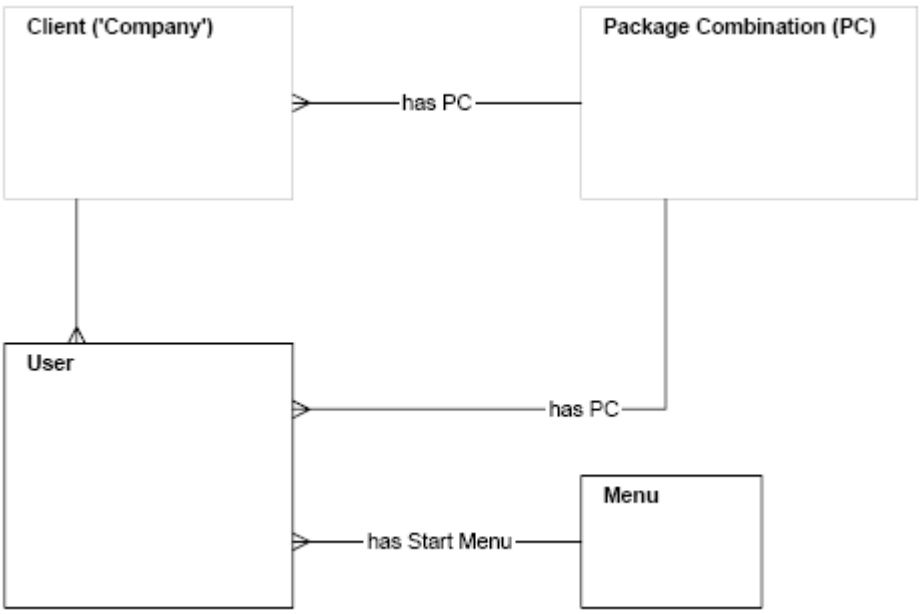
由以下功能组成：

1. 分离数据到两个数据库中。 一个为关键性的表（比如：运行参数和授权）为所有访问用户记录安全日志，另一个为其它的表；
2. 倒出，倒入，删除，检查以及重组表（在杂项下）
3. 检查对特定的表可能修改的数据（在通用表维护下）

SQL 查询

一个数据库的查询可以应用窗体被交互式的创建或直接在 Baan 的文本编辑器中输入 SQL SELECT 语句。 结果集被显示在屏幕上或通过 Baan 的报表打印在纸上。

图 1：默认的用户的公司必须与用户有着相同的子系统组合:



2.3.2 定制用户界面

用户界面中的许多组件可以不通过编程来进行交互式的更改和创建，这包括：

菜单 Menus

组成一组选项引导至应用程序或到子菜单。

标签 Labels

是一个带有名称的短文本在窗体中作为标签到或报表中作为列进行显示；

报表 Reports

由许多布局元素和数据字段与标签所定义。 它们被用作创建复杂的报表如提单，发票，也可用于显示和打印 SQL 查询的结查。

窗体 Forms

由用于显示并允许修改数据的窗体字段， 下拉菜单，通过调用程序而执行特定功能的按键组成。

消息与问题 Messages and Questions

是被命名的短文本在应用程序运行期间用于显示消息或询问

2.3.3 编程

BaanERP 编程开发由版本管理， 程序脚本的创建和维护组成：
版本管理包括：

1. PVRC 的创建与维护
2. 在子系统组合中配置一组子系统版本

真正的程序是指程序脚本，包括模块（也称为功能函数）和 DLL 模块（库）。

程序脚本-Program Scripts

用一种类似 Pascal 的过程语言编写，称为‘Baan 3GL’,.
对于基于窗体事件编程的被称为’4GL， 它是通过扩展了特殊的关键字去指代窗体和数据库中的不同事件。

函数'Functions'

是实际包括于模块之中，允许重复使用的变量定义，功能 和过程。

库 Libraries

允许维护与重用函数与过程代码。与函购数相对应的，DLL 库的功能不需要重新在编译原代码中产生并形成目标代码。

2.4 限制

2.4.1 应用服务器 ('Bshell')

- 最大的字符串缓冲区 4K
- 最大的函数调用堆栈深度是 75

2.4.2 数据表

- 最大记录长度是 3072 字节
- 最大字段数 1024
- 最大字段长度是 3072 字

2.4.3 索引

- 最大索引长充 120 字节
- 最大的字段索引数是 32 (非联合字段)

3 体系结构

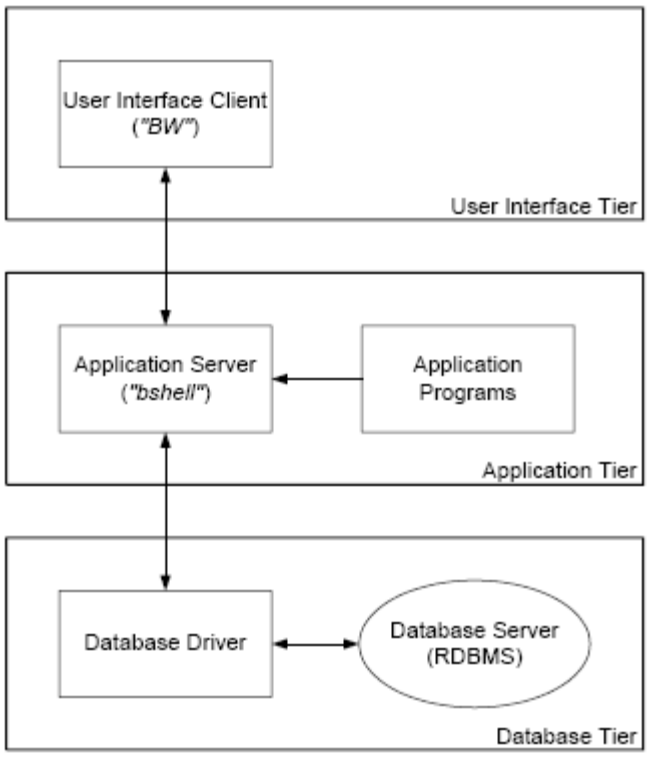
BaanERP 支持 3 层结构，由用户界面层，应用程，数据库层组成。用户界面接口层为用户的交互提供表现和输入服务，应用程由 Baan ERP 应用服务器和应用程序组成。 数据库层包括 Baan ERP 数据库驱动和第三方 RDBMS 产品以执行数据库服务。

图 1： 表示 Baan ERP 结构

本处的重点在于 Baan ERP 数据库驱动。 数据库驱动是 Baan ERP 应用和 RDBMS 的接口。数据库驱动翻译从 Baan ERP 传来的请求至 RDBMS 特定的 SQL 请求然后送至数据库服务器。

在数据库完成检索需求的信息后，数据库驱动将结果传回到 Baan ERP 应用服务.

