

Oracle HTML DB 最佳应用方法

Oracle 白皮书
2004 年 6 月

引言.....	3
应用程序开发目标	3
开发.....	3
数据库设计.....	3
执行完整性约束条件	3
填充主键.....	4
开发效率.....	4
UI 缺省值.....	4
间接引用常用字符串	4
页面组件重复使用	5
使用向导有利于工作	5
错误和异常处理.....	5
部署.....	6
组织工作区	6
部署应用程序.....	6
管理数据库对象.....	7
管理静态文件.....	7
使应用程序可以移植.....	8
用户界面一致性和可用性.....	9
将样式和布局与数据和应用程序逻辑分离	9
维持多个应用程序用户界面的一致性	10
安全性.....	10
由数据库开始确保安全性.....	10
管理用户.....	11
URL 篡改.....	11
跨网站指令码攻击.....	12
SQL 指令植入式攻击.....	13
保护报表列.....	14
性能.....	14
使用连接变量.....	14
恰当地选择分页样式.....	16
尽可能使用已公布的逻辑.....	17
使用重定向.....	17
结论.....	17

Oracle HTML DB 最佳应用方法

引言

Oracle HTML DB 是一种在 Oracle 数据库上快速开发 Web 应用程序的工具。本白皮书提供了一系列使用 Oracle HTML DB 开发及部署应用程序的最佳方法。

应用程序开发目标

有很多特征有利于 web 应用程序的成功。这样的特征包括：

- 易于开发
- 易于部署
- 易于维护
- 易于使用
- 运行速度快
- 安全
- 成功率高

本白皮书提供开发具有这些特征的应用程序的指导原则。本白皮书中描述的内容并不是严格的固定规则。但是，它们是根据在生产中开发及运行 Oracle HTML DB 应用程序的经验总结出来的建议和最佳方法。

开发

Oracle HTML DB 是一种十分高效的工具，使您可以在几小时内，甚至在几分钟内在 Oracle 数据库上构建并部署功能完全的报表和数据入口应用程序。由于此工具非常灵活，因而经常可采用多种方法以实现同一目标。本节中的建议可帮助您以最有效的方法达成目标。

数据库设计

执行完整性约束条件

如果没有正确地设计数据库，则很难创建基于关系数据库的应用程序。正确的数据库设计包括引用完整性和唯一性约束条件，它们确保数据的一致性和逻辑有效性。即使您认为已在应用程序中实施了足够的有效确认或引用完整性规则，往往也无法确保他人没有通过其他方式访问数据库。虽然本白皮书中没有介绍如何对数据库设计进行完整调查，但是，此处建议利用数据库的固有功能维护数据的准确性和完整性。并在适当的地方使用主键、外键、非空限制和检验限制。

填充主键

Oracle HTML DB 中包含在数据库表上创建数据入口表单的向导。该向导创建提供插入、更新和删除功能所需的所有 UI 控件和进程。如果您在使用此表单向导及其生成的自动化 DML 进程，则使用数据库触发器和序列填充主键列通常很有帮助。例如，如果拥有一个表 T，其中包含主键列 ID 和序列 MYSEQ，则您可以创建一个触发器 MY_TRIGGER，如下所示：

```
在表 T 的每行中插入内容之前创建或更换触发器 MY_TRIGGER
begin
    if ( :new.ID is NULL ) then
        select MYSEQ.nextval
        into :new.ID
        from dual;
    end if;
end;
```

开发效率

UI 缺省值

使用 HTML DB 中的 UI（用户界面）缺省值可以将缺省的用户界面属性赋予指定模式中的表、列或视图。这些 UI 属性用于屏幕元素，例如字段标签、格式掩码、列标题和列对齐。当使用向导创建表单或报表时，向导使用该信息为区域和项属性创建缺省值。

根据需要预先设置 UI 缺省值可以节省开发时间，因为不必对 UI 属性进行繁琐的调整。

间接引用常用字符串

如果应用程序经常引用一个或多个可能在开发应用程序期间或之后发生更改的字符串，则应该使用替代字符串。替代字符串是在应用程序级别定义的静态变量。可以在整个应用程序中使用语法 &STRING_NAME 引用这些变量的值。例如，如果应用程序经常引用一个产品名称，而该名称将发生改变，则可以定义一个替代字符串 PRODUCT_NAME 并在区域中引用该替代字符串，如下所示：

```
Welcome to the &PRODUCT_NAME. store!
```

