MANUAL DE ESTÁNDARES DE ORACLE

Junta de Castilla y León

Coordinador de la Información: Área de Desarrollo y Mantenimiento
Dirección General de Telecomunicaciones y Transportes.

ÍNDICE DE CONTENIDOS

CAPÍTULO

1 Introducción

El presente **Manual de Estándares de Oracle** comprende las normas o estándares técnicos aplicables al diseño lógico, diseño físico y construcción de bases de datos y sistema de seguridad bajo el entorno del SGBDR ORACLE, y de aplicaciones bajo el entorno del SGBDR ORACLE y sus lenguajes (SQL, PL/SQL y SQL*Plus) y herramientas de diseño (Designer 6i) y de desarrollo (Oracle Forms y ORACLE Reports).

El contenido del manual ha de ir evolucionando al compás que marquen las necesidades de desarrollo de aplicaciones, el nivel de implantación de metodologías de desarrollo y la propia práctica de las diferentes Consejerías, por lo que su objetivo principal es el de ser el documento marco que en todo momento recoja de forma actualizada las normas que se consideren aplicables.

Se indica a continuación el contenido y objeto de los estándares, el personal responsable de cada una de las áreas en las que se organizan los estándares definidos y por último, la organización general y las normas tipográficas corrientemente utilizadas a lo largo del Manual.

Objeto de los estándares

El **objetivo** o propósito de la definición y aplicación de los Estándares de Oracle son los siguientes:

- Garantizar la aplicación de un nivel mínimo de calidad en los sistemas de información.
- Proporcionar un enfoque consistente y un lenguaje común en entornos con múltiples equipos de desarrollo.
- Conseguir que los componentes de los sistemas de información sean más fáciles de entender y mantener.
- Facilitar el proceso de introducción al nuevo personal de desarrollo.

Personal responsable

Es responsabilidad de cada uno de los miembros de los equipos de trabajo de desarrollo la aplicación consecuente de las normas contenidas en este Manual, así como de su mantenimiento, en función de las nuevas circunstancias o necesidades.

En especial, la correcta interpretación, aplicación y mantenimiento corresponde, según las áreas, a las siguientes personas:

Administrador de Bases de Datos

- Diseño Físico de Bases de Datos
  - Índices
  - Clusters
  - Bases de Datos
  - Espacios de Tablas
  - Segmentos de Rollback
  - Sinónimos

Analista

- Diseño Lógico de Bases de Datos
  - Tablas
  - Columnas
  - Restricciones de tablas
  - Vistas
  - Vistas Materializadas
  - Secuencias
- Diseño de Módulos de Aplicaciones
  - Módulos
  - Parámetros
  - Estructuras de datos
  - Disparadores
  - Uso de datos
  - Interfaces
- Sistema de Seguridad
  - Usuarios
  - Grupos y sus jerarquías
  - Privilegios
  - Auditoría

Programador

- Lenguaje SQL
- Lenguaje PL/SQL
- Lenguaje SQL*Plus
- Oracle Forms
- Oracle Reports

Organización del documento

El presente Manual de Estándares de Administración contiene los siguientes **capítulos**:

- Diseño Lógico de Bases de Datos
- Diseño Físico de Bases de Datos
- Sistema de Seguridad
- Diseño de Módulos de Aplicaciones
- Lenguaje SQL
- Lenguaje PL/SQL
- Lenguaje SQL*Plus
- Oracle Forms
- Oracle Reports

Cada capítulo se divide en **secciones** y éstos se subdividen en **apartados**, que recogen las **normas** o estándares aplicables.

A lo largo de este documento, "ORACLE" se refiere en general al SGBDR ORACLE o al software de la compañía del mismo nombre.

Normas de composición

Se detallan a continuación las **normas tipográficas** utilizadas en el Manual para presentación de contenidos:

MAYÚSCULA:	palabras clave o constantes
cursiva	referencia a documentos o conceptos ORACLE
<*cursiva*>	construcciones sintácticas
negrita	conceptos o aspectos importantes
`courier`	sentencias o componentes de sentencias SQL o PL/SQL
🕮	indica que la característica requerida no afecta a la sintaxis de creación de objetos, sino a su documentación

CAPÍTULO

2 Normas Generales

Los Estándares de Oracle no son reglas rígidas que han de aplicarse de forma inflexible. Cada una de ellas pretende un objetivo concreto y cuando este objetivo no se alcanza mediante la aplicación estricta de la norma, deberá aplicarse de la forma más coherente posible con la norma definida y su objetivo.

En todos los casos en los que se adopten decisiones distintas a las indicadas en las normas se ha de documentar los motivos o razones, así como examinar la posible necesidad de emitir una propuesta de cambio de la norma o normas aplicables.

La asignación de áreas de responsabilidad según el perfil del personal de desarrollo no ha de suponer la división estricta del trabajo de cada uno, sino que se ha de aplicar también la cooperación y la responsabilidad conjunta, a fin de asegurar en todo momento la mejora del entorno de desarrollo definido y la aplicación consecuente de las normas en vigor.

CAPÍTULO

3 Diseño Lógico de Bases de Datos

Se describen a continuación las normas aplicables para la definición de objetos de la base de datos que han de almacenar los datos de las aplicaciones. Contiene la descripción de normas específicas para el diseño de tablas, columnas de tablas, restricciones, vistas y secuencias.

Normas generales

Estimación de espacio Tras la definición de objetos que ocupan espacio físico en la base de datos se ha de proceder a la evaluación o estimación del espacio ocupado aplicando para ello los datos estimativos sobre el volumen de cada objeto.

Enumeración de elementos repetitivos En aquellos casos en los que se solicita la enumeración de elementos repetitivos y sea necesario numerar cada elemento de forma secuencial se deberán dar valores salteados (de dos en dos, de cinco en cinco) a fin de permitir inserciones posteriores sin necesidad de renumerar todos los elementos siguientes al elemento insertado.

Documentación de objetos En todos los casos en los que se determinan las características que deben cumplir las descripciones de objetos, estas características han de documentarse adecuadamente, bien en herramientas CASE o en formatos específicos.

Nombres de objetos

Características Para el nombre de los objetos de la base de datos (tablas, columnas, índices, etc.) deberán utilizarse términos simples, pero lo más descriptivos posible.

Se evitarán nombres de objetos numerados en serie u otros artificios que escondan su significado.

Abreviaturas Se utilizarán términos y abreviaturas tan cortos como sea posible, aunque esta regla no deberá comprometer o dificultar en ningún caso su legibilidad.

Se utilizarán nombres o abreviaturas idénticas cuando los significados son los mismos (por ejemplo, nombres de columnas de diferentes tablas que identifican códigos, nombres o descripciones) y semejantes cuando la funcionalidad es parecida.

Se deberán usar las abreviaturas estándar siempre que sea posible. Si no existe o no es conveniente crear una nueva abreviatura, se utilizará el término o términos que expresen de forma clara e inequívoca el significado.

Separador de nombres compuestos Para facilitar la legibilidad de los diferentes componentes de los nombres de los objetos se utilizará el carácter de subrayado ("_").

Caracteres especiales En los nombres de los objetos no se utilizarán espacios ni caracteres especiales tales como "$", "#" u otros que puedan tener sentido propio en contextos de desarrollo tales como bases de datos, lenguajes y sistemas operativos.