图 1: Baan ERP 三层结构:



3.1 用户接口层

用户接口层由基于 Windows 的 Baan ERP 用户接口(称为 BW)和基于 Internet 的

Brower (称为 BI). 通过 BW 或 BI 录入的用户数据被传递至 Baan ERP 应用服务;从 Baan ERP 应用服务返回的数据显示在 BI 或 BW 上。

3.2 应用层

应用层包括应用 BAAN ERP 服务器和应用程序。 应用程序和 BAAN ERP 应用服务器提供了大量的 BAAN ERP 功能。应用程序包括已被编译的 BAAN ERP 应用程序和数据字典。BAAN ERP 应用为使用 BAAN ERP 工具包的中的开发环境内嵌 SQL 由 4GL 语言编写的程序。

BAAN ERP 应用服务器调度与运行应用程序，发送与接受用用户接口服务器的信息，在必要时初始化数据库驱动实例以为与数据库通信做准备。一个运行着的数据库驱动能支持多个至单个 RDBMS 实例的连接。如果 BAAN ERP 安装数据在多个数据库，则应用服务器必须为每一个它必须访问的数据库创建一个新的数据库驱动以便与数据库进行连接。BAAN ERP 应用服务器传统地称为 BAAN SHELL 或简称为 ‘Bshell’,在本文以后，它指代 BAAN ERP 应用服务器。

3.3 数据库层

数据库层由 BAAN ERP 数据驱动和数据库服务器组成。 数据库驱动提供 BAAN ERP 应用服务器与数据库服务的通用接口。应用程序与数据库驱动之间的通信是通用的，不管使用哪一种 RDBMS 作为数据库服务器。BAAN ERP 支持的每一种 RDBMS 产品有一个数据库驱动，数据库驱动与数据库这间的通信会根据 RDBMS 进行调整。通过 SQL 语句和 RDBMS 本身的应用程序接口 API。

数据库服务器由以下五种第三种 RDBMS 产口这一组成。 Oracle, Informix, Sybase, DB2, 和 MS SQL Server. 所有的 BAAN ERP 应用数据通过 RDBMS 存储在关系数据库中。在同一个 Baan erp 安装中，用着多个 RDBMS 产口是完全可能的，一部分数据存在一个 RDBMS 中，而另外的一些在另一个上。

3.4 经过 Baan ERP 结构的数据流

注意到数据库驱动提供 BAAN ERP 应用服务器与特定 DBMS 服务器之间的接口。 经过系统的数据流在以下进行描述：

当一个用户在 GUI 的工作站上执行一个操作，用户接口服务解释这些操作并把它发送到 BAAN ERP 应用虚拟机。 基于接收到的信息， 应用服务器用适用的对象对执行。

当一个运行着的应用对象请求存储于数据库中的信息时，应用服务器发送请求至数据库驱动。 从客户应用方请求的信息是与 RDBMS 独立无关的以使用 BAAN ERP SQL， 一种与

RDBMS 无关的 SQL 语言。

当应用程序从一个应用中执行一个数据库查询时，它首先要决定是否有一个已运行着的数据库驱动来完成这个查询。 如果此时没有一个运行着的数据库驱动，或者一个运行着的数据库驱动在与一个数据库服务器通信而此数据库并不是需要所数据库时，此时应用服务器会启动一个新的数据库驱动。 应用程序服务器发送它从应用对象接收到的查询 BAAN ERP SQL 数据库查询并以内部形式传到数据库驱动。 数据库驱动收到的以内部形式表现的查询仍然是 RDBMS 独立的。

数据库驱动翻译数据库查询至与特定的 RDBMS 相一致的 SQL 形式，每一个数据库驱动利用特别的 RDBMS 设计支持因此转换成的 SQL 语句对些 RDBMS 有效并且是具有最好的性能。此 RDBMS 特定的 SQL 语句之后被提交至 RDBMS 服务器，在这里进行数据处理。当 RDBMS 服务器处理完毕，它将结果返回至数据库驱动，任何错误的条件被捕获并且被数据库驱动处理。数据库驱动然后将结果与状态信息至应用服务器，就是最初它提交查询请求的地方。应用程序服务器发送一个消息至用户接口服务器，将合适的信息显示在用户的终端屏上。

3.5 Baan ERP 硬件配置

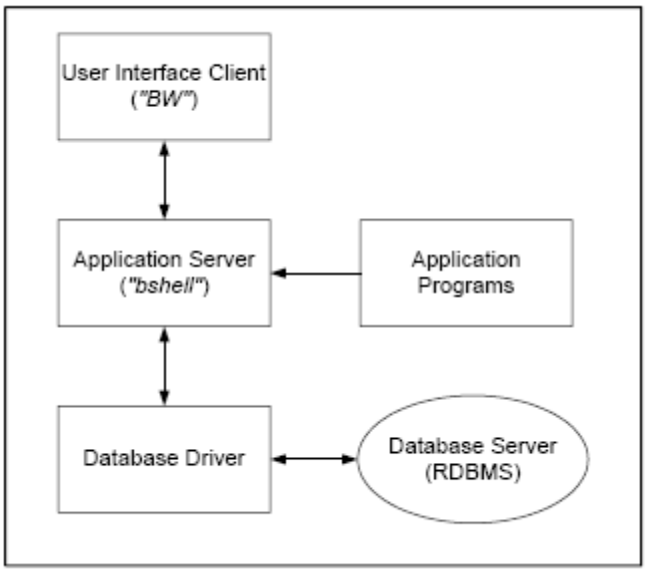
BAAN ERP 支持多种硬件的配置。这些配置包括独立配置模型和多客户/服务器模型。

可能的硬件，数据存档需求，以及性能期望决定着最合适的硬件的配置。

独立(Standalone)模型是指所有的配置在同一计算机上。 在独立模型中，用户可以在主机前面工作也可以通过瘦客户机（X-Terminal）运行 BI。

这种独立模型说明如图 2.

图2 ： 独立模型配置



在 C/S 配置中，BAAN ERP 结构的组件被分布在两个或多个计算机中。 有许多种 C/S 模型配

置。在这里描述最通用的一种。最简单的 C/S 配置是独立模型的一种变种。 在这种配置中，应用层数据库驱动和 RDBMS 在同一台机器上， 然而显示驱动分布在用户工作站上。 一个应用服务器实例和至少一个数据库驱动实例为每一个用户启动。所有的用户访问相同的应用程序和数据库服务器。 这种配置模型如图 3.

当有两台机器可当做服务器时，应用层被安放在一台服务器上，数据驱动与数据库服务放在另一台机器上。与前面配置相比，一个应服务器实例 与至少一个数据库驱区实例为每一用户启动。所有用户访问相同的应用程序和数据库服务器。这种模型说明如图 4 所示。

图 3： 应用和数据库服务位于同一台计算机中:

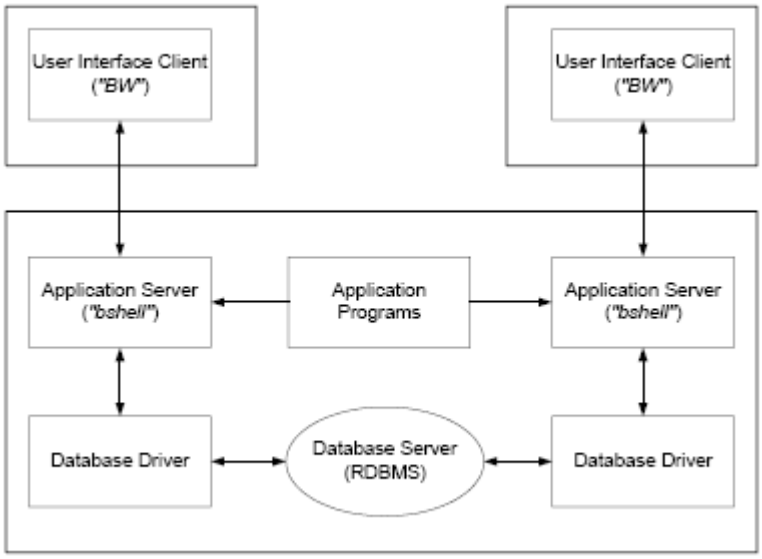
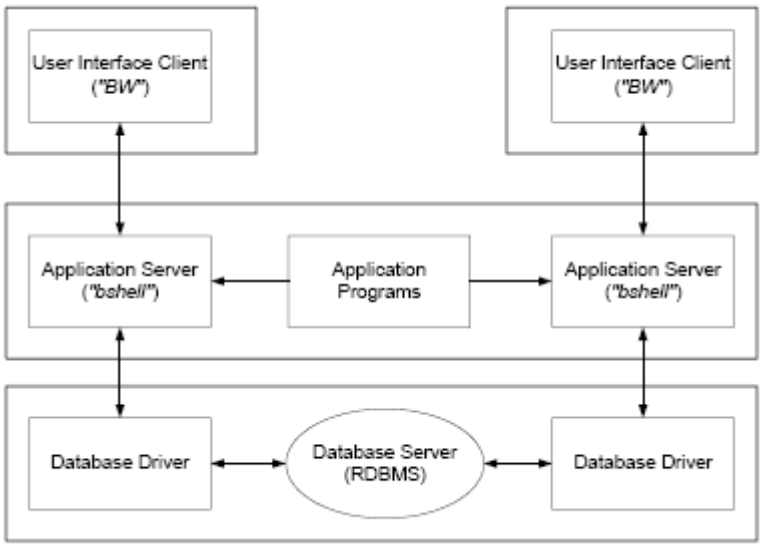


