Oracle9*i* Database: The Power of Globalization Technology

An Oracle White Paper
[January] [2001]

GLOBALIZATION TECHNOLOGY - EXECUTIVE OVERVIEW

Companies today have an unprecedented opportunity to optimize and expand their businesses exponentially.   With the advent of the worldwide web Internet and Intranet applications have the possibility of global exposure. With 92% of the world's population being non-native English speaking and the international Internet community being the fastest growing consumers, opportunity knocks. Statistically a customer is 4 times more likely to make a purchase if the content is displayed in their native language. So the challenge and barrier to succeeding in the global market is to have a multilingual enabled application.

How can Oracle help? First you must have a way to be able to easily store, retrieve and update data in any and all languages. Oracle9i provides full support for Unicode 3.0 the standard for multilingual support. With UTF-8 and UTF-16 support virtually all contemporary languages and scripts of the world can be easily encoded. This allows customers to develop, deploy, and host multiple languages in a single central database. Also Oracle offers the flexibility to have all data stored in a Unicode database or incrementally store select columns in the new Unicode data type. Another key feature is the ability to present information in the users native localization customs. Things like date, time, currency symbols and delimiters and collation order are handled seamlessly in Oracle9*i*.   While Oracle's localization support is nearly comprehensive there is the Oracle Locale Builder Utility a graphical tool to do customization for special support.

Typically most web based applications are multi-tier. Oracle9*i* provides several database access products for inserting and retrieving Unicode data. Oracle offers support for the most commonly used programming environments such as Java and C/C++. Data is transparently converted between the database and client programs, which ensures that client programs are independent of the database character set.

Putting this all together may require migration of legacy data to a Unicode environment. Oracle provides the Character Set Scanner Utility to be able to proactively detect possible migration problems such as loss or truncation of data and overall impact analysis.

THE POWER OF GLOBALIZATION TECHNOLOGY/ INTRODUCTION

So what is the Power of Globalization? Simply speaking it allows companies to reach and expand their business to "the rest of the world". Creating application and software products that support the languages and locale customs of the international consumer community. This is not a trivial task, it requires much more then translating HTML content and imbedded messages.

GLOBALIZATION TECHNOLOGY IS NATIONAL LANGUAGE SUPPORT AND MORE

Most Oracle customers are familiar with the term National Language Support (NLS) and many have a great deal of experience using it. Every customer that orders a particular release of Oracle gets the same version, same binary file and yet Oracle databases are sold through out the world. How is this possible? Through National Language Support. National Language Support allows users to store, process, and retrieve data in their native languages and locales. NLS ensures that database utilities, error messages, sort order, date, time, monetary, numeric and calendar conventions automatically adapt to the users native language and locale. This term is somewhat obsolete as Oracle now allows you to do more then simply handle one language for each database instance.

Built in Universal support

Oracle's National Language Support architecture is implemented with the use of the Oracle NLS Runtime Library (NLSRTL). The NLS Runtime library is not directly exposed to customers but provides a comprehensive suite of API's exposed in SQL, PL/SQL, etc. which allow for proper text and character processing, and culturally and linguistically appropriate data handling. Note, the API's are the same for any locale (universal), but the operations that these API's perform are dependent on the locale settings. This architecture of NLSRTL offers great flexibility in meeting the requirements for software internationalization.

Oracle database applications using SQL, PL/SQL, etc. are built on top of the NLSRTL and the Oracle NLS Architecture, thus inherit the Internationalization model. An application written for example in PL/SQL can be built with one international version that can support any locale. Locale data can be updated without modification to any NLS Runtime Library functions or their usage. Therefore, support for new languages can be added, or the operation of locale-dependent features can be customized by updating the data components. End users and database administrators can then use parameter settings to modify the behavior of the RDBMS to suit their local needs. This provides a flexible architecture that allows language-dependent data to be changed, and data for a new language to be added, without requiring any changes to application code. Also, a single product can support multiple languages.

WHAT IS GLOBALIZATION?

Globalization is the process of developing multilingual applications and software products that can be accessed and run anywhere in the world simultaneously, without modification, rendering content in the native users language and locale preferences. National Language Support is a facilitator for customers to create globalized applications and software. Implementing globalization requires multiple steps. The Process of Globalization can be described as the process of applying Internationalization and Localization.

First let's look at the Internationalization process. In order to have applications that can run anywhere, they must be usable on any language operating system with non-US Keyboards or other country specific hardware. Applications should not have hard-coded dependencies on language strings, and must inter-operate with non-US versions of other products. To be easily translatable applications should isolate language text into separate translatable files. Applications should be codeset independent, be able to handle multi-byte characters and handle differences in a distributed environment. To present the correct information the application must be able to detect the users desired locale.

Localization includes translation of the separated file text. Information must then be presented to the user in a manner consistent with the user's local cultural conventions including data formatting, collation, currency, date, time and directionality of text.

So who needs Globalization?

So is your business selling products or services over the web that have international appeal? Many B2B and B2C companies do, but are they missing out on a huge potential market? Consider a consumer looking for a particular product or service. The consumer enters search keywords in their native language to their web browser but many businesses are missed . If a consumer does find your web site what is the likely hood of there purchasing if they are not addressed in their native language, statistically quite small. Lost sales is one aspect of the revenue picture but a poorly designed global application can be very costly. Take a hosting service that is trying to meet the demands of an international clientele, maintaining multiple servers and databases, one for each of the international sites. Complicated data replication and consolidation must be enforced. Duplicate code must be maintained for each of the applications. Simply a high maintenance headache.