正确使用替代字符串可以使每次产品名称发生改变时，只需在一个地方进行修改。

页面组件重复使用

如果一个页面组件（例如，某个带有静态 HTML 的区域或某个列表）必须在应用程序的多个页面上显示，则考虑使用“零页”。应用程序中的零页不会由最终用户直接运行，但是其中的组件可通过引擎在每个页面视图上显示。可以使用条件有选择地在其他页面中显示零页上的某些组件。

使用向导有利于工作

HTML DB 提供了若干生成功能完全的应用程序组件的向导，从而节省了大量手动构建这些组件的时间。HTML DB 的独特之处在于，使用向导不会牺牲控制或灵活性。即使组件和控件已经生成，也可以对它们进行编辑以修改其外观或运行方式。表向导中的表单十分有用，因为它不但可以提供能够处理插入、更新和删除的功能完全的表单，还可以通过乐观锁定提供丢失更新检测。

其他可以帮助加速开发过程的向导包括快速应用程序向导和创建向导的向导。快速应用程序向导对单个数据库表生成完整的数据入口报表和分析应用程序。“向导向导”按此方式命名的原因是，它生成一个向导，即多页数据集合用户界面。该向导生成一系列包含完整的表单字段、控件和页面流的页面，使用“下一页”和“上一页”按钮在页面之间导航。

强烈建议在从头开始构建应用程序功能之前尝试使用 HTML DB 中的向导。

错误和异常处理

HTML DB 应用程序通常在匿名块或程序包和存储过程中执行 PL/SQL 逻辑，例如在数据库中插入或更新数据。如果调用的 PL/SQL 出现异常，则应该对其进行适当的处理。向用户发出发生此错误的警告，但不中断应用程序流的建议方法是，将错误信息分配给异常事件中的 HTML DB 项。此项可以显示在调用异常进程的页面上。最后，可以使用条件分支将用户重新送到出现错误的页面，使其可以进行更改并再次提交页面。

例如：

第 5 页包含一个数据入口表单，其中的值将使用名为 MY_PROC 的过程插入到数据库中。如果插入操作成功，用户应被引导至第 1 页。如果发生错误，则将给予用户更正错误的机会。要在第 5 页上显示对用户友好的错误信息，请创建名为 P5_ERROR_MSG 的项。定义之后的提交过程，如下所示：

```
BEGIN
```

```
MY_PROC (:P5_ITEM1, :P5_ITEM2);  
:P5_ERROR_MSG := null;  
EXCEPTION when others then  
:P5_ERROR_MSG := 'Error in insert, please correct  
values and try again';  
END;
```

可以使用以下源代码在第 5 页的 HTML 区域显示这些错误信息：

```
&P5_ERROR_MSG.
```

如果未出现异常，P5_ERROR_MSG 将为空，因此该页面上不会显示任何信息。

最后，要想确保用户有机会更正导致出现异常的错误，请创建一个条件分支，在 P5_ERROR_MSG 的值为非空时指向第 5 页，在该值为空时指向第 1 页。

部署

组织工作区

在 HTML DB 中，工作区可供一个或多个开发人员以一种或多种模式在数据库对象上开发应用程序。作为规则，应该对工作区进行组织，以便使它们包含与彼此相关的应用程序。即，包含由公共开发人员小组开发的、在常用数据组和模式上运行的应用程序。

部署应用程序

在 HTML DB 中，应用程序由以下组件构成

- 数据库对象
- HTML DB 应用程序定义
- 图像（可选）
- 层叠样式表（可选）

要将应用程序部署到其他的数据库（未在其上开发此应用程序），您必须部署应用程序定义以及所依赖的数据库对象。根据需要，您可以选择部署图像和层叠样式表 (CSS)。

根据应用程序的可访问性要求、用户群体规模及其他因素，可以确定将开发环境与生产环境分开的行需要达到怎样的准确程度。对于某些应用程序，可以将开发服务器与部署服务器合并，只要最终用户能接受偶尔发生的应用程序无法访问或短暂中断的情况。

其他应用程序可能需要两个（开发与生产）甚至三个（开发、测试与生产）服务器。在 HTML DB 中，可以使用应用程序定义的导出和导入设备在不同环境之间移动应用程序。