Ejemplos:

Incorrecto	Correcto
`GE_56_`	`GE_TP_EXP_56`
`GE_TIPOS_DE_EXPEDIENTES`	`GE_TP_EXPEDIENTES`
`GE_NUEVOS_EXPE`	`GE_EXP_NUEVOS`
`F_FECHA_ALTA_#`	`F_ALTA_NUM`

Alias de la aplicación 🕮 Como nombre abreviado de la aplicación podrán utilizarse alias largos (4 caracteres) o alias cortos (2 caracteres).

Si ya existe una aplicación en la que los dos primeros caracteres coincidan con los correspondientes a la nueva aplicación, el alias corto se formará mediante el primer y tercer carácter, o bien mediante el primer y cuarto carácter, si es necesario.

En lo que sigue, se utilizará el alias corto, salvo que se diga lo contrario. El alias de la aplicación se referencia como <*alias_aplicación*>.

Alias largo 4 caracteres

Ejemplo: `GNOM -> Gestión de Nóminas`

Alias corto 2 caracteres

Ejemplo: `GN -> Gestión de Nóminas`

Fichero global de creación de objetos

Todas las sentencias de creación de objetos de la base de datos (tablas, vistas, secuencias, etc.) deberán almacenarse en un fichero tipo SQL ejecutable bajo SQL*Plus.

El nombre de este fichero se compondrá de la siguiente manera:

<alias_aplicación>_CREAR.SQL

Ejemplo: `GE_CREAR.SQL`

Ficheros particulares de creación de objetos

Todas las sentencias de creación de objetos de la base de datos (tablas, vistas, secuencias, etc.) podrán almacenarse en ficheros particulares, cada uno de ellos referido a un tipo de objeto concreto, y se podrán almacenarse en un fichero tipo SQL ejecutable bajo SQL*Plus.

El nombre de este fichero se compondrá de la siguiente manera:

<alias_aplicación>_CREAR.TAB

<alias_aplicación>_CREAR.SYN

<alias_aplicación>_CREAR.CON

<alias_aplicación>_CREAR.RGR

<alias_aplicación>_CREAR.IND

<alias_aplicación>_CREAR.VW

Ejemplo: `GE_CREAR.TAB → Creación de las tablas de GE.`

Definición de tablas

Sintaxis

Nombre Tabla específica	*<alias_aplicación>_<nombre_tabla>* Ejemplo: `GE_EXPEDIENTES`	máximo 30 caracteres
Nombre Tabla genérica	COMU_< *nombre_tabla*> Ejemplo: `COMU_PROVINCIAS`	máximo 30 caracteres
Alias de la Tabla 🕮	*<alias_tabla>* Ejemplo: `EXP -> Expedientes`	3 caracteres

Descripción general

La descripción general de una tabla debe contener las siguientes características:

Nombre Nombre de la tabla.

Las tablas específicas estarán prefijadas con el nombre abreviado de la aplicación a la que pertenecen. Las tablas genéricas estarán prefijadas con el nombre de la aplicación COMU.

El nombre de la tabla deberá ser un nombre único dentro de cada aplicación. El nombre de las tablas estará en plural.

El nombre de las tablas corresponde al plural de las entidades de donde proceden (excepto en su prefijo).

En Designer, el nombre de la tabla se genera a partir del nombre Plural definido para la Entidad correspondiente.

Alias 🕮 Se ha de definir un alias para cada tabla, de una longitud de tres caracteres (en la medida de lo posible formado por los primeros caracteres del nombre de la tabla), que ha de ser único dentro de la aplicación.

Este alias será utilizado para componer nombres de restricciones, nombres de índices asociados a la tabla y como alias de tabla en sentencias SQL.

Ejemplo: `EXP`

Entidad de procedencia 🕮 Documentar para cada tabla la entidad o entidades de las que procede.

En caso de tablas no derivadas de entidades, documentar igualmente las razones de su creación.

Diario de operaciones 🕮 Indicativo de si se desea que la tabla disponga de una tabla adicional en la que se registren los cambios producidos a fin de mantener un registro o diario de seguridad.

Comentario 🕮 Documentar el objeto, uso o descripción de la tabla.

Parámetros de volumen

Cada tabla deberá llevar una indicación de su volumen inicial y final a fin de poder realizar con posterioridad el dimensionamiento correcto de la base de datos en la que han de residir.

Las indicaciones de volumen de tablas deberán especificar lo siguiente:

Volumen inicial 🕮 Indicar el número de filas estimadas que tendrá la tabla tras la carga inicial de datos.

Volumen final 🕮 Estimar el número final de filas que tendrá la tabla en condiciones normales de explotación.

Parámetros de almacenamiento

La definición completa de los parámetros de almacenamiento de las tablas deberá contener los siguientes aspectos:

Extensión inicial (*Initial extent*) El tamaño de la extensión inicial debe permitir la carga completa de datos inicial, de forma que no se debe crear ninguna extensión adicional durante esta carga inicial de datos.

Extensión siguiente (*Next extent*) Se determinará si la tabla necesita extensiones adicionales de tamaño progresivamente creciente.

En caso de ser necesario, el tamaño de la extensión inicial dividida por el tamaño de la extensión siguiente debe dar como resultado un número entero (doble, triple, etc.)

Número mínimo de extensiones (*Min extents*) Especificar siempre el valor de una extensión (1 extensión por defecto), que deberá tener una dimensión adecuada como mínimo, para alojar la carga inicial de datos.

Número máximo de extensiones (*Max extents*) El máximo número de extensiones de un segmento es siempre una función del tamaño del bloque. Dado que el tamaño del bloque ORACLE oscila entre 1 Kb (1024 bytes) a 8 Kb (8192 bytes) se aplicarán los siguientes valores posibles:

Tamaño del bloque	Número de extensiones
1 Kb	57
2 Kb	121
4 Kb	249
8 Kb	505
16 Kb	1022

No tomar como número máximo de extensiones el resultado exacto de las estimaciones, sino que se debe dejar un 25% adicional para cubrir posibles errores o cambios futuros.

Porcentaje de incremento (*Pct incr)* Se debe especificar siempre un porcentaje de incremento de la siguiente extensión nulo, ya que el uso de este porcentaje hace que se asignen extensiones de diferente tamaño cada vez, originando fragmentación en el espacio de tabla.

Espacio de tabla (*tablespace*) Asignar a cada tabla un espacio de tablas adecuado en función de la cuantía de los cambios que se realizan, diferenciando las tablas estáticas de las tablas dinámicas.

Transacciones iniciales Indicar el número máximo de transacciones que pueden actualizar un bloque de la tabla al mismo tiempo en los momentos iniciales. Oscila entre 1 y 255 (1 por defecto).

No se ha de asignar un número muy diferente del valor por defecto para transacciones iniciales a no ser que se prevea que la tabla vaya a ser actualizada por muchos usuarios a la vez sobre un número pequeño de filas.

Transacciones finales Indicar el número máximo de transacciones que pueden actualizar un bloque de la tabla al mismo tiempo en condiciones de explotación normal. Oscila entre 1 y 255.

No se ha de asignar un número muy diferente del valor por defecto [1] para transacciones finales a no ser que se prevea que la tabla vaya a ser actualizada por muchos usuarios a la vez sobre un número pequeño de filas.

Porcentaje libre (*Pct free*) Indicar el porcentaje de espacio sobre el total de tamaño del bloque que ha de dejarse libre de nuevas inserciones a fin de permitir espacio para las actualizaciones de las filas ya existentes sin que se tenga que recurrir a bloques encadenados de desborde. Por defecto, el valor tomado es 10 %.