WHY GLOBALIZATION?

Go after the largest growing market segment. 92% of the worlds population are non-native English speakers. While the earliest and most fervent users of the Web were of the other 8%, this is quickly changing. By the year 2004 it is estimated that non-English speaking users will make up 70% of the total online population. Making them by far the fastest growing group of Internet users and as we ll discuss next, consumers. Again statistically Non-US e-commerce will shift from 14% of the total worldwide revenues to 37% by 2002 a mere 250% increase. But are companies today turning away business which would make this statistic higher? A recent study showed that major web sites in the US turned away almost half the orders coming from outside the country. Why? An inability to save, access and retrieve multilingual data such as contact and address information as well as currency issues. So will non-native users buy from a site even if they can enter multilingual information. Not likely if the site does not present content in the native users language and locale preferences. Actually a consumer is 4 times more likely to buy if the content is presented in their native language.

WHAT'S NEW IN ORACLE9*I* GLOBALIZATION TECHNOLOGY?

Extended Unicode Enablement

What is Unicode? Unicode is a universal encoded character set that allows you to store information from any language using a single character set. Unicode provides a unique code value for every character, regardless of the platform, program, or language. The Unicode standard has been adopted by many software and hardware vendors, many operating systems and browsers now support Unicode. Unicode is required by modern standards such as XML, Java, JavaScript, LDAP, CORBA 3.0, WML, and it is also compliant to ISO / IEC 10646 standard.

Oracle started supporting Unicode as a database character set in Oracle7. In Oracle9i, Unicode support has been greatly expanded so that customers can find the right solution for their globalization needs. Oracle9*i* supports Unicode 3.0, the third and most recent version of the Unicode standard.

Unicode Encoding

There are two common ways to encode Unicode 3.0 characters:

- UTF-16 Encoding
- UTF-8 Encoding

UTF-8 Encoding

This is the 8-bit encoding of Unicode. It is a variable-width multibyte encoding in which the character codes 0x00 through 0x7F have the same meaning as ASCII.

One Unicode character can be 1-byte, 2-bytes, or 3-bytes in this encoding. Generally characters from the European scripts are represented in either 1 or 2 bytes, while characters from most Asian scripts are represented in 3 bytes.

UTF-16 Encoding

This is the 16-bit encoding of Unicode. It is a 2 byte fixed-width encoding in which the character codes 0x0000 through 0x007F have the same meaning as ASCII. One Unicode character is 2-bytes in this encoding. Characters from all scripts are represented in 2 bytes.

Unicode Databases

The Oracle9i database has the concept of a database character set, which specifies the encoding to be used in the SQL CHAR datatypes as well as the metadata such as table names, column names, and SQL statements. A Unicode database must be defined as UTF-8 as the database character set. There are three Oracle character sets that implement the UTF-8 encoding. The first two are designed for ASCII-based platforms while the third one should be used on EBCDIC platforms.

- UTF8

The UTF8 character set encodes characters in one to three bytes. Surrogate pairs require six bytes.

- AL32UTF8

The AL32UTF8 character set encodes characters in one to three bytes. Surrogate pairs require four bytes.

- UTFE

The UTFE character set should be used as the database character set on EBCDIC platforms to support the UTF-8 encoding.

Did you know?

You can specify the national character set for the SQL NCHAR datatypes when you create a database using the CREATE DATABASE statement. As an example you can create a database with WE8ISO8859P1 as the database character set and AL16UTF16 as the national character set. This example also illustrates the fact that you can store Unicode data in a non-Unicode database by using SQL NCHAR datatypes.

New Unicode Datatypes

The SQL NCHAR datatype now exclusively Unicode provides a flexible alternative for multilingual support for customers that need not convert their entire database to Unicode. You can store Unicode characters into columns of these datatypes regardless of the setting of the database character set. The NCHAR datatype has been redefined in Oracle9i to be a Unicode datatype exclusively. In other words, it stores data in the Unicode encoding only. You can use the SQL NCHAR datatypes in the same way you use the SQL CHAR datatypes.

The encoding used in the SQL NCHAR datatypes is specified as the national character set of the database. You can specify one of the following two Oracle character sets as the national character set:

- AL16UTF16

This is the default character set for SQL NCHAR datatypes. The character set encodes Unicode data in the UTF-16 encoding. Data is counted in 16-bit UTF-16 units.

- UTF8

When UTF8 is specified for SQL NCHAR datatypes, the data stored in the SQL datatypes is in UTF-8, but it is counted as if AL16UTF16 were specified. By default, data is stored in the UTF-16 encoding in the SQL NCHAR datatypes, and the length specified in the NCHAR and NVARCHAR2 columns is always in the number of characters instead of the number of bytes.

Surrogate Pairs

You can extend Unicode to encode more than 1 million characters. These extended characters are called surrogate pairs. Surrogate pairs are designed to allow representation of characters in future extensions of the Unicode standard. Surrogate pairs require 4 bytes in UTF-8 and UTF-16. There is also a reserved area for private use that will never be used by Unicode. This area allows specialized scripts to be stored. As an example perhaps a Star Trek web site wishes to support the language Klingon. Each Klingon character can have a special mapping to the private area using surrogates.

Character Semantics

Character semantics simplifies the task of handling storage requirements and application string manipulation for multibyte strings. Consider character semantics if you are setting up a Unicode database. Unlike say an ASCII character set , UTF-8 and UTF-16 characters are held in multibyte segments. Processing character strings can be much more straight forward especially with variable-width character set such as UTF-8 using character semantics. For example CHAR(10) implies 10 code points or characters of storage. Supported by SQL string functions such as SUBSTR, LENGTH and INSTR.

