SQL Server & Oracle

数据迁移方案

I@chieve

北京信达吉成科技有限公司
Beijing iAchieve S&T Co. Ltd.
北京市金融大街 27 号投资广场 A907 邮编 100032
电话：(010) 66210221 传真：(010) 66212141
电子邮件：xiedong@iachieve.com.cn
主页： www.iachieve.com.cn

目录

前言

随着信息化时代的到来，企业的数据业务日益频繁，数据库的安全性，稳定性，高效性正在经受着以前无法想象的考验。特别是一些迅速壮大的企业其原始开发的程序和数据库产品已经不能满足它的业务需要。升级数据库产品从而进行专业数据迁移和相关程序的适应性维护成为迫切需要解决的问题。

北京信达吉成科技有限公司(iAchieve)是以系统集成和软件工程为主体的高科技企业。公司凭借着雄厚的技术实力与一批对事业执着追求的专业技术人才，紧跟国际先进技术，注重与国际上最优秀企业保持密切的技术合作，在不断开拓进取中得到迅速发展。经过公司全体员工的不断努力，公司在计算机网络通讯技术、网络系统集成，以及专用网络的数据安全和电子商务等方面都打下了坚实的基础。公司凭借深厚的技术实力，完善的开发体系，优秀的人才队伍和出色的管理体制，在金融、能源电力、教育和媒体行业都有极佳的表现，在这几个行业中公司的业务伙伴遍及全中国。

北京信达吉成科技有限公司，定位于提供企业级计算与网络服务，集咨询、应用开发、系统集成和 IT 服务为一体，提供面向金融、电力能源、邮电、税收、交通和商贸等行业和部门的整体解决方案。在为行业用户提供服务的过程中，积累了经验，锻炼了队伍，公司秉承与用户一起成功的理念，始终追求特色、高效和便捷的服务，赢得广大用户的盛誉和信赖。

北京信达吉成科技有限公司是 IBM 的业务伙伴，代理 IBM 公司的全线产品，包括 AS400、RS6000，PC Server 和各种软件，和 IBM 公司建立了密切的合作关系，在许多项目中得到 IBM 公司的大力支持。

北京信达吉成科技有限公司全面代理 Cisco、Marconi （Fore），NortelNetwoks 公司的网络产品，在网络系统集成领域有着非常丰富的经验，与这些厂家有良好的合作，通力合作，为用户提供最好的解决方案。

特别是近两年来信达吉成公司承接了一批电厂数据迁移的工程，这使我们在软件行业特别是数据迁移方面积累了大量的开发经验，给予我们充分的信心拓展计算机数据方面的业务。

Oracle 与 MS SQL Server 的相互转换

本文的目标读者应该具有：

- 坚实的 Oracle RDBMS 基础知识背景。
- 全面的数据库管理知识。
- 熟悉 Oracle SQL 和 PL/SQL 语言。
- 实际使用 C/C++ 编程语言的知识。
- sysadmin 固定服务器角色的成员身份。

本文假定，您熟悉与 Oracle RDBMS 有关的术语、概念和工具。有关 Oracle RDBMS 及其体系结构的详细信息，请参见 *Oracle 7 Server Concepts Manual*（Oracle 7 Server 概念手册）。至于使用 Oracle 脚本和示例，还假定您熟悉 Oracle Server Manager 和 Oracle SQL*Plus 工具。有关这些工具的详细信息，请参见 Oracle 文档。文中讲述了成功地进行转换所需要的工具、过程和技巧。并突出强调了创建高性能、高并发性 SQL Server 应用程序的基本设计原则。

开发和应用程序平台

为了清楚和便于表述，开发和应用程序平台应不低于 Microsoft Visual Studio 6.0 版、Microsoft Windows NT 4 (Service Pack 4)、SQL Server 7.0 和 Oracle 7.3。Oracle 7.3 使用 Visigenic Software ODBC 驱动程序（2.00.0300 版）；SQL Server 7.0 使用 Microsoft Corporation ODBC 驱动程序（3.70 版）。Microsoft SQL Server 7.0 包括用于 Oracle 的 OLE DB 驱动程序，但在本章中不予详细讨论。

概述

应用程序迁移过程似乎很复杂。两种 RDBMS 之间有很多体系结构方面的差异。描述 Oracle 体系结构的词汇和术语在 Microsoft SQL Server 中，其含义常常完全不同。此外，Oracle 和 SQL Server 都有许多专有的 SQL-92 标准扩展。

从应用程序开发人员的角度来看，Oracle 和 SQL Server 管理数据的方式是相似的。但是，Oracle 和 SQL Server 之间内部的差异是相当大的，如果管理得当，它对迁移应用程序造成的影响就会微乎其微。

开发人员面临的最严峻迁移问题是：SQL-92 SQL 语言标准的实现和每种 RDBMS 提供的扩展。一些开发人员只使用标准的 SQL 语言语句，并倾向于使其程序代码尽可能通用。通常，这意味着把程序代码限定在初级 SQL-92 标准，该标准在许多数据库产品中均得到了一致的实现，其中包括 Oracle 和 SQL Server。

这种方法可能给程序代码带来不必要的复杂性,并显著影响程序性能。例如,Oracle 的 DECODE 函数是 Oracle 特有的非标准 SQL 扩展。Microsoft SQL Server 的 CASE 表达式已不止是初级 SQL-92 的扩展，并未在所有的数据库产品上实现。

如果不使用这两个函数，则可以编程方式实现其功能，但可能需要从 RDBMS 检索更多的数据。

此外，SQL 语言的过程扩展也可能带来困难。Oracle PL/SQL 和 SQL Server Transact-SQL 语言功能相似，但语法不同。各 RDBMS 及其过程扩展之间不存在精确的对等关系。因此，您可能会放弃使用存储程序，例如过程和触发器。这是令人遗憾的，因为这些程序能够提供极好的性能和安全性，而这些用任何其它方式均无法实现。

使用专用的开发接口也会带来其它的问题。使用 Oracle OCI（Oracle 调用接口）转换程序，通常需要大量的资源投入。当开发的应用程序可能使用多个 RDBMS 时，应考虑使用开放式数据库连接（ODBC）接口。

ODBC 是专为使用多种数据库管理系统而设计的。ODBC 提供一致的应用程序编程接口（API），它通过数据库特有驱动程序的服务，与不同的数据库一同工作。

一致的 API 是指，不论程序与 Oracle 还是与 SQL Server 交互，它在建立连接、执行命令和检索结果时所调用的函数是相同的。

ODBC 还定义了一个标准调用级接口，并使用标准转义序列，指定执行公用任务的 SQL 函数，但该函数在不同的数据库中语法不同。不需要修改任何程序代码，ODBC 驱动程序就可以自动地把 ODBC 语法转换成原本的 Oracle 或 Microsoft SQL Server SQL 语法。在某些情况中，最好的方法是编写一个程序，使 ODBC 在运行时进行转换。

ODBC 并不是一个神奇的解决方案，不能对所有的数据库均实现完全的数据库独立性、完备的功能以及较高的性能。不同的数据库和第三方厂商提供不同级别的 ODBC 支持。一些驱动程序只实现了映射在其它接口库顶层的核心 API 函数。其它驱动程序，例如 Microsoft SQL Server 驱动程序，在原本的、高性能的驱动程序中提供全面的级别 2 支持。

如果程序只使用核心 ODBC API，它可能放弃了一些数据库带有的功能和性能。再者，并不是所有原本的 SQL 扩展都可以用 ODBC 转义序列表示，例如 Oracle DECODE 和 SQL Server CASE 表达式就是这样。

此外，通过编写 SQL 语句使用数据库优化程序也是通常的做法。在 Oracle 中用来提高性能的技巧和方法，在 SQL Server 中并不一定最好。ODBC 接口无法将技巧从一个 RDBMS 转化到另一个 RDBMS 中。

ODBC 并不禁止应用程序使用数据库特有的功能，也不禁止优化性能，但是应用程序需要一些数据库特有的代码部分。有了 ODBC，要使程序结构和绝大部分程序代码在多个数据库上保持一致，就变得十分简单。

OLE DB 是下一代的数据访问技术。Microsoft SQL Server 7.0 利用了 SQL Server 自身组件中的 OLE DB。此外，应用程序开发人员在 SQL Server 新的开发过程中，应考虑使用 OLE DB。Microsoft 在 SQL Server 7.0 中加入了用于 Oracle 7.3 的 OLE DB 提供程序。

OLE DB 是 Microsoft 的一个战略性系统级编程接口，用于管理整个组织内的数据。OLE DB 是建立在 ODBC 功能之上的一个开放规范。ODBC 是为访问关系型数据库而专门开发的，OLE DB 则用于访问关系型和非关系型信息源，例如主机 ISAM/VSAM 和层次数据库，电子邮件和文件系统存储，文本、图形和地理数据以及自定义业务对象。

OLE DB 定义了一组 COM 接口，对各种数据库管理系统服务进行封装，并允许创建软件组件，实现这些服务。OLE DB 组件包括数据提供程序（包含和表现数据）、数据使用者（使用数据）和服务组件（处理和传送数据，例如，查询处理器和游标引擎）。

OLE DB 接口有助于平滑地集成组件，这样，OLE DB 组件厂商就可以快速地向市场提供高质量 OLE DB 组件。此外，OLE DB 包含了一个连接 ODBC 的“桥梁”，对现用的各种 ODBC 关系型数据库驱动程序提供一贯的支持。

体系结构和术语

要成功地迁移，开始之前应该了解与 Microsoft SQL Server 7.0 有关的基础体系结构和术语。本节中的许多例子均取自 Oracle 和 SQL Server 应用程序示例（附在文中）。

在 Oracle 中，“数据库”指整个 Oracle RDBMS 环境，并包括以下组件：

- Oracle 数据库进程和缓冲区（实例）。
- 包含一个集中系统编录的 SYSTEM 表空间。
- 其它由 DBA 定义的表空间（可选）。
- 两个或多个在线重做日志。
- 存档的重做日志（可选）。
- 各种其它文件（控制文件，Init.ora 等等）。

Microsoft SQL Server 数据库从逻辑上将数据、应用程序和安全机制分离，这一点与表空间非常相似。Oracle 支持多个表空间；SQL Server 则支持多个数据库。表空间还可用于支持数据的物理存放；SQL Server 使用文件组提供相同的功能。

Microsoft SQL Server 还默认安装下列数据库：

- **model** 数据库是所有新创建的用户数据库的模板。
- **tempdb** 数据库与 Oracle 临时表空间相似，它用于临时工作存储和排序操作。与 Oracle 临时表空间不同的是，用户可以创建临时表，并在用户注销时自动删除。
- **msdb** 支持 SQL Server 代理及其计划的作业、警报和复制信息。
- **pubs** 和 **Northwind** 数据库作为培训示例数据库提供。

有关默认数据库的详细信息，请参见 SQL Server Books Online。

每个 Oracle 数据库均在一个集中系统编录或数据字典上运行，它驻留在 SYSTEM 表空间中。每个 Microsoft SQL Server 7.0 数据库均维护其自身的系统编录，它包含下列信息：

- 数据库对象（表、索引、存储过程、视图、触发器等等）。
- 约束。
- 用户和权限。
- 用户定义的数据类型。
- 复制定义。
- 数据库使用的文件。

在 master 数据库中，SQL Server 还加入了一个集中系统编录，它包括系统编录以及有关每个数据库的一些信息：

- 数据库名称和每个数据库的主文件位置。
- SQL Server 登录帐户。
- 系统消息。
- 数据库配置值。
- 远程和/或链接的服务器。
- 当前活动信息。
- 系统存储过程。

与 Oracle 中的 SYSTEM 表空间一样，要访问任何其它数据库，SQL Servermaster 数据库必须可用。因此，当 master 数据库做重大修改后，应对该数据库进行备份以防止数据库出现故障，这一点非常重要。数据库管理员也可以镜像构成 master 数据库的文件。

Oracle RDBMS 由表空间组成，而表空间又是由数据文件组成的。表空间数据文件被格式化为称为“块”的内部单元。块的大小是 DBA 在 Oracle 数据库首次创建时设定的，其范围从 512 到 8192 字节。在 Oracle 表空间中创建一个对象时，用户用称为“扩展盘区”的单位定义其大小（初始扩展盘区、下一扩展盘区、最小扩展盘区和最大扩展盘区）。Oracle 扩展盘区大小是可变的，但必须包括至少五个连续的块。

在数据库一级中，Microsoft SQL Server 使用文件组来控制表和索引的物理存储。文件组是一个或多个文件的逻辑容器，文件组中包含的数据按比例填充到所有属于该组的文件中。

如果没有定义和使用文件组，数据库对象就会被放在一个默认文件组中，该文件组是数据库创建过程中隐式定义的。文件组允许：

- 把大型表分布在多个文件上，以提高 I/O 吞吐量。
- 把索引存储在不同的文件上，而不是它们各自的表上，从而进一步提高了 I/O 吞吐量和磁盘并发性。
- 将 text、ntext、image 列（大对象）从表中存储到不同的文件上。
- 把数据库对象放在特定的磁盘上。
- 备份和恢复文件组中单个表或一组表。

SQL Server 将文件格式化为称为“页”的内部单元。页大小是固定的，为 8192 字节 (8 KB)。扩展盘区由页组成，其大小也是固定的，由 8 个连续的页组成。在 SQL Server 数据库中创建表或索引时，会自动给其分配一个页。与分配一个整个扩展盘区相比，它可更有效地存储较小的表和索引。

对于大多数 Microsoft SQL Server 安装来说，不需要 Oracle 类型的段。相反，SQL Server 可以使用基于硬件的 RAID 或基于 Windows NT 软件的 RAID，更好地分布数据或将数据条带化。基于 Windows NT 软件的 RAID 或基于硬件的 RAID 可以设定条带集，它包括多个磁盘驱动器，看起来就像一个逻辑驱动器一样。如果数据库文件在此条带集上创建，磁盘子系统就负责把 I/O 负载分布到多个磁盘上。建议管理员使用 RAID，把数据分布到多个物理磁盘上。

SQL Server 推荐的 RAID 配置是 RAID 1（镜像）或 RAID 5（带有一个额外的奇偶校验驱动器的条带集，用作冗余）。也建议使用 RAID 10（带有奇偶校验的条带集的镜像），但是它比前两种配置昂贵得多。条带集非常适于分布数据库文件上常常随机产生的 I/O。

如果不能选择 RAID，文件组则是一个有吸引力的替代选择，它提供与 RAID 相同的一些优点。此外，对于可能跨越多个物理 RAID 阵列的大型数据库，文件组是一个很吸引人的方法，它以一种可控的方式，将 I/O 进一步分布到多个 RAID 阵列上。

对于有序 I/O，必须优化事务日志文件，并加以保存，防止单点失败。因此，对于事务日志，建议使用 RAID 1（镜像）。这个驱动器的大小至少要和联机重做日志和回滚段表空间的总计大小一样。应创建一个或多个日志文件，来占用该逻辑驱动器上定义的所有空间。与存储在文件组中的数据不同，事务日志项目总是按顺序地写入，并且不是按比例填充的。

有关 RAID 的详细信息，请参见 SQL Server Books Online、Windows NT Server 文档和 Microsoft Windows NT 资源工具包。

每次启动时，Oracle RDBMS 执行自动恢复。它检验表空间文件的内容是否与联机重做日志文件一致。如果不一致，Oracle 将联机重做日志文件内容应用到表空间文件（前滚），并删除回滚段中发现的任何未提交的事务（回滚）。如果 Oracle 不能从联机重做日志文件中得到它所需要的信息，它就会查询存档重做日志文件。

每次启动时，Microsoft SQL Server 7.0 还通过检查系统中的每个数据库，进行自动数据恢复。它首先检查 master 数据库，然后启动恢复系统中所有其它数据库的线程。对于每个 SQL Server 数据库，自动恢复机制均检查事务日志。如果事务日志包含任何未提交的事务，该事务被回滚。然后，恢复机制在事务日志中，查找已提交但还未写到数据库的事务。如果找到，再次执行这些事务，前滚。

每个 SQL Server 事务日志均有 Oracle 回滚段与 Oracle 联机重做日志的组合功能。每个数据库都有自己的事务日志，它记录了对数据库所作的全部更改，并且由数据库的所有用户共享。当一个事务开始且发生数据修改时，就会在日志中记录一个 BEGIN TRANSACTION 事件（以及修改事件）。在自动故障恢复过程中，这个事件用于确定事务的起始点。在收到每个数据修改语句时，先将更改写入事务日志，然后再写入数据库。有关详细信息，请参见本章后面的“事务、锁定和并发性”一节。

SQL Server 有一个自动检查点机制，确保完成的事务被定期地从 SQL Server 磁盘缓存写入事务日志文件。检查点功能将自上一个检查点之后修改过的任何已被缓存的页面写入数据库。在数据库上对这些被缓存过的页面（称为“脏页”）标出检查点，以确保所有完成的事务均被写到磁盘中。这个过程缩短了从系统故障（如停电）进行恢复所用的时间。通过使用 SQL Server Enterprise Manager 或 Transact-SQL（sp_configure 系统存储过程）修改恢复间隔设置，可对此设置进行修改。

Microsoft SQL Server 给备份数据提供了以下几个选项：

- 完全数据库备份

  要进行完全数据库备份，请使用 BACKUP DATABASE 语句或备份向导。

- 差异备份

  当完成完全数据库备份后，使用 BACKUP DATABASE WITH DIFFERENTIAL 语句或备份向导，只定期备份更改的数据和索引页。

- 事务日志备份

  Microsoft SQL Server 中的事务日志与各自数据库关联。在备份或被截断之前，事务日志都是不断填充的。SQL Server 7.0 的默认配置是，事务日志自动增长，直到用尽了所有磁盘空间或达到最大配置尺寸为止。当事务日志变得太“满”时，它就会产生一个错误，并且在备份或截断之前，禁止对数据进一步修改。其它数据库不受影响。可以使用 BACKUP LOG 或备份向导备份事务日志。

- 文件或文件组备份

  SQL Server 可以备份文件或文件组。有关详细信息，请参见 SQL Server Books Online。

可以在数据库使用过程中对它进行备份，这样就可以对必须连续运行的系统进行备份。SQL Server 7.0 的备份处理和内部数据结构已进行了改进，这样，可将备份的数据传输率提高到最大，同时对事务吞吐量的影响降至最小。

Oracle 和 SQL Server 均需要特定的日志文件格式。在 SQL Server 中，这些文件称为备份设备，它们是使用 SQL Server Enterprise Manager、Transact-SQL sp_addumpdevice 存储过程或相应的 SQL-DMO 命令创建的。

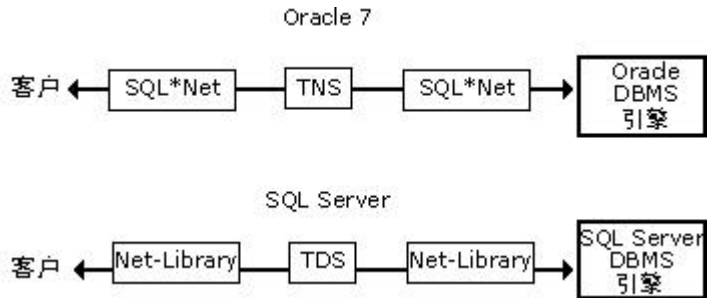
尽管可以手动进行备份，但是，建议使用 SQL Server Enterprise Manager 和/或 Database Maintenance Plan Wizard 计划定期备份或基于数据库活动的备份。

通过在完全数据库备份（设备）中应用事务日志备份和/或差异备份，可以将数据库恢复到某个时点。数据库恢复使用备份中包含的信息来覆盖数据。可以使用 SQL Server Enterprise Manager、Transact-SQL（RESTORE DATABASE）或 SQL-DMO 进行恢复。

正如可以关闭 Oracle 归档文件来覆盖自动备份一样,在 Microsoft SQL Server 中,db_owner 固定数据库角色的成员可以在每次出现检查点时，强制事务日志清除其内容。这一操作可以使用 SQL Server Enterprise Manager（在检查点处截断日志）、Transact-SQL（sp_dboption 存储过程）或 SQL-DMO 来完成。

Oracle SQL*Net 支持 Oracle 数据库服务器及其客户之间的网络连接。它使用透明网络底层 (TNS) 数据流协议进行通信，并允许用户运行多个不同的网络协议，而不必编写专用的代码。核心 Oracle 数据库软件产品并不包括 SQL*Net。

有了 Microsoft SQL Server，*Net-Libraries*（网络库）通过使用表格格式数据流 (TDS) 协议，支持客户和服务器之间的网络连接。它们允许同时连接运行命名管道、TCP/IP 套接字或其它进程间通信 (IPC) 机制的客户。SQL Server CD-ROM 包括所有的客户 Net-Libraries，因此不需要再另行购买。



SQL Server Net-Library 选项可在安装后进行更改。客户网络实用工具为运行 Windows NT、Windows 95 或 Windows 98 操作系统的客户配置默认的 Net-Library 和服务器连接信息。除非在 ODBC 数据源配置过程中更改，或在 ODBC 连接字符串中明确写明，所有的 ODBC 客户应用程序均使用相同的默认 Net-Library 和服务器连接信息。有关 Net-Libraries 的详细信息，请参见 SQL Server Books Online。

要将 Oracle 应用程序完全迁移到 Microsoft SQL Server 7.0，必须了解 SQL Server 数据库安全性和角色的实现。

登录帐户

登录帐户允许用户访问 SQL Server 数据或管理选项。登录帐户只允许用户登录到 SQL Server，并查看允许 guest（来宾）访问的数据库。（guest 帐户不是默认建立的，必须单独创建。）

SQL Server 提供两种类型的登录安全性：Windows NT 身份验证模式（也称为集成模式）和 SQL Server 身份验证模式（也称为标准模式）。SQL Server 7.0 也支持标准和集成安全的组合，称为混合模式。

验证登录连接时，Windows NT 身份验证模式使用 Windows NT 内的安全机制，并且依赖用户的 Windows NT 安全凭据。用户不需要输入 SQL Server 的登录 ID 或密码 - 他们的登录信息直接从网络连接中获取。此时，一个条目被写入 syslogin 表，并在 Windows NT 和 SQL Server

之间进行验证。这称为一个信任连接，就像两个 Windows NT 服务器之间的信任关系一样。它与 Oracle 用户帐户相关的 IDENTIFIED EXTERNALLY 选项作用相似。

SQL Server 身份验证模式要求，用户在请求访问 SQL Server 时，输入登录 ID 和密码。这称为非信任连接。它与 Oracle 用户帐户相关的 IDENTIFIED BY PASSWORD 选项作用类似。使用标准安全模型，登录过程只提供对 SQL Server 数据库引擎的访问，而不提供对用户数据库的访问。

有关这些安全机制的详细信息，请参见 SQL Server Books Online。

组、角色和权限

Microsoft SQL Server 和 Oracle 均使用权限，来实施数据库安全性。SQL Server 语句级权限用于限制创建新的数据库对象（类似于 Oracle 系统级权限）。

SQL Server 还提供对象级权限。与 Oracle 一样，对象级所有权被授予对象的创建者，并且不能被转让。在其他数据库用户访问对象前，必须给他们授予对象级权限。sysadmin 固定服务器角色、db_owner 固定数据库角色或 db_securityadmin 固定数据库角色的成员也可以将一个用户对象上的权限授予其他用户。

可以将 SQL Server 语句级和对象级权限直接授予数据库用户帐户。而管理数据库角色的权限通常要简单得多。SQL Server 角色用于授予或撤销一组数据库用户的权限（与 Oracle 角色非常相似）。角色是与特定数据库相关的数据库对象。对于每种安装，均有相关的专有固定服务器角色，可用于整个数据库。固定服务器角色的一个例子是 sysadmin。当 SQL Server 登录时，可以添增 Windows NT 组或数据库用户。可以给 Windows NT 组或 Windows NT 用户授予权限。

数据库可以有任意数量的角色或 Windows NT 组。在每个数据库中，均可找到默认角色 public，并且该角色不能被删除。public 角色和 Oracle 中 PUBLIC 帐户的作用相似。每个数据库用户始终是 public 角色的一个成员。除了 public 角色之外，数据库用户还可以是任何数量角色的成员。Windows NT 用户和组也可以是任何数量角色的成员，并且始终是 public 角色的成员。

数据库用户和 guest 帐户

在 Microsoft SQL Server 中，要使用数据库及其对象，用户登录帐户必须被授权。登录帐户可以使用下列方法访问数据库：

- 登录帐户可被指定为数据库用户。
- 登录帐户可使用数据库中的 guest 帐户。
- 可以将 Windows NT 组登录映射为一个数据库角色。然后，作为该组成员的各 Windows NT 帐户可以连接到该数据库。

db_owner、db_accessadmin 角色，或 sysadmin 固定服务器角色的成员创建数据库用户帐户角色。帐户可以包括以下几个参数：SQL Server 登录 ID、数据库用户名（可选）和最多一个角色名（可选）。数据库用户名不需和用户的登录 ID 相同。如果没有提供数据库用户名，则用户的登录 ID 和数据库用户名是相同的。如果没有提供角色名，则数据库用户只是 public 角色的成员。创建数据库用户之后，可根据需要赋予该用户相应的角色。

db_owner 或 db_accessadmin 角色的成员还可以创建 guest 帐户。guest 帐户允许任何有效的 SQL Server 登录帐户访问数据库，即便没有数据库用户帐户也可以。默认情况下，guest 帐户继承授予 public 角色的任何权限；但是，这些权限可以更改，使其高于或低于 public 角色的权限。

与 SQL Server 登录一样，Windows NT 用户帐户或组帐户可被授权访问数据库。当作为组成员的 Windows NT 用户连接到该数据库时，此用户就获得授予 Windows NT 组的权限。如果他是多个 Windows NT 组（已授权访问数据库）的成员，则该用户可收到所有这些组的组合权限。

sysadmin 角色

Microsoft SQL Server sysadmin 固定服务器角色成员的权限与 Oracle DBA 的权限相似。在 SQL Server 7.0 中，默认情况下，sa SQL Server 身份验证模式登录帐户是该角色的成员；这就如同当 SQL Server 安装在 Windows NT 计算机上,它就是本地 Administrators 组成员一样。sysadmin 角色的成员可以添加或删除 Windows NT 用户和组，以及 SQL Server 登录。该角色的成员通常有下列职责：

- 安装 SQL Server。
- 配置服务器和客户。
- 创建数据库。*
- 设置登录权限和用户权限。*
- 向 SQL Server 数据库导入数据和从中导出数据。*
- 备份和恢复数据库。*
- 实现和维护复制。
- 计划无值守操作。*
- 监视和优化 SQL Server 性能。*
- 分析系统问题。