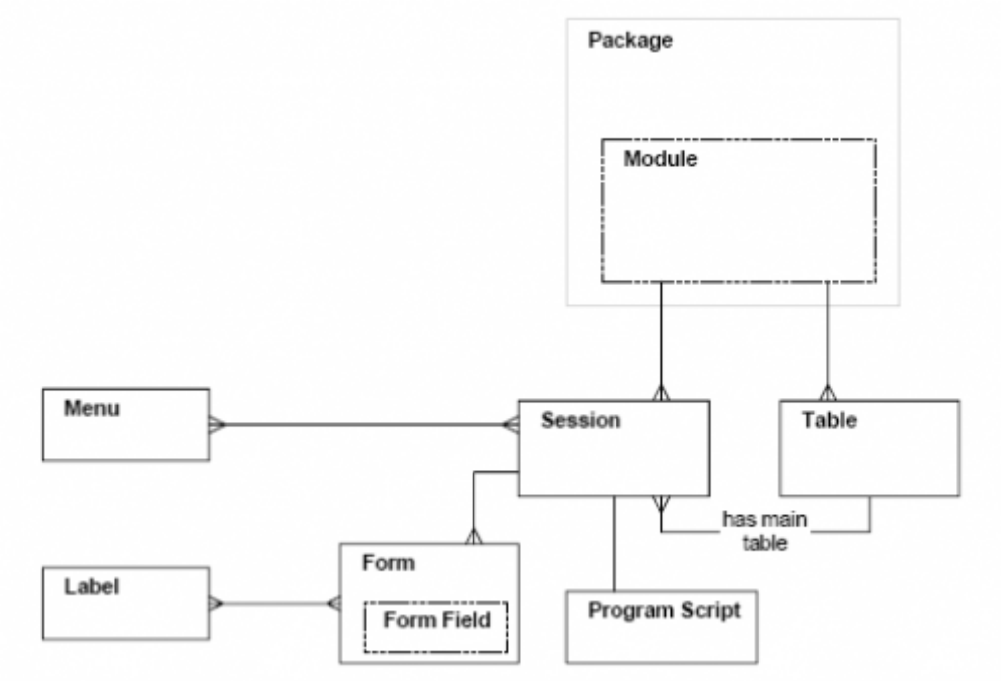
图 4.: 应用与数据位于不同的计算机中



4 软件结构

数据表、应用程序以及它的应用组件都是根据业务功能进行模块划分。

图 1: 业务子系统,模块,进程:



4.1 子系统--Packages

业务划分诸如分销、制造、财务被映射至程序子系统，用两位代号来标识。 这些代号列在表 1 中。

其中最为重要的是: tc/通用, td/分销, ti/制造, 和 tf/财务

表 1: Baan ERP 子系统..

cp	约束计划 Constraint Planning	
ct	控制 Controlling	
ps	流程生产 Process	process-based production
tc	公用 Common	master data

td	分舾 Distribution	purchase/sales/inventory control
ti	制造 Manufacturing	
tf	财务 Finance	financial accounting
tp	项目 Project	project management
tr	运输 Transport	transport industry
ts	服务 Service	service industry
tt	T 工具 Tools	Baan administration and programming
tu	实用程序 Utilities	data exchange interfaces, etc.

4.2 模块--Modules

业务功能按模块进行划分，每模块以三个字母表示。例如：TD 子系统中由以下模块组成

Distribution:Purchase 分销---采购	td pur
Distribution:Sales 分销---销售	td sls
Distribution:Inventory 分销---存货	td inv

4.3 进程 Sessions

所有的业务功能是通过进程来实现的。 一个 BAAN ERP 进程是一个应用程序，它通常与屏幕界面相关联，并且有基本，且通常包含有报表。 一个进程 ID 是由子系统/模块 ID 做为前辍的。 例如：进程 Ti itm 0101m000 显示这个进程是属于模块（ITM）item control 并且存在于 TI （制造子系统中）

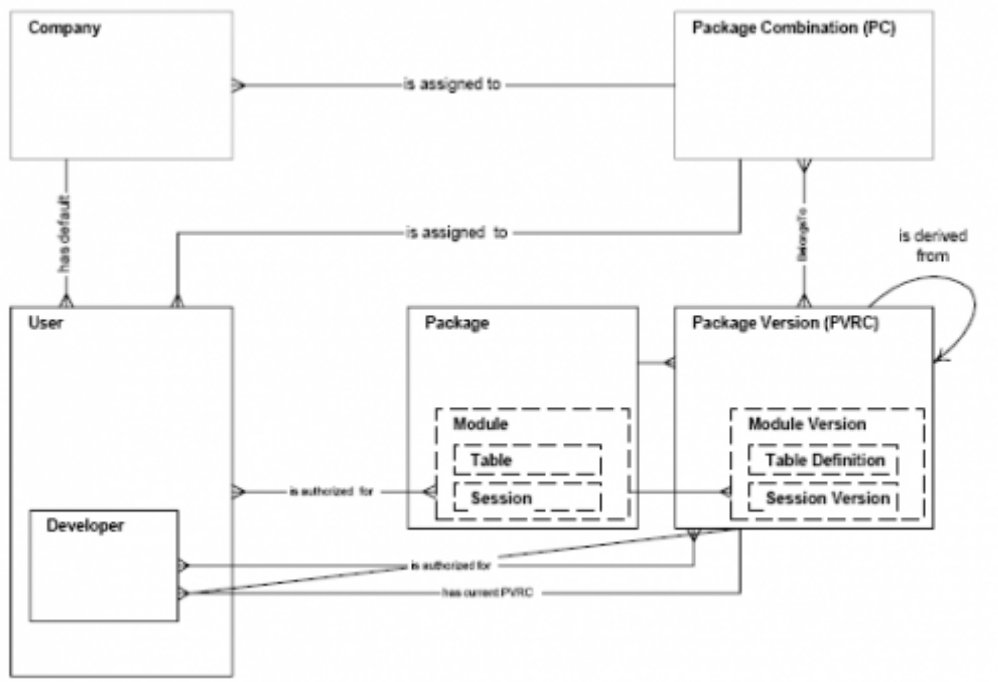
进程被用户通过菜单交互式地调用。有许多用于主数据管理的进程，如维护物料（Maintain items tiitem0101m000）或执行特定的业务过程，如更新标准成本价（ticpr2220m000）

进程可以被定义了子程序，（subprogram, subsession）,那是指在其它程序或进程中进行调用。不用 M（M 表示主进程）而用 S 表示（代表子进程）。如更新成本价（ticpr2220s000）