如果只有一个硬件可以运行该数据库和 HTML DB，您仍然可以通过使用两个访问独立模式的工作区将应用程序的开发版本与其生产版本分离。一个工作区将供开发人员使用，另一个工作区可供生产时部署应用程序。

管理数据库对象

如果将应用程序部署到的工作区和模式与开发该应用程序的工作区和模式相分离，数据库对象可能需要随应用程序移动。如果使用 Oracle HTML DB 执行所有数据库定义语言 (DDL)，那么您有两种方式将更改应用到部署环境。一种方式是在 SQL Workshop 中用脚本管理 DDL。使用 SQL Workshop 可以编辑、创建、导出和导入脚本。如果在 SQL Workshop 中用脚本捕获所有 DDL，您可以通过导入并运行这些脚本在部署环境中安装数据库对象。

第二种方式是使用导出 DDL 功能，该功能也可以在 SQL Workshop 中使用。您可以使用此功能以脚本的形式为模式中的任何数据库对象导出 DDL，而并不考虑其创建方式。因此，重新创建这些生产对象只是导入生成的脚本的一种方式。

管理静态文件

HTML DB 应用程序可以引用其他外部文件，例如层叠样式表 (CSS)、图像和 Javascript 库。可以将这些图像放置在运行 web 服务器的服务器的文件系统上，或在使用 HTML DB 的应用程序构建器上载这些图像时，将其存储在数据库中。

使用应用程序构建器在数据库中管理图像和 CSS 等文件的优点是，仅使用浏览器即可将它们导出并导入到另一个 HTML DB 环境或工作区中。但是，建议仅在具有相对较少的图像以及不期望应用程序承担较高的请求吞吐量时，才将相关文件存储在数据库中。当将图像、CSS 和 Javascript 库存储在数据库中时，每次浏览器请求这些文件之一时都要建立数据库连接。在吞吐量较高的应用程序中，这可能导致对数据库不适当的和不希望的损耗。

使应用程序可以移植

要确保可以轻松地将应用程序从一个 HTML DB 环境移动到另一个环境，您应该避免对可能在环境之间更改的值进行硬编码引用。例如，应用程序 ID 经常在应用程序的 URL 中引用。HTML DB 引擎使用应用程序 ID 唯一标识从应用程序库中获取并运行的应用程序。不要使用实际的应用程序 ID，如下面在 URL 中的操作

```
f?p=100:1:&SESSION.:
```

建议您改用间接的引用：

```
f?p=&APP_ID.:1:&SESSION.:
```

其中，&APP_ID. 是替换硬编码值 100 的内置替代字符串。使用此技术，不必在部署之前对应用程序进行任何修改，即使应用程序 ID 发生更改的情况也是如此。

同样，要确保顺畅的部署，对模板、HTML 区域或 SQL 查询中图像的引用应该使用替代字符串来替换硬编码路径。例如，假设一个应用程序的图像存储在开发环境中的虚拟路径 /dev/images/ 中的文件系统中。当将应用程序部署到另一个服务器时，图像存储在虚拟路径 /prod/images/ 中。使用在应用程序级别定义的替代字符串，可以不必在部署之前更改整个应用程序。例如，定义一个应用程序级别的替代字符串 IMAGE_PATH，并为其赋值 /dev/images/，然后，在查询中引用该字符串，如下所示：

```
SELECT '' icon,  
ename,  
job  
FROM emp
```

要在模板中引用图像，也可使用此方法。例如，页面模板的“标题”部分可能类似以下内容：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN">  
  
<html>  
  
<head>  
  
#HEAD#
```

```
<link rel="stylesheet" href="&IMAGE_PATH.custom.css"
type="text/css">

<title>#TITLE#</title>

</head>

<body bgcolor="#FFFFFF" #ONLOAD#>#FORM_OPEN#
```

在此示例中，`custom.css` 是存储在文件系统上的层叠样式表。如果使用此技术，则在部署时只需对应用程序级别的替代字符串 `IMAGE_PATH` 的值进行一次更改。

用户界面一致性和可用性

将样式和布局与数据和应用程序逻辑分离

HTML DB 提供了大量可选择的组件，以便加速应用程序的构建，其中包括：

- 报表
- 表单
- 列表
- 导航条
- 菜单

这些组件中的每一个都使用模板呈现，允许您定义与样式分离的结构。

要维持样式、逻辑和数据访问之间的完全分离，请避免对这些组件的实施进行自定义编程，除非它确实不可避免。尽可能使用内置的组件呈现 HTML，而不要编写自定义代码来呈现 HTML。这样，就很容易更改应用程序的外观和风格，而不会影响其逻辑。例如，在 HTML DB 中，报表是 SQL 查询的格式化结果。要使 EMP 表中的 `ename` 列以粗体显示在页面上，可以使用以下查询创建报表：