Expanded Locale Coverage

Now over 57 languages and 88 countries and territories worldwide and 200 character sets. Through the use of Unicode databases and datatypes, Oracle*9i* supports most contemporary languages and scripts. Oracle*9i* supports different cultural conventions that are specific to a given geographical location. The default local time format, date format, numeric and monetary conventions are handled based on the local territory setting. By setting different NLS parameters, the database session can utilize different cultural settings. As an example dual currency is supported. In Germany it may be important to display the Deutschmark and the Euro.

Date and Time Formats are also an important consideration to building global applications. The world's various conventions for hour, day, month, and year can be handled in local formats. For example, in the UK, the date is displayed using the format mask 'DD-MON-YYYY' while Japan commonly uses 'YYYY-MON-DD'.

Monetary and Numeric Formats such as currency, credit, and debit symbols also need to be represented in local formats. Radix symbols and thousands separators can be defined by locales. For example, in the US, the decimal separator is a dot ".", while it is a comma "," in France. Therefore, $1,234 could have different meanings in different countries.

Many different calendar systems are in use around the world. Oracle supports seven different calendar systems: Gregorian, Japanese Imperial, ROC Official (Republic of China), Thai Buddha, Persian, English Hijrah, and Arabic Hijrah.

Overview of Oracle’s Sorting Capabilities

Oracle provides linguistic sort capabilities that handle the complex sorting requirements of different languages and cultures. Different languages have different sort orders. What's more, different cultures or countries using the same alphabets may sort words differently. For example, in Danish, the letter Æ is after Z, while Y and Ü are considered to be variants of the same letter. Sort order can be case sensitive or insensitive, and can ignore accents or not. It can also be either phonetic or based on the appearance of the character, such as ordering by the number of strokes or by radicals for East Asian ideographs. Another common sorting issue is when letters are combined. For example, in traditional Spanish, "ch" is a distinct character, which means that the correct order would be: cerveza, Colorado, cheremoya, and so on. This means that the letter "c" cannot be sorted until checking to see if the next letter is an "h". Oracle provides several different types of sort, and can achieve a linguistically correct sort as well as the new multilingual ISO standard (10646) designed to handle many languages at the same time.

Using Binary Sorts

Conventionally, when character data is stored, the sort sequence is based on the numeric values of the characters defined by the character encoding scheme. This is called a binary sort. Binary sorts are the fastest type of sort, and produce reasonable results for the English alphabet because the ASCII and EBCDIC standards define the letters A to Z in ascending numeric value. Note, however, that in the ASCII standard, all uppercase letters appear before any lowercase letters. In the EBCDIC standard, the opposite is true: all lowercase letters appear before any uppercase letters. When characters used in other languages are present, a binary sort generally does not produce reasonable results. For example, an ascending ORDER BY query would return the character strings ABC, ABZ, BCD, ÄBC, in the sequence, when the Ä has a higher numeric value than B in the character encoding scheme. For languages using Chinese characters, a binary sort is not linguistically meaningful.

Using Linguistic Sorts

To produce a sort sequence that matches the alphabetic sequence of characters, another sort technique must be used that sorts characters independently of their numeric values in the character encoding scheme. This technique is called a linguistic sort. A linguistic sort operates by replacing characters with numeric values that reflect each character's proper linguistic order. These numeric values are found in a table containing major and minor values. Oracle makes two passes when comparing strings. The first pass is to compare the major value of entire string from the major table and the second pass is to compare the minor value from the minor table. Each major table entry contains the Unicode codepoint and major value. Usually, letters with the same appearance will have the same major value. Oracle defines letters with diacritic and case differences for the same major value but different minor values. Oracle offers two kinds of linguistic sort: monolingual, commonly used for European languages; and multilingual, commonly used for Asian languages.

Using Monolingual Linguistic Sorts

Oracle offers monolingual linguistic sorts that contain culture-specific sorting order for almost all European languages. Using Multilingual Linguistic Sorts Oracle*9i* extends monolingual linguistic sorts so that you can now sort additional languages as part of one sort. This is useful for certain regions or languages that have complex sorting rules or global multilingual databases. Additionally, Oracle*9i* still supports all the sort orders defined by the previous releases.

For example, in Oracle9i, a French sort is supported, but the new multilingual linguistic sort for French can also be applied by changing the sort order from French to French_M. By doing so, the sorting order will be based on the GENERIC_M sorting order and with the capability to sort secondary level from right to left. Oracle recommends using a multilingual linguistic sort if the tables contain multilingual data. If the tables contain only pure French, for memory usage concern, a French sort may get better performance. There is a trade-off between extensibility and performance.

For Asian language data or multilingual data, Oracle provides a sorting mechanism based on an ISO standard (ISO14651) and the Unicode 3.0 standard. Multilingual linguistic sorting for Asian languages are implemented in a three pass fashion based on the number of strokes, PinYin, or radicals. In addition, handling of canonical equivalence and surrogate codepoint pairs is also implemented with a capacity to define up to 1.1 million codepoints in one sort.

Using Linguistic Indexes

Using linguistic indices you can provide the sophisticated sorting capabilities of a multilingual sort while achieving sorting performance nearly as good as a binary sort (which offers the best performance). Function-based index that uses languages other than English can be created. The index itself does not change the linguistic sort order determined by NLS_SORT. The index simply improves the performance.