*这些项目可以委派给其他安全角色或用户。

在 SQL Server 7.0 中，没有对 sysadmin 固定服务器角色成员的权限进行限制。因此，该角色的成员可以访问 SQL Server 特定实例上的任何数据库及其所有对象（包括数据）。与 Oracle DBA 一样，有一些命令和系统过程，只有 sysadmin 角色的成员可以使用。

db_owner 角色

尽管在使用上，Microsoft SQL Server 数据库与 Oracle 表空间类似，但各个的管理方式不同。每个 SQL Server 数据库都是一个自包含的管理域。每个数据库均被指派一个数据库所有者 (dbo)。该用户始终是 db_owner 固定数据库角色的一个成员。其他用户也可以是 db_owner 角色的成员。作为该角色成员的任何用户，都有能力管理与其数据库有关的管理任务（Oracle 则不同，一个 DBA 可管理所有表空间的管理任务）。这些任务包括：

- 管理数据库访问。
- 更改数据库选项（只读、单用户等等）。
- 备份和恢复数据库内容。
- 授予和撤销数据库权限。
- 创建和删除数据库对象。

db_owner 角色的成员在其数据库中具有所有权限。授予该角色的大多数权限可以分给几个固定数据库角色，或被授予数据库用户。要在数据库中拥有 db_owner 权限，不需要有服务器范围内的 sysadmin 权限。

安装和配置 Microsoft SQL Server

搞清了 Oracle 和 SQL Server 之间基本的结构差异之后，就可以开始进行迁移过程的第一步。应使用 SQL Server 查询分析器，运行以下脚本：

1. 使用基于 Windows NT 软件的 RAID 或基于硬件的 RAID 5，创建一个可容纳所有数据的逻辑驱动器。通过计算 Oracle 系统、临时表空间和应用程序表空间所使用的全部文件空间，来预估空间大小。
2. 使用基于 Windows NT 软件的 RAID 或基于硬件的 RAID 1，来创建用于存放事务日志的第二个逻辑驱动器。此驱动器大小应至少和联机重做与回滚段表空间之和一样大。
3. 使用 SQL Server Enterprise Manager，创建一个与 Oracle 应用程序表空间名称相同的数据库。（示例应用程序使用的数据库名称为 USER_DB。）将数据和事务日志的文件位置分别指定为步骤 1 和 2 创建的磁盘。如果使用多个 Oracle 表空间，不必甚至不建议创建多个 SQL Server 数据库。RAID 会为您分布数据。
4. 创建 SQL Server 登录帐户：

```
USE MASTER
EXEC SP_ADDLOGIN STUDENT_ADMIN, STUDENT_ADMIN
EXEC SP_ADDLOGIN DEPT_ADMIN, DEPT_ADMIN
EXEC SP_ADDLOGIN ENDUSER1, ENDUSER1
GO
```

5. 向数据库中添加角色：

```
USE USER_DB
EXEC SP_ADDROLE DATA_ADMIN
```

```
EXEC SP_ADDROLE USER_LOGON
GO
```

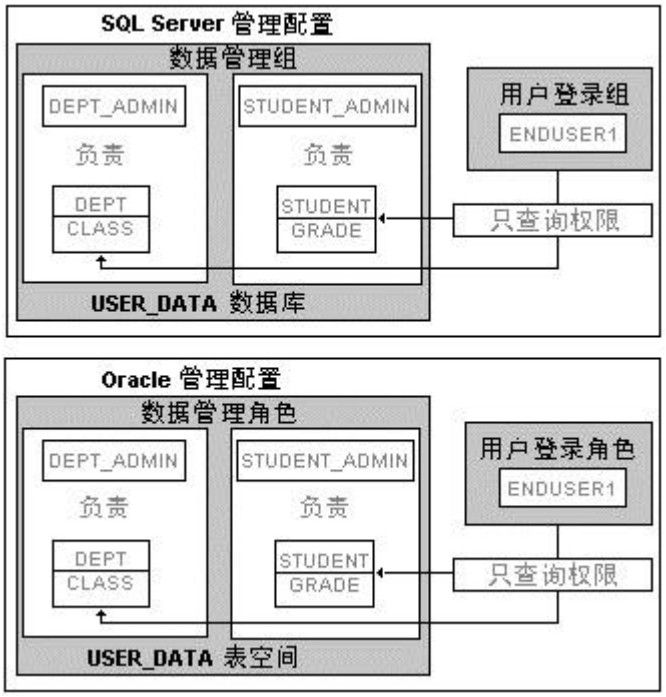
6. **给角色授予权限：**

```
GRANT CREATE TABLE, CREATE TRIGGER, CREATE VIEW,
CREATE PROCEDURE TO DATA_ADMIN
GO
```

7. **把登录帐户添加为数据库用户帐户：**

```
EXEC SP_ADDUSER ENDUSER1, ENDUSER1, USER_LOGON
EXEC SP_ADDUSER DEPT_ADMIN, DEPT_ADMIN, DATA_ADMIN
EXEC SP_ADDUSER STUDENT_ADMIN, STUDENT_ADMIN, DATA_ADMIN
GO
```

此插图给出了此步骤完成后的 SQL Server 和 Oracle 环境。



定义数据库对象

Oracle 数据库对象（表、视图和索引）可以很方便地迁移到 Microsoft SQL Server，因为每种 RDBMS 都严格遵循 SQL-92 标准，该标准是一个关于对象定义的标准。将 Oracle SQL 表、索引和视图定义转换为 SQL Server 表、索引和视图定义，只需要进行相对简单的语法更改即可。下表着重阐述了，Oracle 和 Microsoft SQL Server 数据库对象之间的一些差异。

类别	Microsoft SQL Server	Oracle
列数	1024	254
行大小	8060 字节，加 16 字节指向每个 text 或 image 列	没有限制（但每行只允许一个 long 或 long raw）
最大行数	没有限制	没有限制
BLOB 类型存储	和行一起存储的 16 字节指针。数据存储在其它数据页上。	每表一个 long 或 long raw。必须在行尾。数据存储在与行相同的块上。
聚集的表索引	每表一个	每表一个（索引组织的表）
非聚集的表索引	每表 249 个	没有限制
单索引中索引的最大列数	16	16
索引中列值的最大长度	900 字节	1/2 块
表命名规则	[[[*server*.]*database*.]*owner*.]*table_name*	[*schema*.]*table_name*
视图命名规则	[[[*server*.]*database*.]*owner*.]*table_name*	[*schema*.]*table_name*
索引命名规则	[[[*server*.]*database*.]*owner*.]*table_name*	[*schema*.]*table_name*

假定您从用来创建数据库对象的 Oracle SQL 脚本或程序入手。只要复制这个脚本或程序，并进行下列修改即可。每个更改均在本节的其它部分进行了讨论。该例取自脚本示例程序脚本 Oratable.sql 和 Sstable.sql。

1. 确保数据库对象标识符符合 Microsoft SQL Server 命名规则。可能只需要更改索引名称。
2. 修改数据存储参数，使之用于 SQL Server。如果使用 RAID，则不需要存储参数。
3. 修改 Oracle 约束定义，使之用于 SQL。如有必要，则创建触发器，以支持外键 DELETE CASCADE 语句。如果表跨几个数据库，则使用触发器强制外键关系。
4. 修改 CREATE INDEX 语句，以使用聚集索引。
5. 使用“数据转换服务”，创建新的 CREATE TABLE 语句。检查该语句，注意 Oracle 数据类型与 SQL Server 数据类型是如何对应的。
6. 删除所有 CREATE SEQUENCE 语句。在 CREATE TABLE 或 ALTER TABLE 语句中，使用标识符列，替代序列的使用。
7. 如有必要，修改 CREATE VIEW 语句。
8. 删除任何对同义词的引用。
9. 评估 Microsoft SQL Server 临时表的使用，及其在应用程序中的用途。
10. 把 Oracle 的所有 CREATE TABLEUAS SELECT 命令改成 SQL Server 的 SELECTUINTO 语句。

11. 评估用户定义的规则、数据类型和默认值的潜在用途。

下面图表比较了，Oracle 和 Microsoft SQL Server 处理对象标识符的方式。在大多数情况下，向 SQL Server 迁移时，不需要更改对象名称。

Oracle	Microsoft SQL Server
1-30 个字符长。 数据库名称：最多 8 个字符长。 数据库链接名称：最多 128 个字符长。	1-128 个 Unicode 字符长。 临时表名称：最多 116 个字符长。
标识符名称必须以字母开头，并包含字母、数字字符或 _、$、和 # 字符。	标识符名称可以以字母数字字符或 _ 开头，并且几乎可包含任何字符。 如果标识符以空格开始，并包含除 _、@、# 或 $ 以外的字符，则必须使用 []（分隔符）将标识符名称括起来。 如果对象开始字符是： @ 它是一个局部变量。 # 它是一个局部临时对象。 ## 它是一个全局临时对象。
表空间名称必须唯一。	数据库名称必须唯一。
在用户帐户（架构）中，标识符名称必须是唯一的。	在数据库用户帐户中，标识符名称必须是唯一的。
在表或视图中，列名必须是唯一的。	在表或视图中，列名必须是唯一的。
在用户架构中，索引名称必须是唯一的。	在数据库表名称中，索引名称必须是唯一的。

当访问 Oracle 用户帐户中的表时，仅按其不合格的名称来选定它。访问其它 Oracle 架构中的表时，在表名称前加上架构名称和一个英文句点 (.)。Oracle 同义词可提供其它的位置透明性。

当 Microsoft SQL Server 引用表时，使用了另一套命名规则。因为 SQL Server 登录帐户可以在多个数据库中使用同一名称创建表，所以可使用下列规则访问表和视图：

[[*database_name*.]*owner_name*.]*table_name*

访问以下项中的表	Oracle	Microsoft SQL Server
用户帐户	SELECT * FROM **STUDENT**	SELECT * FROM **USER_DB.STUDENT_ADMIN.STUDENT**
其它架构	SELECT * FROM **STUDENT_ADMIN.STUDENT**	SELECT * FROM **OTHER_DB.STUDENT_ADMIN.STUDENT**

以下是命名 Microsoft SQL Server 表和视图的指导原则：

- 使用数据库名和用户名是可选的。当只按名称来引用表时（例如，STUDENT），SQL Server 在当前数据库的当前用户帐户中查找该表。如果没有找到，它就会在该数据库中查找保留用户名 dbo 拥有的相同名称的一个对象。在数据库的用户帐户中，表名称必须唯一。
- 一个 SQL Server 登录帐户可在多个数据库中拥有名称相同的表。例如，ENDUSER1 帐户拥有下列数据库对象：USER_DB.ENDUSER1.STUDENT 和 OTHER_DB.ENDUSER1.STUDENT。限定符是数据库用户名，而不是 SQL Server 登录名，因为它们并不一定相同。

同时，这些数据库中的其他用户可以拥有相同名称的对象：

- USER_DB.DBO.STUDENT
- USER_DB.DEPT_ADMIN.STUDENT
- USER_DB.STUDENT_ADMIN.STUDENT
- OTHER_DB.DBO.STUDENT

因此，建议把所有者名称作为数据库对象引用的一部分。如果应用程序有多个数据库，建议把数据库名称也作为引用的一部分。如果查询跨多个服务器，也将服务器名称加到引用中。

- 每个 SQL Server 连接都有一个当前的数据库上下文，它是在登录时使用 USE 语句设定的。例如，假定下列场景：

一个用户，使用 ENDUSER1 帐户，登录到 USER_DB 数据库。用户请求 STUDENT 表。SQL Server 查找 ENDUSER1.STUDENT 表。如果找到该表，则 SQL Server 在 USER_DB.ENDUSER1.STUDENT 上执行请求的数据库操作。如果在 ENDUSER1 数据库帐户中没有找到该表，SQL Server 则在此数据库的 dbo 帐户中查找 USER_DB.DBO.STUDENT。如果该表仍没有找到，SQL Server 就会返回一个错误信息，指出该表不存在。

- 如果另一个用户，例如 DEPT_ADMIN，拥有这个表，表名称前面一定加上数据库用户的名称（ DEPT_ADMIN.STUDENT）。否则，数据库名称默认为当前在上下文中的数据库。
- 如果引用的表在另一个数据库中，该数据库名称必须用作引用的一部分。例如，在 OTHERDB 数据库中，要访问 ENDUSER1 拥有的 STUDENT 表时，就要使用 OTHER_DB.ENDUSER1.STUDENT。

可用两个英文句点将数据库和表的名称分隔开，省略对象的所有者名称。例如，如果应用程序引用 STUDENT_DB..STUDENT，SQL Server 进行如下查询：

1. STUDENT_DB.current_user.STUDENT
2. STUDENT_DB.DBO.STUDENT

如果用户一次只使用一个数据库，在对象的引用中省略数据库名称，这样，在其它数据库中使用该应用程序就变得简单了。所有对象引用隐式访问当前使用的数据库。如果在同一服务器上，要维护一个测试数据库和一个生产数据库，这是很有用的。

因为 Oracle 和 SQL Server 均支持标识 RDBMS 对象的 SQL-92 初级规则，所以，CREATE TABLE 语法是相似的。

Oracle	Microsoft SQL Server
CREATE TABLE [*schema.*]*table_name* ({*col_name column_properties* [*default_expression*] [*constraint* [*constraint* [*...constraint*]]]| [[,] *constraint*]} [[,] {*next_col_name* | *next_constraint*}...]) [Oracle Specific Data Storage Parameters]	CREATE TABLE [*server.*][*database.*][*owner.*] *table_name* ({*col_name* *column_properties*[*constraint* [*constraint* [*...constraint*]]]| [[,] *constraint*]} [[,] {*next_col_name* | *next_constraint*}...]) [*ON filegroup_name*]

Oracle 数据库对象名称不区分大小写。在 Microsoft SQL Server 中，取决所选的安装选项，数据库对象名可以是区分大小写的。

SQL Server 第一次安装时，默认的排序次序是字典顺序、不区分大小写。(可以使用 SQL Server 安装程序，设定不同的配置。) 因为 Oracle 对象名称始终是唯一的，所以，把数据库对象迁移到 SQL Server，不应有任何问题。建议在 Oracle 和 SQL Server 中所有的表和列名都使用大写，以避免用户在区分大小写的 SQL Server 上安装时出现问题。

有了 Microsoft SQL Server，使用 RAID 通常可简化数据库对象的存放。与 Oracle 索引组织的表一样，SQL Server 聚集索引被集成到表的结构中。

Oracle	Microsoft SQL Server
CREATE TABLE DEPT_ADMIN.DEPT (DEPT **VARCHAR2**(4) NOT NULL, DNAME **VARCHAR2**(30) NOT NULL, CONSTRAINT DEPT_DEPT_PK PRIMARY KEY (DEPT) **USING INDEX TABLESPACE USER_DATA PCTFREE 0 STORAGE (INITIAL 10K NEXT 10K**	CREATE TABLE **USER_DB.**DEPT_ADMIN.DEPT (DEPT **VARCHAR**(4) NOT NULL, DNAME **VARCHAR**(30) NOT NULL, CONSTRAINT DEPT_DEPT_PK PRIMARY KEY **CLUSTERED** (DEPT), CONSTRAINT DEPT_DNAME_UNIQUE UNIQUE **NONCLUSTERED** (DNAME)

MINEXTENTS 1 MAXEXTENTS UNLIMITED), CONSTRAINT DEPT_DNAME_UNIQUE UNIQUE (DNAME) **USING INDEX TABLESPACE USER_DATA** **PCTFREE 0 STORAGE (INITIAL 10K NEXT 10K** **MINEXTENTS 1 MAXEXTENTS UNLIMITED)**) **PCTFREE 10 PCTUSED 40 TABLESPACE USER_DATA** **STORAGE (INITIAL 10K NEXT 10K** **MINEXTENTS 1 MAXEXTENTS UNLIMITED** **FREELISTS 1)**	)

在 Oracle 中，可以使用任何有效的 SELECT 命令，来创建表。Microsoft SQL Server 可提供相同的功能，但语法不同。

Oracle	Microsoft SQL Server
CREATE TABLE **STUDENTBACKUP** AS SELECT * FROM **STUDENT**	SELECT * INTO **STUDENTBACKUP** FROM **STUDENT**

除非要应用的数据库已将 select into/bulkcopy 数据库配置选项设为 ture，否则，SELECTUINTO 不会生效。（数据库所有者可以使用 SQL Server Enterprise Manager 或 Transact-SQL sp_dboption 系统存储过程设定该选项。）使用 sp_helpdb 系统存储过程检查数据库的状态。如果 select into/bulkcopy 没有设为 true，仍可使用 SELECT 语句，将数据复制到一个临时表中。

```
SELECT * INTO #student_backup FROM user_db.student_admin.student
```

当使用 SELECT..INTO 语句创建新表时，引用完整性定义并没有被转移到新表中。

必须把 select into/bulkcopy 选项设为 true，这一需求可能使迁移过程变得复杂。如果必须使用 SELECT 语句把数据复制到表中，先创建表，然后使用 INSERT INTOUSELECT 语句加载表。Oracle 和 SQL Server 的语法是一致的，并且不需要设定任何数据库选项。

在 Microsoft SQL Server 中，用于创建视图的语法与 Oracle 相似。

Oracle	Microsoft SQL Server
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW [*schema*.]*view_name* [(*column_name* [, *column_name*]...)] AS *select_statement* [WITH CHECK OPTION [CONSTRAINT *name*]] [WITH READ ONLY]	CREATE VIEW [*owner*.]*view_name* [(*column_name* [, *column_name*]...)] [WITH ENCRYPTION] AS *select_statement* [WITH CHECK OPTION]

SQL Server 的视图要求该表存在，并且视图所有者有权访问 SELECT 语句中所指定的表（与 Oracle FORCE 选项类似）。

默认情况下，并不检查视图上的数据修改语句，来确定受影响的行是否在视图的作用域内。要检查所有的修改，则使用 WITH CHECK OPTION。WITH CHECK OPTION 的主要差异在于，Oracle 将其定义为一个约束，而 SQL Server 没有。其它方面，两者是相同的。

定义视图时，Oracle 提供了 WITH READ ONLY 选项。通过将 SELECT 权限仅授予视图用户，SQL Server 应用程序也可获得相同的结果。

SQL Server 和 Oracle 视图均支持使用数学表达式、函数和常量表达式创建派生列。一些 SQL Server 特定的差异有：

- 如果数据修改语句只影响一个基表，则可在多个视图上允许使用数据修改语句（INSERT 或 UPDATE）。在一个语句中，数据修改语句不能用于多于一个表。
- 视图中的 text 或 image 列不能使用 READTEXT 或 WRITETEXT。
- 不能使用 ORDER BY、COMPUTE、FOR BROWSE 或 COMPUTE BY 子句。
- 视图中不能使用 INTO 关键字。

当一个视图是由外部联接定义的，并使用该联接内表一个列上的限定条件进行查询时，SQL Server 和 Oracle 给出的结果可能不同。在大多数情况下，Oracle 视图可以方便地转成 SQL Server 视图。

Oracle	Microsoft SQL Server
CREATE VIEW STUDENT_ADMIN.STUDENT_GPA (SSN, GPA) AS SELECT SSN, ROUND(AVG**(DECODE(grade ,'A', 4 ,'A+', 4.3**	CREATE VIEW STUDENT_ADMIN.STUDENT_GPA (SSN, GPA) AS SELECT SSN, ROUND(AVG**(CASE grade WHEN 'A' THEN 4 WHEN 'A+' THEN 4.3**

,'A-', 3.7 ,'B', 3 ,'B+', 3.3 ,'B-', 2.7 ,'C', 2 ,'C+', 2.3 ,'C-', 1.7 ,'D', 1 ,'D+', 1.3 ,'D-', 0.7 ,0)),2) FROM STUDENT_ADMIN.GRADE GROUP BY SSN	WHEN 'A-' THEN 3.7 WHEN 'B' THEN 3 WHEN 'B+' THEN 3.3 WHEN 'B-' THEN 2.7 WHEN 'C' THEN 2 WHEN 'C+' THEN 2.3 WHEN 'C-' THEN 1.7 WHEN 'D' THEN 1 WHEN 'D+' THEN 1.3 WHEN 'D-' THEN 0.7 ELSE 0 END),2) FROM STUDENT_ADMIN.GRADE GROUP BY SSN

Microsoft SQL Server 提供聚集和非聚集的索引结构。这些索引由页组成，页又构成称为“B 树”的分支结构（类似于 Oracle B 树索引结构）。起始页（根级）指定了表中值的范围。根级页的每个范围都指向另一页（决定节点），它包含的表值范围更窄。依次，这些决定节点可以指向其它决定节点，进一步缩小搜索范围。分支结构的最终级别称为叶级。

聚集索引

在 Oracle 中，聚集索引实现为用索引组织的表。聚集索引与表物理地结合在一起。表和索引共享同一存储区域。聚集索引按照索引的顺序物理地重排数据行，构 成中间决定节点。索引的叶级页包含实际的表数据。这种结构只允许每个表一个聚集索引。一旦在表上施加了 PRIMARY KEY 或 UNIQUE 约束，Microsoft SQL Server 就会自动为该表创建一个聚集索引。聚集索引用于：

- 主键
- 未被更新的列
- 使用 BETWEEN、>、>=、< 和 <= 之类的运算符，返回一个值的范围的查询，例如：

```
SELECT * FROM STUDENT WHERE GRAD_DATE
BETWEEN '1/1/97' AND '12/31/97'
```

- 返回大结果集的查询：

```
SELECT * FROM STUDENT WHERE LNAME = 'SMITH'
```

- 在排序操作（ORDER BY、GROUP BY）中使用的列。

例如，在 STUDENT 表上，主键 ssn 上加入一个非聚集索引可能是很有帮助的，在 lname、fname（姓、名）上可以创建聚集索引，因为这是将学生分组的常用方法。

- 将一个表上的活动进行分布，以防止出现“热点”。热点通常是由于多个用户使用升序键对一个表插入造成的。这种应用场景通常用行级锁定来处理。

在 SQL Server 中，删除和重新创建聚集索引是重组表的一种常用技巧。使用这种方法，可以很容易地保证，在磁盘上页是连续的，且可以在表上方便地重建一些可用空间。这与 Oracle 中的导出、删除和导入表类似。

SQL Server 聚集索引和 Oracle 聚集没有任何相同之处。Oracle 聚集是两个或多个表的物理组合，这些表共享相同的数据块，并使用公共列作为聚集键。在 SQL Server 中，没有与 Oracle 聚集相似的结构。

原则上，在表上定义聚集索引可改善 SQL Server 性能和空间管理。如果不了解给定表的查询或更新模式，可在主键上创建聚集索引。

下表给出了，摘自示例应用程序源代码。请注意 SQL Server 聚集索引的使用。

Oracle	Microsoft SQL Server
CREATE TABLE STUDENT_ADMIN.GRADE (SSN CHAR(9) NOT NULL, CCODE VARCHAR2(4) NOT NULL, GRADE VARCHAR2(2) NULL, CONSTRAINT GRADE_SSN_CCODE_PK PRIMARY KEY (SSN, CCODE) CONSTRAINT GRADE_SSN_FK FOREIGN KEY (SSN) REFERENCES STUDENT_ADMIN.STUDENT (SSN), CONSTRAINT GRADE_CCODE_FK FOREIGN KEY (CCODE) REFERENCES DEPT_ADMIN.CLASS (CCODE))	CREATE TABLE STUDENT_ADMIN.GRADE (SSN CHAR(9) NOT NULL, CCODE VARCHAR(4) NOT NULL, GRADE VARCHAR(2) NULL, CONSTRAINT GRADE_SSN_CCODE_PK PRIMARY KEY **CLUSTERED** (SSN, CCODE), CONSTRAINT GRADE_SSN_FK FOREIGN KEY (SSN) REFERENCES STUDENT_ADMIN.STUDENT (SSN), CONSTRAINT GRADE_CCODE_FK FOREIGN KEY (CCODE) REFERENCES DEPT_ADMIN.CLASS (CCODE))

非聚集索引

在非聚集索引中，索引数据和表数据在物理上是分离的，且表中的行不按照索引的顺序存储。可以把 Oracle 索引定义迁移到 Microsoft SQL Server 非聚集索引定义（如下面例子所示）。但是，出于性能方面的考虑，可能希望选择给定表的一个索引，并把它创建为聚集索引。

Oracle	Microsoft SQL Server
CREATE INDEX **STUDENT_ADMIN.**STUDENT_ MAJOR_IDX ON STUDENT_ADMIN.STUDENT (MAJOR) TABLESPACE USER_DATA **PCTFREE 0** **STORAGE (INITIAL 10K NEXT 10K** **MINEXTENTS 1 MAXEXTENTS UNLIMITED)**	CREATE **NONCLUSTERED** INDEX STUDENT_MAJOR_IDX ON **USER_DB.**STUDENT_ ADMIN.STUDENT (MAJOR)

索引的语法和命名

在 Oracle 中，索引名在用户帐户中是唯一的。在 Microsoft SQL Server 中，索引名在表名称中必须是唯一的，但在用户帐户或数据库中则不一定是唯一的。因此，在 SQL Server 中创建或删除索引时，必须指明表的名称和索引名称。此外，SQL Server DROP INDEX 语句可以同时删除多个索引。

Oracle	Microsoft SQL Server
CREATE [UNIQUE] INDEX [*schema*].*index_name* ON [*schema.*]*table_name* (*column_name* [, *column_name*]...) [INITRANS *n*] [MAXTRANS *n*] [TABLESPACE *tablespace_name*] [STORAGE *storage_parameters*] [PCTFREE *n*] [NOSORT] DROP INDEX ABC;	CREATE [UNIQUE] [CLUSTERED	NONCLUSTERED] INDEX index_name ON table (column [,U*n*]) [WITH [PAD_INDEX] [[,] FILLFACTOR = *fillfactor*] [[,] IGNORE_DUP_KEY] [[,] DROP_EXISTING] [[,] STATISTICS_NORECOMPUTE]] [ON *filegroup*] DROP INDEX USER_DB.STUDENT.DEMO_IDX, USER_DB.GRADE.DEMO_IDX

索引数据存储参数

Microsoft SQL Server 中 FILLFACTOR 选项和 Oracle 中的 PCTFREE 变量的作用基本相同。随着表的增大，索引页就会拆分，以容纳新数据。索引必须重新组织，来容纳新的数据值。填充因子百分比只在索引创建时使用，以后不再维护。

在索引最初创建时，FILLFACTOR 选项（值从 0 到 100 ）控制索引页上保留多少空间。如果没有指定，使用默认的填充因子 0 - 这将完全填充索引叶级页，并在每个决定节点页上保留至少一个条目的空间（对于非唯一的聚集索引保留两个）。

使用较低的填充因子值，最初会减少索引页的拆分，并增加 B 树索引结构中级别的数目。使用较高的填充因子值，可更有效地使用索引页空间，访问索引数据需要较少的磁盘 I/O，并减少了 B 树索引结构中级别的数目。