```
SELECT '<strong>' || ename || '</strong>', job
FROM emp
```

最好使用不包含任何标记的查询，然后使用模板或应用到结果集中的 HTML 表达式来达成同一目标。

使用 HTML DB 的公开组件时，您可以从以下几方面获益。首先，您可以开发应用程序的内容，而不必关心样式和布局。可以在以后通过应用适当的报表模板来添加报表的格式。报表模板可以在以后由同一个开发人员添加，也可以由另一个开发人员同步设计。

维持多个应用程序用户界面的一致性

企业通常要求所有企业内部网或互联网应用程序具有一致的外观。如果需要维持多个应用程序的外观和风格一致，请使用模板预订。模板预订是对在中央位置定义的主模板的引用。最佳方法是定义一个引用应用程序，该应用程序包含模板的主副本。然后，其他应用程序可以向这些主副本预订其模板。每次对主模板进行更改时，这些更改将在预订主模板的模板中实现。

安全性

保证 web 应用程序的安全是付出很多努力仍然难以解决的一个问题，尽管已经出版了很多关于此方面的书籍。以下内容并不是保护应用程序的完整的解决方案，而只是对经常讨论的安全问题的一些建议。

由数据库开始确保安全性

仅依赖应用程序提供安全性和访问控制是远远不够的。就好像，您不会因为房子的前门安了一把好锁，就把成叠的现金放在咖啡桌上，准备让可以通过窗户爬进来的人把现金拿走。相反，您会把现金放在安全的地方。

同样，您构建的应用程序并不是获取数据库中底层数据的唯一方式。还存在其他方法，例如即席查询工具或直接 SQL*Plus 访问。如果在数据库级别构建了正确的访问控制，您就可以放心操作，不必为访问数据库中数据的每个应用程序重新实现安全策略。

要在数据库中实现安全性，请确保用于分析 SQL 和 PL/SQL 的模式只能访问应用程序所必需的数据。如果需要访问模式 A、B 和 C 中的某些数据而不是全部数据，请考虑创建一个附加模式 D，其中包含在 A、B 和 C 中所需对象的顶部定义的视图。

要基于查询表的用户限制对表内行的访问，请使用高精度访问控制 (FGAC)。高精度访问控制，有时称为虚拟专用数据库 (VPD)，是一种基于策略限制对表或视图内行的访问的机制。这些策略基于应用程序上下文修改查询的判定或 WHERE 子句。使用应用程序上下文，可以确定谁在什么时间从哪个应用程序启动了该查询。可以基于这些事实构建 WHERE 子句，以便只向用户返回适当的和允许的数据。

利用 FGAC，只需在数据库对象上定义一次安全策略。访问底层数据库对象的每个应用程序仅继承数据库中的访问控制限制。不必在应用程序中重新实现同一安全策略。

管理用户

在构建和部署 HTML DB 应用程序时，您可以选择管理最终用户凭证的位置和方式。在缺省情况下，最终用户被定义为该应用程序所属的工作区中的用户。部署大的用户群体时，强烈建议您使用外部身份管理解决方案，例如 LDAP 服务器或一次性登录基础架构，例如 Oracle 应用服务器提供的基础架构。一直在企业中集中定义用户定义和凭证可以为应用程序管理员以及最终用户减少密码管理的负担。应用程序管理员不必经常帮助用户重置密码，而最终用户只需记住几个用户名和密码，因为可以使用同一密码登录到多个应用程序。并且，如果集中管理用户的凭证，也简化了用户在离开企业时取消访问所有应用程序的过程。

URL 篡改

基于 web 的应用程序，包括在 Oracle HTML DB 中开发的应用程序，经常通过 URL 将值从一个页面传递到另一个页面。聪明的用户可能会发现这一点，并通过在其浏览器的地址字段中键入自己的值来覆盖某个值。例如，在显示用于编辑员工记录的表单的页面上，特定项的会话状态中的值可以确定显示哪个记录。在 HTML DB 中，可以在包含以下内容的 URL 中设置此项的会话状态：