Multiple Linguistic Indexes

If users wish to store character data of multiple languages into one database, they should create multiple linguistic indexes for one column. This approach improves the performance of the linguistic sort for a specific column for multiple languages and is a powerful feature for multilingual databases.

Date and Time Zones

Applications that support multi-geographical locales will find comprehensive and precision oriented support for time zones, removing the complexity of doing manual calculations. The new datetime data types can store time data with sub-second precision. The datetime data types TSLTZ and TSTZ are time-zone-aware. Datetime values can be specified as local time in a particular region, rather than a particular offset. Using the time zone rules tables for a given region, the time zone offset for a local time is calculated, taking into consideration Daylight Savings time adjustments, and used in further operations.

Character Set Scanner

Migration issues

The Character Set Scanner provides an assessment of the feasibility and potential issues in migrating an Oracle database to a new database character set.   Many customers have discovered and avoided potential migration issues such data truncation, invalid characters and fields that need to be expanded prior to migrating their databases to Unicode using the Character Set Scanner. When migrating to a new database character set, the Export and Import utilities can handle character set conversions from the original database character set to the new database character set. However, character set conversions can sometimes cause data loss or data corruption. For example, if you are migrating from character set A to character set B, the destination character set B should be a superset of character set A. Characters that are not available in character set B will be converted to replacement characters. Another scenario that can cause the loss of data is migrating a database containing data of a different character set from that of the database character set. How can this scenario occur? Users can insert data into the database from another character set if the client `NLS_LANG` character set setting is the same as the database character set. When these settings are the same, Oracle assumes that the data being sent or received is of the same character set, so no validations or conversions are performed. This can lead to two possible data inconsistency problems. One is when a database contains data from another character set but the same codepoints exist in both character sets. The second possibility is having data from mixed character sets inside the database. For example, if the data character set is WE8MSWIN1252, and two separate Windows clients using German and Chinese are both using the `NLS_LANG` character set setting as WE8MSWIN1252, then the database will contain a mixture of German and Simplified Chinese. Obviously for this character set choice i.e. WE8MSWIN1252 the Chinese characters are not expected or supported.

Anticipating Migration Issues

The Scanner checks all character data in the database and tests for the effects and problems of changing the character set encoding. At the end of the scan, it generates a summary report of the database scan. This report provides estimates of the amount of work required to convert the database to a new character set. The Scanner reads the character data and tests for the following conditions on each data cell:

- Do character codes of the data cells change when converted to the new character set?
- Can the data cells be successfully converted to the new character set?
- Will the post-conversion data fit into the current column size?

The Scanner reads and tests for data in CHAR, VARCHAR2, LONG, CLOB, NCHAR, NVARCHAR2, and NCLOB columns only. The Scanner does not perform post-conversion column size testing for LONG, CLOB, and NCLOB columns.

The Oracle Locale Builder

The Oracle Locale Builder allows users to customize virtually any type of locale definition simply and safely through an intuitive graphical interface. The Oracle Locale Builder offers an easy and efficient way to access and define NLS locale data definitions. It provides a graphical user interface through which users can easily view, modify, and define various locale-specific data. It extracts data from the definition files (`*.nlt` and `*.nlb`) and presents them in a readable format, to allow processing of the information without worrying about the specific definition formats used in these files. Users can choose to have the resulting output file be in either text format (.nlt) or the default binary format (.nlb). The tool can be accessed either locally by running it as a local application or remotely with a standard web browser.

The Oracle Locale Builder handles 4 types of locale definitions:

- **Territories** including calendar convention, date and time formats, number and monetary systems
- **Languages** including local month and day names, and writing direction
- **Character Sets** including character set type, character mappings and classifications
- **Collations** including linguistic sort order, and special collation rules

Locale Builder offers 2 approaches to adding new Locale definitions or customizing existing definitions. To do this essentially you build upon an inherited setting and modify it to add additional properties. For example perhaps you wish to build a hybrid language French American. With the Locale Builder you could choose French. Modify month and day names or character rules to create a Unique French American language. As opposed to creating a whole new language you may want to customize a new territory. After defining a new territory for a given language you can then modify date, time, monetary and numeric formats without affecting the Language characteristics. Numeric formats include decimal separator, how numbers are rounded and what measurement standard is used such as the metric system. Monetary formats can be changed such as currency symbol, decimal and group separators grouping and precision, and credit and debit symbols.

You can create new character sets and extend an existing encoded character set definition by using User-defined characters (UDC) to encode special characters representing:

- Proper names

- Historical Han characters that are not defined in an existing character set standard
- Vendor-specific characters
- New symbols or characters you define