PAD_INDEX 选项指定了，将填充因子应用到索引的决定节点页以及数据页中。

尽管在 Oracle 中，必须调整 PCTFREE 参数来优化性能，但在 CREATE INDEX 语句中，一般不需要加入 FILLFACTOR 选项。填充因子可用于优化性能。仅当使用现有数据在表上创建新索引，并且能够准确预估该数据以后的变化时，填充因子才是有用的。

如果已经把 Oracle 索引的 PCTFREE 设为 0，则考虑使用值为 100 的填充因子。它用于不发生插入和更新的表（只读表）。填充因子设为 100 时，SQL Server 创建每页均百分之百填充的索引。

忽略重复的关键字

对于 Oracle 和 Microsoft SQL Server，用户不能向唯一索引的列插入重复的值。如果这样做，就会产生错识消息。但是，使用 SQL Server，开发人员可以选择 INSERT 或 UPDATE 语句对该错误作出何种反应。

如果在 CREATE INDEX 语句中指定了 IGNORE_DUP_KEY，并执行了产生重复键的 INSERT 或 UPDATE 语句，SQL Server 就会发出一条警告消息，并忽略（不插入）此重复的行。如果没有给索引指定 IGNORE_DUP_KEY，SQL Server 就会发出一个错误信息，并回滚整个 INSERT 语句。有关这些选项的详细信息，请参见 SQL Server Books Online。

Oracle 应用程序可能需要创建只存在很短时间的表。应用程序必须确保，在某些时候可以删除所有为此目的创建的表。如果应用程序做不到这一点，表空间很快就会变得混乱和难以管理。

Microsoft SQL Server 提供了临时表数据库对象，它正是为此目的创建的。这些表总是创建在 tempdb 数据库中。表名称决定了它们在 tempdb 数据库中保留多长时间。

表的名称	说明
#table_name	此局部临时表只在创建它的用户会话或过程期间存在。当用户注销或创建表过程完成后，该表自动被删除。这些表不能在多个用户之间共享。其他数据库用户不能访问此类表。不能授予或撤销此类表的权限。

##table_name	此全局临时表通常也在创建它的用户会话或过程期间存在。这个表可以在多个用户间共享。最后一个引用它的用户会话中断时，它自动被删除。所有其他数据库用户均可访问这个表。不能授予或撤销此类表的权限。

可以给临时表定义索引。仅能在 tempdb 中显式创建且不带有 # 或 ## 前缀的表上定义视图。下面的例子给出了，如何创建一个临时表及其相关索引。用户退出时，表和索引被自动删除。

```
SELECT SUM(ISNULL(TUITION_PAID,0)) SUM_PAID, MAJOR INTO #SUM_STUDENT
FROM USER_DB.STUDENT_ADMIN.STUDENT GROUP BY MAJOR

CREATE UNIQUE INDEX SUM STUDENT IDX ON #SUM STUDENT (MAJOR)
```

您可能会发现，使用临时表带来了很多优势，完全有理由为此而修改程序代码。

Microsoft SQL Server 有一些比 Oracle 更为强健的数据类型。Oracle 和 SQL Server 数据类型之间有多种转换方式。建议使用 DTS 向导自动创建新的 CREATE TABLE 语句。必要时，可修改这些语句。

Oracle	Microsoft SQL Server
CHAR	建议使用 char。因为 char 类型的列使用固定的存储长度，所以，访问时比 varchar 列要快一些。
VARCHAR2 和 LONG	varchar 或 text。（如果 Oracle 列中数据值的长度为 8000 字节或更少，则使用 varchar；否则，必须使用 text。）
RAW 和 LONG RAW	varbinary 或 image。（如果 Oracle 列中数据值的长度为 8000 字节或更少，则使用 varbinary；否则，必须使用 image。）
NUMBER	如果整数在 1 和 255 之间，使用 tinyint。 如果整数在 -32768 和 32767 之间，使用 smallint。 如果整数在 -2,147,483,648 和 2,147,483,647 之间，则使用 int。 如果需要浮点类型数，使用 numeric（有精度和小数位）。 注意：不要使用 float 或 real，因为可能会产生舍入（Oracle NUMBER 和 SQL Server numeric 均不舍入）。 如果不确定，则使用 numeric；它最接近 Oracle NUMBER 数据类型。
DATE	datetime。
ROWID	使用 identity 列类型。
CURRVAL, NEXTVAL	使用 identity 列类型以及 @@IDENTITY、IDENT_SEED() 和 IDENT_INCR() 函数。
SYSDATE	GETDATE()。

USER	USER。

使用 Unicode 数据

Unicode 规范给世界各地广泛使用的几乎所有字符定义了统一的编码方案。所有计算机使用该 Unicode 规范，将 Unicode 数据中的位模式统一转换为字符。这就保证了在所有的计算机上，相同的位模式总是转换成相同的字符。数据可以从一个数据库或计算机自由地传输到另一个上，而不必担心接收系统不能把位模式正确转换为字符。

对于每个字符使用 1 个字节编码的数据类型来说，一个问题是这种数据类型只能表示 256 个不同的字符。这就要求对于不同的字母表，必须采用多个编码规范（或代码页）。它也不能处理像日语 Kanji 或韩国语 Hangul 字母表这样有几千个字符的系统。

Microsoft SQL Server 使用与 SQL Server 一起安装的代码页中的定义，将 char、varchar 和 text 列中的位模式转换成字符。客户计算机使用与操作系统一起安装的代码页解释位模式。有许多种不同的代码页。有些字符在一些代码页中，但不在其它代码页中。有些字符在一些代码页中用一种位模式定义，在其它代码页中则使用另一种位模式。当创建必须处理各种语言的国际化系统时，要为所有计算机挑选满足多个国家语言要求的代码页，就变得十分困难。而且也很难保证，所有计算机与使用不同代码页的系统交互时，能够进行正确转换。

在 Unicode 规范 中，每个字符使用 2 字节编码，从而解决了这一问题。两字节中有足够多的不同模式 (65,536)，可以使单一规范涵盖大多数通用的商务语言。因为所有的 Unicode 系统均使用相同的位模式表示所有字符，当字符从一个系统迁移到另一个系统时，不会出现字符转换错误的问题。

在 SQL Server 中，nchar、nvarchar 和 ntext 数据类型均支持 Unicode 数据。有关 SQL Server 数据类型的详细信息，请参见 SQL Server Books Online。

用户定义的数据类型。

可为 model 数据库或单个用户数据库创建用户定义的数据类型。如果用户定义的数据类型是为 model 定义的，此后创建的所有新用户数据库均可使用该数据类型。用户定义的数据类型是用 sp_addtype 系统存储过程定义的。有关详细信息，请参见 SQL Server Books Online。

可以在 CREATE TABLE 和 ALTER TABLE 语句中使用用户定义的数据类型，并将其与默认值和规则绑定在一起。表创建过程中，如果使用用户定义的数据类型时，明确地定义了为空性，则它优先于数据类型创建时定义的为空性。

此例给出了，如何创建一个用户定义的数据类型。参数为用户类型名称、数据类型和为空性。

```
sp_addtype gender_type, 'varchar(1)', 'not null'
go
```

乍看起来，此功能解决了 Oracle 表创建脚本向 SQL Server 迁移的问题。比如，可以方便地增添 Oracle DATE 数据类型：

```
sp_addtype date, datetime
```

但对于需要大小可变的数据类型，例如 Oracle 数据类型 NUMBER，则没有什么用处。返回的错误消息表明，长度也必须指定。

```
sp_addtype varchar2, varchar
Go
Msg 15091, Level 16, State 1
You must specify a length with this physical type.
```

Microsoft timestamp（时间戳）列

timestamp 列允许 BROWSE 模式更新，并使游标更新操作更为有效。timestamp 是一种数据类型，每次包含 timestamp 列的行被插入或更新时，它都会自动更新。

timestamp 中的值不是作为实际的日期或时间存储的，而是以 binary(8) 或 varbinary(8) 存储的，它表示表中行的事件序列。一个表只能有一个 timestamp 列。

有关详细信息，请参见 SQL Server Books Online。

Microsoft SQL Server 对象的权限可以授予给其它数据库用户、数据库组和 public 角色，也可以被其拒绝或撤销。与 Oracle 不同，SQL Server 不允许对象所有者给对象授予 ALTER TABLE 和 CREATE INDEX 权限。这些权限必须只属于对象所有者。

GRANT 语句在安全系统中创建一个条目，允许当前数据库中的用户处理当前数据库中的数据或执行特定的 Transact-SQL 语句。在 Oracle 和 SQL Server 中，GRANT 语句的语法是相同的。

DENY 语句在安全系统中创建一个条目，拒绝当前数据库中安全帐户的权限，并禁止安全帐户以组或角色成员身份继承权限。Oracle 没有 DENY 语句。REVOKE 语句撤销以前授予当前数据库中一个用户的权限或被其拒绝的权限。

Oracle	Microsoft SQL Server
GRANT {ALL [PRIVILEGES][column_list] | permission_list [column_list]} ON {table_name [(column_list)] | view_name [(column_list)] | stored_procedure_name} TO {PUBLIC | name_list } [WITH GRANT OPTION]	GRANT {ALL [PRIVILEGES] | *permission*[,Un]} { [(*column*[,Un])] ON {*table* | *view*} | ON {*table* | *view*}[(*column*[,Un])] | ON {*stored_procedure* | *extended_procedure*} }

	TO *security_account*[,Un]
	[WITH GRANT OPTION]
	[AS {*group* | *role*}]
	REVOKE [GRANT OPTION FOR]
	{ALL [PRIVILEGES] | *permission*[,Un]}
	{
	[(*column*[,Un])] ON {*table* | *view*}
	| ON {*table* | *view*}[(*column*[,Un])]
	| {*stored_procedure* | *extended_procedure*}
	}
	{TO | FROM}
	security_account[,Un]
	[CASCADE]
	[AS {*group* | *role*}]
	DENY
	{ALL [PRIVILEGES] | permission[,Un]}
	{
	[(*column*[,Un])] ON {*table* | *view*}
	| ON {table | view}[(*column*[,Un])]
	| ON {*stored_procedure* | *extended_procedure*}
	}
	TO *security_account*[,Un]
	[CASCADE]

有关对象级权限的详细信息，请参见 SQL Server Books Online。

在 Oracle 中，REFERENCES 权限只能授予一个用户。SQL Server 则允许将 REFERENCES 权限授予数据库用户和数据库组。在 Oracle 和 SQL Server 中，INSERT、UPDATE、DELETE 和 SELECT 权限授予的方式相同。

实施数据完整性和业务规则

实施数据完整性确保了数据库中数据的质量。在表的规划中，有两个重要的步骤，即识别列的有效值，以及确定如何在列中实施数据完整性。数据完整性可以分为四个类别，并用不同的方法来实施。

完整性类型	实施方式
实体完整性	PRIMARY KEY 约束 UNIQUE 约束

	IDENTITY 属性
范围完整性	DEFAULT 定义 FOREIGN KEY 约束 CHECK 约束 为空性
引用完整性	范围 DEFAULT 定义 FOREIGN KEY 约束 CHECK 约束 为空性
用户定义的完整性	在 CREATE TABLE 中的列级和表级约束 存储过程 触发器。

实体完整性把一行定义为特定表的一个单独实体。实体完整性通过索引、UNIQUE 约束、PRIMARY KEY 约束或 IDENTITY 属性，来实施表的标识符列或主键的完整性。

约束的命名

应该始终显式命名约束。如果没有，则 Oracle 和 Microsoft SQL Server 使用不同的命名规则隐式命名约束。这些命名上的差别会给迁移过程带来不必要的麻烦。在删除或禁用约束时，就会造成不一致，因为必须使用名称，约束才能被删除。对于 Oracle 和 SQL Server 来说，显式命名约束的语法是相同的。

```
CONSTRAINT constraint_name
```

主键和唯一列

SQL-92 标准要求，主键中的所有值应该唯一，并且该列不允许有空值。一旦定义了 PRIMARY KEY 或 UNIQUE 约束，Oracle 和 Microsoft SQL Server 通过自动创建唯一索引，来实施唯一性。此外，主键列自动定义为 NOT NULL。每个表只允许一个主键。

对主键来说，默认地创建一个 SQL Server 聚集索引，尽管也可以请求非聚集索引。主键上的 Oracle 索引可以通过删除或禁用该约束来删除，而 SQL Server 索引只能通过删除该约束来删除。

在两种 RDBMS 中，均可使用 UNIQUE 约束来定义备用键。在任一表上，均可定义多个 UNIQUE 约束。UNIQUE 约束列可为空。在 SQL Server 中，除非另外指定，默认创建非聚集索引。

在迁移应用程序时，要注意，对于完全唯一键（单个或多个列索引），SQL Server 只允许一行包含 NULL 值，而 Oracle 允许任意数量的行包含 NULL 值。

Oracle	Microsoft SQL Server

CREATE TABLE DEPT_ADMIN.DEPT (DEPT **VARCHAR2**(4) NOT NULL, DNAME **VARCHAR2**(30) NOT NULL, CONSTRAINT DEPT_DEPT_PK **PRIMARY KEY (DEPT)** **USING INDEX TABLESPACE USER_DATA** **PCTFREE 0 STORAGE (** **INITIAL 10K NEXT 10K** **MINEXTENTS 1 MAXEXTENTS UNLIMITED),** CONSTRAINT DEPT_DNAME_UNIQUE **UNIQUE (DNAME)** **USING INDEX TABLESPACE USER_DATA** **PCTFREE 0 STORAGE (** **INITIAL 10K NEXT 10K** **MINEXTENTS 1 MAXEXTENTS UNLIMITED)** **)**	CREATE TABLE **USER_DB.**DEPT_ADMIN.DEPT (DEPT **VARCHAR**(4) NOT NULL, DNAME **VARCHAR**(30) NOT NULL, CONSTRAINT DEPT_DEPT_PK **PRIMARY KEY CLUSTERED (DEPT),** CONSTRAINT DEPT_DNAME_UNIQUE **UNIQUE NONCLUSTERED (DNAME)**)

增加和删除约束

禁用约束可以提高数据库性能和简化数据复制过程。例如，在远程站点重建或复制表数据时，不需要再重复约束检查，因为数据最初插到表中时，数据完整性已经检查过了。可以编写一个 Oracle 应用程序，禁用或启用约束（除 PRIMARY KEY 和 UNIQUE 外）。在 Microsoft SQL Server 中，将 ALTER TABLE 语句与 CHECK 和 WITH ONCHECK 选项一起使用，也可实现上述过程。

此插图给出了，这一过程的对比。

在 SQL Server 中，可以使用 NOCHECK 子句和 ALL 关键字延迟所有的表约束。

如果 Oracle 应用程序要使用 CASCADE 选项禁用或删除 PRIMARY KEY 或 UNIQUE 约束，则可能需要重写一些代码，因为 CASCADE 选项禁用或删除父约束及其相关的任何子完整性约束。

下面是该语法的一个示例：

```
DROP CONSTRAINT DEPT_DEPT_PK CASCADE
```

必须修改 SQL Server 应用程序，使其先删除子约束，然后再删除父约束。例如，要删除 DEPT 表上的 PRIMARY KEY 约束，必须删除列 STUDENT.MAJOR 和 CLASS.DEPT 上的外键。下面是该语法的一个示例：

```
ALTER TABLE STUDENT
DROP CONSTRAINT STUDENT_MAJOR_FK
ALTER TABLE CLASS
DROP CONSTRAINT CLASS_DEPT_FK
ALTER TABLE DEPT
DROP CONSTRAINT DEPT_DEPT_PK
```

ALTER TABLE 语法（用于增添和删除约束）对 Oracle 和 SQL Server 几乎是相同的。

产生连续的数值

如果 Oracle 应用程序使用 SEQUENCE，则可以方便地对它进行修改，来使用 Microsoft SQL Server IDENTITY 属性。

类别	Microsoft SQL Server IDENTITY
语法	CREATE TABLE new_employees (Empid int IDENTITY (1,1), Employee_Name varchar(60), CONSTRAINT Emp_PK PRIMARY KEY (Empid)) **如果增加间隔为 5：** CREATE TABLE new_employees (Empid int IDENTITY (1,5), Employee_Name varchar(60), CONSTRAINT Emp_PK PRIMARY KEY (Empid))
每个表中的标识符列	一个
允许空值	否
可否使用默认约束、值	不能使用
实施唯一性	是
在 INSERT 、SELECT INTO 或大容量复制语句完成之后，查询当前最大的标识编号	@@IDENTITY (function)
返回在标识符列创建过程中指定的种	IDENT_SEED('*table_name*')

子值	
返回在标识符列创建过程中指定的增量值	IDENT_INCR('*table_name*')
SELECT 语法	当引用具有 IDENTITY 属性的列时，在 SELECT、INSERT、UPDATE 和 DELETE 语句中，可以使用关键字 IDENTITYCOL 代替列的名称。

尽管 IDENTITY 属性在一个表中自动完成行编号，但不同的表（每个表均有其自己的标识符列）的属性值可能会相同。这是因为，IDENTITY 属性只保证在使用它的表中唯一。如果应用程序必须生成一个标识符列，其在整个数据库中、或者甚至每个联网计算机上的每个数据库中都是唯一，则使用 ROWGUIDCOL 属性、uniqueidentifier 数据类型和 NEWID 函数。SQL Server 使用全局唯一标识符列，来合并复制，确保在表的多个副本中，行被唯一地标识。

有关创建和修改标识符列的详细信息，请参见 SQL Server Books Online。

对于给定列，范围完整性实施了有效的条目。范围完整性是通过限制可能值的类型（通过数据类型）、格式（通过 CHECK 约束）或范围（通过 REFERENCE 和 CHECK 约束）实施的。

DEFAULT 和 CHECK 约束

Oracle 把默认值作为列属性，而 Microsoft SQL Server 把默认值作为约束。SQL Server DEFAULT 约束可以包含常量、不带参数的内置函数（niladci 函数）或 NULL。

要方便地迁移 Oracle DEFAULT 列属性，应该在 SQL Server 列级中定义 DEFAULT 约束，而不必使用约束名称。对于每个 DEFAULT 约束，SQL Server 均生成一个唯一的名称。

在 Oracle 和 SQL Server 中，定义 CHECK 约束的语法是相同的。搜索条件必须对一个布尔表达式进行求值，并且不能包括子查询。列级 CHECK 约束只能引用受约束的列，表级 CHECK 约束只可以引用受约束表中的列。可以为一个表定义多个 CHECK 约束。在 CREATE TABLE 语句中，SQL Server 语法规定，在一个列上只允许创建一个列级 CHECK 约束，约束可以有多个条件。

测试修改后的 CREATE TABLE 语句的最好方法是，使用 SQL Server 中的 SQL Server 查询分析器，并分析该语法。结果窗格给出所有的错误。有关约束语法的详细信息，请参见 SQL Server Books Online。

Oracle	Microsoft SQL Server
CREATE TABLE STUDENT_ADMIN.STUDENT (SSN CHAR(9) NOT NULL, FNAME **VARCHAR2**(12) NULL, LNAME **VARCHAR2**(20) NOT NULL,	CREATE TABLE **USER_DB.**STUDENT _ADMIN.STUDENT (SSN CHAR(9) NOT NULL, FNAME **VARCHAR**(12) NULL, LNAME **VARCHAR**(20) NOT NULL,

GENDER CHAR(1) NOT NULL **CONSTRAINT STUDENT_GENDER_CK CHECK (GENDER IN ('M','F')),** MAJOR **VARCHAR2**(4) **DEFAULT 'Undc' NOT NULL,** BIRTH_DATE **DATE** NULL, TUITION_PAID **NUMBER**(12,2) NULL, TUITION_TOTAL **NUMBER**(12,2) NULL, START_DATE **DATE** NULL, GRAD_DATE **DATE** NULL, LOAN_AMOUNT **NUMBER**(12,2) NULL, DEGREE_PROGRAM CHAR(1) **DEFAULT 'U' NOT NULL CONSTRAINT STUDENT_DEGREE_CK CHECK (DEGREE_PROGRAM IN ('U', 'M', 'P', 'D')),** **...**	GENDER CHAR(1) NOT NULL **CONSTRAINT STUDENT_GENDER_CK CHECK (GENDER IN ('M','F')),** MAJOR **VARCHAR**(4) **DEFAULT 'Undc' NOT NULL,** BIRTH_DATE **DATETIME** NULL, TUITION_PAID **NUMERIC**(12,2) NULL, TUITION_TOTAL **NUMERIC**(12,2) NULL, START_DATE **DATETIME** NULL, GRAD_DATE **DATETIME** NULL, LOAN_AMOUNT **NUMERIC**(12,2) NULL, DEGREE_PROGRAM CHAR(1) **DEFAULT 'U' NOT NULL CONSTRAINT STUDENT_DEGREE_CK CHECK (DEGREE_PROGRAM IN ('U', 'M', 'P','D')),** **...**

有关用户定义的规则和默认值的说明 :出于向后兼容的考虑,仍保留 Microsoft SQL Server 规则和默认值，但对于新的应用程序开发，建议使用 CHECK 和 DEFAULT 约束。有关详细信息，请参见 SQL Server Books Online。

为空性

Microsoft SQL Server 和 Oracle 创建列约束，来实施为空性。Oracle 列默认为 NULL，除非在 CREATE TABLE 或 ALTER TABLE 语句中指定了 NOT NULL。在 Microsoft SQL Server 中，数据库和会话设置可以覆盖列定义中使用的数据类型为空性。

所有的 SQL 脚本（不论是 Oracle 还是 SQL Server）都应该为每一列显式定义 NULL 和 NOT NULL。要了解这一策略是如何实施的，请参见 Oratable.sql 和 Sstable.sql 示例表创建脚本。如果没有显式定义，列的为空性遵循下列规则。

Null 设置	说明
使用用户定义的数据类型定义列	SQL Server 使用该数据类型创建时指定的为空性。使用 sp_help 系统存储过程，来获取数据类型默认的为空性。
使用系统提供的数据	如果系统提供的数据类型只有一个选择，则它

类型定义列	优先。现在，bit 数据类型只能定义为 NOT NULL。 如果任何会话设置是 ON（使用 SET 打开），那么： 如果 ANSI_NULL_DFLT_ON 为 ON，则赋值 NULL。 如果 ANSI_NULL_DFLT_OFF 为 ON，则赋值 NOT NULL。 如果配置了任何数据库设置（用 sp_dboption 系统存储过程更改），那么： 如果 ANSI null default 为 true，则赋值 NULL。 如果 ANSI null default 为 false，则赋值 NOT NULL。
NULL/NOT NULL 没有定义	如果没有显式地定义（没有设置任何 ASNI_NULL_DFLT 选项），会话没有改变，并且数据库设为默认值（ANSI null default 为 false），那么，SQL Server 赋值为 NOT NULL。

下表提供了一个用于定义引用完整性约束的语法比较。

约束	Oracle	Microsoft SQL Server
PRIMARY KEY	[CONSTRAINT *constraint_name*] PRIMARY KEY (*col_name* [, *col_name2* [..., *col_name16*]]) [USING INDEX *storage_parameters*]	[CONSTRAINT *constraint_name*] PRIMARY KEY [CLUSTERED | NONCLUSTERED] (*col_name* [, *col_name2* [..., *col_name16*]]) [ON *segment_name*] [NOT FOR REPLICATION]
UNIQUE	[CONSTRAINT *constraint_name*] UNIQUE (*col_name* [, *col_name2* [..., *col_name16*]]) [USING INDEX *storage_parameters*]	[CONSTRAINT *constraint_name*] UNIQUE [CLUSTERED | NONCLUSTERED](*col_name* [, *col_name2* [..., *col_name16*]]) [ON *segment_name*] [NOT FOR REPLICATION]
FOREIGN KEY	[CONSTRAINT *constraint_name*] [FOREIGN KEY (*col_name* [, *col_name2* [..., *col_name16*]])] REFERENCES [*owner.*]*ref_table* [(*ref_col*	[CONSTRAINT *constraint_name*] [FOREIGN KEY (*col_name* [, *col_name2* [..., *col_name16*]])] REFERENCES [*owner.*]*ref_table* [(*ref_col* [, *ref_col2* [..., *ref_col16*]])]

	[, *ref_col2* [..., *ref_col16*]])] [ON DELETE CASCADE]	[NOT FOR REPLICATION]
DEFAULT	**列属性，但不是约束** DEFAULT (*constant_expression*)	[CONSTRAINT *constraint_name*] DEFAULT {*constant_expression* | *niladic-function* | NULL} [FOR *col_name*] [NOT FOR REPLICATION]
CHECK	[CONSTRAINT *constraint_name*] CHECK (*expression*)	[CONSTRAINT *constraint_name*] CHECK [NOT FOR REPLICATION] (*expression*)

NOT FOR REPLICATION 子句用于复制过程中暂停列级、FOREIGN KEY 和 CHECK 约束的使用。

外键

在每种 RDBMS 中，定义外键的规则是相似的。在外键子句中指定的列的数目和每个列的数据类型必须和 REFERENCES 子句相符。在该列输入的非空值在 REFERENCES 子句中定义的表和列中必须存在，并且被引用表的列中，必须有一个 PRIMARY KEY 或 UNIQUE 约束。

Microsoft SQL Server 约束提供了引用同一数据库中表的能力。要实现跨数据库的引用完整性，须使用基于表的触发器。

Oracle 和 SQL Server 都支持自引用的表，在该表中引用（外键）指向同一表的一个或多个列。例如，CLASS 表中的 prereq 列可以引用 CLASS 表中的 ccode 列，以保证输入了有效的课程代码（作为先决条件）。

Oracle 使用 CASCADE DELETE 子句，来实现级联删除和更新，而 SQL Server 使用表触发器，来提供相同的功能。有关详细信息，请参见本章后面的“SQL 语言支持”。

用户定义的完整性允许定义不属于任何其它完整性类别的业务规则。

存储过程

Microsoft SQL Server 存储过程使用 CREATE PROCEDURE 语句，接受和返回用户提供的参数。除临时存储过程外，其它存储过程均在当前数据库中创建。下表给出了其 Oracle 和 SQL Server 语法。

Oracle	**Microsoft SQL Server**
CREATE OR REPLACE PROCEDURE [*user.*]*procedure*	CREATE PROC[EDURE] *procedure_name* [;*number*]

[(*argument* [IN | OUT] *datatype* [, *argument* [IN | OUT] *datatype*] {IS | AS} block	[{*@parameter data_type*} [VARYING] [= *default*] [OUTPUT]] [,U*n*] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}] [FOR REPLICATION] AS *sql_statement* [U*n*]

在 SQL Server 中，临时过程创建在 tempdb 数据库中，对于局部临时过程，在 *procedure_name* 前加一个数字符（#*procedure_name*），全局临时过程前加两个数字符（##*procedure_name*）。