5 版本管理

复杂的应用软件系统，比如 ERP 系统，受**持续性的改进**。更改可能产生于发现程序的错误而进行，或者新的技术，或者出于法律法规的要求需对 ERP 进行更改。 并且，标准软件的不同部分必须可被定制以适应不同的国家，分公司或公司级的特定流程需求。为了能够适应这些更改的需求， 以一种系统化的方法，不同的软件组件版本（表与进程）能被维护进 BAAN ERP 系统的**定制版本。**因此所有的组件存在同一个逻辑命名层(Logical naming level）并且在特定实例的层（Level of special instance ）或版本。 如图 1 所示。

图 1　BAAN ERP 中的版本管理



5.1 子系统版本-Package Versions (PVRCs)

版本管理在 BAAN ERP 中是继承方式的。 一个子系统的新版本可以从已经存在的一个版本中继承。 在这个新版本中，仅有那些**新创建的**，或**经过修改过的组件存在**。

所有的其它组件自动从前一个版本中进行执行。应用程序的实际版本取决于登录的子系统组合。 例 如，物料维护（tiitem0101m000）的运行版本取决于 TI 子系统的版本，而 TI 子系统的版本又取决于子系统组合， 这个子系统给合的版本在登录时确定。子系统的版本也称这 **P-VRC**（Package VRC），其中 VRC 代表 Version 版本，Release 发布 Customization 定制。

这种命名在 BAAN ERP 中反应出不同的版本层次。BAAN ERP 软件系统通过持续的改进与扩展，会定期地发布更新的版本。这会采用两级的 V-R 模式（Version-release）。 比如，从版本名称 B40S 和发行名称 C3，子系统 VRC TiB40Sc3 形成了 TI 子系统的 BAAN 软件 40 版本的标准版本（S）的第三次发布（C3）。BAAN ERP 标准软件的两级版本附加第三版本名称，此级的名称是为企业定制化修改的标准版本。例 如，特定的版本 TI 子系统从 BAAN IV 标准版本 C3 继承并为 Wagner World-wide 企业定制可以命名为 tiB40Cc3wag. BAAN 强烈建议所有的定制符合它的命 名标准。比如对企业定制的 BAAN ERP 是 B40C 而不是 B40S，由 C 表示是经过定制的。

版本代号的意义如下：

S	standard version 标准版本
U	update version (service pack)更新版
L	country-specific 'localisation' 特定国别的
B	branch-specific version 特定分支的
C	enterprise-specific customization 特定企业定制的

BAAN 必须在它的标准系统中提供许名按国别定制的附加/修改以能够在全球进行销售。如表 1 所示。

特定的子系统版本（称为现行 PVRC）指定给每一个开发者作为当前编程的环境。如果开发者有权限可以在以后进行更改。

5.2 子系统版本的发源

一个新的子系统版本（PVRC）可以从一个已存的在子系统版本中继承。这种机制允许只有那些全新的或是与以前版本不相同的组件存在新的版本中（即在新版本中只有新的组件，或某些组件，这些组件与旧版本中不相同的功能，尽管名称可以一样）。那些完全相同的组件是不会在新版本中出现的，当执行时，会顺着这种继承的链去查找一直查找到最初的组件。顺着这种逻辑，我们可以得知，仅有最特殊的（或最近的）组件版本存在于继承链的最末端。

表 11: 按国别定制的例子.

国家 / 区域	国别定制化项目
Argentina	central invoicing, Argentinean VA
Brazilia	central invoicing, fiscal receipts, receipt schedule
Italy	Libro Giornale, withholding tax, LIFO by Year, BAM, VAT book
Japan	statement of accounts, customer approvals flow

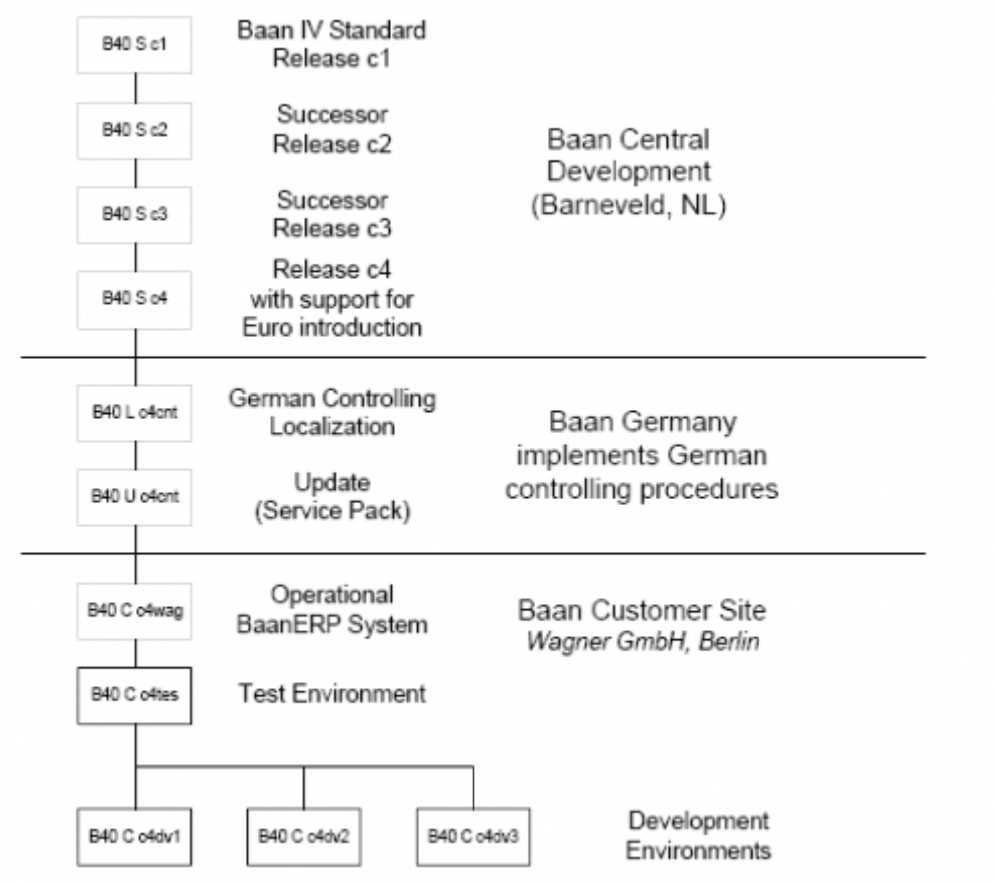
Nordic countries	flow for interest invoice
Switzerland	central invoicing
India	excise invoice, MODVAT, personal ledger account, tax deduction at source
Australia	sales tax, prescribed payments system

5.3 子系统组合—PC（Package Combinations）

一个子系统组合是一个特定的多个变化的 PVRC，它们形成一个复杂的应用软件包。 每一个 PC 都与特定的数据据库模式，定义所有表结构的数据字典相关联。

每一个 ERP 用户都将会指定一个 PC 给他以决定他可以使用哪些程序。继然这些程序只为特定的数据模式而工作，因此用户只能从客户访问与相同 PC 相关联的的那些数据表。