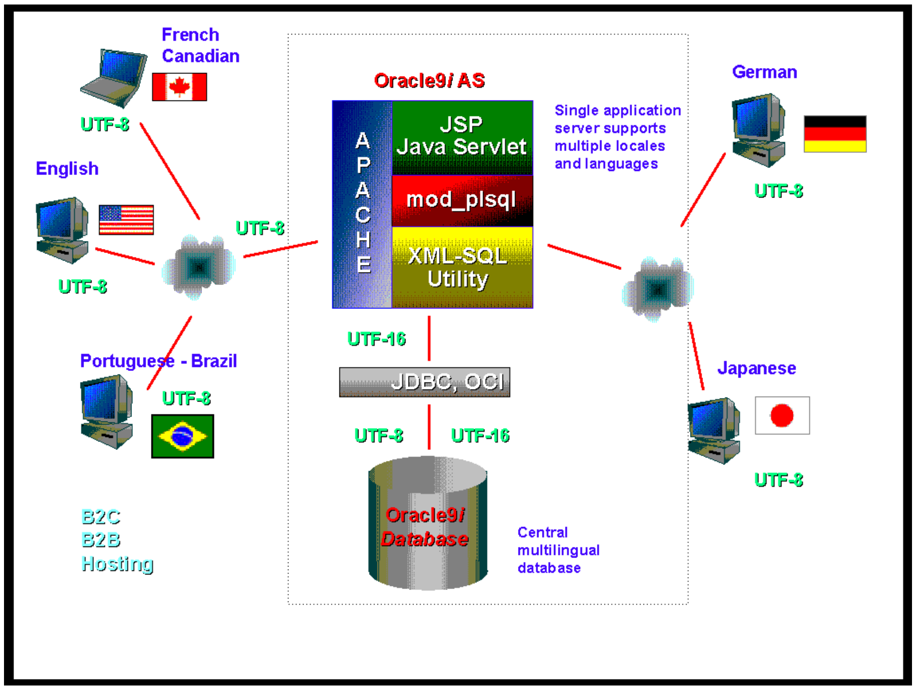
User-defined characters are typically supported within such as Unicode and East Asian character sets. Character sets that support User-defined characters have at least one range of reserved codepoints for use as user-defined characters. For example, Japanese Shift JIS preserves 1880 codepoints for user-defined characters.

Collation order can be changed in many ways. Code point ranges can be reconfigured so for example numbers sort after letters or you can change the order of individual code points. Accented characters can be customized individually. Canonical

Building a Central Multilingual Server

Architectural Diagram

An e-business runs on one globalized computer system. Everybody is connected. And all the information is shared in one place.



Taking a Central Server Approach

Here is what Oracle faced prior to creating a single point solution and saving over a billion dollars in the process. Does this sound like your organization? Information was scattered across hundreds of separate databases and that was the problem! Each organizations, marketing, sales, service, etc. had its own computer system. Each computer system had its own database. Oracle had hundreds of databases around the world. Data was so fragmented, it was difficult for people to find the information they needed to do their jobs. Separate databases also made it difficult to share information between the organizations. And if groups can't share information, they don't cooperate. So, marketing didn't cooperate with sales. Germany didn't cooperate with France. And the lack of cooperation led to duplication of effort and inefficiency. To eliminate this inefficiency, Oracle had to make information easier to find and easier to share. But how? The solution was quite simple become an e-business. But what is an e-business anyway? It's all about the Internet and globalization. An e-business uses a global network and a global database to integrate all aspects of doing business. Every business function: marketing, sales, supply chain, manufacturing, customer service, accounting, human resources, everything uses the same global network and the same global database. An e-business runs on one unified computer system. Everybody is connected. And all the information is in one place. While conceptually simple, this single, unified database approach required fundamental changes to application software. Most companies whether doing B2B, B2C or hosting face the challenge of handling multilingual data and presenting information in their users native languages and locales. Middle tier software must be able to support web traffic coming from anywhere in the world, and identify the customers language and locale preferences, and create the proper translated pages. If we are to consolidate servers and data to a unary approach then all components must be multilingual ready.

Implementing a Unicode Solution in the Database

The first step in building a central multilingual server is to have a way to be able to easily store, retrieve and update data in any and all languages. With UTF-8 and UTF-16 support virtually all contemporary languages and scripts of the world can be easily encoded. This allows customers to develop, deploy, and host multiple languages in a single central database. Multilingual information can be shared and yet each user can be served in their own locale preferences.

Single Multilingual Application Server

Supporting multiple locales simultaneously from a single binary and a single code base has many benefits including reducing the cost of hardware, maintenance and faster support of additional locales. Unicode lets your applications support multiple languages simultaneously. And the single multilingual server approach provides full Unicode support in all tiers of the application architecture.

Just as Oracle9i manages all your data, Oracle9i Application Server (Oracle9i AS) runs all your applications. For Internet and intranet applications, Oracle9i Application Server offers the most innovative and comprehensive set of middle-tier services. While offering unique capabilities to quickly develop web applications a second key feature is the tight integration of Oracle9i Application Server with Oracle database technologies. Each Oracle9i Application Server development environment provides programming methods that minimizes the complexities of inserting and retrieving Unicode data regardless of the client character set. For example let's look at JSP, Java Servlets, and PSPs.

Oracle9i AS implements the standard Java Servlet API that allows the deployment of Java Servlets. It also implements the JSP compiler according to the standard Java ServerPages specification that allows users to compile standard JSPs to Java Servlets. As a result, applications can fully utilize the internationalization support provided in the Java (JDK), Java Servlet and JSP technologies. Oracle JDBC drivers transparently convert the data from the database character set if needed to UTF-16, for all Char types, as well as LONG and CLOB. As a result of this transparent conversion, JSPs and Java Servlets calling Oracle JDBC drivers may bind and define database columns with Java strings, and fetch data into Java strings from the result set of a SQL execution.

PSP and PL/SQL Web Toolkit Oracle iAS provides a Web gateway that allows PL/SQL stored procedures to generate dynamic web content and deliver it to the client browser in the same way as Java Servlets. Oracle9*i* provides an API called the PL/SQL Web Toolkit for the development of Internet applications in PL/SQL stored procedures. The API provides methods for you to format web pages in PL/SQL and send them out to the client browser via the Web gateway. In addition to the Web Toolkit, you may also use the SQL functions, such as SUBSTRING(), TO_DATE() and LENGTH(), provided by Oracle9*i* to manipulate strings. All string variables, such as VARCHAR and CHAR strings, and the SQL functions used in the PL/SQL stored procedures operate on Unicode.