```
f?p=&APP_ID.:1:.....:P1_EMPID:1234
```

其中 100 是应用程序 ID、1 是页面 ID、P1_EMPID 是会话状态变量，1234 表示员工表中的主键值。

当用户单击此类链接时，它将显示在浏览器的地址字段中。此时，用户可以试用 URL 并将其修改为：

```
f?p=&APP_ID.:1:.....:P1_EMPID:5678
```

以便使用主键值 5678 检索员工记录。可以允许也可以拒绝未确定身份的员工查看员工 5678 的详细信息。如果不允许用户查看，则应该对页面上的逻辑应用相应的授权规则，甚至还可以在数据库级别防止该用户查看此数据。通过编码或加密表单遮蔽在 URL 中传递的值是远远不够的。

即，不要尝试通过遮蔽在 URL 中传递的值来保护应用程序。而应该使用数据库级别安全策略保护 URL 转到的目标。此外，还可以使用 HTML DB 授权方案。

跨网站指令码攻击

跨网站指令码攻击，有时称为 XSS，是利用动态生成的 Web 页的安全漏洞。在 XSS 攻击中，web 应用程序使用在信任用户的浏览器进行读取操作时激活的脚本发送。该脚本可以窃取数据甚至会话凭证，并将其发送给黑客。这些攻击很少是由应用程序开发人员发起的，但是防御这些攻击却是应用程序开发人员的职责。由于动态网站依赖用户的输入，因此具有不良企图的用户可以通过将恶意脚本隐藏在合法请求中的方式在页面中输入恶意脚本。如果开发人员不能防止应用程序发送伪装成善意用户输入的恶意脚本，那么善意用户可能会遭受该脚本的攻击。

要抵御这些攻击，首先要过滤所有用户输入中的异常字符和恶意输入。其次，应过滤所有浏览器输出以防止不小心将内容呈现为 HTML 或脚本。HTML DB 提供大量创建应用程序组件的向导，它们自身不会向应用程序暴露这些安全问题。但是，当您使用附加功能和自定义代码扩展应用程序时，必须始终严密防范，例如，通过对每个数据元素进行端到端的分析，从作为输入引入一直到发出供浏览器显示都要进行防范。使用以下语句在页面上的 PL/SQL 动态内容区域发出名为 P1_ITEM 的应用程序项的情况就是后者中的一个示例：

```
http.p(:P1_ITEM);
```

如果 P1_ITEM 的值（在通常情况下，可能由某些内部应用程序逻辑建立）被黑客使用以下 URL 引入该页：

```
f?p=APP:PAGE:::::P1_ITEM:<some malicious javascript>
```

发出页将向浏览器发送恶意脚本以执行其破坏活动。开发人员应该意识到这种可能性，并过滤在该页中使用的输出。建议的方法是使用 `htf.escape_sc` 等函数，提供 `mod_plsql` web 工具包。此函数将 `<` 和 `&` 等标记和字符转换为实体引用 `<` 和 `&`；例如：

```
http.p(htf.escape_sc(:P1_ITEM));
```

SQL 指令植入式攻击

SQL 指令植入式攻击是对数据库驱动的网站的一种攻击形式，攻击者通过利用网站代码中的缺陷执行未经授权的 SQL 命令。要防止此类攻击，不要在应用程序运行的 SQL 查询中包括任何用户输入。考虑将以下匿名块作为报告源使用。此处，查询是基于用户输入动态构建的，存储在名为 P1_USER_INPUT 的 HTML DB 项中。根据下面的 PL/SQL 逻辑，如果用户为该项提供了一个值，该值将被附加到查询中。

```
DECLARE
q varchar2(4000);
BEGIN
q := 'SELECT ename, job, sal
FROM emp
WHERE deptno = 20 `;
IF :P1_USER_INPUT is not NULL then
q := q || :P1_USER_INPUT;
END IF;
return q;
END;
```

假设 P1_USER_INPUT 的值为 “AND SAL < 2000”，执行的查询将是：