局部临时过程只能由创建它的用户使用。运行局部临时过程的权限不能授予其他用户。用户会话结束后，局部临时过程自动被删除。

所有的 SQL Server 用户均可使用全局临时过程。如果创建了全局临时过程，所有的用户均可以访问它，权限不能被显式地撤销。使用过程的最后一个用户会话结束后，全局临时过程被删除。

SQL Server 存储过程最多可嵌套 32 级。嵌套级在调用过程开始执行时递增，调用过程执行结束时递减。

下面的例子给出了，如何使用 Transact-SQL 存储过程替代 Oracle PL/SQL 打包的函数。Transact-SQL 版本要简单得多，因为 SQL Server 可以直接从存储过程中的 SELECT 语句返回结果集，而无需使用游标。

Oracle	**Microsoft SQL Server**
CREATE OR REPLACE PACKAGE STUDENT_ADMIN.P1 AS ROWCOUNT NUMBER :=0; CURSOR C1 RETURN STUDENT%ROWTYPE; FUNCTION SHOW_RELUCTANT_STUDENTS (WORKVAR OUT VARCHAR2) RETURN NUMBER; END P1; / CREATE OR REPLACE PACKAGE BODY STUDENT_ADMIN.P1 AS CURSOR C1 RETURN STUDENT%ROWTYPE IS SELECT * FROM STUDENT_ADMIN.STUDENT	CREATE PROCEDURE STUDENT_ADMIN.SHOW_ RELUCTANT_STUDENTS AS SELECT FNAME+" +LNAME+', social security number'+ SSN+' is not enrolled in any classes!' FROM STUDENT_ADMIN.STUDENT S WHERE NOT EXISTS (SELECT 'X' FROM STUDENT_ADMIN.GRADE G WHERE G.SSN=S.SSN)

WHERE NOT EXISTS
(SELECT 'X' FROM STUDENT_ADMIN.GRADE
WHERE GRADE.SSN=STUDENT.SSN) ORDER BY SSN;
 	
FUNCTION SHOW_RELUCTANT_STUDENTS
(WORKVAR OUT VARCHAR2) RETURN NUMBER
IS
WORKREC STUDENT%ROWTYPE;
BEGIN
IF NOT C1%ISOPEN THEN OPEN C1;
ROWCOUNT :=0;
ENDIF;
FETCH C1 INTO WORKREC;
IF (C1%NOTFOUND) THEN
CLOSE C1;
ROWCOUNT :=0;
ELSE
WORKVAR := WORKREC.FNAME||'
'||WORKREC.LNAME||
', social security number '||WORKREC.SSN||' is not enrolled
in any classes!';
ROWCOUNT := ROWCOUNT + 1;
ENDIF;
RETURN(ROWCOUNT);
	ORDER BY SSN
	RETURN@@ROWCOUNT
	GO

EXCEPTION WHEN OTHERS THEN IF C1%ISOPEN THEN CLOSE C1; ROWCOUNT :=0; ENDIF; RAISE_APPLICATION_ERROR(-20001,SQLERRM); END SHOW_RELUCTANT_STUDENTS; END P1; /	

SQL Server 不支持类似 Oracle 包或函数的结构，也不支持创建存储过程的 CREATE OR REPLACE 选项。

延迟存储过程的执行

Microsoft SQL Server 提供了 WAITFOR，它允许开发人员指定触发语句块、存储过程或事务执行的时间、时间间隔或事件。对于 Transact-SQL 来说，它就相当于 Oracle 的 dbms_lock.sleep。

```
WAITFOR {DELAY 'time' | TIME 'time'}
```

此处

- DELAY

  通知 Microsoft SQL Server 等待，直到指定的时间结束，最多 24 小时。

- *'time'*

  等待的时间。*time* 可用 datetime 数据可接受的一种格式来指定，或指定为一个局部变量。不能指定日期；因此，不允许 datetime 值的数据部分。

TIME

通知 SQL Server 等到指定的时间。

例如：

```
BEGIN
WAITFOR TIME '22:20'
EXECUTE update_all_stats
END
```

指定存储过程的参数

要指定存储过程的参数，请使用以下语法：

Oracle	Microsoft SQL Server
Varname datatype DEFAULT <value>;	{@*parameter* *data_type*} [VARYING] [= *default*] [OUTPUT]

触发器

Oracle 和 Microsoft SQL Server 都有触发器，但它们的实现方式有些不同。

说明	Oracle	Microsoft SQL Server
每个表的触发器数	没有限制	没有限制

量		
触发器在 INSERT、UPDATE 和 DELETE 之前执行	是	否
触发器在 INSERT、UPDATE 和 DELETE 之后执行	是	是
语句级触发器	是	是
行级触发器	是	否
在执行前检查约束	是，除非触发器被禁用。	是。此外，这是数据转换服务的一个选项。
引用 UPDATE 或 DELETE 触发器中旧的或以前的值。	:old	DELETED.*column*
引用 INSERT 触发器中的新值	:new	INSERTED.*column*
禁用触发器	ALTER TRIGGER	数据转换服务中的选项

DELETED 和 INSERTED 是 SQL Server 为触发器语句创建的逻辑（概念）表。它们与定义触发器的表结构上是相似的，并保存用户操作可能改变的行的旧值或新值。这些表跟踪 Transact-SQL 中的行级更改。这些表提供了与 Oracle 行级触发器相同的功能。当在 SQL Server 中执行 INSERT、UPDATE 或 DELETE 语句时，行被同时加入到触发器表、INSERTED 和 DELETED 表中。

INSERTED 和 DELETED 表与触发器表是完全相同的。它们具有相同的列名和数据类型。例如，如果一个触发器放置在 GRADE 表上，INSERTED 和 DELETED 表就具有这样的结构。

GRADE	INSERTED	DELETED
SSN CHAR(9) CCODE VARCHAR(4) GRADE VARCHAR(2)	SSN CHAR(9) CCODE VARCHAR(4) GRADE VARCHAR(2)	SSN CHAR(9) CCODE VARCHAR(4) GRADE VARCHAR(2)

触发器可以检查 INSERTED 和 DELETED 表，来决定触发器应该采取什么样的操作。INSERTED 表与 INSERT 和 UPDATE 语句一起使用。DELETED 表与 DELETE 和 UPDATE 语句一起使用。

UPDATE 语句使用 INSERTED 和 DELETED 表，因为进行 UPDATE 操作时，SQL Server 总是删除旧行，并插入新行。因此，执行完 UPDATE 操作，INSERTED 表中的行与 DELETED 表中的行总是完全相同的。

下面的示例使用 INSERTED 和 DELETED 表，来代替 PL/SQL 行级触发器。一个完全外部联接用于查询每个表中的所有行。

Oracle	Microso

```
CREATE TRIGGER STUDENT_ADMIN.TRACK_GRADES
AFTER
INSERT OR UPDATE OR DELETE
ON STUDENT_ADMIN.GRADE
FOR EACH ROW
BEGIN
INSERT INTO GRADE_HISTORY(
TABLE_USER, ACTION_DATE,
OLD_SSN, OLD_CCODE,
OLD_GRADE, NEW_SSN,
NEW_CCODE, NEW_GRADE)
VALUES (USER, SYSDATE,
:OLD.SSN, :OLD.CCODE, :OLD.GRADE, :NEW.SSN, :NEW.CCODE, :NEW.GRADE),
END;
```

```
CREATE
STUDENT
ON STUD
FOR INSE
AS
INSERT IN
TABLE_U
OLD_SSN
OLD_GRA
NEW_SSN
NEW_GRA
SELECT U
OLD.SSN,
OLD.GRA
NEW.SSN
NEW.GRA
FROM
OUTER JO
DELETED
OLD.SSN
```

只能在当前数据库中创建触发器，但可以引用当前数据库之外的对象。如果使用所有者名称来限定触发器，应使用相同的方法限定表的名称。

触发器可以嵌套 32 级。如果触发器更改了一个其上有另一触发器的表，第二个触发器就被启动，并调用第三个触发器，依此类推。如果此链中的某一触发器产生了一个死循环，就会超出嵌套级，触发器被取消。此外，如果表中一列上的更新触发器导致对另一列的更新，该更新触发器只被启用一次。

Microsoft SQL Server 声明引用完整性（DRI）不提供数据库之间的引用完整性。如果需要数据库之间的引用完整性，则使用触发器。

Transact-SQL 触发器不允许下列语句：

- CREATE **语句**（DATABASE、TABLE、INDEX、PROCEDURE、DEFAULT、RULE、TRIGGER、SCHEMA **和** VIEW）
- DROP **语句**（TRIGGER、INDEX、TABLE、PROCEDURE、DATABASE、VIEW、DEFAULT、RULE）
- ALTER **语句**（DATABASE、TABLE、VIEW、PROCEDURE、TRIGGER）
- TRUNCATE TABLE
- GRANT, REVOKE, DENY
- UPDATE STATISTICS
- RECONFIGURE
- UPDATE STATISTICS
- RESTORE DATABASE, RESTORE LOG

- LOAD LOG, DATABASE
- DISK 语句
- SELECT INTO（因为它创建一个表）

有关触发器的详细信息，请参见 SQL Server Books Online。

事务、锁定和并发性

此节解释了，在 Oracle 和 Microsoft SQL Server 中，事务是如何执行的，并揭示两种数据库中锁定过程及并发性问题的差异。

在 Oracle 中，进行插入、更新或删除操作时，事务自动启动。要把所有的更改保存到数据库中，应用程序必须执行 COMMIT 命令。如果没有执行 COMMIT，所有的更改都被回滚或自动取消。

默认情况下，Microsoft SQL Server 在每个插入、更新或删除操作之后，自动执行一个 COMMIT 语句。因为数据是自动存储的，所以不能回滚任何更改。可以使用隐性或显性事务模式，改变这种默认行为。

隐性事务模式允许 SQL Server 与 Oracle 一样运作，它可通过 SET IMPLICIT_TRANSACTION ON 语句来启动。如果这个选项是 ON，并且没有任何待处理事务，每个 SQL 语句就会自动启动一个事务。如果有打开的事务，就不能启动新的事务。要使所有的更改生效并释放所有的锁定，用户必须使用 COMMIT TRANSACTION 语句显式地提交开启的事务。

显性事务是一组 SQL 语句，它用下列事务分隔符括起来：

- BEGIN TRANSACTION [*transaction_name*]
- COMMIT TRANSACTION [*transaction_name*]
- ROLLBACK TRANSACTION [transaction_name | savepoint_name]

在下列示例中，English 系被改为 Literature 系。请注意 BEGIN TRANSACTION 和 COMMIT TRANSACTION 语句的用法。

Oracle	Microsoft SQL Server
INSERT INTO DEPT_ADMIN.DEPT (DEPT, DNAME) VALUES ('LIT', 'Literature') / UPDATE DEPT_ADMIN.CLASS SET MAJOR = 'LIT' WHERE MAJOR = 'ENG' / UPDATE	**BEGIN TRANSACTION** INSERT INTO DEPT_ADMIN.DEPT (DEPT, DNAME) VALUES ('LIT', 'Literature') UPDATE DEPT_ADMIN.CLASS SET DEPT = 'LIT' WHERE DEPT = 'ENG'

STUDENT_ADMIN.STUDENT SET MAJOR = 'LIT' WHERE MAJOR = 'ENG' / DELETE FROM DEPT_ADMIN.DEPT WHERE DEPT = 'ENG' / **COMMIT** /	UPDATE STUDENT_ADMIN.STUDENT SET MAJOR = 'LIT' WHERE MAJOR = 'ENG' DELETE FROM DEPT_ADMIN.DEPT WHERE DEPT = 'ENG' **COMMIT TRANSACTION** **GO**

所有显性事务都必须包含在 BEGIN TRANSACTION...COMMIT TRANSACTION 语句中。SAVE TRANSACTION 和 Oracle SAVEPOINT 语句的作用相同，即为事务设定一个保存点，来允许部分回滚。

事务之间可以嵌套。如果是这样，最外面的嵌套对创建和提交事务，里面的嵌套对跟踪嵌套级。当遇到嵌套的事务时，@@TRANSACTION 函数递增。通常，当带有 BEGINUCOMMIT 对的存储过程或触发器互相调用时，此明显的事务嵌套发生。尽管事务可以嵌套，但它们对 ROLLBACK TRANSACTION 语句的行为没有什么影响。

在存储过程或触发器中，BEGIN TRANSACTION 语句的数目必须和 COMMIT TRANSACTION 语句的数目一致。包含不成对的 BEGIN TRANSACTION 和 COMMIT TRANSACTION 语句的存储过程或触发器执行时就会产生错误信息。如果包含 BEGIN TRANSACTION 和 COMMIT TRANSACTION 语句，语法允许从事务内部调用存储过程和触发器。

只要可能，应把大的事务分为较小的事务。确保在一批作业中，每个事务都是显式定义的。要将可能的并发性冲突降到最低，事务不应该跨越多批作业，也不应等待用户输入。把多个 Transact-SQL 语句合到一个长时间运行的事务，会负面地影响恢复时间，并引起并发性问题。

当使用 ODBC 编程时，可以使用 SQLSetConnectOption 函数，选择隐性或显性事务模式。ODBC 程序选择这种还是那种模式，取决于 AUTOCOMMIT 连接选项。如果 AUTOCOMMIT 设为 ON（默认值），则处于显性模式。如果 AUTOCOMMIT 设为 OFF，则处于隐性模式。

如果使用 SQL Server 查询分析器或其它查询工具执行一个脚本，则可以在脚本上包含前面所示的显式 BEGIN TRANSACTION 语句，或者用 SET IMPLICIT _TRANSACTIONS ON 语句启动脚本。BEGIN TRANSACTION 方法更灵活，但隐式方法与 Oracle 的兼容性更好。

Oracle 和 Microsoft SQL Server 的锁定和隔离策略大不相同。把应用程序从 Oracle 迁移到 SQL Server 时，必须考虑这些差异，确保应用程序的可伸缩性。

对于所有读取数据的 SQL 语句,Oracle 均显式或隐式地使用多版本一致性模型。在此模型中，默认情况下，在读取数据行之前，数据读取者既不获取锁定也不等待其它锁定被释放。当读取者要请求的数据已经被其他写入者更改但还未提交，Oracle 使用其回滚段重建一个行快照，来重新创建旧数据。

Oracle 中的数据写入者请求锁定被更新、删除或插入的数据。这些锁定一直保持到事务结束时为止，可防止其他用户覆盖未提交的更改。

Microsoft SQL Server 具有多粒锁定功能，允许一个事务锁定各种类型的资源。要将锁定开销降至最低，SQL Server 自动在任务相应的级别上锁定资源。在较小粒度上（例如行）进行锁定，可提高并发性，但开销更大，因为如果要对多行锁定，就必须保持更多的锁定。在较大粒度（例如表）上进行锁定，从并发性角度讲是代价很高的，因为对整个表的锁定限制了其它事务对表任何部分的访问，但是它有较小的开销，因为只需要维护较少的锁定。SQL Server 可以锁定以下资源（按照粒度增大的顺序排列）。

资源	说明
RID	行标识符。用于单独地锁定一个单行表。
Key	键，索引内的行锁定。用于保护串行事务中键的范围。
Page	8 KB 数据页或索引页。
Extent	相邻的八个数据页或索引页的组。
Table	整个表，包括所有数据和索引。
DB	数据库。

SQL Server 使用不同的模式锁定资源，这些模式决定了并发事务如何访问资源。

锁定模式	说明
共享 (S)	用于不更改或更新数据的操作(只读操作),例如 SELECT 语句。
更新 (U)	用于可更新的资源。防止一种通常形式的死锁，它在多个会话被读取，锁定并随后可能更新资源时发生。
排它 (X)	用于数据修改操作，例如 UPDATE、INSERT 或 DELETE。确保不能同时对同一资源进行多个更新。
Intent	用于建立一个锁定层次结构。
架构	在执行依赖于表架构的操作时使用。有两种类型的架构锁定：架构稳定性 (Sch-S) 和架构修改 (Sch-M)。

在任何 RDBMS 中，快速释放锁定来提供最大并发性，是很重要的。可以将事务尽可能缩短，以确保快速释放锁定。如果可能，事务不应跨到服务器的多个往返操作，也不应该包括用户“思考”时间。如果使用游标，还需编写应用程序来快速地提取数据，因为未提取的数据扫描可能占据服务器上的共享锁定，从而阻碍更新者进行更新。有关详细信息，请参见本章后面的“使用 ODBC”。

Microsoft SQL Server 和 Oracle 都允许开发人员请求非默认的锁定和隔离行为。在 Oracle 中，最通常的机制是 SELECT 命令上的 FOR UPDATE 子句、SET TRANSACTION READ ONLY 命令和显式 LOCK TABLE 命令。

因为 Oracle 和 SQL Server 的锁定和隔离机制是完全不同的，所以很难将这些锁定选项在两者之间直接对应起来。要更好地理解这一过程，了解 SQL Server 提供的改变其默认锁定行为的选项，是很重要的。

在 SQL Server 中，改变默认锁定模式最常用的机制是 SET TRANSACTION ISOLATION LEVEL 语句，以及 SELECT 及 UPDATE 语句中支持的锁定提示。SET TRANSACTION ISOLATION LEVEL 语句设定用户会话期间的事务隔离级别。这是会话的默认行为，除非在一个 SQL 语句的 FROM 子句中在表级指定了锁定提示。事务隔离这样设定：

```
SET TRANSACTION ISOLATION LEVEL
{
READ COMMITTED
| READ UNCOMMITTED
| REPEATABLE READ
| SERIALIZABLE
}
```

- READ COMMITTED

SQL Server 的默认隔离级别。使用这个选项时，应用程序不能读取还未被其它事务提交的数据。但是，在这个模式中，只要数据被从页中读取，共享锁定就被释放。如果应用程序在同一事务中重新读取相同的数据范围，它会看到其他用户的更改。

- SERIALIZABLE

设定了这个选项，事务就被彼此隔离。如果在查询中不想看到其他用户的更改，应把事务隔离级别设为 SERIALIZBLE。SQL Server 保持所有的共享锁定，直到事务结束为止。在 SELECT 语句中，在表名之后使用 HOLDLOCK 提示，可以在更细微的级别上获得这一效果。使用这两个选项，在保证严格的一致性的同时，也降低了并发性，因此只有在必要时才使用。

- READ UNCOMMITTED

设定了这个选项，SQL Server 读取者就会畅通无阻，这一点与 Oracle 一样。这个选项实现脏读或隔离级别 0 锁定，这意味着不发出共享锁定，并且不支持排它锁定。这个选项设定时，就可以读取未提交的或“脏”数据；在事务结束之前，数据中的值可以被改变，数据集中的行可能出现或消失。这个选项与在事务的所有 SELECT 语句中将所有表上均设定 ONLOCK 有相同的效果。这是四个隔离级别中限制最少的。只有当彻底分析了它怎样影响应用程序中结果的准确性之后，才能使用这个隔离级别。

SQL Server 用两种方式支持 Oracle READ ONLY 功能：

- 如果应用程序中的一些事务需要可重复读取行为，则可能需要使用 SQL Server 提供的 SERIALIZABLE 隔离级别。
- 如果所有的数据库访问都是只读的，可以把 SQL Server 数据库选项设为 READ ONLY，以提高性能。

Oracle 的 SELECTUFOR UPDATE 语句主要在应用程序需要用 WHERE CURRENT OF 语法执行定位更新或删除游标时使用。在这种情况下，可有选择地删除 FOR UPDATE 子句，因为 Microsoft SQL Server 游标是可默认更新的。

默认情况下，SQL Server 游标不保留提取行的锁定。SQL Server 使用一个优化的并发性策略，防止更新彼此覆盖。如果用户要更新或删除一行，但该行自读入游标后已被更改，则 SQL Server 发出一个错误消息。应用程序可以捕获该错误消息，相应地重试更新或删除。要替代这种行为，开发人员可在游标声明中使用 SCROLL_LOCKS。

在一般情况下，更新者之间冲突很少，乐观的并发性策略可支持较高的并发性。如果应用程序的确需要确保行在提取后不能被更改，则使用 SELECT 语句中的 UPDLOCK 提示。这个提示并不阻碍其他读取者读取，但是它防止其他潜在写入者获得数据上的更新锁定。使用 ODBC 时，可以使用 SQLSETSTMTOPTION (U,SQL_CONCURRENCY)= SQL_CONCUR_LOCK 达到这一效果。但是，这两个选项均降低并发性。

Microsoft SQL Server 可以使用 SELECTU*table_name* (TABLOCK) 语句锁定整个表。它与 Oracle LOCK TABLEUIN SHARE MODE 语句执行的操作相同。该锁定允许他人读取表，但是禁止他们更新。默认情况下，锁定一直保持到语句结束为止。如果还加入了关键字 HOLDLOCK (SELECTU*table_name* (TABLOCK HOLDLOCK))，表锁定一直保持到事务结束为止。

可以使用 SELECTU*table_name* (TABLOCKX) 语句，在 SQL Server 表上设置排它锁定。该语句请求表上的排它锁定。它用于防止他人读取或更新表，并一直保持到命令或事务结束时为止。它和 Oracle LOCK TABLEUIN EXCLUSIVE MODE 语句的功能相似。

在显式锁定请求中，SQL Server 并不提供 NOWAIT 选项。

当查询从表中请求行时，Microsoft SQL Server 自动生成页级锁定。但是，如果查询请求占表中很大比例的行，SQL Server 就把锁定从页级升级到表级。这一过程叫做“锁定升级”。

锁定升级使表扫描和对大结果集的操作更有效，这是因为它降低了锁定开销。没有 WHERE 子句的 SQL 语句通常引起锁定升级。

在读操作过程中，当将共享页锁定升级为表锁定时，就会使用共享表锁定 (TABLOCK)。共享表锁定应用于：

- 使用 HOLDLOCK 或 SET TRANSACTION ISOLATION LEVEL SERIALIZABLE 语句时。
- 优化程序选定一个表扫描时。
- 表中共享锁定的累计数目超过锁定升级极限时。

锁定升级极限默认为每表 200 页，但也可以根据表大小，自定义最小和最大极限。共享表锁定还在创建非聚集索引时使用。有关锁定升级极限的详细信息，请参见 SQL Server Books Online。

在写操作过程中，一个 UPDATE 锁定被升级为表锁定时，则应用排它表锁定 (TABLOCKX)。排它表锁定应用于：

- 更新或删除操作没有可用索引时。
- 有排它锁定的表中页的数目超过了锁定升级极限时。
- 创建聚集索引时。

如果 Oracle 不能升级行级锁定，在某些包含 FOR UPDATE 子句的查询中，就可能出现问题。例如，假定 STUDENT 表有 100,000 行数据，并且一个 Oracle 用户执行下列语句：

```
SELECT * FROM STUDENT FOR UPDATE
```

这个语句迫使 Oracle RDBMS 一次锁定 STUDENT 表一行；这可能要花很长时间。它从不升级请求，来锁定整个表。

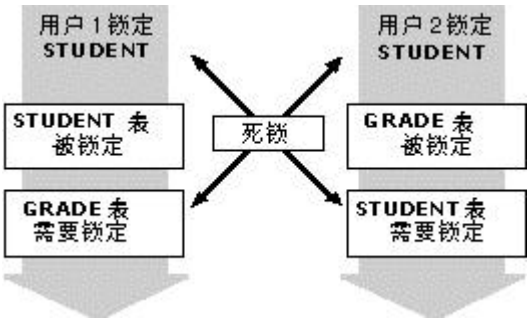
在 SQL Server 中，相同的查询是：

```
SELECT * FROM STUDENT (UPDLOCK)
```

当查询运行时，页级锁定升级为表级锁定，它查询更为有效，并且速度快得多。

死锁

当一个进程锁定另一个进程需要的页或表，而后者又锁定了前者所需要的页时，死锁就发生了。死锁也称为“僵局”。SQL Server 自动检测并处理死锁。如果发现了死锁，服务器就会终止处于“僵局”的用户进程。



每次数据修改后，程序代码都应检测消息编号 1205，它用于指示死锁。如果返回了这个消息编号，则发生了死锁，并且事务被回滚。在这种情况下，应用程序必须重新启动事务。

使用一些简单的技巧，就可以避免死锁：

- 应用程序的各个部分均使用相同的顺序访问表。
- 在每个表上使用聚集索引，来实施显式行排序。
- 使事务保持简短。

有关详细信息，请参见 Microsoft Knowledge Base 文章：Detecting and Avoiding Deadlocks in Microsoft SQL Server（Microsoft SQL Server 死锁的检测和避免）。

在 Oracle 中，要执行远程事务，必须能够使用数据库链接，来访问远程数据库节点。在 SQL Server 中，则必须能够访问一个“远程服务器”。远程服务器是在网络上运行 SQL Server 的一个服务器，用户可使用本地服务器对它进行访问。当一个服务器被设置为远程服务器时，用户无须显式登录，就可以使用其上的系统过程和存储过程。

远程服务器是成对设置的。必须对两个服务器进行配置，使之均将对方作为远程服务器。必须使用 sp_addlinkedserver 系统存储过程或 SQL Server Enterprise Manager，把每个服务器的名称加入它的伙伴名称中。

设置远程服务器之后，使用 sp_addremotelogin 系统存储过程或 SQL Server Enterprise Manager，为必须访问远程服务器的用户设定远程登录 ID。这一步完成之后，必须授予此执行存储过程的权限。

然后，使用 EXECUTE 语句，运行远程服务器上的过程。以下示例执行了远程服务器 STUDSVR1 上的 validate_student 存储过程，并把指示成功或失败的返回状态存储在 @retvalue1 中：

```
DECLARE @retvalue1 int
EXECUTE @retvalue = STUDSVR1.student_db.student_admin.validate_student '111111111'
```

有关详细信息，请参见 SQL Server Books Online。

如果在两个或多个网络数据库节点上对表进行更改，Oracle 就会自动启动一个分布式事务。SQL Server 分布式事务则使用包含在 SQL Server 中的 Microsoft 分布式事务处理协调器（MS DTC）的两阶段提交服务。

默认情况下，必须指示 SQL Server 参与分布式事务。可以使用以下方法之一，使 SQL Server 开始参与 MS DTC 事务：

- 使用 BEGIN DISTRIBUTED TRANSACTION 语句。这个语句开始一个新的 MS DTC 事务。
- 使用直接调用 DTC 事务接口的客户应用程序。

在此例中，请注意对本地表 GRADE 和远程表 CLASS（使用 class_name 过程）的分布式更新：

```
BEGIN DISTRIBUTED TRANSACTION
UPDATE STUDENT_ADMIN.GRADE
SET GRADE = 'B+' WHERE SSN = '111111111' AND CCODE = '1234'
DECLARE @retvalue1 int
EXECUTE @retvalue1 = CLASS_SVR1.dept_db.dept_admin.class_name '1234', 'Basketweaving'
COMMIT TRANSACTION
GO
```