PSPs are HTML pages with embedded PL/SQL code. PSP relates to PL/SQL stored procedure in the same way as JSP relates to Java Servlet. Oracle9*i* provides a PSP compiler to compile PSPs into a PL/SQL stored procedures and load them into the database. When using PSP and PL/SQL stored procedures in an application, direct access to UTF-8 and UTF-16 data comes automatically by using SQL or PL/SQL from inside the database.

For more detailed information on building multilingual Internet applications see the online globalization white papers at:
http://technet.oracle.com/products/oracle8i/

Summary

Flexible

With Oracle9*i* customers can create a locale environment that meets their business needs, from basic monolingual, native language support to centralized multilingual databases. Localization parameter settings can be specified from the client session, server, or explicitly within SQL functions. Customization features are also available to help tailor the environment. Customers needing to move to a Unicode solution, can choose between using a Unicode database or the Unicode datatypes to best suit their business needs.

Compatible

Support for 200 different national, multinational, and vendor-specific character sets, provides consistency and interoperability with existing data. The most popular single-byte, multi-byte, and fixed-width encodings used in the industry today including UTF-8 and UTF-16 for full Unicode 3.0 support. For customers needing to move to a Unicode solution will find, the Character Set Scanner utility can help identify compatibility issues to assure a smooth migration. Oracle9i further supports deployment of multi-tier multilingual applications by automatically and transparently performing any necessary character set conversions on the database.

Integrated

Oracle makes use of a fully internationalized single-binary model which ensures that the same Oracle9*i* release can service users all around the globe. UI translations are also included in many different languages. Plan to support multilingual applications and a Unicode database? All access programming interfaces to Oracle9*i* are enabled for both UTF-16 and UTF-8, thus providing excellent native integration for applications written in these Unicode forms.

APPENDIX A

Locale Data

This appendix lists the languages territories, character sets, and other locale data supported by the Oracle server. It includes the following topics:

- Languages and Translated Messages
- Linguistic Sorting
- Multilingual Sorting
- Territories
- Calendar Systems
- Character Sets

Bold indicates new in Oracle9i

Languages

American English
Arabic †
Assamese
Bangla
Bengali
Brazilian Portuguese †
Bulgarian
Canadian French
Catalan †
Croatian
Czech †
Danish †
Dutch †
Egyptian
English
Estonian
Finnish †
French †
German †
German Din
Greek †
Gujarati
Kannada
Hebrew †
Hindi
Hungarian †
Icelandic
Indonesian
Italian †
Japanese †
Korean †
Latin American Spanish †
Latvian
Lithuanian
Malay
Malayalam
Marathi
Mexican Spanish
Norwegian †
Oriya
Polish †
Portuguese †
Punjabi
Romanian †
Russian †
Simplified Chinese †
Slovak †
Slovenian
Spanish †
Swedish †
Telugu
Tamil
Thai
Traditional Chinese †
Turkish †
Ukrainian
Vietnamese

†Translated software messages available.

Linguistic Sorts

Arabic
Arabic_ABJ_Match
Arabic_ABJ_SORT
Arabic_Match
ASCII7
Bengali
Big5
Bulgarian
Canadian French
Catalan**
Croatian**
Czech**
Czech_Binary**
Danish**
Dutch**
EEC_EURO
EEC_EUROPA3
Estonian
Finnish
French**
GBK
German**
German_Din**
Greek
Hebrew
HKSCS

Icelandic
Indonesian
Italian
Japanese
Latin
Latvian
Lithuanian
Malay
Norwegian
Polish
Punctuation**
Romanian
Russian
Slovak**
Slovenian**
Spanish**
Swedish
Swiss**
Thai_Dictionary
Thai_Telephone
Turkish**
Ukrainian
Unicode_Binary*
Vietnamese
West_European**

Multilingual Linguistic Sorts

GENERIC_M
SPANISH_M
FRENCH_M
THAI_M
SCHINESE_STROKE_M
SCHINESE_PINYIN_M
TCHINESE_RADICAL_M
TCHINESE_STROKE_M
JAPANESE_M
KOREAN_M
CANADIAN_M
DANISH_M
CYRILLIC_M

Hungarian**

Territories

Algeria
America
Austria
Australia
Bahrain
Bangladesh
Belgium
Brazil
Bulgaria
Canada
Catalonia
China
Chile
CIS
Colombia
Costa Rica
Croatia
Cyprus
Czech Republic
Czechoslovakia
Denmark
Djibouti
Egypt
El Salvador
Estonia
Finland
France
Germany
Greece
Guatemala
Hong Kong
Hungary
Iceland
India
Indonesia
Iraq
Ireland
Israel
Italy
Japan
Jordan
Kazakhstan
Persian
Korea
Kuwait
Latvia
Lebanon
Libya
Lithuania
Luxembourg
Macedonia
Malaysia
Mauritania
Mexico
Morocco
New Zealand
Nicaragua
Norway
Oman
Panama
Peru
Poland
Portugal
Puerto Rico
Qatar
Romania
Saudi Arabia
Singapore
Slovakia
Slovenia
Somalia
South Africa
Spain
Sudan
Sweden
Switzerland
Syria
Taiwan
Thailand
The Netherlands
Tunisia
Turkey
Ukraine
United Arab Emirates
United Kingdom
Uzbekistan
Venezuela
Vietnam
Yemen
Yugoslavia