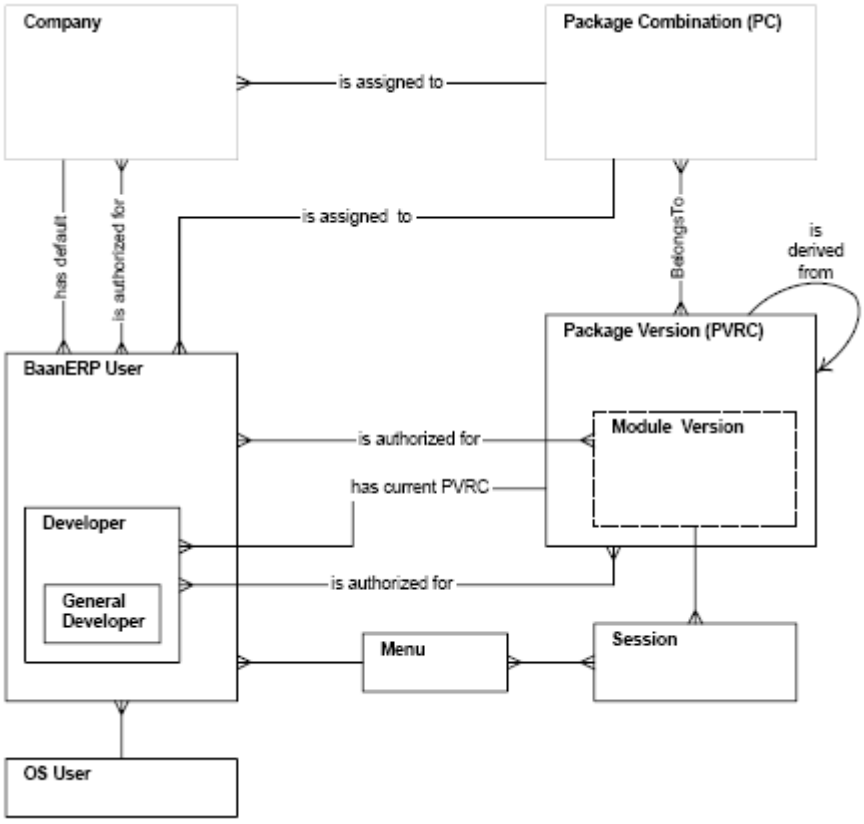
图 2: 不同版本是如何继承的



6 公司与用户

BAAN 的公司由公司代号，公司名称，币种和 PC 共同而定义。 PC 与特定的数据模式相关联。

图 1： 用户与开发者



一个 BAAN ERP 用户通来一个 BAAN ERP 登录名称来标识，此登录名称必须与操作系统的登录相关联。BAAN ERP 用户最重要的属性有： **子系统组合 PC，默认公司号，启动菜单**。为了让同一个人在用不同的子系统组合下工作，多个 ERP 用户帐号可以与同一个操作系统用户相关联。 即（同一个 OS 帐户）----（多个 ERP 用户帐号）------（每一 ERP 用户帐号对应不同的 PC）。

每一个 ERP 用户帐号被分类为：正常（NORMAL）与超级（SUPER）用户。超级用户有着最高的权限，可以访问它子系统组合下的所有进程，所有的表，所有的公司。就正常用户来说，对进程的授权，公司以及表必须明确地定义，通过角色与授权模板。

一个 ERP 用户也可以被标识为 BAAN ERP 开发者，它有权去定制用户接口组件，过程与数据表。开发者的当前 PVRC 定义着它的默认开发环境，它可以定制这个 PVRC 下的组件。

7 数据库处理

所有的被 BAAN ERP 使用的应用数据都在存在下层的 RDBMS 数据表中。为了使 BAAN ERP 处理数据的独立性于 RDBMS，BAAN ERP 使用了自己的数据字典。这个数据字典（以下简称为 DD）包括了域（domain）,模式（schema）参考完整性的以一种独立的方式存存在数据库中的信息。

BAAN ERP 系统提供了 RDBMS 的接口，被称为数据库驱动（database driver）,以连接重要的数据库系统如（Oracle, Informix, Sybase,DB2, and MS SQL server）。BAAN ERP 数据库驱动有一个内置的机制维护参考完整性,它不依赖于下面的 RDBMS 的参考完整性功能。

7.1 Baan ERP 数据库概念

7.1.1 数据库表

关系统数据库以表的形式向用户表现信息。在一个表中，数据是按行与列进行组织的。 每一**列**（也称为**字段，field**）代表一组数据。 每一**行**（也称为一个**记录，record**）代表一个唯一的由这些列定义的一个实列。

字段总是参考为**域（domain）**,它是定义为一系列的值可供一个或多个字段取用。 例如：tcweek 域是一系列的整数，从 1 到 53.

7.1.2 主键-PK （Primary keys）

每一个数据表有一个字段，或者有多个字段的组合，用来唯一地标识这个表中的每一个记录。 这个标识被称为 PK（主键）。主键是数据库中操作的基础，它提供了在关系型数据库模型 RDBMS 中唯一的记录级的寻址机制。主键在一个表中作为一条记录的参考（reference）

7.1.3 参考--References

在关系数据库中，你可以在多个表中存储数据，你也可以在这些表间定义关系。这意味着每一个表可以保存少量的数据以消除数据的冗余。两个表间存在一个或多个共有字段时便产生了关系。因此，例 如，一个客户详细表可以链到一个订单表，通过包括在这两个表中共用客户代号这个字段。 在客户详细表中，客户 ID 字段是主健 PK。在订单表中，它作为一个外键 FK。这样，就没有必要在订单中记录客户的详细信息了。

需要注意的是：在一个表中的外键 FK 始终是另一个表中的 PK。

7.1.4 索引--Indexes

索引是为了更快地在数据表中查找数据与排序而设置的。一个索引是一种特殊类型的文件（或者文件的一部分），在这这种文件中，每一个实体有两个值，一个是数据值，另一个为指针。 这个数据值即为索引表中进行索引字段的数据值，而那个指针为为指向包括此特定索引字段值的记录位置。 这与我们常用的图书的目录类似，在目录中，指出了要查找内容的页码以方便从书中检索到需要的内容。 需注意的是，可以基于两个或多个字段值去构建一个索引。

每一个表必须至少有一个索引，即 PK 上的索引，这也称为主索引。任何一个基于其它字段的索引被称为第二索引。

7.1.5 事务处理-Transaction handling

事务，是一系列相关的动作，它们必须被当作一个单元。即它们要么全部被执行，要么根本就不执行，不执行部分执行的情况。

一个事务以 commit.transaction()来结束，（此时，有事务中所有的更改都将提交写入到数据库中）或由 abort.transaction()（此时，不会在 DB 中产生任何的更改）来结束。

事务在一个过程的开始时通过函数 set.tranaction.readonly(), db.lock.table(),或进前一个事务结束后开始。中间被取消的事务并且程序结束时没有显示提交取放充调用的将自动进行回滚，提交用 commit.transaction(),放弃为 abort.transaction(). 这种机制的前提为下层的 DBMS 系统支持这种事务机制。

有些数据库行为不可以放在事务中，因为它们不可以被回滚。这么行为如：db.create.table(),db.drop.table(), set.tranaction.readonly(). 这些只能在程序开始时调用或在前一个事务结束后进行调用。

你可以在一个事务前设置 reset point 回滚点。在出错情况下，系统会回退到此处并从那里重新执行。

只读事务是只允许你从 DB 中读取的事务，在整个事务中会保持读一致。这是什么意思呢，就是在整个事务中，你所看到的数据都是一样的，即使别的人改变了它也是如此。只读事务以 Set.transaction.readonly() (这个调用必须是在程序开始或者是前一个事务结束之后)并以是以 commit.transaction() or abort.transaction()结束。由于一致视图将消费大量的内存，因此，保持只读事务尽可能的短，不推建在此事务期间使用用户交互。

7.1.6 锁-Locking

数据库在有多个程序或进程试图访问，更改或删除时会产生数据库的不一致性问题。读不一致会在事务最终提交前对期它处理来说是可见的情况下出现。比如事务必须被依次放弃。

为了避免这种不一致的出夙，BAAN ERP 支持以下的锁机：
记录 / 页锁定，**表锁定**以及**应用锁定**。

为了确保在一个时间内仅有一个处理可以修改一条记录，数据库驱动在第一次处理试图去修改些记录时就进行锁定，在这个锁没有被释放以前，其它的任何处理不可以去更改或删除此记录。然而，他们可以读取此记录。当一个处理在更新一个表，对其它的处理来说保持读取的一致性是非常重要的。读一致性就是说不能看到那些未提交的数据更改，只有当这个事务提交了修改后这个更改才可以被其它处理所看到。