如果应用程序不能完成此事务，应用程序就会使用 ROLLBACK TRANSACTION 语句取消它。如果应用程序失败或参与的资源管理器失败，MS DTC 就会取消此事务。MS DTC 不支持分布式保存点或 SAVE TRANSACTION 语句。如果一个 MS DTC 事务终止或回滚，整个事务被回滚到分布式事务的起始处，无论有多少个保存点都是如此。

Oracle 和 MS DTC 两阶段提交机制在操作上是类似的。在 SQL Server 两阶段提交的第一阶段中，事务管理器请求每个参加的资源管理器为提交做准备。如果任何资源管理器不能准备，则事务管理器向事务所涉及的每个人广播终止决定。

如果所有资源管理器都可以成功地准备，则事务管理器广播提交决定。这是提交过程的第二阶段。当资源管理器在准备时，它不知道事务将被提交还是被终止。MS DTC 有一个有序日志，这样就可以永久保存提交或终止决定。如果资源管理器或事务管理器失败，它们重新连接时，就可重新处理有疑问的事务。

SQL 语言支持

这一部分概述了 Transact-SQL 和 PL/SQL 语言语法之间的相同点和不同点，并给出转换策略。

要将 Oracle DML 语句和 PL/SQL 程序迁移到 SQL Server 时，请按下列步骤执行：

1. 验证所有 SELECT、INSERT、UPDATE 和 DELETE 语句的语法是有效的。进行任何必要的修改。
2. 把所有外部联接改为 SQL-92 标准外部联接语法。
3. 用相应 SQL Server 函数替代 Oracle 函数。
4. 检查所有的比较运算符。
5. 用“ +”字符串串联运算符代替“ ‖”字符串串联运算符。
6. 用 Transact-SQL 程序代替 PL/SQL 程序。
7. 把所有 PL/SQL 游标改为非游标 SELECT 语句或 Transact-SQL 游标。
8. 用 Transact-SQL 过程代替 PL/SQL 过程、函数和包。
9. 把 PL/SQL 触发器转换为 Transact-SQL 触发器。
10. 使用 SET SHOWPLAN 语句，优化查询性能。

SELECT 语句

Oracle 和 Microsoft SQL Server 使用的 SELECT 语句语法类似。

Oracle	Microsoft SQL Server
SELECT **[/*+ optimizer_hints*/]** [ALL | DISTINCT] *select_list* [FROM {*table_name* | *view_name*	SELECT *select_list* [INTO *new_table_*] FROM *table_source* [WHERE *search_condition*]

select_statement}] [WHERE clause] [GROUP BY *group_by_expression*] [HAVING *search_condition*] [START WITH U CONNECT BY] [{UNION | UNION ALL | INTERSECT | MINUS} SELECT U] [ORDER BY clause] [FOR UPDATE]	[GROUP BY [ALL] *group_by_expression* [,U*n*] [WITH { CUBE | ROLLUP }] [HAVING *search_condition*] [ORDER BY *order_expression* [ASC | DESC]] In addition: UNION Operator COMPUTE Clause FOR BROWSE Clause OPTION Clause

SQL Server 不支持 Oracle 特定的基于开销的优化程序提示，它必须被删除。建议使用的技术是，使用 SQL Server 基于开销的优化程序。有关详细信息，请参见本章后面的“SQL 语句优化”。

SQL Server 不支持 Oracle 的 START WITHUCONNECT BY 子句。在 SQL Server 中，可以创建完成相同任务的存储过程替代它。

SQL Server 不支持 Oracle 的 INTERSECT 和 MINUS 集合运算符。可使用 SQL Server EXISTS 和 NOT EXISTS 子句，实现相同的结果。

在下面示例中，使用 INTERSECT 运算符，用于查找学生登记的所有课程的代码和名称。注意，EXISTS 运算符是如何代替 INTERSECT 运算符的。返回的数据是相同的。

Oracle	Microsoft SQL Server
SELECT CCODE, CNAME FROM DEPT_ADMIN.CLASS **INTERSECT** **SELECT C.CCODE, C.CNAME** **FROM** **STUDENT_ADMIN.GRADE G,** **DEPT_ADMIN.CLASS C** **WHERE C.CCODE = G.CCODE**	SELECT CCODE, CNAME FROM DEPT_ADMIN.CLASS C **WHERE EXISTS** **(SELECT 'X' FROM** **STUDENT_ADMIN.GRADE G** **WHERE C.CCODE = G.CCODE)**

在此例中，使用 MINUS 运算符，查找那些没有任何学生登记的课程。

Oracle	Microsoft SQL Server
SELECT CCODE, CNAME FROM DEPT_ADMIN.CLASS **MINUS** **SELECT C.CCODE, C.CNAME** **FROM** **STUDENT_ADMIN.GRADE G,**	SELECT CCODE, CNAME FROM DEPT_ADMIN.CLASSC **WHERE NOT EXISTS** **(SELECT 'X' FROM** **STUDENT_ADMIN.GRADE G** **WHERE C.CCODE = G.CCODE)**

DEPT_ADMIN.CLASS C WHERE C.CCODE = G.CCODE	

INSERT 语句

Oracle 和 Microsoft SQL Server 使用的 INSERT 语句语法类似。

Oracle	Microsoft SQL Server

INSERT INTO
*{table_name | view_name | **select_statement**} [(column_list)]*
*{**values_list** | select_statement}*
	INSERT [INTO]
	{
	table_name [[AS] *table_alias*] WITH
	(<table_hint_limited> [U*n*])
	| *view_name* [[AS] *table_alias*]
	| *rowset_function_limited*
	}
	
	{ [(*column_list*)]
	{ VALUES ({ DEFAULT
	| NULL
	| *expression*
	}[,U*n*]
	)
	| *derived_table*
	| *execute_statement*
	}
	}
	| DEFAULT VALUES

Transact-SQL 语言支持对表和视图的插入，但不支持对 SELECT 语句的 INSERT 操作。如果 Oracle 应用程序代码执行对 SELECT 语句的插入操作，则必须对它进行修改。

Oracle	Microsoft SQL Server
INSERT INTO (**SELECT SSN, CCODE, GRADE FROM GRADE**) VALUES ('111111111', '1111',NULL)	INSERT INTO **GRADE** (**SSN, CCODE, GRADE**) VALUES ('111111111', '1111',NULL)

Transact-SQL ***values_list*** 参数提供了 SQL-92 标准关键字 DEFAULT，但 Oracle 不支持。此关键字指定了，执行插入操作时使用列的默认值。如果指定列的默认值不存在，则插入 NULL。如果该列不允许 NULL，则返回一个错误消息。如果该列数据类型定义为 **timestamp**，则插入下一个有序值。

标识符列不能使用 DEFAULT 关键字。要生成下一个序列号，拥有 IDENTITY 属性的列不能列在 *column_list* 或 *values_clause* 中。不需使用 DEFAULT 关键字，来获取列的默认值。正如在 Oracle 中，如果列没有在 *column_list* 中引用，并且它有默认值，则默认值存放在列中。这是迁移时可使用的最兼容的方法。

一个有用的 Transact_SQL 选项（EXECute *procedure_name*）是，执行一个过程并将其结果用管道输出到目标表或视图中。Oracle 不允许这样做。

UPDATE 语句

因为 Transact SQL 支持 Oracle UPDATE 命令使用的绝大多数语法，所以只需要极少的修改。

Oracle	Microsoft SQL Server

UPDATE
{*table_name* | *view_name* | ***select_statement***}
SET *[column_name(s)* =
{*constant_value* | *expression* | *select_statement* | *column_list* | ***variable_list****]*
{*where_statement*}
	UPDATE
	{
	table_name [[AS] *table_alias*] WITH
	(<table_hint_limited> [U*n*])
	view_name [[AS] *table_alias*]
	| *rowset_function_limited*
	}
	SET
	{*column_name* = {*expression* | DEFAULT | NULL}
	| *@variable = expression*
	| *@variable = column = expression* }
	[,U*n*]
	
	{{[FROM {<table_source>} [,U*n*]]
	
	[WHERE
	<search_condition>] }
	|
	[WHERE CURRENT OF
	{ { [GLOBAL] *cursor_name* } | *cursor_variable_name*}
	] }
	[OPTION (<query_hint> [,U*n*])]

Transact-SQL UPDATE 语句不支持对 SELECT 语句的更新操作。如果 Oracle 应用程序代码对 SELECT 语句进行更新，则可以把 SELECT 语句转换成一个视图，然后在 SQL Server UPDATE 语句中使用该视图名称。请参见前面“ INSERT 语句 ”中的示例。

Oracle UPDATE 命令只能使用一个 PL/SQL 块中的程序变量。要使用变量，Transact-SQL 语言并不需要使用块。

Oracle	Microsoft SQL Server

DECLARE
VAR1 NUMBER(10,2);
BEGIN
VAR1 := 2500;
UPDATE
STUDENT_ADMIN.STUDENT
SET TUITION_TOTAL = **VAR1;**
END;
	DECLARE
	@VAR1 NUMERIC(10,2)
	SELECT @VAR1 = 2500
	UPDATE
	STUDENT_ADMIN.STUDENT
	SET TUITION_TOTAL=**@VAR1**

在 SQL Server 中，DEFAULT 关键字可用于将一列设为其默认值。但不能使用 Oracle UPDATE 命令，将一列设为默认值。

Transact-SQL 和 Oracle SQL 均支持在 UPDATE 语句中使用子查询。但是，Transact-SQL FROM 子句可用来创建一个基于联接的 UPDATE。这一功能使 UPDATE 语法可读性更好，在某些情况下还能改善性能。

Oracle	Microsoft SQL Server
UPDATE STUDENT_ADMIN.STUDENT S SET TUITION_TOTAL = 1500 WHERE SSN IN (SELECT SSN FROM GRADE G WHERE G.SSN = S.SSN AND G.CCODE = '1234')	Subquery: UPDATE STUDENT_ADMIN.STUDENT S SET TUITION_TOTAL = 1500 WHERE SSN IN (SELECT SSN FROM GRADE G WHERE G.SSN = S.SSN AND G.CCODE = '1234') FROM clause: UPDATE STUDENT_ADMIN.STUDENT S SET TUITION_TOTAL = 1500 FROM GRADE G WHERE S.SSN = G.SSN AND G.CCODE = '1234'

DELETE 语句

在大多数情况下，不需要修改 DELETE 语句。如果要对 Oracle 中的 SELECT 语句执行删除操作，则必须修改 SQL Server 语法，因为 Transact-SQL 不支持这一功能。

Transact-SQL 支持在 WHERE 子句中使用子查询，以及在 FROM 子句中使用联接。后者可产生更有效的语句。请参见前面“UPDATE 语句”中的示例。

Oracle	Microsoft SQL Server

DELETE [FROM]
{*table_name* | *view_name* | ***select_statement***}
[WHERE clause]
	DELETE
	[FROM]
	{
	table_name [[AS] *table_alias*] WITH
	(<table_hint_limited> [U*n*])
	| *view_name* [[AS] *table_alias*]
	| *rowset_function_limited*
	}
	
	[FROM {<table_source>} [,U*n*]]
	[WHERE
	{ <search_condition>
	| { [CURRENT OF
	{
	{ [GLOBAL] *cursor_name* }
	cursor_variable_name
	}
	]
	}
	]
	[OPTION (<query_hint> [,U*n*])]

TRUNCATE TABLE 语句

Oracle 和 Microsoft SQL Server 使用的 TRUNCATE TABLE 语句语法类似。TRUNCATE TABLE 用于从表中删除所有的行，并且不能回滚。表结构及其所有索引继续存在。DELETE 触发器不执行。如果表被一个 FOREIGN KEY 约束引用，则它不能被截断。

Oracle	Microsoft SQL Server
TRUNCATE TABLE *table_name* **[{DROP | REUSE} STORAGE]**	TRUNCATE TABLE *table_name*

在 SQL Server 中，此语句只能由表的所有者执行。在 Oracle 中，如果是表的所有者或拥有 DELETE TABLE 系统权限，就可以执行此命令。

Oracle TRUNCATE TABLE 命令可以有选择地释放表中行所占用的存储空间。SQL Server TRUNCATE TABLE 语句总是收回表数据及其相关索引所占用的空间。

标识符列和时间戳列中数据的处理

Oracle 序列是与任何给定的表或列均不直接相关的数据库对象。列和序列之间的关系是在应用程序中实现的，即通过编程的方法将序列值赋给列。因此，Oracle 使用序列时，并不实施任

何规则。但是，在 Microsoft SQL Server 标识符列中，值不能被更新，并且不能使用 DEFAULT 关键字。

默认情况下，数据不能直接插入到标识符列。标识符列自动给表中插入的每个新行生成一个唯一的序列号。可以使用下列 SET 语句改写这种默认设置：

```
SET IDENTITY_INSERT table_name ON
```

将 IDENTUTY_INSERT 设为 ON，用户就可以向新行的标识符列插入任何值。要防止出现有重复号码的条目，必须为该列创建唯一索引。这条语句的目的是，允许用户给无意中删除的行重新创建一个值。@@IDENTITY 函数可用来获取上一个标识值。

TRUNCATE TABLE 语句将标识符列重置为其起始 SEED 值。如果不想重置列的标识值，则不使用 TRUNCATE TABLE 语句，而使用不带 WHERE 子句的 DELETE 语句。必须评估它对 Oracle 迁移造成的影响，因为 ORACLE SEQUENCE 在 TRUNCATE TABLE 命令之后不被重置。

处理 timestamp 列时，只能执行插入和删除。如果要更新一个 timestamp 列，会收到以下的错误信息：

```
Msg 272, Level 16, State 1 Can't update a TIMESTAMP column.
```

锁定请求的行

Oracle 使用 FOR UPDATE 子句来锁定 SELECT 命令中指定的行。不需要在 Microsoft SQL Server 中使用对等的子句，因为这是默认行为。

行合计和 COMPUTE 子句

SQL Server COMPUTE 子句用于生成行合计函数（SUM、AVG、MIN、MAX 和 COUNT），它们在查询结果中作为附加行出现。它允许查看一组结果的详细和汇总信息行。可以计算子组的汇总值，以及计算同一组的多个合计函数。

Oracle SELECT 命令语法不支持 COMPUTE 子句。但是，SQL Server COMPUTE 子句与 Oracle SQL*Plus 查询工具中的 COMPUTE 命令作用相似。

联接子句

Microsoft SQL Server 7.0 允许在一个联接子句中最多可联接 256 个表，包括临时表和永久表。在 Oracle 中，则没有联接限制。

在 Oracle 中使用外部联接时，外部联接运算符（+）通常放在该联接的子列（外键）旁边。(+) 标识了具有较少唯一值的列。情况一般是这样，除非外键允许空值，在这种情况下，(+) 被放在父（PRIMARY KEY 或 UNIQUE 约束）列上。(+) 不能放在等号 (=) 两边。

在 SQL Server 中，可以使用 *= 和 =* 外部联接运算符。* 用于标识有较多唯一值的列。如果子（外键）列不允许空值，则 * 放在等号的父（PRIMARY KEY 或 UNIQUE 约束）列一边。* 的位置和 Oracle 完全相反。* 不能放在等号 (=) 两边。

= 和 = 被认为是旧式联接运算符。SQL Server 也支持下面列出的 SQL-92 标准联接运算符。建议使用这种语法。SQL-92 标准语法功能更强大，并且比 * 运算符的限制要少。

联接操作	说明
CROSS JOIN	这是两个表的交叉乘积。它与旧式联接中未指定 WHERE 子句而返回的行相同。在 Oracle 中，这种类型联接称为笛卡尔联接。
INNER	这种联接指定，所有内部行均要返回。丢弃任何不匹配的行。这和标准的 Oracle 表联接相同。
LEFT[OUTER]	这种类型的联接指定，所有左表的外部行均要返回，即使没找到匹配的列，也就如此。这和 Oracle 外部联接 (+) 的操作类似。
RIGHT[OUTER]	这种类型的联接指定，所有右表的外部行均要返回，即使没找到匹配的列，也是如此。这和 Oracle 外部联接 (+) 的操作类似。
FULL [OUTER]	如果两个表中的一行不符合选择标准，则指定将这一行加到结果集中，并且将其对应于另一表的输出列设为 NULL。这和将 Oracle 外部联接运算符放在“=”号两边 (col1(+)=col2(+)) 的效果是一样的，但后者在 Oracle 中是不允许的。

下面的代码示例返回所有学生登记的课程列表。外部联接定义在学生 (student) 和成绩 (grade) 表之间，成绩表允许所有的学生出现在上面，甚至那些没有登记任何课程的学生也是如此。外部联接也加到课程表上，以返回课程名称。如果外部联接不加到课程表上，就不会返回那些没有登记任何课程的学生，因为他们的课程代码 (CCODE) 为空。

Oracle	Microsoft SQL Server
SELECT S.SSN AS SSN, FNAME, LNAME **FROM** **STUDENT_ADMIN.STUDENT S,** **DEPT_ADMIN.CLASS C,** **STUDENT_ADMIN.GRADE G** **WHERE S.SSN = G.SSN(+)** **AND G.CCODE = C.CCODE(+)**	SELECT S.SSN AS SSN, FNAME, LNAME **FROM** **STUDENT_ADMIN.GRADE G** **RIGHT OUTER JOIN** **STUDENT_ADMIN.STUDENT S** **ON G.SSN = S.SSN** **LEFT OUTER JOIN** **DEPT_ADMIN.CLASS C** **ON G.CCODE = C.CCODE**

将 SELECT 语句做为表名使用

Microsoft SQL Server 和 Oracle 均支持在执行查询时，把 SELECT 语句作为表的来源使用。SQL Server 需要一个别名；对 Oracle，别名的使用是可选的。

Oracle	Microsoft SQL Server
SELECT SSN, LNAME, FNAME, TUITION_PAID, SUM_PAID FROM STUDENT_ADMIN.STUDENT, **(SELECT SUM(TUITION_PAID) SUM_PAID FROM STUDENT_ADMIN.STUDENT)**	SELECT SSN, LNAME, FNAME, TUITION_PAID, SUM_PAID FROM STUDENT_ADMIN.STUDENT, **(SELECT SUM(TUITION_PAID) SUM_PAID FROM STUDENT_ADMIN.STUDENT) SUM_STUDENT**

读取和修改 BLOB

Microsoft SQL Server 使用 text 和 image 列，来实现二进制大型对象 (BLOB)。Oracle 使用 LONG 和 LONG RAW 列实现 BLOB。在 Oracle 中,SELECT 命令可以查询 LONG 和 LONG RAW 列中的值。

在 SQL Server 中，可以使用标准的 Transact-SQL 语句或专门的 READTEXT 语句读取 text 和 image 列中的数据。READTEXT 语句允许读取 text 或 image 列的部分片段。Oracle 没有提供处理 LONG 和 LONG RAW 的对等语句。

READTEXT 语句使用 *text_pointer*，它可以用 TEXTPTR 函数获得。TEXTPTR 函数返回一个指针，它指向指定行中的 text 或 image 列，或如果返回不止一行，则它指向查询返回的最后一行的 text 或 image 列。因为 TEXTPTR 函数返回一个 16 字节的二进制字符串，最好声明一个局部变量来存放文本指针，然后由 READTEXT 使用该变量。

READTEXT 语句指定要返回的字节数。@@TEXTSIZE 函数中的值是要返回的字符或字节数的限度，如果它小于 READTEXT 指定的大小，就会替代 READTEXT 语句指定值。

可使用带 TEXTSIZE 参数的 SET 语句，指定 SELECT 语句返回的文本数据的大小（字节数）。如果指定 TEXTSIZE 为 0，其大小被重置为默认值 (4 KB)。设置 TEXTSIZE 参数，会影响 @@TEXTSIZE 函数。当 SQL_MAX_LENGTH 语句选项更改时，SQL Server ODBC 驱动程序就会自动设置 TEXTSIZE 参数。

在 Oracle 中,UPDATE 和 INSERT 命令用于更改 LONG 和 LONG RAW 列中的值。在 SQL Server 中，可以使用标准的 UPDATE 和 INSERT 语句，也可以使用 UPDATETEXT 和 WRITETEXT 语句。UPDATETEXT 和 WRITETEXT 均允许无日志记录的选项,并且 UPDATETEXT 允许对 text 或 image 列进行部分更新。

UPDATETEXT 语句可用于替换现有数据、删除现有数据或插入新数据。新插入的数据可以是常量、表名、列名或文本指针。

WRITETEXT 语句可完全覆盖受其影响的列中的任何现有数据。使用 WRITETEXT 可替换文本数据；使用 UPDATETEXT 可修改文本数据。UPDATETEXT 语句更加灵活，因为它只更改一部分文本或图像值，而不是更改全部。

有关详细信息，请参见 SQL Server Books Online。

本节中的表给出了 Oracle 和 SQL Server 标量值函数和合计函数之间的关系。尽管名称看起来是相同的，但要注意，函数参数的数量和类型是不同的，这一点非常重要。此外，在这个列表中，没有给出仅由 Microsoft SQL Server 提供的函数，因为本章仅限于讲述，如何方便地实现从现有 Oracle 应用程序的迁移。Oracle 不支持函数的例子有：角度 (DEGREES)、圆周率 (PI) 和随机数 (RAND)。

数字/数学函数

下面是 Oracle 支持的数字/数学函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
绝对值	ABS	ABS
反余弦	ACOS	ACOS
反正弦	ASIN	ASIN
n 的反正切	ATAN	ATAN
m/n 的反正切	ATAN2	ATN2
>=值的最小整数	CEIL	CEILING
余弦	COS	COS
双曲余弦	COSH	COT
指数值	EXP	EXP
<=值的最大整数	FLOOR	FLOOR
自然对数	LN	LOG
以任何为底的对数	LOG(N)	暂缺
以 10 为底的对数	LOG(10)	LOG10
模数（余数）	MOD	USE MODULO (%) OPERATOR
幂	POWER	POWER
随机数	暂缺	RAND
舍入	ROUND	ROUND
数的符号	SIGN	SIGN
正弦	SIN	SIN
双曲正弦	SINH	暂缺
平方根	SQRT	SQRT
正切	TAN	TAN
双曲正切	TANH	暂缺

截尾	TRUNC	暂缺
列表中的最大数	GREATEST	暂缺
列表中的最小数	LEAST	暂缺
如果为 NULL,转换成数字	NVL	ISNULL

字符函数

下面是 Oracle 支持的字符函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
把字符转换成 ASCII	ASCII	ASCII
字符串串联	CONCAT	（表达式 + 表达式）
把 ASCII 转换成字符	CHR	CHAR
返回字符串中的起始字符（从左）	INSTR	CHARINDEX
将字符转换成小写	LOWER	LOWER
将字符转换成大写	UPPER	UPPER
在字符串的左边填充字符	LPAD	暂缺
删除前导空格	LTRIM	LTRIM
删除尾空格	RTRIM	RTRIM
字符串中模式的起始点	INSTR	PATINDEX
多次重复字符串	RPAD	REPLICATE
字符串的语音表示	SOUNDEX	SOUNDEX
重复空格的字符串	RPAD	SPACE
从数字数据转换而来的字符数据	TO_CHAR	STR
子串	SUBSTR	SUBSTRING
字符替换	REPLACE	STUFF
字符串中每个词的第一个字母大写	INITCAP	暂缺
字符串转换	TRANSLATE	暂缺
字符串长度	LENGTH	DATELENGTH 或 LEN
列表中的最大字符串	GREATEST	暂缺
列表中的最小字符	LEAST	暂缺

串		
如果为 NULL,则转换字符串	NVL	ISNULL

日期函数

下面是 Oracle 支持的日期函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
日期加	（日期列 +/- 值）或 ADD_MONTHS	DATEADD
日期间的间隔	（日期列 +/- 值）或 MONTHS_BETWEEN	DATEDIFF
当前日期和时间	SYSDATE	GETDATE()
月的最后一天	LAST_DAY	暂缺
时区转换	NEW_TIME	暂缺
该日期后的第一个工作日	NEXT_DAY	暂缺
日期的字符串表示	TO_CHAR	DATENAME
日期的整数表示	TO_NUMBER (TO_CHAR))	DATEPART
日期舍入	ROUND	CONVERT
日期截尾	TRUNC	CONVERT
字符串转换为日期	TO_DATE	CONVERT
如果为 NULL,则转换日期	NVL	ISNULL

转换函数

下面是 Oracle 支持的转换函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
数字到字符	TO_CHAR	CONVERT
字符到数字	TO_NUMBER	CONVERT
日期到字符	TO_CHAR	CONVERT
字符到日期	TO_DATE	CONVERT
十六进制到二进制	HEX_TO_RAW	CONVERT
二进制到十六进制	RAW_TO_HEX	CONVERT

其它行级函数

下面是 Oracle 支持的其它行级函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
返回第一个非空表达式	DECODE	COALESCE
当前序列值	CURRVAL	暂缺
下一个序列值	NEXTVAL	暂缺
如果表达式 1 = 表达式 2，则返回空	DECODE	NULLIF
用户的登录 ID 号	UID	SUSER_ID
用户的登录名	USER	SUSER_NAME
用户的数据库 ID 号	UID	USER_ID
用户的数据库名	USER	USER_NAME
当前用户	CURRENT_USER	CURRENT_USER
用户环境（审核记录）	USERENV	暂缺
CONNECT BY 子句的级别	LEVEL	暂缺

合计函数

下面是 Oracle 支持的合计函数及其 Microsoft SQL Server 对等函数。

函数	Oracle	Microsoft SQL Server
平均值	AVG	AVG
计数	COUNT	COUNT
最大值	MAX	MAX
最小值	MIN	MIN
标准偏差	STDDEV	STDEV 或 STDEVP
汇总	SUM	SUM
方差	VARIANCE	VAR 或 VARP

条件测试

Oracle DECODE 语句和 Microsoft SQL Server CASE 表达式都执行条件测试。当 *test_value* 中的值符合下列任何表达式时，就会返回相关的值。如果不符合，则返回 *default_value*。如果没有指定 *default_value*，且不符合任何表达式，则 DECODE 和 CASE 返回 NULL。下表给出了语法以及一个转换的 DECODE 命令的示例。