```
SELECT ename, job, sal
FROM emp
WHERE deptno = 20 AND SAL < 2000
```

此查询的结果将包含部门 20 中工资低于 2,000 的所有员工的姓名、工作描述和工资。但是，如果 P1_USER_INPUT 项中的用户类型 “OR 1=1”，查询将类似以下内容：

```
SELECT ename, job, sal
```

```
FROM emp

WHERE deptno = 20 OR 1=1
```

现在，结果将是 emp 表的所有内容！不要将任何用户输入附加到应用程序执行的查询中，除非目的是允许对基本表的完全即席访问。相反，应严格控制查询的结构，并小心合并用户输入。

保护报表列

可以将 HTML DB 中的报表定义为对不同的用户显示不同的列，具体取决于用户的权限。授权方案是集中定义的访问控制规则，可以应用到应用程序的所有元素中。访问控制规则可以使用 SQL 或 PL/SQL 定义，也可以在已公布的逻辑中构建。例如，假设您创建了一个授权用户名为 PRIVILEGED_USERS 的表，您可以基于 EXISTS 查询实现授权方案，如下所示：

```
SELECT 1

FROM privileged_users

WHERE username = :APP_USER
```

在本示例中，当前登录用户的用户名自动存储在会话状态变量 APP_USER 中，并与 PRIVILEGED_USERS 表中的所有条目相比较。授权方案的方式被定义为，仅在存在匹配项时，允许访问适用于该项的所有元素。

适用于报表列的授权方案允许您在实施访问控制规则的同时一次为所有用户定义报表。

性能

Oracle HTML DB 的体系结构是在数据库内部进行大部分处理。HTML DB 引擎自身利用很少的开销，具体取决于处理器的速度，可能只是每页请求占用 CPU 0.04 秒。当然，向页面中添加执行处理的报表查询和 HTML DB 引擎的指令时，CPU 时间的要求将增加。下面是开发应用程序时最小化数据库负载的一些建议。

使用连接变量

Oracle 数据库执行 SQL 查询或 PL/SQL 代码之前，它将执行一系列包含分析、名称解析、安全检查和优化的步骤。要在经常执行同一 SQL 或 PL/SQL 的情况下避免不必要地重复这些步骤，数据库使用存储“预编译”

SQL 和 PL/SQL 的共享池或库缓存。在执行查询期间，数据库将首先在此缓存中查看该查询是否已经运行，以及它的版本是否可以重复使用。要尽可能增加查询可重复使用的次数，应该使用连接变量。连接变量是 SQL 或 PL/SQL 中的占位符，在执行期间为其赋值。例如，请考虑此 SQL：

```
SELECT empno, sal
FROM emp
WHERE empno = 1234
```

执行此查询后，该查询在共享池中可供重复使用，使用次数取决于老化例程，该例程会在需要空间时清除此部分的内存。当然，仅在应用程序恰好执行同一查询时，才需要重复使用特定的查询。即，目标是获得员工编号和员工编号 (EMPNO) 为 1234 的员工的工资的查询。

现在，请考虑带有连接变量的同一 SQL：

```
SELECT empno, sal
FROM emp
WHERE empno = :empno
```

该查询可供任意编号的员工重复使用，因为实际的值将在执行查询后分配。

确保查询可以从共享池重复使用对数据库应用程序的可伸缩性至关重要。执行“预编译的”查询，从共享池出发运行查询比将该查询视为一个从未见过的新查询所花费的时间要少。因此，您的用户将消耗数据库中确保并发控制所需的更少的宝贵的资源。

如何确保查询可在 HTML DB 应用程序中重复使用？可以通过在包含 SQL 和 PL/SQL 中会话状态中的值时使用连接变量来达到此目的。作为规则，请避免在 SQL 和 PL/SQL 中使用替代字符串。请考虑此 SQL 查询：

```
SELECT '<a href="f?p=100:1:&APP_SESSION.">edit</a>'
link,
empno, sal
FROM emp
```

乍一看，该查询好像是准确并无害的，但是它并不是经过优化可供重复使用的查询。事实上，它只能由一个用户重复使用。执行此查询之前，HTML DB 将当前用户的会话标识符替换为 &APP_SESSION。数据库将看到如下所示的查询：