有些 DB 系统不支持读一致性，因此，读脏数据仍是可能的。读脏数据是一个处理更新了数据但还没有提交而另一个处理进程进行数据读取，如果这个更改被放弃了，则被第二个处理进程读取的数据则无效了。有些数据库，如 SYBASE , MS SQL SERVER 6.5,使用一种页锁定技术而不是记录锁定。 那就是说，他们锁定整个的一页而不是仅一条记录，一页是预先定义大小的一个数据块，这个块中的记录数量取决于记录的大小。

7.2 延迟锁--Delayed locks

锁定一个记录比需求的时间还要长时会导致一些不必要的长时间等待，可以用延迟锁在一定程度上解决这个问题。 延迟锁是只有当更改或删除记录时，仅当在执行之前的时刻点才加上去的锁。当一条记录被读取时，它被临时地保存着，当要更新它时，这条记录会从 DB 中重新读取并加锁于它，此时如果记录已有锁存在，则处理进程会返回到 RESET 点并且重新尝试这个操作。当记录没有锁时，系统会对第一次读的数据与第二次读的进行比较，如果发现数据有变化，即在此期间此记录被其它处理进程进行了更改，则 ROWCHANGED 错误会出现并返回到事务，此事务将不会被完成。 如果在比较时没有发现改变，则对 DB 进行更改。

你通过关键字 FOR UPDATE 至 SELECT 语名来标识它是一个延迟锁。
例 子如下：

```
table tpctst999
db.retry.point()
SELECT pctst999.* FROM pctst999 FOR UPDATE
SELECTDO
pctst999.dsca = "...."
....
db.update(tpctst999, DB.RETRY)
ENDSELECT
```

Retry 点在程序中标识一个位置，意思是如要在事务中出现了错误，程序将退回到这个点上。然后，这个事务被重新执行。
在许多情况下，RETRY 点都是很有用的：

- 在记录/页锁的情况下应用延迟锁，一个错误出现会导致系统搪行 abort.transaction(). 在这种情况下，BAAN ERP 只有通知程序那个事务被放弃。然而，如果使用的 RESET 点，系统会自动地再次执行这个事务而用户根本不知发生的这些。
- 有些 DB 在读取脏数据时会产生 abort.transaction(). 在有两个或多个处理进程试图去更改，删除同一记录时会产生 Abort.transaction()。在这些情况下，BAAN ERP 工具可以通过 RETRY 点消除这些部碹同。 它仅是重试这个事务。如果没有设置 RETRY 点，这个事务放弃，而这个 ERP 进程将中止。
- 在 BAAN ERP 中，更新是存放在 BUFFER 中的。因此更新是否成功直到最后的 commit.transaction()时才会确切地知道。如果更新失败，提交事务也会失败，而整个事务必须从头再来。 如查使用 RETRY 点，系统会自动执行这个事务。
- RETRY 点也可以解决死锁问题。 例如系统不能去锁定一个记录，它会回退这个事务后再进行尝试。

RETRY 点在所有的更新程序中被包括是很重要的。RETRY 点必须设置在事务的开始。 下面的例 子说明了如何在你的程序中设置 RETRY 点：

```
db.retry.point() | set retry point
if db.retry.hit() then
...... | code to execute when the system
| goes back to retry point
else
...... | initialization of retry point
endif
```

当 RETRY 点在第一次代码执行时产生后，函数 db.tetry.hit()返回 0。 当系统通过 DB 级返回到 RETRY 点时，这个返回一个非 0 值。 用这种方法来区分是首次执行还是重新尝试。

当系统返回至一个 RETRY 点，它会清空内部的堆栈，局部变量，以及那些在事务中的调用。程序会继续从 RETRY 点执行。全局变量不会被重置。

当事务提交失败，DB 会自动回到事务开始时的状态。 程序会回到最后一个 RETRY 点。 因此，RETRY 位于事务开始是至关重要的。 Db.retry.hit()必须紧接着 db.retry.point()调用。不要把它放在 SQL 的 LOOP 中，否则会让代码看起来很费解。一旦一个 RETRY 点被放在事务中，系统产生一个消息并结束这个进程。

7.2.1 表锁 Table locks

BAAN ERP 提供了表锁定机制，它能够 将整个表中的记录进行锁定。 一个表锁可以阻止其它进程对这个表中的记录进行修改或删除，但是不可以阻止读取记录。
当一个事务需要大量的对记录进行加锁时，用表锁将非常有用。
加表锁的方法：db.lock.table()

7.2.2 应用锁

在某些操作中，需要防止其它进程对数据进行读取删除与修改，应用事务锁可以达到此要求。它不是事务的一部分，因此在事务处理结束后，它不会自动被消除。然而，它可以在应用程序结束后或使用 appl.delete()来删除。

7.3 MS SQL 数据库驱动

这一节中我们描述以 MS SQL SERVER 的 RDBMS 接口的相关内容。

由于需要大量的表，因此需要一种转换去命名表，表中的列以及表中的索引。在这一节中我们介绍在 DB 接口中的数据字典 DD 以及在 DB 接口中的名字转换，同时也讨论如何将 BAAN ERP 的数据类型映射到 MS SQL SERVER 中的数据类型上去。

BAAN ERP 数据字典 DD 映射 BAA 民 ERP 数据类型，域 DOMAIN ，模式 schema 以及参考完整性信息。 当在 RDBMS 中检索数据时，数据库驱动程序映射数据字黄信息到数据库表定义中。 BAAN ERP 数据字典 DD 信息保存在共享的内存中，它可以被运行的所有 BAAN ERP 应用服务使用。 在一个数据库驱动中它为所有的进程所共享使用。

BAAN ERP 中的数据类型不能在 RDBMS 中直接使用，这是因为并非所有的 BAAN ERP 数据类型在 RDBMS 中都可以直接找到确切的对应关系。 为了在 RDBMS 中创建一个表，数据库接口必须要做一些转换或叫做翻译。 当进行这种转换时，数据表名，列名索引名的转换将被使用。

7.3.1 表名的转换

存在 SQL SERVER 中的表对外部的名称有着如下的格式：

```
t(PackageCode)(TableName)(CompanyNumber)
```

T+ 子系统代号+ 表名+ 公司名

PackageCode 子系统代号➔**两字母**

用两位字母表示，代表此表归属于哪一个子系统。 例如，属于 TOOLS 部分的表用 TT 表示。

TableName 表名字➔**三字母+三数字**

用三位字母代表模块，后面再跟上三位数字。模块表示此表的归属，而后三位的数字就是顺序号。

CompanyNumber 公司代号➔**三位数字**

用三位数字代表不同的公司实例 。

000 公司代表系统的元数据，它代表对其它公司均有用的一些变量和系统值。

7.3.2 列名的转换

BAAN ERP 数据字黄中的每一列代表着在 SQL 数据库中的一外或多列。列适用如下规则：

通用➔General

在 SQL 中的列名前加 t_. 例如，BAAN ERP 中列如 cpac 在 SQL 中的列名为 t_cpac. 如果 BAAN ERP DD 中的列如含用句点（period）它将用下划线代替。

长文本字段➔Long string columns

BAAN ERP DD 中的列可以超过 SQL 中的列长度的限制。SQL 中的 CHAR 字段类型有着 254 字符长度的限制。 当超过长度的字段转换时，第一字段长度为 254 位，字段名后用"_1"来表示。 第二个字段用"_2"表示另外的 254 个字符依次类推。

例 如字段 desc 在 BAAN ERP DD 中有 300 个长度，在 SQL 将用两个字段来表示。它们的字段名分别为"t_desc_1" 254 长度，第二个为"t_desc_2" 长度为（300-254=46）

表 1： BAAN ERP 与 SQL 数据类型的关系对比.