Oracle	Microsoft SQL Server

DECODE (*test_value, expression1, value1* [[,*expression2, value2*] [U]] [,*default_value*])	**CASE** *input_expression* **WHEN** *when_expression* **THEN** *result_expression* [[**WHEN** *when_expression* **THEN** *result_expression*] [...]] [**ELSE** *else_result_expression*] **END**
CREATE VIEW STUDENT_ADMIN.STUDENT_GPA (SSN, GPA) AS SELECT SSN, ROUND(AVG**(DECODE(grade** **,'A', 4** **,'A+', 4.3** **,'A-', 3.7** **,'B', 3** **,'B+', 3.3** **,'B-', 2.7** **,'C', 2** **,'C+', 2.3** **,'C-', 1.7** **,'D', 1** **,'D+', 1.3** **,'D-', 0.7** **,0)**),2) FROM STUDENT_ADMIN.GRADE GROUP BY SSN	CREATE VIEW STUDENT_ADMIN.STUDENT_GPA (SSN, GPA) AS SELECT SSN, ROUND(AVG(CASE grade WHEN 'A' THEN 4 WHEN 'A+' THEN 4.3 WHEN 'A-' THEN 3.7 WHEN 'B' THEN 3 WHEN 'B+' THEN 3.3 WHEN 'B-' THEN 2.7 WHEN 'C' THEN 2 WHEN 'C+' THEN 2.3 WHEN 'C-' THEN 1.7 WHEN 'D' THEN 1 WHEN 'D+' THEN 1.3 WHEN 'D-' THEN 0.7 ELSE 0 END),2) FROM STUDENT_ADMIN.GRADE GROUP BY SSN

CASE 表达式可以支持使用 SELECT 语句进行布尔测试，这是 DECODE 命令所不允许的。有关 CASE 表达式的详细信息，请参见 SQL Server Books Online。

将值转换为不同的数据类型

Microsoft SQL Server CONVERT 和 CAST 函数均是多用途的转换函数。它们提供了相似的功能，把一种数据类型的表达式转换为另一种数据类型，并支持多种特殊的数据格式：

- CAST(expression AS data_type)
- CONVERT (data type[(length)], expression [, style])

CAST 是一个 SQL-92 标准函数。这些函数执行与 Oracle TO_CHAR、TO_NUMBER、TO_DATE、HEXTORAW 和 RAWTOHEX 函数相同的操作。

数据类型是指该表达式要转换成为的任何系统数据类型。不能使用用户定义的数据类型。*length* 参数是可选的，它与 char、varchar、binary 和 varbinary 数据类型一起使用。可允许的最大长度是 8000。

转换	Oracle	Microsoft SQL Server
字符到数字	TO_NUMBER('10')	CONVERT(**numeric**, '10')
数字到字符	TO_CHAR(10)	CONVERT(**char**, 10)
字符到日期	TO_DATE('04-JUL-97') TO_DATE('04-JUL-1997', 'dd-mon-yyyy') TO_DATE('July 4, 1997', 'Month dd, yyyy')	CONVERT(**datetime**, '04-JUL-97') CONVERT(**datetime**, '04-JUL-1997') CONVERT(**datetime**, 'July 4, 1997')
日期到字符	TO_CHAR(sysdate) TO_CHAR(sysdate, 'dd mon yyyy') TO_CHAR(sysdate, 'mm/dd/yyyy')	CONVERT(**char**, GETDATE()) CONVERT(**char**, GETDATE(), 106) CONVERT(**char**, GETDATE(), 101)
十六进制到二进制	HEXTORAW('1F')	CONVERT(**binary**, '1F')
二进制到十六进制	RAWTOHEX (binary_column)	CONVERT(**char**, *binary_column*)

注意，字符串是如何转换成日期的。在 Oracle 中，默认的日期格式模型为“DD-MON-YY”。如果使用任何其它格式，必须提供相应的日期格式模型。CONVERT 函数自动转换标准日期格式，而无须格式模型。

当把日期转换成字符串时，CONVERT 函数默认输出为“dd mon yyyy hh:mm:ss:mmm(24h)”。一种数字类型的编码用于设定到其它日期格式模型输出的格式。有关 CONVERT 函数的详细信息，请参见 SQL Server Books Online。

下表给出了 Microsoft SQL Server 日期的默认输出。

不带世纪	带有世纪	标准	输出
-	0 或 100 (*)	默认	mon dd yyyy hh:miAM（或 PM）
1	101	美国	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	英国/法国	dd/mm/yy
4	104	德国	dd.mm.yy

5	105	**意大利**	dd-mm-yy
6	106	-	dd mon yy
7	107	-	mon dd, yy
8	108	-	hh:mm:ss
-	9 **或** 109 (*)	**默认毫秒**	mon dd yyyy hh:mi:ss:mmm **（** AM **或** PM**）**
10	110	**美国**	mm-dd-yy
11	111	**日本**	yy/mm/dd
12	112	ISO	yymmdd
-	13 **或** 113 (*)	**欧洲默认**	dd mon yyyy hh:mm:ss:mmm(24h)
14	114	-	hh:mi:ss:mmm(24h)

用户定义的函数

Oracle PL/SQL 函数可用于 Oracle SQL 语句中。在 Microsoft SQL Server 中，这一功能通常以其它方式实现。

在下面的示例中，Oracle 用户定义的函数 GET_SUM_MAJOR 用于获取按专业（major）交纳的学费总和。在 SQL Server 中，可通过把查询作为表使用，以替代这一函数。

Oracle	**Microsoft SQL Server**
SELECT SSN, FNAME, LNAME,) TUITION_PAID, TUITION_PAID/**GET_SUM_ MAJOR(MAJOR)** AS PERCENT_MAJOR FROM STUDENT_ADMIN.STUDENT	SELECT SSN, FNAME, LNAME, TUITION_PAID, TUITION_PAID/**SUM_MAJOR** AS PERCENT_MAJOR FROM STUDENT_ADMIN.STUDENT, **(SELECT MAJOR, SUM(TUITION_PAID) SUM_MAJOR FROM STUDENT_ADMIN.STUDENT GROUP BY MAJOR) SUM_STUDENT WHERE STUDENT.MAJOR = SUM_STUDENT.MAJOR**
CREATE OR REPLACE FUNCTION GET_SUM_MAJOR (INMAJOR VARCHAR2) RETURN NUMBER	**不需要 CREATE FUNCTION 语法；使用 CREATE PROCEDURE 语法。**

AS SUM_PAID NUMBER; BEGIN SELECT SUM(TUITION_PAID) INTO SUM_PAID FROM STUDENT_ADMIN.STUDENT WHERE MAJOR = INMAJOR; RETURN(SUM_PAID); END GET_SUM_MAJOR;	

Oracle 和 Microsoft SQL Server 比较运算符几乎是相同的。

运算符	**Oracle**	**Microsoft SQL Server**
等于	(=)	(=)
大于	(>)	(>)
小于	(<)	(<)
大于或等于	(>=)	(>=)
小于或等于	(<=)	(<=)
不等于	(!=, <>, ^=)	(!=, <>, ^=)
不大于，不小于	**暂缺**	!> , !<
在集合的任何成员中	IN	IN
不在集合的任何成员中	NOT IN	NOT IN
集合中的任一值	ANY, SOME	ANY, SOME
引用集合中的所有值	!= ALL, <> ALL, < ALL, > ALL, <= ALL, >= ALL, != SOME, <> SOME, < SOME, > SOME, <= SOME, >= SOME	!= ALL, <> ALL, < ALL, > ALL, <= ALL, >= ALL, != SOME, <> SOME, < SOME, > SOME, <= SOME, >= SOME
与模式相似	LIKE	LIKE
与模式不相似	NOT LIKE	NOT LIKE
x **和** y **之间的值**	BETWEEN x AND y	BETWEEN x AND y
不在两者之间的值_	NOT BETWEEN	NOT BETWEEN
值存在	EXISTS	EXISTS
值不存在	NOT EXISTS	NOT EXISTS
值{**为**|**不为**} NULL	IS NULL, IS NOT NULL	**相同。也可以使用** = NULL、 !=NULL，**用于向后兼容性（不推荐使用）。**

模式匹配

SQL Server LIKE 关键字提供了一些 Oracle 不支持的、有用的通配符搜索选项。除了支持两个 RDBMS 通用的 % 和 _ 通配符外，SQL Server 还支持 [] 和 [^] 字符。

[] 字符用于在给定范围内搜索某一单个字符。例如，如果在单字符位置搜索从 a 到 f 的字符，可以用 LIKE '[a-f]' 或 LIKE '[abcdef]' 指定。此表给出了这些附加通配符的用法。

Oracle	Microsoft SQL Server
SELECT * FROM STUDENT_ADMIN.STUDENT WHERE LNAME LIKE '**A%**' **OR LNAME LIKE 'B%'** **OR LNAME LIKE 'C%'**	SELECT * FROM STUDENT_ADMIN.STUDENT WHERE LNAME LIKE '**[ABC]%**'

[^] 通配符集合用于指定不在给定范围内的字符。例如,如果接受除 a 到 f 以外的任何字符，则使用 LIKE '[^a - f]' 或 LIKE '[^abcdef]'。

有关 LIKE 关键字的详细信息，请参见 SQL Server Books Online。

NULL 用法对比

尽管 Microsoft SQL Server 传统上支持 SQL-92 标准以及其它一些非标准的 NULL 行为，但是它也支持 Oracle 中 NULL 的用法。

要执行分布式查询，SET ANSI_NULLS 应该设为 ON。

SQL Server ODBC 驱动程序和 SQL Server 的 OLE DB 提供程序连接时，就会自动把 SET ANSI_NULLS 设为 ON。此设置可以在 ODBC 数据源、ODBC 连接属性设定，或者连接 SQL Server 前，在应用程序中设置的 OLE DB 连接属性中设定。对来自 DB-Library 应用程序的连接，SET ANSI_NULLS 默认为 OFF。

当 SET ANSI_DEFAILTS 为 ON 时，就会启用 SET ANSI_NULLS。

有关使用 NULL 的详细信息，请参见 SQL Server Books Online。

字符串串联

Oracle 将两个管道符号 (||) 作为字符串串联运算符，而 SQL Server 则使用加号 (+)。这种差别只需要对应用程序代码进行小小的修改即可。

Oracle	Microsoft SQL Server

SELECT FNAME||' '||LNAME AS NAME FROM STUDENT_ADMIN.STUDENT	SELECT FNAME +' '+ LNAME AS NAME FROM STUDENT_ADMIN.STUDENT

控制流语言控制 SQL 语句、语句块和存储过程的执行数据流。PL/SQL 和 Transact SQL 提供了许多相同的结构，但在语法上有一些差异。

关键字

以下是每种 RDBMS 支持的关键字。

语句	Oracle PL/SQL	Microsoft SQL Server Transact-SQL
声明变量	DECLARE	DECLARE
语句块	BEGIN...END;	BEGIN...END

条件处理	IFUTHEN,
	ELSIFUTHEN,
	ELSE
	ENDIF;
		IFU[BEGINUEND]
		ELSE <condition>
		[BEGINUEND]
		ELSE IF <condition>
		CASE expression

无条件退出	RETURN	RETURN
无条件退出到当前程序块结束后紧接着的那条语句	EXIT	BREAK
重新开始一个 WHILE 循环	暂缺	CONTINUE
等待指定的间隔	暂缺 (dbms_lock.sleep)	WAITFOR
循环控制	WHILE LOOPUEND LOOP; LABELUGOTO LABEL; FORUEND LOOP; LOOPUEND LOOP;	WHILE <condition> BEGINU END LABELUGOTO LABEL
程序注释	/* U */, --	/* U */, --
打印输出	RDBMS_OUTPUT.PUT_LINE	PRINT
提出程序错误	RAISE_APPLICATION_ERROR	RAISERROR
执行程序	EXECUTE	EXECUTE
语句终止符	分号 (;)	暂缺

声明变量

Transact-SQL 和 PL/SQL 变量是使用 DECLARE 关键字来创建的。Transact-SQL 变量用 @ 来标识，并且像 PL/SQL 变量一样，第一次创建时该变量被初始化为空值。

Oracle	Microsoft SQL Server
DECLARE VSSN CHAR(9); VFNAME VARCHAR2(12); VLNAME VARCHAR2(20); VBIRTH_DATE DATE; VLOAN_AMOUNT NUMBER(12,2);	DECLARE @VSSN CHAR(9), @VFNAME VARCHAR2(12), @VLNAME VARCHAR2(20), @VBIRTH_DATE DATETIME, @VLOAN_AMOUNT NUMERIC(12,2)

Transact-SQL 不支持 %TYPE 和 %ROWTYPE 变量数据类型定义。在 DECLARE 命令中，不能对 Transact-SQL 变量进行初始化。Oracle NOT NULL 和 CONSTANT 关键字不能用在 Microsoft SQL Server 数据类型定义中。

与 Oracle LONG 和 LONG RAW 数据类型一样，text 和 image 数据类型不能用于变量声明。此外，不支持 PL/SQL 类型的记录和表定义。

变量赋值

Oracle 和 Microsoft SQL Server 提供以下方法，给局部变量赋值。

Oracle	Microsoft SQL Server
赋值运算符 (:=)	SET *@local_variable* = value
用于从一行中选择列值的 SELECT...INTO **语法。**	SELECT *@local_variable* = *expression* [FROMU] **用于为字面值、涉及其它局部变量的表达式或一行中的列值赋值。**
FETCHUINTO **语法**	FETCHUINTO **语法**

以下是一些语法示例。

Oracle	Microsoft SQL Server
DECLARE VSSN CHAR(9); VFNAME VARCHAR2(12); VLNAME VARCHAR2(20); BEGIN **VSSN := '123448887';** SELECT **FNAME, LNAME INTO**	DECLARE @VSSN CHAR(9), @VFNAME VARCHAR(12), @VLNAME VARCHAR(20) SET **@VSSN = '12355887'** SELECT **@VFNAME=FNAME,** **@VLNAME=LNAME** FROM

VFNAME, **VLNAME** FROM STUDENTS WHERE SSN=VSSN; END;	STUDENTS WHERE SSN = @VSSN

语句块

Oracle PL/SQL 和 Microsoft SQL Server Transact-SQL 支持使用 BEGINUEND 术语，来指定程序块。Transact-SQL 不要求在 DECLARE 语句后面使用语句块。在 Microsoft SQL Server 中，如果 IF 语句和 WHILE 循环执行不止一个语句，需要使用 BEGINUEND 语句块。

Oracle	Microsoft SQL Server
DECLARE DECLARE VARIABLES ... **BEGIN -- THIS IS REQUIRED SYNTAX** PROGRAM_STATEMENTS ... **IF ...THEN** STATEMENT1; STATEMENT2; STATEMENTN; **END IF;** WHILE ...**LOOP** STATEMENT1; STATEMENT2; STATEMENTN; **END LOOP;** **END; -- THIS IS REQUIRED SYNTAX**	DECLARE DECLARE VARIABLES ... **BEGIN -- THIS IS OPTIONAL SYNTAX** PROGRAM_STATEMENTS ... IF ... **BEGIN** STATEMENT1 STATEMENT2 STATEMENTN **END** WHILE ... **BEGIN** STATEMENT1 STATEMENT2 STATEMENTN **END** **END -- THIS IS REQUIRED SYNTAX**

条件处理

Microsoft SQL Server Transact-SQL 条件语句包含 IF 和 ELSE 语句,而不是 Oracle PL/SQL 中的 ELSIF 语句。可以嵌套多个 IF 语句，来达到同样的效果。对于大量的条件测试，CASE 表达式更容易阅读。

Oracle	Microsoft SQL Server
DECLARE VDEGREE_PROGRAM CHAR(1); VDEGREE_PROGRAM_NAME VARCHAR2(20);	DECLARE @VDEGREE_PROGRAM CHAR(1), @VDEGREE_PROGRAM_NAME VARCHAR(20)

BEGIN
VDEGREE_PROGRAM := 'U';
IF VDEGREE_PROGRAM = 'U'
THEN
VDEGREE_PROGRAM_NAME :=
'Undergraduate';
ELSIF VDEGREE_PROGRAM =
'M' THEN
VDEGREE_PROGRAM_
NAME := 'Masters';
ELSIF VDEGREE_PROGRAM =
'P' THEN VDEGREE_PROGRAM_
NAME := 'PhD';
ELSE VDEGREE_PROGRAM_
NAME := 'Unknown';
END IF;
END;
	SELECT @VDEGREE_PROGRAM =
	'U'
	SELECT @VDEGREE_PROGRAM_
	NAME = CASE
	@VDEGREE_PROGRAM
	WHEN 'U' THEN 'Undergraduate'
	WHEN 'M' THEN 'Masters'
	WHEN 'P' THEN 'PhD'.
	ELSE 'Unknown'
	END

重复的语句执行（循环）

Oracle PL/SQL 提供了无条件的 LOOP 和 FOR LOOP。而 Transact-SQL 则提供了 WHILE 循环和 GOTO 语句，来达到循环的目的。

```
WHILE Boolean_expression
{sql_statement | statement_block}
[BREAK] [CONTINUE]
```

对于一个或多个语句的重复执行，WHILE 循环测试一个布尔表达式。只要给定的表达式求值为 TRUE，语句就会重复执行。如果要执行多个语句，它们必须放在一个 BEGINUEND 块中。

Oracle	Microsoft SQL Server

DECLARE
COUNTER NUMBER;
BEGIN
COUNTER := 0
WHILE (COUNTER <5) LOOP
COUNTER := COUNTER + 1;
END LOOP;
END;
	DECLARE
	@COUNTER NUMERIC
	SELECT@COUNTER = 1
	WHILE (@COUNTER <5)
	BEGIN
	SELECT @COUNTER =
	@COUNTER +1
	END

可以使用 BREAK 和 CONTINUE 关键字，从循环的内部控制语句的执行。BREAK 关键字导致从 WHILE 循环中无条件退出，CONTINUE 关键字使 WHILE 循环跳过后面的语句，并重新开始循环。BREAK 关键字和 Oracle PL/SQL EXIT 关键字等同。Oracle 没有 CONTINUE 的对等关键字。

GOTO 语句

Oracle 和 Microsoft SQL Server 均有 GOTO 语句，但是语法不同。遇到 GOTO 语句，Transact-SQL 批处理执行就会跳到标号处。GOTO 语句和标号之间的语句不执行。

Oracle	Microsoft SQL Server
GOTO label; <<label name here>>	GOTO *label*

PRINT 语句

Transact-SQL PRINT 语句和 PL/SQL RDBMS_OUTPUT.*put_line* 过程所执行的操作相同。它用于打印用户指定的消息。

PRINT 语句的消息限度为 8,000 个字符。使用 char 或 varchar 数据类型定义的变量可以嵌入打印语句中。如果使用了任何其它数据类型，必须使用 CONVERT 或 CAST 函数。可以打印局部变量、全局变量和文本。可用单引号和双引号将文本括上。

从存储过程返回

Microsoft SQL Server 和 Oracle 均有 RETURN 语句。使用 RETURN 语句，程序可从查询或过程无条件退出。RETURN 是一条可立即执行的完整语句，并可在任何时候用于从过程、批处理或语句块中退出。RETURN 后面的语句均不执行。

Oracle	Microsoft SQL Server
RETURN *expression*:	RETURN [*integer_expression*]

提出程序错误

Transact-SQL RAISERROR 语句返回一个用户定义的错误信息，并设定一个系统标志，来记录已发生了一个错误。它和 PL/SQL *raise_application_error* 异常错误处理程序的功能相似。

RAISERROR 语句允许客户从 sysmessages 表检索一个条目，或使用用户定义的严重性和状态信息动态地创建一条消息。定义后，此消息作为服务器错误信息返回给客户。

```
RAISERROR ({msg_id | msg_str}, severity, state
[, argument1 [, argument2]])
[WITH options]
```

转换 PL/SQL 程序时，可能不需要使用 RAISERROR 语句。在下面的代码示例中，PL/SQL 程序使用 *raise_application_error* 异常错误处理程序，而 Transact-SQL 程序什么也不使用。加

入 *raise_application_error* 异常错误处理程序，可避免 PL/SQL 程序返回二义性的 *unhandled exception* 错误信息。相反，当发生意外问题时，它总是返回 Oracle 错误信息 (SQLERRM)。

当 Transact-SQL 程序失败时，它总是给客户程序返回详细的错误信息。因此，除非需要进行专门错误处理，否则，并不总是需要 RAISERROR 语句。

Oracle	Microsoft SQL Server
CREATE OR REPLACE FUNCTION DEPT_ADMIN.DELETE_DEPT (VDEPT IN VARCHAR2) RETURN NUMBER AS BEGIN DELETE FROM DEPT_ADMIN.DEPT WHERE DEPT = VDEPT; RETURN(SQL%ROWCOUNT); EXCEPTION WHEN OTHER THEN RAISE_APPLICATION_ERROR (-20001,SQLERRM); END DELETE_DEPT; /	CREATE PROCEDURE DEPT_ADMIN.DELETE_DEPT @VDEPT VARCHAR(4) AS DELETE FROM DEPT_DB.DBO.DEPT WHERE DEPT = @VDEPT RETURN @@ROWCOUNT GO

游标的实现

不管从数据库中请求行数的多少，Oracle 始终要求游标和 SELECT 语句一起使用。在 Microsoft SQL Server 中，未包含在游标内的 SELECT 语句把行作为默认结果集，返回给客户。这是一种将数据返回给客户程序的有效方法。

SQL Server 给游标函数提供了两个接口。当在 Transact-SQL 批处理或存储过程时使用游标时，SQL 语句可用来声明、打开游标和从游标以及定位更新和删除中提取。当使用来自 DB-Library、ODBC 或 OLE DB 的游标时，SQL Server 客户机库透明地调用内置的服务器函数，以更有效地处理游标。

当从 Oracle 导入 PL/SQL 过程时,首先确定游标是否需要执行和 Transact-SQL 中相同的功能。如果游标只给客户程序返回一组行，则使用 Transact-SQL 中无游标的 SELECT 语句，返回一个默认的结果集。如果使用游标向局部过程变量每次加载一行数据,则必须使用 Transact-SQL 中的游标。

下表给出了游标的使用语法。

操作	Oracle	Microsoft SQL Server
声明游标	**CURSOR *cursor_name* [(*cursor_parameter*(*s*))] IS *select_statement;***	**DECLARE *cursor_name* CURSOR [LOCAL | GLOBAL] [FORWARD_ONLY | SCROLL] [STATIC | KEYSET | DYNAMIC | FAST_FORWARD] [READ_ONLY | SCROLL_LOCKS | OPTIMISTIC] [TYPE_WARNING] FOR *select_statement* [FOR UPDATE [OF *column_name* [,Un]]]**
打开游标	OPEN *cursor_name* [(***cursor_parameter***(***s***))];	OPEN *cursor_name*
从游标中提取	**FETCH *cursor_name* INTO *variable(s)***	**FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE {*n* | @nvar} | RELATIVE {*n* | @nvar}] FROM] *cursor_name* [INTO @variable(s)]**
更新提取的行	UPDATE *table_name* SET *statement(s)U* WHERE CURRENT OF *cursor_name*;	UPDATE *table_name* SET *statement(s)U* WHERE CURRENT OF *cursor_name*
删除提取的行	DELETE FROM *table_name* WHERE CURRENT OF *cursor_name*;	DELETE FROM *table_name* WHERE CURRENT OF *cursor_name*
关闭游标	CLOSE *cursor_name*;	CLOSE *cursor_name*
删除游标数据结构	**暂缺**	**DEALLOCATE *cursor_name***

尽管 Transact-SQL DECLARE CURSOR 语句不支持使用游标参数，但它的确支持局部变量。当游标打开时，这些局部变量的值可在游标中使用。Microsoft SQL Server 在其 DECLARE CURSOR 语句中提供了许多额外的功能。

INSENSITIVE 选项用于定义游标，使之创建一个要使用的数据的临时副本。对游标的所有请求均由此临时表应答。因此，对基表的修改不会反映在对该游标提取所返回的数据中。这种类型游标访问的数据不能被修改。

应用程序可以请求一种游标类型，然后执行一条不被所请求类型服务器游标支持的 Transact-SQL 语句。SQL Server 就会返回一个错误，指出游标类型已经更改；或者如给定了一组要素，SQL Server 就会隐式转换游标。有关使 SQL Server 7.0 隐式地将游标从一种类型转换到另一种类型的完整要素列表，请参见 SQL Server Books Online。

除向前提取外，SCROLL 选项还允许向后、绝对和相对提取。滚动游标使用键集游标模型，在该模型中，任何用户对基表的已提交删除和更新都会反映在以后的提取中。仅当没有使用 INSENSITIVE 选项来声明该游标时，才成立。

如果选定了 READ ONLY 选项，则禁止对游标中的任何行进行更新。该选项将改写对游标更新的默认功能。

UPDATE [OF *column_list*] 语句用于定义游标中可更新的列。如果给出了 [OF *column_list*]，则只有列出的列允许修改。如果没有给出列表，所有的列均可更新，除非游标已定义为 READ ONLY。

注意到，SQL Server 游标的名称作用域就是连接本身这一点，是很重要的。这和局部变量的名称作用域不同。在同一用户连接上，在第一个游标释放之前，不能声明与现有游标名称相同的第二个游标。

与 PL/SQL 不同，在一个游标打开时，Transact-SQL 不支持向该游标传递参数。当 Transact-SQL 游标打开时，结果集成员身份和次序是固定的。对于其他用户已提交的对基表的更新和删除，均会反映在所有未使用 INSENSITIVE 选项定义的游标提取中。对于 INSENSITIVE 游标，还会生成一个临时表。

Oracle 游标只能向前移动--不能向后或相对滚动。SQL Server 可以使用下表所示的提取选项，向前和向后滚动。只有当游标使用 SCROLL 选项声明时，这些提取选项才可以使用。

滚动选项	说明
NEXT	如果是对游标的提取，返回结果集的第一行，如果不是，在结果集中移动游标一行。NEXT 是在结果集中移动所使用的主要方法。NEXT 是默认的游标提取。
PRIOR	在结果集中，返回上一行。
FIRST	把游标移动到结果集的第一行，并返回第一行。
LAST	把游标移动到结果集的最后一行，并返回最后一行。
ABSOLUTE *n*	返回结果集中的第 *n* 行。如果 *n* 是负值，则返回的行是从结果集最后一行向回数的第 n 行。
RELATIVE *n*	返回当前提取行后的第 *n* 行。如果 *n* 是负值，返回的行是从游标的相对位置向回数的第 n 行。

Transact-SQL FETCH 语句不需要 INTO 子句。如果没有指定返回变量，该行就会作为单行结果集，自动返回给客户。但是，如果过程必须给客户提供行，则使用无游标的 SELECT 语句，更为有效。

在每个 FETCH 之后，@@FETCH_STATUS 函数均被更新。它和 PL/SQL 中使用的 CURSOR_NAME%FOUND 和 CURSOR_NAME%NOTFOUND 变量用法类似。每次成功提取后，@@FETCH_STATUS 函数值被设为 0。如果该提取要读取游标结尾之外的地方，则返回值 -1。如果游标打开后，请求的行已被从表中删除，则 @@FETCH_STATUS 函数返回 -2。通常，返回值 -2 只在使用 SCROLL 选项声明的游标中出现。每次提取后，必须检查该变量，以保证数据的有效性。