Porcentaje usado (*Pct used*) Indicar el porcentaje de utilización del bloque necesario para volver a poder hacer nuevas inserciones en una tabla que no las permitía por haber llegado a ocupar su *Pct free.*

Secuencia de bloqueo Indicar el número correspondiente a la secuencia de bloqueo de la tabla en el momento de su actualización, si le corresponde.

Diario de operaciones Indicar si la tabla utilizará una tabla journal en la que queden registradas las operaciones realizadas con ella a fin de mantener un registro o diario de seguridad.

Ejemplo

```
CREATE TABLE      ge_expedientes
(c_numero         NUMBER
, ... ... ...
)
TABLESPACE        ge_tsd1
STORAGE           (INITIAL        50K
                   NEXT           50K
                   MINEXTENTS     1
                   MAXEXTENTS     5
                   PCTINCREASE    5
                   INITRANS       2
                   MAXTRANS       10);
```

[1] El valor por defecto depende del tamaño del bloque elegido en el momento de crear la base de datos; por ejemplo, para bloques de longitud 2048, el valor por defecto es de 255.

Definición de columnas

Sintaxis

Nombre de la Columna <*nombre_columna*> (máximo 30 caracteres)

Utilizar los siguientes abreviaturas para prefijar nombre de columnas de uso frecuente:

- C: Código
- D: Descripción
- I: Importe
- Q: Cantidad
- F: Fecha
- Y: Año
- H: Horas
- B: Valores booleanos (S/N)
- W: Frecuencia
- M: Longitud, superficie o volumen
- E: Estado
- L: Nivel de dependencia jerárquica
- N: Números que no sean cantidades ni importes
- A: Alfanumérico que no corresponda a ninguno de los casos anteriores
- P: Porcentaje
- X: Contenido de un fichero de texto.
- V: Contenido de un fichero de vídeo.
- S: Contenido de un fichero de sonido.
- G: Contenido de un fichero de imágenes gráficas.
- FIC: Contenido de un tipo no incluido en los anteriores.
- DIR: Objetos tipo directorio.
- OBJ: Objetos definidos como objetos Oracle.
- ARR: Objetos tipo varray.

Utilizar como sufijo ID, precedido del carácter "_" como indicativo de claves primarias o ajenas

Ejemplos **Columna no clave:** d_corta

Claves principales: c_empleado_id

Claves ajenas: c_poblacion_id

obj_coche: para un objeto que almacenase información relativa a vehículos (cada una de los componentes del objeto, serán nombrados como siempre, según su tipo de contenido.

x_doc_manual*:* para una columna que almacenase el contenido de un fichero de Word con un documento.

g_jpg_foto: para una columna que almacenase el contenido de un fichero JPG con una fotografía.

fic_prj_plan: para una columna que almacenase el contenido de un fichero PRJ (Proyect) de un proyecto.

fic_exe_juego**:** para una columna que almacenase el contenido de un fichero ejecutable con un juego

Descripción general

La descripción general de cada columna deberá contener las siguientes características:

Nombre Nombre de la columna.

El nombre de la columna deberá ser único dentro de la tabla. El nombre de la columnas se pondrá en singular.

Ejemplo: `A_NOM_EMPLEADO`

Las columnas que se derivan de atributos deben conservar, en la medida de lo posible, el nombre del atributo.

En lo posible, el nombre de las columnas que forman parte de la clave primaria de tablas que han de ser referenciadas por otras columnas deberá incluir el alias o referencia al nombre de la tabla.

Ejemplo:

En la tabla `GE_POBLACION`, la columna que soporta la clave primaria se denominará `C_POBLACION_ID`.

Las columnas que son claves ajenas deberán nombrarse en la medida de lo posible con el nombre de la clave primaria de la tabla referenciada.

Ejemplo: `C_TP_EXPEDIENTE_ID`

Secuencia 🕮 Numerar cada columna de forma que refleje el orden de su creación en la base de datos.

Utilizar la siguiente secuencia de ordenamiento según el tipo de columnas:

- columnas que forman parte de la clave primaria
- columnas que forman parte de la clave o claves únicas
- columnas obligatorias que forman parte de la clave o claves ajenas
- resto de columnas obligatorias
- resto de columnas opcionales

Formato Seleccionar el formato deseado.

Longitud media 🕮 Indicar la longitud media de la columna en número de posiciones. En caso de no poder estimar de antemano su longitud media utilizar las siguientes estimaciones:

- Columnas de tipo alfanuméricas

– si la longitud máxima de la columna es menor de 10, se tomará la longitud máxima.

– si es mayor de 10, la longitud media será el 50% de la longitud máxima.

- Columnas numéricas: (longitud / 2) + 1

Longitud máxima 🕮 Indicar la longitud máxima de la columna en número de posiciones. Se darán las longitudes necesarias en cada caso.

Decimales Definir para cada columna que utilice el tipo de dato *NUMBER* el número de posiciones decimales.

Opcionalidad Indicar si la columna es obligatoria u opcional.

Los siguientes tipos de columnas han de definirse como *NOT NULL*:

- columnas que forman parte de la clave primaria de la tabla.
- columnas que forman parte de una clave ajena obligatoria.
- columnas que toman un valor booleano. Estas columnas de tipo booleano deben ir asociadas a una restricción de tipo CHECK que fuerce a tomar uno de los valores posibles.
- cualquier columna que debe tomar un valor obligatoriamente.

En el resto de los casos, sólo se definirán como obligatorias las columnas que requieran siempre un valor.

Comentario 🕮 Documentar el objeto, uso o descripción de la columna, así como las notas asociadas al uso o aspectos técnicos de la columna.

Parámetros de volumen

Volumen inicial 🕮 Indicar el porcentaje de filas que tendrán algún valor en esta columna en el momento de la carga inicial.

Volumen final 🕮 Indicar el porcentaje de filas que tendrán algún valor en esta columna cuando la tabla esté en producción estable.

Requisitos de inserción de datos

Mayúsculas 🕮 Indicar si los valores insertados en estas columnas deberán ser convertidos a mayúsculas.

Valor por defecto Definir el valor por defecto que tomará la columna en el caso de inserciones en vacío, si ha de tomar alguno, sobre todo en el caso de columnas obligatorias.

Puede ser una constante o bien expresión como *ROWNUM*. Las funciones *SYSDATE, USER, USERENV y UID* también son aceptadas.

Secuencia utilizable 🕮 Indicar el nombre de la secuencia de la que se obtendrán valores para la columna en el momento de la inserción.

Se definirá una secuencia de tipo ORACLE o controlada por la aplicación para todos aquellas columnas que lo requieran. Se especificará si la secuencia se puede utilizar por otras columnas o debe ser usada exclusivamente por determinada columna.

Se estudiará la necesidad de crear secuencias para columnas en los siguientes casos:

- las columnas candidatas por clave primaria van a ser actualizadas con frecuencia.
- las columnas candidatas par clave primaria requieren un gran volumen de

espacio de almacenamiento.

- la tabla no tiene columnas que puedan ser candidatas a clave primaria.

Secuencia respecto al padre 🕮
Indicar si la columna ha de llevar una secuencia con respecto a la fila de la tabla maestra (por ejemplo, en la columna `N_LINEA` de la tabla `CO_LINEA_FACTURA`).

Columnas derivadas o calculadas 🕮
Se indicarán las reglas de derivación, en el caso de columnas calculadas derivadas o que almacenen acumulados o contadores.