BaanERP 数据类型	MSQL 数据类型
INT	Smallint
LONG	Integer
FLOAT	Real
DOUBLE	Float
CHAR	Binary(1)
STRING(N)	Char(n)
TIME	DateTime
DATE	DateTime
TEXT	Integer
BITSET	Integer
ENUM	Binary(1)

数组列

在 BAAN ERP DD 中，数组列可以被定义。 一个数组即为多个元素组成的一列。 元素的个素表示为深度。 例如，一个列包括有日期，可以被定义为包括三个元素的一个数组，三个元互分别代表年，月，日。 在 SQL SERVER 中，这三个元素被当成三个独立的字段进行定义。 三个字段的名称为字段的名称加上后辍，后辍用数据中的元素号来代替。

例 如：一个数组叫做 DATE，包括三个元素，在 SQL SERVER 中分别是

T_Date_1 代表第一个数据元素的列，t_Date_2 代表第二个元素的列。

7.3.3 数据类型的映射

表 1 显示了 BAAN ERP DD 中的数据类型与 SQL SERVER 中的数据类型的对比。
注意： 在 MSQL 数据库驱动中用 SQL SERVER 中的 CHAR 数据类型，因为用 CHAR 可以进行数据的比较而不用 VARCHAR 以达到以下结果。

7.3.4 数据的映射规则

- 除了上述的规则外，在 BAAN ERP 数据映射到 SQL SERVER 数据中时，以下规则适用：
- SQL SERVER 对待对象名称大小写区分，因为要采用二进制排序；
- 所有的由 MSQL 驱动创建的列有 NOT NULL 的约束，BAAN ERP 不支 SQL 中的 NULL 值概念。
- BAAN ERP 中的日期范围与 SQL 中不尽相同。 BAAN ERP 中的日期在 SQL 中有些无效。BAAN ERP 中最早的日期在 SQL 中表示为 1753-1-2 而最晚为 999-12-31.

7.4 ODBC 接口

ODBC 是一个应用于与 DB SERVER 通信的 API 接口。它包括许名库函数的调用，外部程序调用 ODBC 中的函数以执行 SQL 语句来达到与数据源通信的目的。

被 MSQL 数据驱动调用的 ODBC 功能可以完成以下功能：

- 连接到 Microsoft SQL Server (open session)
- 分配语句句柄；
- 分解 SQL 语句
- 绑定输入的变量；
- 定义结果变量
- 执行 SQL 语句；
- 读取结果行；
- 提交或放弃事务
- 关闭，解开或删除光标
- 断开 MSQL (close session)

8 编程

8.1 介绍

随 BAAN ERP TOOLS 中所自带的开发工具可以让开者人员去为已存的 BAAN ERP 开发附加的功能，也可以去开发一个全新的应用系统。这个开发工具提供了如下的功能：

- BAAN ERP 3GL 开发语言；
- 4GL 语言功能，你可以去增加或修改进程、报表、DAL 的默认行为；
- BAAN SQL， 你可以从 DB 中检索数据；
- 对 DLL 的支持；

3GL 是开发程序的脚本，它用于那些没有 FORM 的进程或者根本不连接到进程的程序。它与 4GL 引擎没有任何关系。当你要创建这类程序时，你必须指定整个地程序流程，包括主程序在这里面，你不可以使用 4GL 里的事件以及函数。

在 BAAN ERP 应用系统中，4GL 提供了一个进程许多默认的功能。你可以对连接到一个进程的 4GL 脚本进行增加或修改它以改变它们默认的功能。4GL 脚本是面向事件的。它们包括一个多个事件段，4GL 引擎在特定的状态去那个事件段中相对应的功能。在以前的软件版本中，对一个进程的默认功能进行更改或增加是在与这个进程相关联的单独的一个文件中进行的。 在 Baan ERP 中，用户的接口行国（UI）与数据库的行为进行了分离。现在，DAL（数据访问层）处理与 DB 的接口行为。 开发人员创建一个 UI（用户接口）脚本去更改一个进程的默认行为。它们创建了一个 DAL 脚本去处理了与这个特定表相关的所有逻辑规则，即与 DB 中的表相关的所有事务逻辑规则已全部在那个 DAL 中封装完成了。在事件段中的程序语句，你可以使用 3GL/4GL 的一个组合去完成。

报表脚 本是连接到一个报表的 4GL 脚本以增加或修改它的输出内容。在一个报表脚本中，你可以进行编程以在便在报表执行的特定阶段去执行你所期望的功能。

你可以在 4GL 脚本中使用 BAAN SQL 语句去从数据表中检索数据。这里提供了两种使用的方法：

将它嵌入在语言中；

通过动态 SQL 去调用；

Bshell (BAAN ERP 应用服务器)为 BAAN ERP 应用提供了多任务执行环境。每一个 bshell 可以执行与调度多个并行的处理。

8.2 3GL 编程语言特性

8.2.1 数据类型

共有五种变量的数据类型：long,double, string, table, domain variables.

Long

变量可以是从-2147483648 到 2147483647 的任何值。对于超过这个范围的，你需要使用 double 型。对于 LONG 型的，分配两个字节的长度的存贮区域。

Double

任何包括一位小数点的有效长度为 15 位的值。 分配的存贮空间为 8 字节。

String

用于符号，描述，短文本的变量。 最大长度为 1024. 它可以被申明为多字节字串，以处理多字节和双向字符。在单字节字串中，每一个字节包含一个字符。
然而在多字节的字串中，一个字符可以存在一个至四个字节中。

Table

在程序中申明用于访问数据表。这个表必须是在 DD 中已定义过的。

Domain

在 DD 中已定义过的 domain 的变量。它可以以任何以下的数据类型：long, byte,integer, date, enumerate, set, float, double, string, text. 每一个在 DD 中定义过的 domain 都可以在你的程序中进行申明。

8.2.2 编程语句

BAAN 3GL 由以下语句组成：

- 变量赋值；
- **IF ...THEN ...ELSE ...ENDIF ON CASE** 语句
- **WHILE-, REPEAT- and FOR- Loops with BREAK and CONTINUE** options
- 过程调用

除了标准的 LOOP 结构外，还有一个内嵌的 SQL LOOP 结构用于 SQL 查询表达式中。（通过标准的（SELECT ...FROM...WHERE ... 语名）后面跟随 **SELECTDO ...SELECTEMPTY ...ENDSELECT**。

在 SELECTDO 中的块会对选择的每一行进行一次重复，然而，SELECTEMPY 仅当结果为空时执行一次。

SELECT XX
FROM XXX

WHEREXXX

SELECTDO

每个记录重复执行的语句块

SELECTEMPTY

仅当结果集为空时执行一次

ENDSELECT

8.2.3 例子

```
FUNCTION LONG compnr_check( LONG new_compnr )
{
TABLE tpctst999
SELECT pctst999.* FROM pctst999
WHERE pctst999.compnr = :new_compnr
ORDER BY pctst999.compnr


SELECTDO
compnr = pctst999.compnr
SELECTEMPTY
RETURN(FALSE)
ENDSELECT
RETURN(TRUE)
}
```

8.3 4GL 编程语言特性

当你创建一个进程，这个进程产生器会产生一段标准的源代码。这个标准的源代码提供了进程的默认功能。如果你所需要的功能没有在标准的源代码中实现，你可以在 4GL 中进行增加。