SQL Server 不支持 Oracle 的游标 FOR 循环语法。

在 PL/SQL 和 Transact-SQL 中，用于更新和删除的 CURRENT OF 子句语法和函数是相同的。定位 UPDATE 或 DELETE 用于对指定游标内的当前行进行更新和删除操作。

Transact-SQL CLOSE CURSOR 语句关闭游标，但数据结构仍可用于重新打开游标。PL/SQL CLOSE CURSOR 语句关闭并释放所有的数据结构。

Transact-SQL 需要使用 DEALLOCATE CURSOR 语句，删除游标数据结构。DEALLOCATE CURSOR 语句与 CLOSE CURSOR 的不同之处在于，关闭的游标可以重新打开。DEALLOCATE CURSOR 语句释放所有与游标有关的数据结构，并删除游标的定义。

下面示例给出了，PL/SQL 和 Transact-SQL 中对等的游标语句。

Oracle	Microsoft SQL Server
DECLARE VSSN CHAR(9); VFNAME VARCHAR(12); VLNAME VARCHAR(20);	DECLARE @VSSN CHAR(9), @VFNAME VARCHAR(12), @VLNAME VARCHAR(20)

CURSOR CUR1 IS
SELECT SSN, FNAME, LNAME
FROM STUDENT ORDER BY LNAME;
BEGIN
OPEN CUR1;
FETCH CUR1 INTO VSSN, VFNAME, VLNAME;
WHILE (CUR1%FOUND) LOOP
FETCH CUR1 INTO VSSN, VFNAME, VLNAME;
END LOOP;
CLOSE CUR1;
END;
	DECLARE curl CURSOR FOR
	SELECT SSN, FNAME, LNAME
	FROM STUDENT ORDER BY SSN
	OPEN CUR1
	FETCH NEXT FROM CUR1
	INTO @VSSN, @VFNAME, @VLNAME
	WHILE (@@FETCH_STATUS <> -1)
	BEGIN
	FETCH NEXT FROM CUR1 INTO @VSSN, @VFNAME, @VLNAME
	END
	CLOSE CUR1
	DEALLOCATE CUR1

优化 SQL 语句

本节提供了用于优化 Transact-SQL 语句的一些 SQL Server 工具的信息。有关优化 SQL Server 数据库的详细信息，请参见本卷前面的“性能优化”。

可以使用 SQL Server 查询分析器的图形显示计划功能，了解优化程序处理语句的详细信息。

此图形工具可以实时地捕获服务器活动的连续记录。SQL Server 事件探查器监视多个不同的服务器事件和事件类别，使用用户定义的标准筛选这些事件，并把跟踪记录输出到屏幕、文件或其它 SQL Server。

SQL Server 事件探查器可用于：

- 监视 SQL Server 的性能。
- 调试 Transact-SQL 语句和存储过程。
- 确定执行缓慢的查询。
- 解决 SQL Server 中的问题。通过捕获引起某一特定问题的所有事件，然后在测试系统上重现这些事件，以重复和分离该问题，达到解决问题的目的。
- 在项目开发阶段，通过单步执行语句，每次一行，测试 SQL 语句和存储过程，来确认代码是否按照预期结果执行。
- 在生产系统上捕获事件，并在测试系统上重现所捕获的那些事件，从而为测试或调试重建了生产环境中所发生的事件。通过在其它系统中重现所捕获的事件，用户可继续使用生产系统，而不会影响正常工作。

SQL Server 事件探查器给一组扩展存储过程提供了图形用户界面。也可以直接使用这些扩展存储过程。因此，可以创建自己的应用程序，它使用 SQL Server 事件探查器扩展存储过程监视 SQL Server。

SET 语句可以为工作会话期、触发器或存储过程运行期设定 SQL Sever 查询处理选项。

SET FORCEPLAN ON 语句强制优化程序按照表在 FROM 子句中出现的顺序处理联接，类似 Oracle 优化程序中使用的 ORDERED 提示。

SET SHOWPLAN_ALL 和 SET SHOWPLAN_TEXT 语句只返回查询或语句的执行计划信息，但不执行查询或语句。要运行查询或语句，将相应的显示计划语句设为 OFF。然后，查询或语句就会执行。SHOWPLAN 选项与 Oracle EXPLAIN PLAN 工具提供的结果类似。

使用 SET STATISTICES PROFILE ON，每个执行的查询返回标准的结果集，然后，返回附加结果集（给出查询执行的事件探查）。其它选项包括 SET STATISTICS IO 和 SET STATISTIECS TIME。

Transact-SQL 语句处理包括分两步，即编译和执行。NOEXEC 选项编译每个查询，但不执行。NOEXEC 设为 ON 时，不执行随后的语句（包括其它 SET 语句），直到 NOEXEC 设为 OFF 为止。

```
  SET SHOWPLAN ON
SET NOEXEC ON
go
SELECT * FROM DEPT_ADMIN.DEPT,
STUDENT_ADMIN.STUDENT
WHERE MAJOR = DEPT
go
STEP 1
The type of query is SETON
STEP 1
The type of query is SETON
STEP 1
The type of query is SELECT
FROM TABLE
DEPT_ADMIN.DEPT
Nested iteration
Table Scan
FROM TABLE
STUDENT_ADMIN.STUDENT
Nested iteration
Table Scan
```

Oracle 需要使用提示，来调整基于开销的优化程序的操作和性能。Microsoft SQL Server 基于开销的优化程序不需要使用提示，来协助其查询评估过程。但是在某些情况下，确有使用它们的必要。

INDEX = { *index_name* | *index_id* } 提示指定了该表使用的索引名或 ID。 *index_id* 为 0，就会强制一个表扫描，而当 *index_id* 为 1，则强制使用聚集索引（如存在）。这和 Oracle 中使用的索引提示类似。

如果其列顺序和 ORDER BY 子句匹配，SQL Server FASTFIRSTROW 提示就会指示优化程序使用非聚集索引。这个提示的运行方式和 Oracle FIRST_ROWS 提示类似。

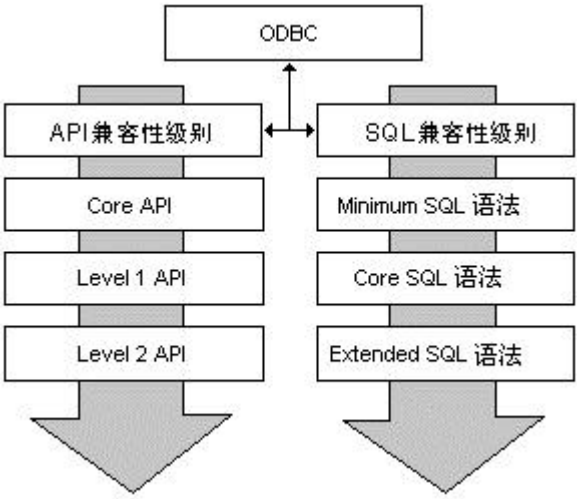
使用 ODBC

本节提供 Oracle 和 SQL Server 使用 ODBC 的方式的信息，以及使用 ODBC 开发和迁移应用程序的信息。

将应用程序代码从 Oracle 转换到 SQL Server 时，请使用下面的过程：

1. 如果应用程序用 Oracle Pro*C 或 Oracle 调用接口 (OCI) 写成，请考虑将应用程序转换为 ODBC。
2. 了解 SQL Server 默认结果集和游标选项，并选择对应用程序最有效的提取策略。

3. 如必要，将 Oracle ODBC SQL 数据类型重新映射到 SQL Server ODBC SQL 数据类型。
4. 使用 ODBC Extended SQL 扩展，创建通用的 SQL 语句。
5. 确定 SQL Server 应用程序是否需要手动提交模式。
6. 测试应用程序的性能，并在需要时修改程序。

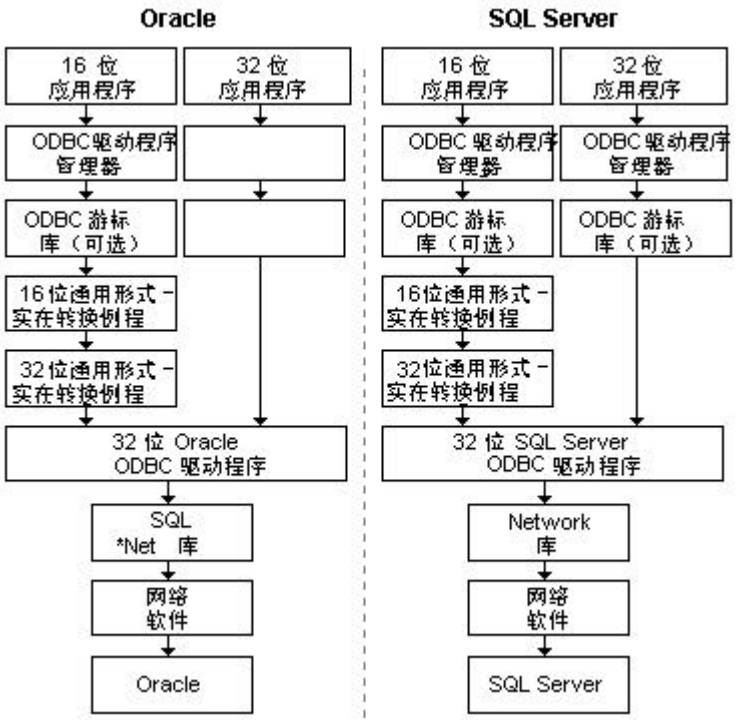


Microsoft 提供 16 位和 32 位版本的 ODBC SQL Server 驱动程序。32 位的 ODBC SQL Server 驱动程序是线程安全的。驱动程序将多个线程共享的访问串行送到共享的语句句柄（hstmt），连接句柄（hdbc）和环境句柄（henv）。即使程序使用多个线程时，ODBC 程序仍负责将语句和连接空间中的操作保持为正确的顺序。

因为 Oracle ODBC 驱动程序可由多个可能的厂商之一提供，对于体系结构和操作，就会有许多可能的方案。必须与厂商联系，确保 ODBC 驱动程序能够满足您的应用程序的需要。

在大多数情况下，Oracle ODBC 驱动程序使用 SQL*Net，与 Oracle RDBMS 连接。与 Personal Oracle 连接时，可以不使用 SQL*Net。

下图给出了 32 位环境的应用程序/驱动程序体系结构。



“形式-实在转换”一词是指一个函数调用，它是一个特殊处理，在 16 位和 32 位代码间进行转换，然后将控制转移给目标函数。注意，ODBC 游标库是如何有选择地在驱动程序管理器及其驱动程序之间驻留的。此库在仅支持只能前进的游标的驱动程序上，提供了可滚动游标服务。

Oracle 和 SQL Server 处理结果集和游标方式不同。要成功地将客户应用程序从 Oracle 迁移到 SQL Server，并以最佳状态运行，了解这些差异是至关重要的。

在 Oracle 中，当在客户应用程序中被提取时，任何 SELECT 命令的结果集都被作为只能前进的游标处理。不管使用 ODBC、OCI，还是嵌入式 SQL 作为开发工具都是如此。

默认情况下，要返回一行，客户程序（例如，ODBC 中的 SQLFetch）执行的每个 Oracle FETCH 命令都产生一个通过网络到达服务器的往返。如果客户应用程序要每次通过网络提取不止一行，则必须在其程序中建立一个数组，并使用数组提取。

由于 Oracle 的多版本并发性模型，对于只读游标，在提取间隙服务器上不保持锁定。当程序使用 FOR UPDATE 子句指定一个可更新的游标时，语句打开时，SELECT 命令中请求的所有行均被锁定。在程序发出 COMMIT 或 ROLLBACK 请求前，这些行级锁定均有效。

在 SQL Server 中，SELECT 语句并不总是与服务器上的游标关联。默认情况下，SQL Server 只是把 SELECT 语句的结果集合行依次返回给客户。SELECT 一执行，数据流就开始了。结果集数据流也可以由存储过程的 SELECT 语句返回。此外，对于单个 EXECUTE 语句，单个存储过程或一批命令可能返回多个结果集。

一旦这些默认结果集可用，SQL Server 客户就负责提取它们。对于默认结果集，客户的提取不产生到服务器的往返。相反地，对默认结果集的提取可将数据从网络缓冲区读取到程序变量中。

默认结果集模型创建了一种有效的机制，在通过网络的一次往返，向客户机返回多行数据。将网络往返次数最小化，通常是改善客户/服务器应用程序性能最重要的因素。

和 Oracle 游标相比，默认结果集赋予了 SQL Server 客户应用程序更多的职责。SQL Server 客户应用程序必须立即提取 EXECUTE 语句返回的所有结果集行。如果应用程序需要逐步地将行提供给程序的其它部分，它必须将行缓存在一个内部数组中。如果它未能提取所有的结果集行，则与 SQL Server 连接仍然保持繁忙。

如果发生这种情况,在所有结果集行被提取或客户取消请求之前,在连接上不能执行其它工作。而且，在提取完成之前，服务器继续保持表数据页上的共享锁定。正是基于提取完成之前，这些共享锁定始终保持这一事实，您必须尽快提取所有的行。这与 Oracle 应用程序中常见的逐步提取方式形成了鲜明的对比。

Microsoft SQL Server 提供了“服务器游标”，来满足通过网络逐步提取结果集的需要。可通过调用 SQLSetStmtOption 设定 SQL_CURSOR_TYPE 选项，在应用程序中请求服务器游标。

当 SELECT 语句作为服务器游标执行时，EXECUTE 语句只返回游标标识符。随后的提取请求把游标标识符和指定一次提取行数的参数一起传回给服务器。服务器返回请求的行数。

在提取请求的间隙，连接保持空闲，以执行其它命令，包括其它游标的 OPEN 或 FETCH 请求。在 ODBC 术语中，它是指服务器游标允许 SQL Server 驱动程序在单个连接上支持多个活动语句。

此外，在提取请求间隙，服务器游标通常不保持锁定，所以在提取间隙可随时暂停等候用户输入，而不会影响其他用户。可以使用乐观开放冲突检测或悲观滚动锁定并发性选项，来更新服务器游标。

尽管有了这些功能，对 Oracle 开发人员而言，使用服务器游标编程比默认结果集更熟悉，但是这也不是没有代价的。和默认结果集相比：

- 从服务器资源角度讲，服务器游标更昂贵，因为在服务器上使用了临时存储空间，来维护游标状态信息。
- 用服务器游标检索给定结果集的数据更昂贵,因为 EXECUTE 语句和服务器游标中的每个检索请求均需要一个不同的到服务器的往返。
- 在支持批处理和存储过程方面，服务器游标灵活性较差。这是因为服务器游标每次只执行一个 SELECT 语句，而默认结果集可用于批处理和存储过程，其返回多个结果集或包含 SELECT 语句以外的其它语句。

由于这些原因，最好将服务器游标的使用限定在需要其功能的应用程序部分。使用服务器游标的一个例子可以在 Ssdemo.cpp 示例 SQL Server ODBC 程序文件的 LIST_STUDENTS 函数中找到。

Oracle RDBMS 只支持前滚游标。每行均是按照查询中指定的顺序提取到应用程序中。Oracle 不支持后移到上一个提取行的请求。向后移动的唯一办法是，关闭游标再重新打开。但遗憾的是，您会被重定位到活动查询集的第一行。

因为 SQL Server 支持可滚动游标，所以可将 SQL Server 游标定位到任何行。可以向前和向后滚动。对于许多涉及用户界面的应用程序，可滚动性是一个很有用的功能。有了可滚动游标，应用程序可以一次提取一整屏的行，并且可按照用户请求，只提取附加行。

尽管 Oracle 并不直接支持可滚动游标，但是可以使用一个 ODBC 选项，将这一限制减至最小。例如，一些 Oracle ODBC 驱动程序（例如与 Microsoft Developer Studio 可视化开发系统一起发行的 Oracle ODBC 驱动程序）可在驱动程序中提供基于客户的可滚动游标。

另外，对于任何符合兼容性级别 Level One 的 ODBC 驱动程序，ODBC 游标库均支持块可滚动游标。通过使用 RDBMS 仅向前提取，以及将结果集数据缓存在内存或磁盘上，这两种客户游标选项均能支持滚动。当请求数据时，驱动程序根据需要从 RDBMS 或本地缓存中检索数据。

对于 SELECT 语句产生的结果集，基于客户的游标还支持定位 UPDATE 和 DELETE 语句。游标库使用 WHERE 子句构造 UPDATE 或 DELETE 语句，该子句为行中的每列均指定了缓存值。

如果需要可滚动游标，且要为 Oracle 和 SQL Server 实现保持相同的源代码，那么，ODBC 游标库就是一个很有用的选择。有关 ODBC 游标库的详细信息，请参见 ODBC 文档。

由于 SQL Server 提供的提取数据选项很多，有时很难决定使用哪个选项，以及何时使用。以下是一些有用的指导原则：

- 默认结果集始终是将整个数据集从 SQL Server 移到客户最快捷的方法。在应用程序中寻找可使用这一功能的可能性。例如，批量报告生成通常将整个结果集处理完，在处理过程中没有用户交互和更新。
- 如果程序需要可更新的游标，使用服务器游标。使用定位 UPDATE 或 DELETE 语句时，默认结果集绝不可以更新。此外，服务器游标比基于客户的游标更适于更新，后者必须构建对等的 UPDATE 或 DELETE 语句，来模拟定位 UPDATE 或 DELETE。
- 如果程序需要可滚动的、只读的游标，ODBC 游标库和服务器游标都是很好的选择。ODBC 游标库提供 SQL Server 与 Oracle 之间的兼容行为；在每次通过网络提取多少数据方面，服务器游标则具有更大的灵活性。
- 当使用默认结果集或构建在默认结果集之上的 ODBC 游标库游标时，要保证尽快地将结果集提取完，应避免在服务器上保持共享锁定。
- 使用服务器游标时，要确保使用 **SQLExtendedFetch**，每次提取整块的行，而不是每次一行。这和 Oracle 应用程序中的数组类型提取是一样的。服务器游标上的每个提取请求均需要在网络上从应用程序到 RDBMS 的一个往返。

购买日用杂货提供了一个类比。假定您在食品杂货店购买了 10 袋食品，把一袋食品装上车，开回家，卸下来，再返回食品杂货店取下一袋。在现实生活中，这种场景是不太可能发生的，但是，SQL Server 和程序从服务器游标单行提取数据，就是这种情形。

- 如果程序只要求只能向前、只读的游标，但依赖一个连接上多个打开的游标，当知道可将整个结果集立即提取到程序变量中时，则使用默认结果集。当不知道可否立刻提取所有的行时，使用服务器游标。

这个策略不像听起来那么困难。大多数程序员知道，何时使用单独选择，该选择至多返回一行数据。对于单独提取，使用默认结果集比使用服务器游标更有效。

有关这一技巧的例子，请参见 Ssdemo.cpp 示例 SQL Server ODBC 程序文件中的 LIST_STUDENTS 函数。如果 SELECT 语句返回不止一行时，注意服务器游标是如何被请求的。在此执行步之后，行集大小被设为一个合理的批处理大小。这允许同一个 SQLExtendedFetch 循环在默认结果集或服务器游标中均能有效地工作。

有关实现游标的详细信息，请参见 SQL Server Books Online。

ODBC 驱动程序使用语句句柄 (hstmt)，跟踪程序中的每个活动 SQL 语句。语句句柄总是与 RDBMS 连接句柄 (hdbc) 关联。ODBC 驱动程序管理器使用连接句柄，把请求的 SQL 语句送到指定的 RDBMS。大多数 Oracle ODBC 驱动程序允许每个连接上有多个语句句柄。但是，当使用默认结果集时，SQL Server ODBC 驱动程序只允许每个连接上有一个活动的语句句柄。当使用 SQL_ACTIVE_STATEMENTS 选项查询时，此 SQL Server 驱动程序的 SQLGetInfo 函数返回值 1。当语句选项设定为，使用服务器游标时，则在每个连接句柄上支持多个活动语句。

有关设定语句选项，请求服务器游标的详细信息，请参见 SQL Server Books Online。

与任何可用的 Oracle ODBC 驱动程序相比，SQL Server ODBC 驱动程序提供的一组数据类型映像更丰富。

SQL Server 数据类型	ODBC SQL 数据类型
binary	SQL_BINARY
bit	SQL_BIT
char, character	SQL_CHAR
datetime	SQL_TIMESTAMP
decimal, dec	SQL_DECIMAL
float, double precision, float(n) for n = 8-15	SQL_FLOAT
image	SQL_LONGVARBINARY
int, integer	SQL_INTEGER
money	SQL_DECIMAL
nchar	SQL_WCHAR
ntext	SQL_WLONGVARCHAR
numeric	SQL_NUMERIC
nvarchar	SQL_WVARCHAR
real, float(n) for n = 1-7	SQL_REAL
smalldatetime	SQL_TIMESTAMP
smallint	SQL_SMALLINT
smallmoney	SQL_DECIMAL
sysname	SQL_VARCHAR
text	SQL_LONGVARCHAR

timestamp	SQL_BINARY
tinyint	SQL_TINYINT
uniqueidentifier	SQL_GUID
varbinary	SQL_VARBINARY
varchar	SQL_VARCHAR

timestamp 数据类型被转换为 SQL_BINARY 数据类型。这是因为 timestamp 列中的值不是 datetime 数据，而是 binary(8) 数据。它们用于指示行上 SQL Server 活动的顺序。

下表给出了 Oracle 数据类型和用于 Oracle 的 Microsoft ODBC 驱动程序的对应关系。

Oracle 数据类型	ODBC SQL 数据类型
CHAR	SQL_CHAR
DATE	SQL_TIMESTAMP
LONG	SQL_LONGVARCHAR
LONG RAW	SQL_LONGVARBINARY
NUMBER	SQL_FLOAT
NUMBER(P)	SQL_DECIMAL
NUMBER(P,S)	SQL_DECIMAL
RAW	SQL_BINARY
VARCHAR2	SQL_VARCHAR

其他厂商的 Oracle ODBC 驱动程序其数据类型对应可能不同。

ODBC Extended SQL 标准给 ODBC 提供了 SQL 扩展，它支持 Oracle 和 SQL Server 中提供的非标准高级 SQL 功能。该标准允许 ODBC 驱动程序将通用的 SQL 语句转换为原本的 Oracle 和 SQL Server SQL 语法。

这个标准处理外部联接，例如谓词转义符、标量函数、日期/时间/时间戳值和存储程序。使用这一语法确定这些扩展：

```
--(*vendor(Microsoft), product(ODBC) extension *)--
OR
{extension}
```

转换在运行时发生，并且不需要修改任何程序代码。在大多数应用程序开发过程中，最好的方法是编写一个程序，然后在程序运行时允许 ODBC 执行 RDBMS 转换。

Oracle 和 SQL Server 没有兼容的外部联接语法。可使用 ODBC Extended SQL 外部联接语法，解决这一问题。Microsoft SQL Server 语法和 ODBC Extended SQL/SQL-92 语法是一致的。唯一的差别是 {oj } 容器。

ODBC Extended	Oracle	Microsoft SQL Server

SQL 和 SQL-92		
SELECT STUDENT.SSN, FNAME, LNAME, CCODE, GRADE FROM {oj STUDENT LEFT OUTER JOIN GRADE ON STUDENT.SSN = GRADE.SSN}	SELECT **SUBSTR**(LNAME,1,5) FROM STUDENT	SELECT **SUBSTRING**(LNAME,1,5) FROM STUDENT

ODBC 为日期、时间和时间戳值提供了三个转义子句。

类别	简写语法	格式
日期	{d 'value'}	"yyyy-mm-dd"
时间	{t 'value'}	"hh:mm:ss"
时间戳	{Ts 'value'}	"yyyy-mm-dd hh:mm:ss[.fU]"

与 SQL Server 应用程序相比，日期格式对 Oracle 应用程序的影响更大。Oracle 期望的日期格式是“DD-MON-YY”。如果不是这样，使用 TO_CHAR 或 TO_DATE 函数与日期格式模型，执行格式转换。

Microsoft SQL Server 自动转换最常用的日期格式，当自动转换不能执行时，它还提供 CONVERT 函数。

正如表中所示，ODBC Extended SQL 可用于两种数据库中。SQL Server 不需要转换函数。但是，ODBC 简写语法对 Oracle 和 SQL Server 是通用的。

ODBC Extended SQL	Oracle	Microsoft SQL Server
SELECT SSN, FNAME, LNAME, BIRTH_DATE FROM STUDENT WHERE BIRTH_DATE < {D '1970-07-04'}	SELECT SSN, FNAME, LNAME, BIRTH_DATE FROM STUDENT WHERE BIRTH_DATE < **TO_DATE**('1970-07-04', **'YYYY-MM-DD'**)	SELECT SSN, FNAME, LNAME, BIRTH_DATE FROM STUDENT WHERE BIRTH_DATE < '1970-07-04'

用于调用存储程序的 ODBC 简写语法支持 Microsoft SQL Server 存储过程、Oracle 存储过程、函数和包。可选的“?=”捕获 Oracle 函数或 SQL Server 过程的返回值。参数语法用于向被调用的程序传递值和从被调用的程序返回值。在大多数情况中，同一语法对 Oracle 和 SQL Server 应用程序是通用的。

在下面的例子中，SHOW_RELUCTANT_STUDENTS 函数是 Oracle 包 P1 的一部分。包中必须有该函数，因为它从 PL/SQL 游标中返回多个行。当调用存在于包中的函数或过程时，包的名称必须放在程序名称之前。

包 P1 中的 SHOW_RELUCTANT_STUDENTS 函数使用包游标，检索多行数据。必须调用该函数来请求每一行。如果不再检索其它行，函数返回值 0，表示不再检索其它行。这个示例 Oracle 包及其函数的性能可能不够满意。对于这种类型的操作，SQL Server 过程更为有效。

通用 ODBC Extended SQL	Oracle	Microsoft SQL Server
{?=} call *procedure_name*[(*parameter*(*s*))]} SQLExecDirect(hstmt1,(SQLCHAR *)"{?= call *owner.procedure*(?)}", SQL_NTS);	SQLExecDirect(hstmt1, (SQLCHAR*)"{**?= call STUDENT_ADMIN.P1. SHOW_RELUCTANT _STUDENTS(?)}**", SQL_NTS);	SQLExecDirect(hstmt1, (SQLCHAR*)"**{?= call STUDENT_ADMIN. SHOW_RELUCTANT _STUDENTS}**", SQL_NTS);

由于 Oracle 和 SQL Server ODBC 驱动程序的多样性，所以，并非总能得到相同的扩展 SQL 函数转换字符串。为了便于应用程序调试，可考虑使用 SQLNativeSql 函数。这个函数返回驱动程序转换的 SQL 字符串。