Reglas de validación o tratamientos 🕮
Se indicarán las reglas de validación y de tratamiento lógico asociadas a la columna, así como los dominios asociados a la columna, tales como rangos y conjuntos de valores válidos.

Características de visualización

Se definirán cómo se visualizarán las columnas en pantallas e informes de las aplicaciones.

Visualizable 🕮
Indicar si columna deberá ser visualizada. Por ejemplo, no se visualizan las palabras de paso, ni ciertas columnas de uso interno de la aplicación.

Tipo de dato 🕮
Indicar el tipo de dato empleado para la visualización. Puede ser diferente del tipo de dato de almacenamiento.

Longitud 🕮
Indicar la longitud de la columna en pantallas e informes. Puede ser diferente del tipo de dato de almacenamiento.

Mayúscula o Minúscula 🕮
Indicar si se ha de visualizar en mayúscula o minúscula.

En general, los datos de texto se visualizarán en mayúscula, excepto los nombres propios, en los que se utilizarán los caracteres introducidos por el usuario.

Representar las fechas en mayúscula.

Alineación 🕮
Indicar si se ha de visualizar alineado a la derecha, a la izquierda o centrado.

Visualizar los números que representan importes en pesetas alineados a la derecha sin decimales; el resto, alineados a la derecha con dos decimales.

Altura 🕮
Indicar la altura de la columna en pantallas e informes.

Secuencia 🕮
Indicar la secuencia en la que aparece la columna en una pantalla o informe.

Formato 🕮
Indicar la máscara que modificará la forma de visualizar el dato.

Resalte 🕮
Especificar el tipo de resaltado o color a utilizar.

Descriptor 🕮
Especificar el descriptor que se usará para describir el campo de las pantallas en las que se utilice la columna.

Línea de ayuda 🕮
Definir una breve descripción del campo que aparecerá como ayuda en la última línea de las pantallas.

Ordenamiento 🕮
Especificar la secuencia de ordenamiento de la columna respecto a las demás cuando sea incluida en una cláusula ORDER BY.

En este caso se examinará la necesidad de definir un índice que facilite la presentación ordenada.

No se especificarán columnas en primera posición de la cláusula ORDER BY si la columna no aparece como primera en un índice.

Sentido del ordenamiento 🕮
Definir el sentido (ascendente o descendente) en que se deben ordenar los valores en una cláusula ORDER BY.

Descriptor de clave 🕮
A menudo, las filas de una tabla se identifican mediante una columnas que almacena un código al que va asociada una o más columnas de descripción.

Esta propiedad es un número entero que representa la prioridad de la columna cuando se usa como descriptor de un código. Las listas de valores utilizarán esta propiedad para visualizar la descripción en vez del (o además del) propio código.

Columnas de auditoría 🕮
Indica si se desea que la tabla tenga las siguientes columnas adicionales y cuyo valor de columna sea automáticamente generado por las aplicaciones:

- Creado por
- Modificado por
- Fecha de creación
- Fecha de modificación

Ayuda 🕮
Definir el texto para la ayuda de pantalla plena.

Restricciones de clave primaria [1]

Sintaxis

Nombre *<alias_aplicación>_<alias_ tabla>_PK*
Ejemplo: GE_EXP_PK

Descripción General

Nombre 🕮 Indicar un nombre único para la clave primaria.

En Designer, el nombre de la PK se genera a partir del nombre corto definido para la entidad.

Mensaje de error 🕮 Indicar el mensaje de error usado por las aplicaciones cuando se intente insertar o modificar una clave ya existente o nula.

Tabla de excepciones 🕮 Indicar la tabla donde se almacenarán todas las excepciones que se produzcan con la clave primaria en tiempo de ejecución.

Nivel de validación 🕮 Indicar el nivel de implementación de la clave primaria. Utilizar las opciones *Servidor, Cliente y Ambos.*

Activar Indicador de si la clave primaria se crea activada o desactivada.

Actualizable? 🕮 Indicador de si una vez insertada la clave primaria puede ser actualizada.

Columnas que la conforman Especifican las columnas que componen la clave primaria. Indicar el orden y el nombre de cada una de ellas.

Definir todas las columnas que forman parte de la clave primaria como *NOT NULL.*

Parámetros de almacena-miento Indicar los parámetros de almacenamiento del índice que implementará la clave primaria. Ver la sección **Parámetros de almacenamiento** en la sección relativa a tablas.

1 en inglés, primary constraint

Ejemplo

```
CREATE TABLE ge_empleados
(
,        c_empleado       NUMBER    (7)   NOT NULL
,        a_apellido       VARCHAR2 (35)   NOT NULL
,        a_nombre         VARCHAR2 (35)   NOT NULL
,        a_usuario        VARCHAR2  (8)   NOT NULL
,        n_sueldo         NUMBER   (15)   NOT NULL
,        c_depart         NUMBER    (7)   NOT NULL
,        CONSTRAINT       ge_emp_pk
                          PRIMARY KEY (c_empleado)
);
```

Restricciones de clave única [1]

Sintaxis

Nombre *<alias_aplicación>_<alias _tabla>_<columna>_UK*

donde *<columna>* puede contener el nombre completo o la abreviatura de la columna.

Ejemplo: `GE_EXP_C_DPTO_UK`

Descripción General

Nombre Indicar un nombre único para la clave única.

En Designer, el nombre de la UK se genera a partir del nombre corto definido para la entidad (*<alias_aplicación>_<alias _tabla>*) más el nombre dado al identificador único (*<columna>*).

Mensaje de error 🕮 Indicar el mensaje de error usado por las aplicaciones cuando se intente insertar o modificar una clave única ya existente.

Tabla de excepciones 🕮 Indicar la tabla donde se almacenarán todas las excepciones que se produzcan con la clave única en tiempo de ejecución.

Nivel de validación 🕮 Indicar el nivel de implementación de la clave única. Utilizar las opciones *Servidor, Cliente y Ambos.*

Activar Indicador de si la clave única se crea activada o desactivada.

Actualizable? 🕮 Indicador de si una vez insertada la clave única puede ser actualizada.

Columnas que la conforman Especifican las columnas que componen la clave única. Indicar el orden y el nombre de cada una de ellas.

Parámetros de almacenam. Indicar los parámetros de almacenamiento del índice que implementará la clave única. Ver la sección **Parámetros de almacenamiento** en la sección relativa a tablas.

1 en inglés, *unique constraint*

Ejemplo

```
CREATE TABLE ge_empleados
(
,         c_empleado       NUMBER    (7)    NOT NULL
,         a_apellido       VARCHAR2 (35)    NOT NULL
,         a_nombre         VARCHAR2 (35)    NOT NULL
,         a_usuario        VARCHAR2  (8)    NOT NULL
,         n_sueldo         NUMBER   (15)    NOT NULL
,         c_depart         NUMBER    (7)    NOT NULL
,         CONSTRAINT       ge_emp_usuario_uk
                           UNIQUE (a_usuario)

);
```

Restricciones de clave ajena [1]

Sintaxis

Nombre <alias_aplicación>_<alias _tabla>_<alias_aplicación>_<alias _tabla>_FK (si hubiera más de una con el mismo nombre añadir _1, _2, …
donde el *<alias_aplicación>_<alias_tabla>* hace referencia a la tabla que cubre la restricción. Si hay más de unas sobre la misma tabla se añade e sufijo *_1, _2*.

Ejemplo: `GE_EXP_GE_DPTO_FK`

Descripción General

Nombre Indicar un nombre único para la restricción de clave ajena.