在 BAAN IV 中，对进程默认功能的修改是在与此进程相关联的唯一的脚本文件中。 与用户的接口 UI 以及与 DB 数据库的访问接口 DAL 都是在同一个脚本文件中。在 BAAN ERP 中，UI 为和 DB 的行为进行了分离。数据访问层（DAL）现在处理 DB 的交互行为。 程序员创建的 UI 脚本去处理进程的的交互行为。开发人员创建了 DAL 已封装了对于这个特定的数据表的所有的相关的逻辑与规则。

8.3.1 4GL 脚本类型

共有四种 4GL 脚本：

- 类型 1：单个（详细）进程作用一个主表。
- 类型 2：多个（概述）带有分组字段的进程，作用于一个主表
- 类型 3： 多个（概述）不带分组的进程，作用于一个主表
- 类型 4： 打印/处理进程，不带主表

4GL 脚本是面向事件的。它们包括一个或多个事件段，在执行程序的特定阶段去执行特定的相对应的程序段。 比如，当一个进程开始时，窗体激活前，字段输入前，字段输入以后等等。没有必要去对所有的事件段进行编程，仅仅去对那个在标准代码中没有提供所要求功能的地方去进行编程。 你可以混合使用 3GL/4GL 进行编程。

8.3.2 例子

```
field.pctst099.item:
check.input:
SELECT pctst001.* FROM pctst001
WHERE pctst001.item = :pctst099.item
AS SET WITH 1 ROWS

SELECTDO
....
SELECTEMPTY
pctst001.dsca = "*****"
set.input.error(".....")
ENDSELECT


before.input:
if ..... then
attr.input = false
endif

after.display:
SELECT pctst001.* FROM pctst001
WHERE pctst001.item = :pctst099.item
AS SET WITH 1 ROWS
```

```
SELECTEMPTY
pctst001.dsca = "*****"
ENDSELECT
field.pctst099.date:

before.input:
attr.oformat$ = "%D002,2"
```

8.4 报表脚本

BAAN ERP 报表是用于将 DB 中的数据输出到各种形式的外部设备上（比如：打印机，显示屏，文件等） 报表的布局与内容在 DD 中已被定义。 你可以连接一个报表脚本结报表。 在报表脚本中，你可以在报表执行的特定阶段去执行特定的程序以更改或增加默认的功能。比如，你可以创建一个脚本去对报表上的一个数据进行计算，或者从相关的数据表中读取关联的数据，而这些是在默认的报表中是没有的功能。 报表脚本与 4GL 编程脚本相同，除了报表的事件段和一些特殊的函数功能外。

报表脚本包括一个或多事事件段，在那里你可以增加需要在特定阶段执行的脚本。语名由 3GL 与报表函数组合而成。 报表脚本支持以下的事件段：

程序段

报表段

文本字段等

8.5 数据访问层 Data Access Layer (DAL)功能

8.5.1 概述

在 BAAN 应用中，标准的程序提供了对一个进程的许多默认功能。 在以前的版本中，所有的处理均在与进程相关联的一个脚本文件中。 在 BAAN ERP 中，用户接品 UI 与数据访问 DAL 进行了分离。 DAL 现在仅处理与数据库的交互行为。因此，DAL 确保了数据库逻辑的完整性，在以前的软件版本中，DBMS 来确保数据参照的完整性。

对特定的表的 DAL 脚本与这个表有着相同的名字。它是以 DLL 的方式实现，并且可以被 UI 中进行调用(通过标准程序)，或被其它 DAL 调用，或被其它的外部程序（通过 CDAS，common Data Access Server）.下图说明了这整个的关系：

8.5.2 数据库集成检查

下面是一些逻辑完整性规则的一些可被编入 DAL 程序脚本的例子：

- 当一个客户达到了它的信用限制，则他就不可以再订购更多的货物;
- 如果一个订单的发票已被打印，则此订单不可以再被修改；
- 如果当前的 PVRC 不等于程序脚本的 PVRC，脚本不可以被编译；

在分离的 DAL 中进行编程进行数据库完整性的检查，有着以下两个好处：

- 代码重用： 这些完整性规则不需要在使用这个表的每个一进程中进行复制；
- 外部访问： 外部的应用程序可以通过 CDAS 和 DAL 进行 DB 的访问；

对于 UI， 标准代码和 DAL 的交互的概述，请参考 8.5.4.

8.5.3 业务方法 Business methods

除了执行数据完整性检查，DAL 提供了一些业务方法，它用于处理非数据库更改交互比如打印销售订单，存档所有的订单。 一个业务方法即是一个执行一项特定功能或在数据库中检查一个或多个数据表的一个函数。这个函数编程在 DAL 中，它可以在 UI 中直接调用。 对于大多数据数据表，它必须编程在 DAL 中。

用户可以通过一个简单的命令去激活一个业务方法，然后不需要交互的参与。在 UI 中调用 DAL 中的函数，它执行相应的功能去完成此项任务。 用户必须等到这项任务完成后才可以去执行另一项内容。然而，如果提供了操作的窗体，用户也可以通过 CANCEL 按键来取消它。 UI 脚本调用 DAL 中的业务方法是通过调用 dal.start.business.method()函数的。

8.5.4 UI, DAL 以及标准程序间的交互

DAL 脚本中包括着对特定对象的所有逻辑完整性规则。 这些规则被称为勾子（Hook）并且它们可以在对对象集中的每一个对象的处理中被执行。对每一个有一个主表的进程，标准程序确信对这个表的完整性规则会在每一次的更新，删除插入以及读取操作中进行执行。假设没有 DAL 与那个对象相连，则没有逻辑完整性被检查。(险非他们在 UI 程序中自己去处理这类事情）

DAL 脚本可以包含有这样的 HOOK，它阻止数据库的访问，并且数据被传回到 UI 中。 前者（阻止对数据的访问）在执行前检查，后者（阻止数据传回到 UI）会在数据库行为后执行检查。

如果一个用户在窗体上更改了客户的地址，标准程序通过 DAL 去更改客户的地址，如果你

要求了特定的限制在那个更新的动作上，则你编写一个勾子 HOOK 属性在 DAL 用以检查这个以实现那个特定的限制。

8.5.5 UI 函数调用

UI 脚本可以使用 DB 写函数和 DAL 方法（DAM）去处理数据库。 数据库写函数是对 DB 的直接调用。它们不使用 DAL。 在这种情况下，任何对于逻辑的完整性检查要在 UI 中自己去进行。 这就就是这些编写在 UI 中的对逻辑完整性约束的代码不会被其它的地方重用。如果通过 DAL，则所有必须的检查均会自动执行。

当用 DAL 去处理 DB 时，下面的主要步骤将被包括：

1. 用户发布一个命令，通过 UI 去访问 DB 中的一个记录。
2. 标准程序加载相应的 DAL 并且执行 DAL 的 HOOK 去检查那个对象的完整性。
3. 如果特定的行为被允许，程序产生相应的数据库调用。
4. 当 DB 被更新，DAL 执行进一步的检查去决定是需要数据返回到 UI。
5. 标准程序将数据返回到 UI（假定那个完整性规则允许这个）

8.5.6 DAL hooks

勾子 HOOK 就是一个函数，有着预定义的名称，DAL 开发人员在 DAL 中进行了编程。它的用处是用于对据库访问进行逻辑完整性检查。DAL 脚本被编译成 DLL。

当一个用户发出一个命令去访问 DB 时，标准程序装载将要访问对象的 DAL 的 DLL 文件并且调用 HOOK 去执行完整性检查。假设这个要访问对象的 DAL 脚本存在，标准程序永远会确保在正确的时间上去调用 DAL 的 HOOK。

DAL 脚本包括两类 HOOK， **属性 HOOK** 和**对象 HOOK**。

《末完待续》