Calendars

Republic of China Official
Thai Buddha
Japanese Imperial
Arabic Hijrah
English Hijrah
Gregorian

Character Sets

AL16UTF16 16-bit Unicode
AL32UTF8 32-bit Unicode
APTEC 715 Client 8-bit Latin/Arabic
APTEC 715 Server 8-bit Latin/Arabic
Arabic MS-DOS 710 Client 8-bit Latin/Arabic
Arabic MS-DOS 710 Server 8-bit Latin/Arabic
Arabic MS-DOS 720 Client 8-bit Latin/Arabic
Arabic MS-DOS 720 Server 8-bit Latin/Arabic
ASCII 7-bit American
ASCII 7-bit Finnish
ASCII 7-bit Yugoslavian
ASMO 708 Plus 8-bit Latin/Arabic
ASMO Extended 708 8-bit Latin/Arabic
Bangladesh National Code 8-bit BSCII
BESTA 8-bit Latin/Cyrillic
BIG5 16-bit Traditional Chinese
BLT8ISO8859P13 8-bit Baltic
Bull EBCDIC GCOS7 8-bit Greek
Bull EBCDIC GCOS7 8-bit West European
CEL8ISO8859P14 Celtic
CL8EBCDIC1025R 8-bit Cyrillica
CL8ISOIR111 8-bit Cyrillic
CL8KOI8U Ukrainian Cyrillic
CGB2312-80 16-bit Simplified Chinese
DEC 8-bit Latin/Greek
DEC 8-bit Turkish
DEC 8-bit West European
DEC VT100 7-bit Dutch
DEC VT100 7-bit Finnish
DEC VT100 7-bit French
DEC VT100 7-bit German
DEC VT100 7-bit Italian
DEC VT100 7-bit Norwegian/Danish
DEC VT100 7-bit Spanish
DEC VT100 7-bit Swedish
DEC VT100 7-bit Swiss (German/French)
DEC VT100 7-bit Turkish
DG 8-bit West European
EBCDIC 16-bit Simplified Chinese
EBCDIC Code Page 1025 (Modified) 8-bit Cyrillic
EBCDIC Code Page 1025 8-bit Cyrillic
EBCDIC Code Page 1026 8-bit Turkish
EBCDIC Code Page 1086 8-bit Hebrew
EBCDIC Code Page 1112 8-bit Baltic Multilingual
EBCDIC Code Page 273/1 8-bit Austrian German
EBCDIC Code Page 277/1 8-bit Danish
EBCDIC Code Page 278/1 8-bit Swedish
EBCDIC Code Page 280/1 8-bit Italian
EBCDIC Code Page 284 8-bit Latin American Spanish
EBCDIC Code Page 285 8-bit West European
EBCDIC Code Page 297 8-bit French
EBCDIC Code Page 37 8-bit Oracle/c
EBCDIC Code Page 37 8-bit West European
EBCDIC Code Page 424 8-bit Latin/Hebrew
EBCDIC Code Page 500 8-bit Oracle/c
EBCDIC Code Page 500 8-bit West European
EBCDIC Code Page 870 8-bit East European
EBCDIC Code Page 871 8-bit Icelandic
EBCDIC Code Page 875 8-bit Greek
EBCDIC Code Page 1141 8-bit Austrian German *†
EBCDIC Code Page 1142 8-bit Danish *†
EBCDIC Code Page 1143 8-bit Swedish *†
EBCDIC Code Page 1147 8-bit French *†
EBCDIC XBASIC Server 8-bit Latin/Arabic
EEC EUROPA3 8-bit West European/Greek
EE8BS2000 8-bit East European
EL8EBCDIC875R EBCDIC 8-bit Greek
EEC Targon 35 ASCI West European / Greek
EUC 16-bit Japanese
EUC 16-bit Japanese (Modified Yen)
EUC 32-bit Traditional Chinese
German Government Printer 8-bit All-European Latin
HP ARABIC8 8-bit Latin/Arabic
HP CCDC 16-bit Traditional Chinese
HP LaserJet 8-bit West European
HP Roman8 8-bit West European
Hungarian 8-bit CWI-2
Hungarian 8-bit Special AB Mod
IBM DBCS 16-bit Japanese
IBM DBCS 16-bit Korean
IBM DBCS 16-bit Traditional Chinese
IBM DBCS Code Page 290 16-bit Japanese
IBM-PC Alternative Code Page 8-bit Latvian (Latin/Cyrillic)
IBM-PC Code Page 1117 8-bit Latvian
IBM-PC Code Page 1507/862 8-bit Latin/Hebrew
IBM-PC Code Page 437 8-bit (Bulgarian modification)
IBM-PC Code Page 437 8-bit (Greek modification)
IBM-PC Code Page 437 8-bit American
IBM-PC Code Page 737 8-bit Greek/Latin
IBM-PC Code Page 772 8-bit Lithuanian
IBM-PC Code Page 774 8-bit Lithuanian (Latin)
IBM-PC Code Page 775 8-bit Baltic
IBM-PC Code Page 850 8-bit West European
IBM-PC Code Page 851 8-bit Greek/Latin
IBM-PC Code Page 852 8-bit East European
IBM-PC Code Page 855 8-bit Latin/Cyrillic
IBM-PC Code Page 857 8-bit Turkish
IBM-PC Code Page 860 8-bit West European
IBM-PC Code Page 861 8-bit Icelandic
IBM-PC Code Page 863 8-bit Canadian French
IBM-PC Code Page 865 8-bit Norwegian
IBM-PC Code Page 866 8-bit Latin/Cyrillic
IBM-PC Code Page 869 8-bit Greek/Latin
ICL EBCDIC 8-bit American
ICL EBCDIC 8-bit West European
ICL special version ISO8859-1
ISO 6937 8-bit Coded Character Set for Text