En Designer, el nombre de la FK se genera a partir del nombre corto definido para ambas entidades.

Tabla referenciada El nombre de la tabla referenciada por la restricción.

Mensaje de error 🕮 Mensaje de error usado por las aplicaciones cuando se intenta introducir una clave ajena que no existe en la tabla referenciada.

Tabla de excepciones 🕮 Indicar la tabla donde se almacenarán todas las excepciones que se produzcan con la clave ajena en tiempo de ejecución.

Nivel de validación 🕮 Indicar el nivel de implementación de la clave ajena. Utilizar las opciones *Servidor, Cliente y Ambos.*

Activar Indicador de si la clave ajena se crea activada o desactivada.

Obligatoria 🕮 Indicar si es obligatorio o no introducir un valor para la clave ajena.

Transferible 🕮 Indicar si la clave ajena puede ser actualizada.

1 en inglés, *foreign constraint*

Reglas de borrado 🕮

Se deberán indicar las reglas de integridad referencial aplicables al borrar una fila de la tabla referenciada:

- Cascada: borrado en cascada de las filas de otras tablas que contienen el valor borrado.
- Restringido: impide el borrado si hay otras tablas que contienen el valor borrado
- Anular: permite el borrado y pone a nulo el valor en las tablas que contienen el valor borrado.

Reglas de actualización 🕮

Se deberán indicar las reglas de integridad referencial aplicables para actualizar la clave primaria de una tabla referenciada por otras tablas:

- Cascada: actualización en cascada de las filas de otras tablas que contienen el valor modificado.
- Restringido: impide la actualización si hay otras tablas que contienen el valor actualizable.
- Anular: permite la actualización y pone a nulo el valor en las tablas que contienen el valor modificado.

Columnas que la conforman

Especifican las columnas que componen la clave ajena. Indicar el orden y el nombre de cada una de ellas. Indicar también las columnas de la tabla referenciada con las que se corresponde.

Índice 🕮

Indicar si la clave ajena se apoyará en un índice.

Ejemplo

```
CREATE TABLE ge_empleados
(
,         c_empleado       NUMBER    (7)    NOT NULL
,         a_apellido       VARCHAR2 (35)    NOT NULL
,         a_nombre         VARCHAR2 (35)    NOT NULL
,         a_usuario        VARCHAR2  (8)    NOT NULL
,         n_sueldo         NUMBER   (15)    NOT NULL
,         c_depart_id      NUMBER    (7)    NOT NULL
,         CONSTRAINT       ge_emp_ge_departamentos_fk
          FOREIGN KEY      (c_depart_id)
          REFERENCES        ge_departamentos
                           (c_depart)

);
```

Restricciones de comprobación [1]

Sintaxis

Nombre *<alias_aplicación>_<alias_ tabla>_<columna>_CK*
Ejemplo: `GE_EXP_ESTADO_CK`

Descripción General

Nombre Nombre único de la restricción.

En Designer, las CK deben definirse una vez generadas las tablas.

Mensaje de error 🕮 Mensaje de error usado por las aplicaciones cuando se intenta introducir un valor que viola la restricción de comprobación.

Tabla de excepciones 🕮 Indicar la tabla donde se almacenarán todas las excepciones que se produzcan con la restricción de comprobación en tiempo de ejecución.

Nivel de validación 🕮 Indicar el nivel de implementación de la restricción. Utilizar las opciones *Servidor, Cliente y Ambos.*

Activar Indicador de si la restricción se crea activada o desactivada.

Texto de la restricción Indicar la condición o texto que conforma la restricción.

Ejemplo

```
CREATE TABLE ge_empleados
(
,         c_empleado       NUMBER    (7)    NOT NULL
,         a_apellido       VARCHAR2 (35)    NOT NULL
,         a_nombre         VARCHAR2 (35)    NOT NULL
,         a_usuario        VARCHAR2  (8)    NOT NULL
,         n_sueldo         NUMBER   (15)    NOT NULL
,         c_depart         NUMBER    (7) NOT NULL
,         CONSTRAINT       ge_emp_sueldo_ck
                           CHECK (n_sueldo > 0)
);
```

1 en inglés, *check constraint*

Definición de vistas

Sintaxis

Nombre <*alias_aplicación*>_V_<*alias_ vista*>
Ejemplo: `GE_V_EXP_ESTADO`

Descripción general

La descripción general de una vista debe contener las siguientes características:

Nombre Aplicar las normas definidas para los nombres de las tablas.

Alias 🕮 Aplicar las normas definidas para el alias de tablas.

Comentario 🕮 Documentar el objeto, uso o descripción de la tabla.

Definición de columnas

Aplican las mismas normas que para la definición de columnas de tablas.

Los nombres de la columnas de la vista han de ser idénticos a los nombres de las columnas de las tablas en las que se basan. En el caso de columnas que no basadas directamente en columnas de tablas aplican las normas de denominación de columnas de tablas.

Tablas base de la vista

Indicar las tablas y vistas en las que se basa la vista mediante las siguientes características:

Tabla o vista Tabla o vista en la que se basa la vista tratada.

Alias 🕮 Alias de la tabla o vista.

Columnas base de la vista

Indicar las columnas de las tablas que constituyen la vista mediante las siguientes características:

Nombre de la columna Nombre de la columna en la vista.

Columna base Nombre de la columna en la tabla o vista a la que pertenece.

Expresión SQL Una columna de una vista puede estar basada en una expresión en cuyo caso se reflejará en esta característica.

Ejemplo

```
CREATE VIEW ge_v_empleados
(
,         c_codigo          NUMBER     (7)    NOT NULL
,         a_apellido        VARCHAR2  (35)    NOT NULL
,         a_nombre          VARCHAR2  (35)    NOT NULL
)
AS SELECT c_empleado
,         a_apellido
,         a_nombre
FROM      ge_empleados
WHERE     d_tipo_puesto     = 'ADMINISTRATIVO'
WITH READ ONLY;
```

Definición de vistas materializadas

Sintaxis

Nombre <*alias_aplicación*>_VM_<*alias_ vista*>
Ejemplo: `GE_VM_EXP_ESTADO`

Descripción general

La descripción general de una vista materializada debe contener las siguientes características:

Nombre Aplicar las normas definidas para los nombres de las tablas.

Alias 🕮 Aplicar las normas definidas para el alias de tablas.

Comentario 🕮 Documentar el objeto, uso o descripción de la tabla.

Definición de columnas

Aplican las mismas normas que para la definición de columnas de tablas.

Los nombres de la columnas de la vista han de ser idénticos a los nombres de las columnas de las tablas en las que se basan. En el caso de columnas que no basadas directamente en columnas de tablas aplican las normas de denominación de columnas de tablas.

Tablas base de la vista

Indicar las tablas y vistas en las que se basa la vista mediante las siguientes características:

Tabla o vista Tabla o vista en la que se basa la vista tratada.

Alias 🕮 Alias de la tabla o vista.

Columnas base de la vista

Indicar las columnas de las tablas que constituyen la vista mediante las siguientes características:

Nombre de la columna Nombre de la columna en la vista.

Columna base Nombre de la columna en la tabla o vista a la que pertenece.

Expresión SQL Una columna de una vista puede estar basada en una expresión en cuyo caso se reflejará en esta característica.