```
SELECT '<a href="f?p=&APP_ID.:1:9878768768759">edit</a>'
link,
empno, sal
FROM emp
```

由于 9878768768759 是一个唯一的会话标识符，上述查询将是一个唯一的查询，除非它恰好要被该会话的所有者执行多次。构建此查询较常用的方法是：

```
SELECT '<a href="f?p=&APP_ID.:1:' || :APP_SESSION ||
''>edit</a>'
link,
empno, sal
FROM emp
```

此处，会话标识符作为连接变量引用，该变量将在数据库从共享池中检索到先前执行的唯一查询后分配。注意，可以使用替代变量 &APP_ID，因为该变量始终保持同一个值，除非在同一数据库中有同一应用程序的多个副本（具有不同的应用程序 ID）在运行。

恰当地选择分页样式

在 HTML DB 中创建的报表可以自动配置为通过结果集提供分页。5,000 行的结果集可以分成每页 25 行的页面。有不同的分页样式，有些使用“下一页”和“上一页”链接以及表示当前显示的行范围的数字，有些使用选择列表允许用户跳转到特定的行范围。

当用户不需要确切知道结果集中有多少行时，可以使用不预先计算结果集大小的分页样式。例如，使用“行范围 X 到 Y”等分页样式可以节省宝贵的数据库资源。

尽可能使用已公布的逻辑

Oracle HTML DB 在很多情况下为开发人员提供了构建自定义逻辑的机会。例如，在应用于呈现或处理的条件中，或者在表单输入的验证中。要在提高开发人员效率的同时优化性能，可以使用许多预定义的条件类型和验证。

由于这些预定义的逻辑单元在本质上是“烙入”HTML DB 引擎中的，因此强烈建议您尽可能地使用它们以提高应用程序性能。例如，要验证用户的输入是否为数字，您可以编写自己的 PL/SQL 逻辑以实现验证。选择预先构建的“项目为数字”验证类型将花费更少的时间，并将导致更好的性能。

使用重定向

HTML DB 应用程序中的大多数页面在运行时经历两个不同的阶段。第一个阶段是呈现阶段，在呈现阶段期间，页面基于其存储在 HTML DB 应用程序库中的定义以及它所引用的其他组件和模板进行组装。然后，当用户触发事件时，例如单击某个按钮在数据库中插入数据，HTML DB 引擎将使该页转到处理阶段。在此阶段中，运行验证并执行处理以修改会话状态或数据库中的数据。最后，执行分支，将用户带到另一页（或同一页）。

某些用户触发事件不需要处理。例如，单击“取消”按钮从数据入口表单导航到另一页通常不需要对会话状态进行任何修改。对于此情况，HTML DB 提供一种不需要执行任何处理的按钮，从而通过使用 URL 重定向将用户导航到远离当前页面的位置而提供短路。

结论

Oracle HTML DB 是一种十分高效的公开工具，开发人员可以使用它对应用程序的行为、样式和布局进行精确的控制，并且不会牺牲灵活性。这意味着 HTML DB 开发人员在开发过程中可以有所选择。

在本白皮书中，我已经讨论了使用 Oracle HTML DB 确保简化具有以下特征的 web 应用程序的开发和部署的指导原则和最佳应用方法：安全、性能良好稳定以及使用一致的并易于维护的用户界面。



Oracle HTML DB 最佳应用方法

2004年6月

作者: Sergio Leunissen

甲骨文公司

全球总部

500 Oracle Parkway

Redwood Shores, CA 94065

U. S. A.

全球咨询热线:

电话: +1.650.506.7000

传真: +1.650.506.7200

www.oracle.com

版权所有 © 2003, Oracle。保留所有权利。

本文档只用于提供信息,

其中的内容如有更改恕不通知。

不保证本文档中没有错误, 也不提供任何其它

保证或条件 (无论是口头表达还是法律暗示), 包括

商用的隐含保证和条件或者对特殊目的的适

用性。我们明确拒绝与本文档有关的任何责任, 并且本文

档不构成任何直接或间接的契约义务。未经

我们事先的书面许可, 不得以任何形式或

方法 (电子或机械方法) 为任何目的复制或传输本文档。

Oracle 是甲骨文公司和/或其会员的注册商标。

其他名称可能是其各自所有者的商标。