对于包含标量函数 CONVERT 的下列输入 SQL 字符串，其可能结果如下。SSN 列定义为类型 CHAR(9)，并被转换成数值。

原始语句	转换后的 Oracle 语句	转换后的 SQL Server 语句
SELECT **{fn CONVERT (SSN, SQL_INTEGER)}** FROM STUDENT	SELECT **TO_NUMBER(SSN)** FROM STUDENT	SELECT **CONVERT(INT, SSN)** FROM STUDENT

Common.cpp 示例程序没有使用 ODBC Extended SQL 语法。相反地，它使用一系列视图和过程，将不是 Oracle 和 SQL Server 共有的语句和函数隐藏起来。这个程序尽管是使用 ODBC 编写的，但其目的在于，让应用程序编程人员知道在编写通用程序时，如何轻而易举地克服任何显而易见的障碍。

在非 ODBC 开发环境中使用时，这些技巧和策略获得了最佳效果。如果正在使用 ODBC，则考虑使用 ODBC Extended SQL 语法，克服 Oracle 和 SQL Server 之间的任何语法上的差异。

一旦用户修改数据时，Oracle 就会自动进入事务模式。必须紧跟着一个显式 COMMIT，将更改写入数据库。如果用户要撤消更改，可执行 ROLLBACK 语句。

默认情况下，在更改发生时，SQL Server 自动提交更改。在 ODBC 中，这称为自动提交模式。如果不想使用此模式，可使用 BEGIN TRANSACTION 语句，指定包含事务的语句块的起始处。这个语句执行后，接下来是一个显式 COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句。

要保证与 Oracle 应用程序的兼容性，建议使用 SQLConnectionOption 函数，将 SQL Server 应用程序置为隐性事务模式。要实现这一点，SQL_AUTOCMMIT 选项必须设置为 SQL_AUTOCOMMIT_OFF。从示例程序中截取的这段代码例证了这个概念：

```
SQLSetConnectOption(hdbc1, SQL_AUTOCOMMIT,-sql_AUTOCOMMIT_OFF);
```

SQL_AUTOCOMMIT_OFF 选项指示驱动程序使用隐性事务。默认选项 SQL_AUTOCOMMIT_ON 选项指示驱动程序使用自动提交模式，其中每条语句在执行后均自动提交。从手动提交模式转为自动提交模式，就会将连接上任何打开的事务进行提交。

如果设定了 SQL_AUTOCOMMIT_OFF 选项，应用程序必须使用 SQLTransact 函数显式地提交或回滚事务。对于与一个连接句柄关联的所有语句句柄上的所有活动操作，此函数均请求一个提交或回滚操作。它还可以请求，对所有与环境句柄有关的连接句柄执行一个提交或回滚操作。

```
SQLTransact(henv1, hdbc1, SQL_ROLLBACK);
(SQLTransact(henv1, hdbc1, SQL_COMMIT);
```

当自动提交模式关闭时，驱动程序向服务器发出 SET IMPLICIT_TRANSACTION ON 语句。从 SQL Server 6.5 开始，这种模式支持 DDL 语句。

在手动提交模式中，要提交或回滚一个事务，应用程序必须调用 SQLTransact。SQL Server 驱动程序发出 COMMIT TRANSACTION 语句提交事务；发出 ROLLBACK TRANSACTION 语句回滚一个事务。

注意，手动提交模式可能会负面地影响 SQL Server 应用程序的性能。每个提交请求都需要一个不同的到服务器的往返，以发送 COMMIT TRANSACTION 字符串。

如果有单个原子事务处理（一个 INSERT、UPDATE 或 DELETE 语句紧跟着一个 COMMIT），则使用自动提交模式。

在示例程序中，手动提交模式也已经打开（即便对单原子事务也是如此），例证了开发 SQL Server 应用程序，来精确模拟用于 Oracle RDBMS 的类似应用程序的操作，是多么方便。

开发和管理数据库复制

本节解释 Oracle 和 Microsoft SQL Server 复制支持之间的不同点。

SQL Server 实现了快照复制，替代 Oracle 的只读快照。顾名思义，快照复制指，对数据库中某一时刻发布的信息及时地拍一张照片或快照。快照复制比事务复制需要的持续处理器开销要少，因为它不需要连续地监视源服务器上的数据更改。不同于复制 INSERT、UPDATE 和 DELETE 语句（事务复制的特征），或数据修改（合并复制的特征），订户由数据集的完全刷新来更新。因此，快照复制向订户传送所有的数据，而不是只传送更改。

SQL Server 还提供事务复制，这种类型的复制在发行者的数据库事务日志中，标记已选定的要复制事务，然后把它们作为增量变化异步地发布给订户，同时保持事务的一致性。

SQL Server 实现合并复制，替代 Oracle 的可更新快照和 Oracle 的多主控复制模型。合并复制是一种复制类型，它允许站点对复制的数据进行自主更改，并随后将所有站点更改进行合并。合并复制并不保证事务的一致性。

SQL Server 提供异类复制，这是将数据发行给异类订户的最简单方法，它使用 ODBC 并创建一个从发行者到 ODBC 订户的强制订阅。但是，作为替代方法，您可以创建一个发行，然后用嵌入式分布控制创建一个应用程序。嵌入式控制实现了从发行者到订户的强制订阅。对于 ODBC 订户，订阅数据库没有所执行复制的管理能力。

分发服务器作为 ODBC 客户，连接到所有的订阅服务器。复制要求所有分发服务器上均要安装 32 位 ODBC 驱动程序。SQL Server 安装程序在基于 Windows NT 的计算机上自动安装所需的驱动程序。

不需为 SQL Server 订阅服务器预先配置 ODBC 数据源，因为分发进程只使用订户的网络名称来建立连接。

SQL Server 也包括一个 ODBC 驱动程序，它支持 Oracle 对 SQL Server 的订阅。该驱动程序只用于基于 Intel 的计算机。要向 Oracle ODBC 订户复制，必须从 Oracle 或软件厂商那里获取相应的 Oracle SQL*Net 驱动程序。

如果在 Windows NT 注册表中提供了密码，Oracle ODBC 就会连接到 Oracle，而不需请求密码。如果 Windows NT 注册表中没有提供密码，则在 SQL Server Enterprise Manager 的新建 ODBC 订户对话框中指定 DSN 时，必须输入 Oracle ODBC 数据源的用户名和密码。

当向 Oracle ODBC 订户复制时，就会有以下限制：

- **datetime** 数据类型对应为 DATE。Oracle DATE 数据类型的范围是 4712 B.C. 到 4712 A.D.。如果向 Oracle 复制，检查复制列的 SQL Server **datetime** 条目是否在这个范围内。
- 复制的表只能有一个 **text** 或 **image** 列。
- **datetime** 数据类型对应为 Oracle CHAR 数据类型。
- SQL Server **float** 和 **real** 数据类型的范围与 Oracle 不同。

其他 ODBC 订户类型的驱动程序必须符合通用 ODBC 订户的 SQL Server 复制要求。ODBC 驱动程序：

- 必须符合 ODBC Level 1。
- 分布进程运行的处理器体系结构必须是 32 位，并且是线程安全的。
- 必须能够完成事务处理。
- 必须支持数据定义语言 (DDL)。
- 不能是只读的。

迁移数据和应用程序

本节阐述将数据在 Oracle 数据库和 Microsoft SQL Server 数据库之间互相迁移所使用的各种方法。

从 SQL Server 向 Oracle 迁移的四种可行方法

1、使用 MS SQL 自带的 Import/Export 工具 Import/Export 工具可以方便的把数据移植到 Oracle。你需要通过定义 ODBC For Oracle 作为目的源。这样的方法可以保证 SQL7 的绝大部分数据移植到 Oracl e 中去，但预先你必须在 Oracle 建立 user 和 相应的 tablespace。因为 SQL7 中有 一 些特殊的 datatype，如 text 、image 等。当一个 table 中有多于一个 text 或 image 的字段时，将出现错误，不能执行。这是你需要做出选择，或者把 text 镜像为 v archar2(4000)，或者镜像为 Long datatype，但 long datatype 一个 table 里只能 有一个。而且，还有可能遇到字符集的问题，最好用第三种方法或者第四种。所 以你遇到这样的情况，可以结合使用第 3 种方法。

2、使用 Oracle Migration Workbanch。 这个工具可以在 http://technet.oracle.com 免费下载。 它是 Oracle 提供的一个代替 SQl*Loader 的工具，当然目前该工具仍然不能完全取 代 SQL*Loader。使用 OMWB，只要你定义了 ODBC for MS SQL7 或 Access 或 Sysbase ，就可以很方便的把 tabels、views、triger、procedure、shortnaps、users 等 完全转到 Oracle 中去，对于 text，可以镜像为 CLOB 类型，CLOB 类型可以在一个 Or acle table 里有多列。 image 可以镜像为 BLOB。但是遗憾的是,OMWB2.2 不支持 中文 CLOB,无论我如何调整,数据 migrate 到 Oracle 后，都变成了????，如果谁有 解 决的方法，别忘了 email 给我。我对 OMWB 对数据流（如 image,video,sound）的控 制非常的欣赏。

3、使用 Oracle 的 SQL*Loader 使用 SQL*Loader，也许是最不方便的方法，但是是最有效的方法。可以使用 各种方法把源数据导到一个外部分件中。我使用了 MS SQL 带的 BCP 工具，可以把 那些特殊多 text 字段的 tables 导出作为外部文件。然后使用 SQL*Loader 在把这些 数据导到 Oracle 的一个临时表里，在对第 2 中方法出现的????字段进行 update。

4、使用程序进行移植 例子：从 SQL7.0 向基于 Linux 下的 Oracle 数据库倒入数据： 程序语言：java 与数据库的连接 SQL7.0：jdbc-odbc 桥，java 自带。 Oracle：jdbc，Oracle 提供。代码如下：

```
import java.lang.*;

import java.sql.*;

import oracle.jdbc.driver.*; //倒入要用到的包

public class hhw extends Object
```

```
{ public static void main(String args[]) throws SQLException, ClassNotFoundException
//抛出 SQLException 异常 { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Class.forName ("oracle.jdbc.driver.OracleDriver"); // 登记驱动程序，准备联接数据库

Connection cn1 =DriverManager.getConnection"jdbc:oracle:thin:@192.16
8.1.52:1521:SONIC", "sadly", "sadly");

Connection cn2 =DriverManager.getConnection"jdbc:odbc:sql", "sa", "" ); //联接到数
据库，建立到两个数据库的连接

Statement s1=cn1.createStatement();

Statement s2=cn2.createStatement();

ResultSet rs1=s2.executeQuery("select * from users where id>0 and id<4 0000 "); //
从数据源中取得数据，定义一些中间变量

int id;

String name=new String("1");

String passwd=new String("1");

String email=new String("1"); //执行另一个查询，向目的数据库插入数据

while(rs1.next())

{id=rs1.getInt(1);

name=rs1.getString("name");

passwd=rs1.getString("passwd");

email=rs1.getString("email"); //System.out.print(id+name+passwd+email);

s1.executeQuery("insert into bbsuser values(\\"+id+"\,\\"+name+"\,\\"+pas
swd+"\,\\"+email+"\)"); } }}
```

从 Oracle 向 SQL Server 迁移的方法

在 Oracle 和 SQL Server 之间迁移的最简单的方法是，使用 Microsoft SQL Server 7.0 中的数据转换服务 (DTS) 功能。DTS 向导指导您将数据移到 SQL Server。

如果应用程序是使用 Oracle 调用接口写成的，则可考虑使用 ODBC 重写。OCI 是 Oracle RDBMS 特有的，不能用于 Microsoft SQL Server 或其它数据库。

在大多数情况下，可以用相应 ODBC 函数代替 OCI 函数，并对支持程序代码进行相关的修改。剩下的非 OCI 程序代码应该只需要较小的修改。下例给出了，建立到 Oracle 数据库连接所需 OCI 和 ODBC 语句的对比。

Oracle 调用接口	Oracle ODBC
rcl = olog(&logon_data_area, &host_data_area, user_name, -1, (text*) 0, -1, (text) 0, -1, OCI_LM_DEF);	rcl = SQLConnect(hdbc1, (SQLCHAR*) ODBC_dsn, (SQLSMALLINT) SQL_NTS, (SQLCHAR*) user_name, (SQLSMALLINT) SQL_NTS, (SQLCHAR*) user_password, (SQLSMALLINT) SQL_NTS);

该表建议了下面 Oracle OCI 函数调用和 ODBC 函数之间的转换。这些建议的转换是近似的。可能并不存在转换过程的精确匹配。要获得类似的功能，可能需要对程序代码作进一步修改。

OCI 函数	ODBC 函数
Obindps	SQLBindParameter
Obndra	SQLBindParameter
Obndrn	SQLBindParameter
Obndrv	SQLBindParameter
Obreak	SQLCancel
Ocan	SQLCancel, SQLFreeStmt
Oclose	SQLFreeStmt
Ocof	SQLSetConnectOption
Ocom	SQLTransact
Ocon	SQLSetConnectOption
Odefin	SQLBindCol
Odefinps	SQLBindCol
Odescr	SQLDescribeCol
Oerhms	SQLError
Oexec	SQLExecute, SQLExecDirect
Oexfet	SQLExecute, SQLExecDirect, and SQLFetch
Oexn	SQLExecute, SQLExecDirect
Ofen	SQLExtendedFetch
Ofetch	SQLFetch
Oflng	SQLGetData
Ogetpi	SQLGetData

Olog	**SQLConnect**
Ologof	**SQLDisconnect**
Oopen	**SQLExecute**, **SQLExecDirect**
Oparse	**SQLPrepare**
Orol	**SQLTransact**

许多应用程序是使用 Oracle 编程接口（Pro*C、Pro*Cobol 等等）写成的。这些接口支持 SQL-92 标准嵌入式 SQL。它们还包括非标准的 Oracle 编程扩展。

可以使用 Microsoft 用于 C 语言的嵌入式 SQL (ESQL) 开发环境，将 Oracle 嵌入式 SQL 应用程序迁移到 SQL Server。与 ODBC 应用程序相比，这个环境对性能和 SQL Server 功能的使用提供了充分的、但绝非最优的控制。

有些 Oracle Pro*C 功能在 Microsoft ESQL 预编译器中不予支持。如果 Oracle 应用程序要广泛使用这些功能，重写为 ODBC 可能是一个较好的迁移选择。这些功能包括：

- 宿主数组变量。
- 将数据类型同等化的 VAR 和 TYPE 语句。
- C++ 模块中对嵌入式 SQL 的支持。
- 对嵌入式 PL/SQL 或 Transact-SQL 块的支持。
- 游标变量。
- 多线程应用程序支持。
- Oracle 通信区域 (ORACA) 支持。

如果应用程序是用 Cobol 开发的，可以从 Micro Focus 迁移到用于 Cobol 语言的嵌入式 SQL。在 Cobol 中，可能会遇到与用于 C 语言的 Microsoft ESQL 预编译器相同的一些限制。

可以把 Oracle 嵌入式 SQL 应用程序转换到 ODBC 环境。这一迁移过程很简单，并且提供许多优势。ODBC 像嵌入式 SQL 一样，不需要使用预编译器。因此，就节省了许多与程序开发有关的费用。

下表给出了嵌入式 SQL 语句和 ODBC 函数的近似关系。

嵌入式 SQL 语句	**ODBC 函数**
CONNECT	**SQLConnect**
PREPARE	**SQLPrepare**
EXECUTE	**SQLExecute**
DECLARE CURSOR 和 OPEN CURSOR	**SQLExecute**
EXECUTE IMMEDIATE	**SQLExecDirect**
DESCRIBE SELECT LIST	**SQLNumResultCols, SQLColAttributes, SQLDescribeCol**
FETCH	**SQLFetch**
SQLCA.SQLERRD[2]	**SQLRowCount**

CLOSE	**SQLFreeStmt**
COMMIT WORK, ROLLBACK WORK	**SQLTransact**
COMMIT WORK RELEASE, ROLLBACK WORK RELEASE	**SQLDisconnect**
SQLCA, SQLSTATE	**SQLError**
ALTER, CREATE, DROP, GRANT, REVOKE	**SQLExecute, SQLExecDirect**

将嵌入式 SQL 程序转换为 ODBC 时，最大的改动是有关 SQL 语句错误的处理。开发嵌入式 SQL 程序时，常常使用 MODE=ORACLE 选项。当使用这个选项时，SQL 通信区域（SQLCA）通常用于错误处理操作。

SQLCA 结构提供：

- Oracle 错误代码。
- Oracle 错误信息。
- 警告标志。
- 关于程序事件的信息。
- 最近一个 SQL 语句处理的行数。

在大多数情况下，在每个 SQL 语句执行后，应该检查 sqlca.sqlcode 变量中的值。如果值小于 0，就出现错误。如果值大于 0，请求语句执行，但带有警告。可从 sqlca.sqlerrm.sqlerrmc 变量，检索 Oracle 错误信息文本。

在 ODBC 中，函数在请求操作之后，返回一个指示其成功或失败的数字状态代码。状态代码被定义为字符串文字，包括 SQL_SUCCESS、SQL_SUCCESS_WITH_INFO、SQL_NEED_DATA、SQL_ERROR 等等。每次函数调用后，应检查这些返回值。

通过调用 SQLError 函数，可以获得关联的 SQLSTATE 值。此函数返回 SQLSTATE 错误代码、原本的错误代码（它是数据源所特有的错误代码）和错误信息文本。

当上一个 ODBC 函数调用返回 SQL_ERROR 或 SQL_SUCCESS_WITH_INFO 时，应用程序通常调用这个函数。但是，每次调用时，任何 ODBC 函数可能不发布错误或发布更多的错误，因此每次 ODBC 函数调用之后，应用程序可调用 SQLError。

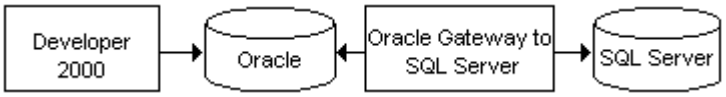
以下是每种环境中错误处理的示例。

Oracle Pro*C 和 EMBEDDED SQL	**Oracle ODBC**
EXEC SQL DECLARE CURSOR C1 CURSOR FOR SELECT SSN, FNAME, LNAME FROM STUDENT	if (SQLExecDirect(hstmtl, (SQLCHAR*)"SELECT SSN, FNAME, LNAME FROM STUDENT ORDER BY SSN", SQL_NTS) !=

ORDER BY SSN; EXEC SQL OPEN C1; if (sqlca.sqlcode) != 0 { /* handle error condition, look at sqlca.sqlerrm.sqlerrmc for error description...*/}	SQL_SUCCESS) { /* handle error condition, use SQLError for SQLSTATE details regarding error...*/}

如果应用程序是使用 Oracle Developer 2000 开发的，并且要在 SQL Server 上运行，则考虑将其转换到 Microsoft Visual Basic。Visual Basic 是一个功能强大的开发系统，可很好地用于这两种数据库。也应考虑其它平台上的开发工具，如 Microsoft Visual Studio、PowerBuilder、SQL Windows 等等。

如果不能直接从 Developer 2000 迁移，则考虑使用 Oracle Gateway to SQL Server。它可以作为从 Oracle 向 SQL Server 迁移的中间步骤。这个“通路”允许 Oracle RDBMS 连接到 SQL Server。所有的 SQL Server 数据请求都经由这个“通路”自动地转换。从 Developer 2000 应用程序的角度看，这个连接是透明的。SQL Server 数据被视作 Oracle 数据处理。几乎不需要对应用程序代码进行修改。



另外一个中间步骤是，将 Developer 2000 应用程序直接用于 SQL Server。Developer 2000 可以使用 Oracle 开放客户适配器 (OCA) 直接访问 SQL Server。OCA 符合 ODBC Level 1 规范，并有限支持 ODBC Level 2 功能。

OCA 建立一个与 SQL Server ODBC 驱动程序的连接。当把 Developer 2000 工具连接到 SQL Server 时，必须将 ODBC 数据源名称指定为数据库连接字符串的一部分。当退出 Developer 2000 应用程序时，OCA 与 ODBC 数据源的连接被断开。

下面的示例给出了登录连接字符串的语法。在此例中，用户登录到 SQL Server STUDENT_ADMIN 帐户。SQL Server ODBC 数据源的名称是 STUDENT_DATA。

```
STUDENT_ADMIN/STUDENT_ADMIN@ODBC:STUDENT_DATA
```

使用 ODBC 驱动程序并不保证，Developer 2000 应用程序在 SQL Server 上使用正常。要处理非 Oracle 数据源，必须修改应用程序代码。例如，列的安全属性是 Oracle 特有的，对 SQL Server 无效。

必须修改用于标识每行数据的键模式。当使用 Oracle 作为数据源时，用 ROWID 标识每行数据。当使用 SQL Server 时，必须使用唯一主键值，来保证行值的唯一性。

锁定模式也必须更改。使用 Oracle 时，在对行进行任何更改后，Developer 2000 就会试图立即锁定那行数据。使用 SQL Server 时，锁定模式应该被设定为延迟的，这样，记录只在被写入数据库时才被锁定。

还有许多其它要解决的问题，包括在 PL/SQL 程序块中，如果一个表上的多个插入访问同一页数据，就会有产生死锁的可能性。有关详细信息，请参见本章前面介绍的“事务、锁定和并发性”部分。

Microsoft SQL Sever 包括 SQL Server Web 助手，它是一个从 SQL Server 数据中生成标准 HTML 文件的向导。此向导可以将 Web 页配置为静态的、定期更新的或当数据更新时更新。向导帮助您创建 Web 页。

数据库示例

一个示例的大学 RDBMS 应用程序已经创建，用于支持本章中通篇引用的示例应用程序和代码。这个应用程序是专门创建的，用于阐述将 Oracle 7 应用程序转换到 SQL Server 7.0 应用程序所涉及的要点、问题和窍门。

这个示例应用程序使用四个表，记录大学中的所有活动。DEPT 表用于记录大学所设的系。CLASS 表用于记录每个系开设的课程。STUDENT 表用于记录大学中每个学生的情况。GRADE 表用于记录每门课登记的学生。

在此示例应用程序中，社会安全号码（ssn）用作 STUDENT 表的主键。DEPT 表用系代码（dept）作为主键；而课程代码（ccode）用作 CLASS 表的主键。社会安全号码（ssn）和课程代码（ccode）组成 GRADE 表的主键。

列 major 定义为 STUDENT 表的外键。选择专业（major）时，学生必须从 DEPT 表中选择一个有效的系代码（dept）。CLASS 表中的系别（dept）一列也被定义为外键。当一门课程插入这个表中时，它必须和 DEPT 表中有效的系别（dept）联系起来。

GRADE 表有两个外键。当一个学生登记一门课程时，在 STUDENT 表中必须有其社会安全号码（ssn）；在 CLASS 表中必须有课程代码（ccode）。这就保证了，学生不会登记表中不存在的课程，也不会允许表中不存在的学生来登记课程。

在本章中，对这些示例应用程序通篇引用。

- **Orademo.cpp**

  访问 Oracle 7.3 数据库中示例大学表的 Oracle ODBC 应用程序。这个程序是转换过程的起点。它允许用户对示例大学应用程序进行数据输入和制作报告。

- **Ssdemo.cpp**

  使用 ODBC 写成的 SQL Server 应用程序。这个程序是转换过程的终点。所有的 Oracle SQL 命令、过程、包和函数都已被转化为 SQL Server Transact-SQL 命令和过程。这个程序例证了许多和 SQL Server 7.0 有关的优点。

- **Common.cpp**

可用于 Oracle 和 SQL Server 的 ODBC 应用程序。要连接到 Oracle 或 SQL Server，用户只要提供 ODBC 数据源名称（DSN）即可。该程序然后登录到请求的 RDBMS。该程序包括编程技巧方面的一些极好示例，可供开发多 RDBMS 程序时使用。

- **Orauser.sql**

创建示例 Oracle 程序所需的数据库用户帐户和数据库角色。

- **Oratable.sql**

创建示例 Oracle 程序所需的表和视图。

- **Oraproc.sql**

创建示例 Oracle 程序所需的存储过程、函数和包。

- **Oracommn.sql**

创建支持 Common.cpp 程序需的所有其它 Oracle 数据库对象。

- **Oradata.sql**

将示例应用程序数据加载到示例 Oracle 程序所需的表中。

- **Ssuser.sql**

创建示例 SQL Server 程序所需的用户帐户和数据库角色。

- **Sstable.sql**

创建示例 SQL Server 程序所需的 SQL Server 表和视图。

- **Ssproc.sql**

创建示例 SQL Server 程序所需的存储过程。

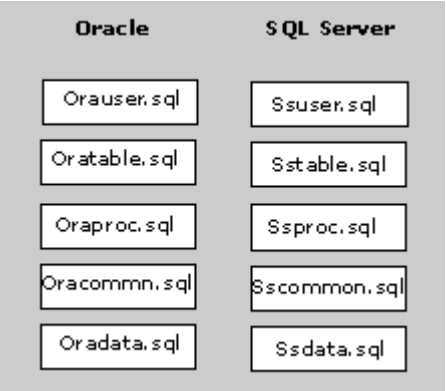
- **Sscommon.sql**

创建支持 Common.cpp 应用程序所需的所有其它 SQL Server 数据库对象。

- **Ssdata.sql**

将示例应用程序数据加载到示例 SQL Server 程序所需的表中。

要在目标 RDBMS 平台上创建示例应用程序，示例脚本必须按照下面的顺序运行。



对于 SQL Server 数据库，运行这些脚本之前，要使这些脚本和示例 SQL Server 程序能够工作，必须创建一个应用程序数据库（称为 USER_DB）。可以使用 SQL Server Enterprise Manager 或 Transact-SQL CREATE DATABASE 语句，来创建该数据库。数据库创建后，以系统管理员（sa SQL Server 用户，或 sysadmin 固定服务器角色的成员）身份登录到 SQL Server 7.0，使用 SQL Server 查询分析器并按照指定顺序运行这些脚本。

在为 Oracle 7.3 数据库运行这些脚本前，示例脚本假定 USER_DATA 和 TEMPORARY_DATA 表空间存在。它们通常是在 Oracle 7.3 的默认安装过程中创建的。如果这些表空间不存在，必须添加它们；或者修改提供的示例脚本，以使用其它表空间。

检验并确认这些表空间存在后，使用 SYSTEM 帐户登录到 SQL*Plus。如果密码不是默认值 MANAGER，则在 Oracle SQL 脚本中更改密码。

为此应用程序创建了三个用户帐户：

- STUDENT_ADMIN

该帐户是 STUDENT 和 GRADE 表的管理所有者。

- DEPT_ADMIN

该帐户是 DEPT 和 CLASS 表的管理所有者。

- ENDUSER1

该帐户是一个只查询帐户，可以访问 STUDENT、GRADE、DEPT 和 CLASS 表。