Ejemplo

```
CREATE MATERIALIZED VIEW ge_vm_empleados
(
,         c_codigo          NUMBER    (7)   NOT NULL
,         a_apellido        VARCHAR2 (35)   NOT NULL
,         a_nombre          VARCHAR2 (35)   NOT NULL
)
AS SELECT c_empleado
,         a_apellido
,         a_nombre
FROM      ge_empleados
WHERE     d_tipo_puesto     = 'ADMINISTRATIVO';
```

Definición de secuencias

Sintaxis

Nombre *<alias_aplicación>_<alias_tabla>_<identificador>_SEQ*

El segmento <identificador> hará referencia al nombre de la columna o, en caso de utilizar una misma secuencia para varias columnas de diferentes tablas, a la funcionalidad que la identifique.
Ejemplo: `GE_EXP_REF_SEQ`

Descripción general

Se definirán secuencias siempre que una columna de una tabla requiera el almacenamiento de números enteros únicos generados automáticamente, por ejemplo, para generar valores para claves primarias.

La definición completa de una secuencia deberá contener los siguientes aspectos:

Incremento En general tomará un valor de uno.

Ciclo No utilizar secuencias cíclicas. En caso contrario, documentar los motivos e indicar los procedimientos para evitar conflictos con valores generados en el ciclo previo.

Orden No generar secuencias en su orden de petición.

Inicio Usar secuencias ascendentes.

Máximo, Mínimo No dar valores máximos o mínimos superiores a la longitud de almacenamiento del la columna o columnas que utilizarán la secuencia.

Valor cache Tomar el valor por defecto (20).

Método de implantación 🕮 En general, se utilizarán secuencias generadas por el SGBDR ORACLE. En caso de necesitar secuencias con valores consecutivos (por ejemplo, número de factura), se definirán secuencias controladas por la aplicación.

Comentario 🕮 Objeto o uso de la secuencia

Ejemplo

```
CREATE SEQUENCE ge_exp_ref_seq
INCREMENT BY      1
START WITH        1
NOCYCLE
NOCACHE
```

Resumen de sintaxis de objetos

Objeto	Sintaxis	Longitud
Alias de la aplicación	<alias_aplicación> Por defecto se utilizará el alias corto, excepto cuando se indique lo contrario.	largo: 4 car corto: 2 car
Fichero de creación de objetos	<alias_aplicación>_CREAR.SQL	
Ficheros particulares de creación de objetos	<alias_aplicación>_CREAR.TAB (tablas) <alias_aplicación>_CREAR.SYN (sinónimos) <alias_aplicación>_CREAR.CON (constraints) <alias_aplicación>_CREAR.RGR (roles a roles y a tablas) <alias_aplicación>_CREAR.IND (indices) <alias_aplicación>_CREAR.VW (vistas)	
Tabla específica	<alias_aplicación>_<nombre_tabla>	máximo 30 caracteres
Tabla genérica	COMU_<nombre_tabla>	máximo 30 caracteres
Alias de la Tabla	<Alias_tabla>	3 caracteres
Columna	<nombre_columna>	máximo 30 caracteres
Restricción de clave primaria	<alias_aplicación>_<alias_ tabla>_PK	
Restricción de clave única	<alias_aplicación>_<alias _tabla>_<columna>_UK	
Restricción de clave ajena	<alias_aplicación>_<alias _tabla>_<alias_aplicación>_<alias _tabla>_FK (si hubiera más de una con el mismo nombre añadir _1, _2, …	
Restricción de comprobación	<alias_aplicación>_<alias_ tabla>_<columna>_CK	
Vistas	<alias_aplicación>_V_<alias_ vista>	
Vistas Materializadas	<alias_aplicación>_VM_<alias_ vista>	
Secuencias	<alias_aplicación>_<alias_tabla>_<identificador>_SEQ	

Anexo: Abreviaturas de uso común

1	PRIMERO
2	SEGUNDO
ACC	ACCION
ACT	ACTIVIDAD
ACTA	ACTA
ACTL	ACTUAL
ACUM	ACUMULADO
ADM	ADMINISTRACION
AGT	AGENTE
ANT	ANTERIOR
ANTG	ANTIGÜEDAD
ANUL	ANULADO
APELL	APELLIDO
ARG	ARGUMENTO
AUT	AUTOR
BCO	BANCO
BD	BASE DE DATOS
BENEF	BENEFICIO
CAJ	CAJA
CAMP	CAMPO
CARAC	CARACTERISTICA
CLI	CLIENTE
C	CODIGO
COEF	COEFICIENTE
COLECT	COLECTIVO
COM	COMENTARIO
COMERC	COMERCIAL
COMIS	COMISION
COMP	COMPRA
COMUNIC	COMUNICACION
COND	CONDICION
CONS	CONSEJERÍA
CONT	CONTADOR
CONV	CONVERSION
CP	CODIGO POSTAL
CRTO	CONTRATO
CTA	CUENTA
DEF	DEFINICION
DENOM	DENOMINACION
DPTO	DEPARTAMENTO
DEPEND	DEPENDENCIA
D	DESCRIPCION
DEST	DESTINO
DEVOL	DEVOLUCION
DGRAL	DIRECCIÓN GENERAL
DIF	DIFERENCIA
DIP	DIPUTADO
DIR	DIRECCION
DOC	DOCUMENTO
DOMIC	DOMICILIO
DT	DATO
EJEC	EJECUCION
ELAB	ELABORACION
EMIS	EMISION
EMP	EMPLEADO
EMPR	EMPRESA
ENTI	ENTIDAD
ENTR	ENTRADA
ENTR	ENTREGA
ENV	ENVIO
EQU	EQUIPO
E	ESTADO
EVOL	EVOLUCION
EXPTE	EXPEDIENTE
EXT	EXTERNO
EXTN	EXTENSION
EXTR	EXTRAORDINARIO
FACT	FACTURA
FDO	FIRMADO
F	FECHA
FIN	FINAL
FINAL	FINALIDAD
FINAN	FINANCIERO
GEST	GESTION
GRP	GRUPO
HIST	HISTORICO
H	HORA
IDENT	IDENTIFICADOR
I	IMPORTE
IMPR	IMPRESIÓN
INCID	INCIDENCIA
INCR	INCREMENTO
IND	INDICE
INDIC	INDICADOR
INDIV	INDIVIDUAL
INF	INFERIOR
ING	INGRESO
INI	INICIAL
INSERC	INSERCION
INTERV	INTERVENCION
INV	INVERSION
LAB	LABORAL
LEG	LEGAL
LIM	LIMITE