ISO 8859-1 West European
ISO 8859-10 North European
ISO 8859-2 East European
ISO 8859-3 South European
ISO 8859-4 North and North-East European
ISO 8859-5 Latin/Cyrillic
ISO 8859-6 Latin/Arabic
ISO 8859-7 Latin/Greek
ISO 8859-8 Latin/Hebrew
ISO 8859-9 West European & Turkish
ISO 8859-15 *†
Israeli Standard 960 7-bit Latin/Hebrew
JVMS 16-bit Japanese
KSC5601 16-bit Korean
KSCCS 16-bit Korean
Latvian Standard LVS8-92(1) Windows/Unix 8-bit Baltic
Latvian Version IBM-PC Code Page 866 8-bit Latin/Cyrillic
Mac Client 8-bit Central European
Mac Client 8-bit Croatian
Mac Client 8-bit Extended Roman8 West European
Mac Client 8-bit Greek
Mac Client 8-bit Hebrew
Mac Client 8-bit Icelandic
Mac Client 8-bit Latin/Arabic
Mac Client 8-bit Latin/Cyrillic
Mac Client 8-bit Latin/Thai
Mac Client 8-bit Turkish
Mac Client CGB2312-80 16-bit Simplified Chinese
Mac Client Shift-JIS 16-bit Japanese
Mac Server 8-bit Central European
Mac Server 8-bit Croatian
Mac Server 8-bit Extended Roman8 West European
Mac Server 8-bit Greek
Mac Server 8-bit Hebrew
Mac Server 8-bit Icelandic
Mac Server 8-bit Latin/Arabic
Mac Server 8-bit Latin/Cyrillic
Mac Server 8-bit Latin/Thai
Mac Server 8-bit Turkish
MS Windows 8-bit Bulgarian Cyrillic
MS Windows Code Page 921 8-bit Lithuanian
MS Windows Code Page 923 8-bit Estonian
MS Windows Code page 949 Korean*
MS Windows Code page 950 Traditional Chinese*
MS Windows Code page 950 Traditional Chinese with Hong Kong Supplementary Character Set*
MS Windows Code Page 1250 8-bit East European †
MS Windows Code Page 1251 8-bit Latin/Cyrillic †
MS Windows Code Page 1252 8-bit West European †
MS Windows Code Page 1253 8-bit Latin/Greek †
MS Windows Code Page 1254 8-bit Turkish †
MS Windows Code Page 1255 8-bit Latin/Hebrew †
MS Windows Code Page 1256 8-Bit Latin/Arabic †
MS Windows Code Page 1257 8-bit Baltic †
MS Windows Code Page 1258 8-bit Vietnamese *†
Multiple-Script Indian Standard 8-bit Latin/Indian
Mussaď Alarabi/2 768 Client 8-bit Latin/Arabic
Mussaď Alarabi/2 768 Server 8-bit Latin/Arabic
Nafitha Enhanced 711 Client 8-bit Latin/Arabic
Nafitha Enhanced 711 Server 8-bit Latin/Arabic
Nafitha International 721 Client 8-bit Latin/Arabic
Nafitha International 721 Server 8-bit Latin/Arabic
NCR 4970 8-bit West European
NeXTSTEP PostScript 8-bit West European
No-op character set prohibiting conversions
RELCOM Internet Standard 8-bit Latin/Cyrillic
SAKHR 706 Server 8-bit Latin/Arabic
SAKHR 707 Client 8-bit Latin/Arabic
SAKHR 707 Server 8-bit Latin/Arabic
Shift-JIS 16-bit Japanese
Shift-JIS 16-bit Japanese (Modified Yen)
Siemens 9750-62 EBCDIC 8-bit American
Siemens 9750-62 EBCDIC 8-bit Danish
Siemens 9750-62 EBCDIC 8-bit French
Siemens 9750-62 EBCDIC 8-bit German
Siemens 9750-62 EBCDIC 8-bit Spanish
Siemens 97801/97808 7-bit Danish
Siemens 97801/97808 7-bit French
Siemens 97801/97808 7-bit German
Siemens 97801/97808 7-bit Italian
Siemens 97801/97808 7-bit Norwegian
Siemens 97801/97808 7-bit Spanish
Siemens 97801/97808 7-bit Swedish
Siemens EBCDIC.DF.04 8-bit East European *
Siemens EBCDIC.DF.04 8-bit West European
Siemens EBCDIC.DF.04.L5 8-bit West European/Turkish
Siemens EBCDIC.EHC.LC 8-bit Cyrillic
SOPS 32-bit Traditional Chinese
Taiwan Taxation 16-bit Traditional Chinese
Thai Industrial Standard 620-2533 - ASCII 8-bit †
Thai Industrial Standard 620-2533 - EBCDIC 8-bit
TRIS 32-bit Traditional Chinese
Unicode 3.0 UTF-8 * †
Unicode 3.0 EBCDIC UTF-8 *
VN3 8-bit Vietnamese
Windows95 16-bit PRC version Chinese
XBASIC Right-to-Left Arabic

The Power of Globalization
[Month] 2000
Author:
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2000 Oracle Corporation
All rights reserved.