LIQ	LIQUIDACION
LOC	LOCAL
LLAM	LLAMADA
MANT	MANTENIMIENTO
MATR	MATRICULA
MAX	MAXIMO
MAY	MAYOR
MED	MEDIO
MENS	MENSAJE
MENS	MENSUAL
MERC	MERCADO
MESA	MESA DEL CONGRESO
MIN	MINIMO
MKT	MARKETING
MOD	MODALIDAD
MOD	MODELO
MODIF	MODIFICACION
MOT	MOTIVO
MOV	MOVIMIENTO
MRG	MARGEN
MULT	MULTIPLE
MUNI	MUNICIPIO
NIV	NIVEL
NOM	NOMBRE
NOTIF	NOTIFICACION
NUE	NUEVO
N	NUMERO
OBJ	OBJETO
OBLI	OBLIGATORIO
OBSERV	OBSERVACION
OD	ORDEN DEL DIA
OFIC	OFICINA
OPER	OPERACION
ORD	ORDEN
ORG	ORGANIZACIÓN
ORI	ORIGEN
OTR	OTROS
PANT	PANTALLA
PARAM	PARAMETRO
PARC	PARCIAL
PART	PARTICULAR
PEND	PENDIENTE
PER	PERIODO
PERS	PERSONA
POBL	POBLACION
P	PORCENTAJE
POSIC	POSICION
PREF	PREFERENCIA
PRESUP	PRESUPUESTO
PREV	PREVIO
PROC	PROCEDENCIA
PROD	PRODUCTO
PROP	PROPIO
PROV	PROVINCIA
PROVR	PROVEEDOR
PROX	PROXIMO
PTAS	PESETAS
Q	CANTIDAD
REAL	REALIZACION
RECEP	RECEPCION
REF	REFERENCIA
REG	REGIMEN
REG	REGION
REL	RELACION
RENOV	RENOVACION
REP	REPRESENTACION
RESP	RESPUESTA
REV	REVISION
RGTRO	REGISTRO
SAREA	SUBAREA
SDO	SALDO
SECC	SECCION
SEC	SECUENCIA
SEG	SEGMENTO
SEM	SEMANA
SEN	SENADO
SERV	SERVICIO
SIST	SISTEMA
SIT	SITUACION
SOLIC	SOLICITUD
STIPO	SUBTIPO
STND	ESTANDAR
SUC	SUCURSAL
SUP	SUPERIOR
SUPERF	SUPERFICIE
TAB	TABLA
TARJ	TARJETA
TEL	TELEFONO
TEMP	TEMPORAL
TERM	TERMINAL
TIT	TITULAR
TIT	TITULO
TMP	TIEMPO
TOT	TOTAL
TP	TIPO
TRAB	TRABAJO
TRANS	TRANSACCION
TRANSF	TRANSFERENCIA
TRASP	TRASPASO
UBIC	UBICACION

ULT	ULTIMO	VOL	VOLUMEN
UNID	UNIDAD	VTO	VENCIMIENTO
USR	USUARIO	W	FRECUENCIA
VAL	VALOR	Y	AÑO

CAPÍTULO

4 Diseño Físico de Bases de Datos

Se definen a continuación las normas aplicables para la definición de objetos de la base de datos relacionados con su estructura física de almacenamiento y funcionamiento tales como índices, clusters, espacios de tablas, segmentos de rollback y sinónimos. Aparece como anexo la relación de abreviaturas de uso común para la denominación de objetos.

Entorno de Trabajo

Se pretende definir una serie de estándares para conseguir una Administración de Base de datos uniforme para toda la JCyL. Los aspectos a considerar son los siguientes:

Sistema Operativo

El sistema operativo soporte de la Base de Datos será Unix.

Versión de Base de Datos

Se adoptará como versión de base de datos Oracle la última versión que presente cierta estabilidad y permita la interacción con las bases de datos existentes.

Esta versión estará publicada en las páginas web del servicio de informática corporativa, al igual que para el resto de productos Oracle.

Cuentas de S.O.

oracle Se utilizará para la instalación del producto.

Pertenece al grupo “dba”.

Si existen diferentes máquinas, se utilizará el mismo UID y GID en todas las máquinas para evitar problemas de protecciones.

adminsis Se encarga de centralizar la ejecución de todos los procedimientos administrativos asociados a Unix, bases de datos y aplicaciones.

Pertenece al grupo “adminsis”, al “dba” y a los grupos de las distintas aplicaciones que convivan con el servidor.

operador Se encarga de realizar operaciones administrativas guiadas por menú a través de un procedimiento cautivo.

Estructura Física

La definición de la estructura de directorios y ficheros es como sigue:

Sintaxis

Estructura El esquema general de la estructura deberá ser como sigue:

- /<oracle>: Software Oracle
- /ENTORNO1: Estructura principal del ENTORNO1
- /ENTORNO2: Estructura principal del ENTORNO2
- /bak: Estructura de Respaldo

donde el entorno será el código identificativo para desarrollo, explotación,...

Ficheros El esquema general de nombrado dentro de la estructura deberá ser como sigue:

- BASE: Nombre de la base de datos.
- BASE<*_extensión fichero*>: Nombre del fichero.
- APLI: Nombre de la aplicación.
- ENTORNO: Nombre del Entorno

Descripción general

Se pretende que la estructura definida:

- Facilite la realización de copias de seguridad y su recuperación
- Agrupe lógicamente los registros seleccionados
- Sea adaptable a entornos diferentes a una máquina independiente: sistemas en cluster, parallel servers, etc.
- Facilite la recuperación hasta la última transacción a través de una estructura de respaldo.

Estructura Principal de Directorios

/ENTORNO/BASE/archive → /bak/BASE/archive
/audit
/comandos/crear_base
/replica
/snapshots
/scripts
/control/BASE_fc1.ctl
/BASE_fc2.ctl
/BASE_fc3.ctl → /bak/BASE/control/BASE_fc3.ctl
/copias → /bak/BASE/copias
(Copias de fichero de BD realizadas antes de ejecutar procesos críticos)
/core
/datos/BASE_APLI_tsd11.dbf
/BASE_syst_tsd11.dbf
/BASE_rbck_tsr11.dbf
/BASE_temp_tst11.dbf
/BASE_util_tsd11.dbf
/BASE_user_tsd11.dbf
(Tablespace por defecto para todos los usuarios)
/indices/BASE_APLI_tsi11.dbf
/BASE_util_tsi11.dbf
/export → /bak/BASE/export
(Ficheros de export generados en momentos puntuales)
/log
/redo/espejo → /bak/BASE/redo/espejo
/origen/BASE_frd1.log
/BASE_frd2.log
/…
/sid/initBASE.ora ←
/$ORACLE_HOME/dbs/initBASE.ora
/sort
/tmp
/trace/usuario
/sistema
/utl (Salidas de exportaciones a ficheros generadas con UTL)

En caso de que sea necesario repartir esta estructura en varios discos lógicos (discos “reales”, raidsets...), se mantendrá la misma nomenclatura para la estructura, cambiando únicamente el nombre del directorio raíz sobre el que se monta. Puede ser especialmente interesante, en caso de que existan varios discos, separar el directorio de índices para mejorar el rendimiento, o el de redo logs para obtener una mayor fiabilidad.

Estructura de Respaldo BD

Existe una única estructura de éste tipo para todas las bases de datos del servidor. Su finalidad es permitir la recuperación de la base de datos en el supuesto de corrupción o pérdida de la estructura principal.

/bak/BASE/archive/arch_BASE_log_xxxxx_x.arc→ (xxxx→%s x→ %t)
/control
/copias/archive
/control
/datos
/export
/redo
/sid
/redo/espejo/BASE_frd1.log
/BASE_frd2.log
/BASE_frd3.log
/…
/export
/bak/BASE/…

Procedimientos S.O.

Los procedimientos de s.o. de uso general se depositarán en la siguiente estructura de directorios:

/etc/oracomun/bin	(Procedimientos comunes)
/dat	(Ficheros de parametrización comunes)
/log	(Ficheros de log generados por scripts)
/err	(Ficheros de eventos detectados)
/ora	(Ficheros sqlnet, tnsnames, listener…)
/mai	(Ficheros de mail)
/tmp	(Ficheros temporales)

Los procedimientos de uso general que existirán en todas las instalaciones serán:

- subir y bajar bases de datos

- exports

- procedimientos de copia

- procedimientos de generación de estadísticas

- monitorización

- cálculo de espacio

- regeneración de un esquema

para procedimientos que no estén en esta lista, cuando surga la necesidad, se verá si alguien ya lo ha necesitado, si no, se desarrollará en SQL o PL/SQL y se notificará al resto de los administradores.

La necesidad de más procedimientos se verá en conjunción con la herramienta de Administración estándar PATROL

Definición de Bases de Datos

La definición completa de una base de datos deberá contener los siguientes aspectos:

Sintaxis

Nombre del Entorno

El nombre del entorno de trabajo deberá ser como sigue:

- Base de datos de desarrollo: DE
- Base de datos de desarrollo Web: DW
- Desarrollo paquetes cerrados: DQ
- Base de datos de explotación: EX
- Base de datos explotación Web: EW
- Explotación paquetes cerrados: EQ → Se incluyen aquí paquetes de terceros que no se adapten a los estándares definidos.
- Base de datos de preexplotación: PX → Se utilizará para la realización de pruebas de aceptación (usuario) y de carga (informático).
- Base de datos preexplotación Web: PW
- Preexplotación paquetes cerrados: PQ
- Administración: AD → Repositorios Oracle Enterprise Manager, PATROL, ...
- Herramientas CASE: CA → Oracle Designer 6i
- Formación: FO → Para apoyo a cursos
- Pruebas Software: PS

Nombre Base de Datos

El nombre de la base de datos se codificará como sigue:

EEAAAANN EE -> Entorno

AAAA -> Agrupación lógica que identifica la Consejería:

CPAT -> C. de Presidencia y Admón. Territorial
CEH_ -> C. de Economía y Hacienda
CAG_ -> C. de Agricultura y Ganadería
CF__ -> C. de Fomento
CSBS -> C. de Sanidad y Bienestar Social
CMA_ -> C. de Medio Ambiente

CEC_ -> C. de Educación y Cultura
CICT -> C. de Industria, Comercio y Turismo
GSS_ -> Gerencia de Servicios Sociales
JCYL -> Junta de Castilla y León
NN -> Número secuencial, comenzando por 01

Cuando se trate de bases de datos provincializadas, después del nombre de la consejería deberán seguir las 2 primeras letras del nombre de la provincia (AV,BU,...)

Descripción General

Reutilización del *control file*
Indicación de si el fichero de control se ha de reutilizar o no. A fin de evitar la pérdida errónea de bases de datos no se reutilizará ningún fichero de control.

Compartida (*Shared)*
Indicar si la base de datos ha de ser compartida por múltiples instancia a la vez o no. Indicar siempre que es compartida.

Archivado de *redo files*
Indicar que se desea copiar los *redo log files*. Se deberán mantener copias en espejo de cada uno de los archivos del grupo de *redo log files*.

Máximo número de *log files*
Indicar el número máximo (255 ficheros).

Máximo número de *data files*
Indicar el número máximo (depende de la versión instalada de Oracle8i, pero por defecto es de 30).

Máximo número de instancias
Indicar un número máximo de 255 instancias que pueden acceder a la vez a la base de datos.

Nodo 🕮
Equipo en el que se instala la base de datos.

Cadena de conexión (*connect string)*
Cadena de conexión utilizada para la conexión remota a la base de datos.

Comentario 🕮
Objeto o uso de la base de datos.

Características de los ficheros de bases de datos

La definición completa de los ficheros utilizados por cada base de datos deberá contener los siguientes aspectos:

Nombre
Nombre del fichero de sistema operativo. El nombre del fichero de base de datos se formará como sigue:

- fichero de datos: Ver apartado “Definición de Espacios de Tablas”,

"Ficheros"

- fichero de control: <*nombre_base_datos*>_*FCx*
- fichero de redo log: <*nombre_base_datos*>_*FRDx*

donde x es un número secuencial identificador de cada uno de ellos.

Ejemplo: `DES_FC1.ctl, DES_FRD1.dbf`

Base de datos 🕮 Nombre de la base de datos que lo utiliza.

Tamaño Tamaño del fichero

Unidad Utilizar kilobytes ("K") como unidad de tamaño del fichero.

Número de grupo Número de grupo para el *redo log file.*

Comentario 🕮 Objeto o uso del fichero.

Conexión a una base de datos remota

En el caso de que una base de datos deba conectarse con otra u otras bases de datos situadas en diferentes nodos se definirá un enlace [1] de base de datos Oracle, con las siguientes características:

Nombre Nombre del enlace de base de datos. Deberá componerse de la siguiente manera:

<nodo>_<usr>_DBL

donde *<nodo>* es una referencia al nodo al que se conecta y *<usr>* es una referencia al usuario de la base de datos remota con que se hace la conexión.

Public Indicar si el enlace es público o privado.

Usuario Indicar el usuario y la palabra de paso de conexión a la base de datos remota.

Cadena de conexión Identificar la cadena de conexión con la que se realiza el acceso a la base de datos remota.

Sinónimos de objetos remotos A fin de facilitar el acceso a los objetos remotos y ocultar la necesidad de incluir en el nombre de los objetos el nombre del enlace se deberán definir los correspondientes sinónimos.

En caso de resultar colisión de nombres con los objetos locales se añadirá el sufijo ´_R´.

Ejemplos

```
CREATE DATABASE LINK nodo_usr_dbl
   CONNECT TO     usuario_remoto
   IDENTIFIED BY      palabra_paso
   USING          ´t:unix5/1525:des´

CREATE SYNONYM   ge_empleados_r
   FOR usuario_remoto.ge_empleados@nodo_usr_dbl
```

[1] En inglés, *link.*

Definición de Espacios de Tablas

Se definirán como mínimo los siguientes espacios de tablas:

- uno para las tablas de cada aplicación
- uno para los índices de cada aplicación
- uno para las tablas de uso común
- uno para los índices de tablas de uso común
- uno para segmentos temporales
- uno para segmentos de rollback
- uno para uso por defecto de usuarios

El espacio de tablas por defecto para los usuarios se deberá calcular en función del número de usuarios que deberán utilizarlo.

No se deberá utilizar el espacio de tabla *SYSTEM* para ningún uso relacionado con las aplicaciones.

Sintaxis

Nombre Nombre del espacio de tabla. El nombre del espacio de tabla se formará como sigue:

- espacio de tablas de sistema : SYSTEM
- espacio de tablas de datos de aplicación: <*alias_aplicación*>*_TSDx*
- espacio de tablas de índices de aplicación: <*alias_aplicación*>*_TSIx*
- espacio de tablas de datos de uso común: *COMU_TSDx*
- espacio de tablas de índices de tablas de uso común: *COMU_TSIx*
- espacio de tablas de datos de uso común para una determinada Consejería: *COMU_AAAA_TSDx*
- espacio de tablas de índices de tablas de uso común para una determinada Consejería: *COMU_AAAA_TSIx*
- espacio de tablas temporal: *TEMP_TSTx*
- espacio de tablas de segmentos de rollback: *RBCK_TSRx*
- espacio de tablas por defecto de usuarios: *USER_TSDx*

donde X es un número secuencial identificador de cada uno de ellos, <*alias_aplicación*> el alias largo (cuatro caracteres) de la aplicación y AAAA -> Agrupación lógica que identifica la Consejería.

Descripción general

On line Indicación de si el espacio de tablas se ha de poner inmediatamente después de su creación en uso.

Read only Indicación de si el espacio de tabla no ha de permitir escritura de datos.

Base de datos 🕮 Nombre de la base de datos a la que pertenece el espacio de tabla.

Parámetros de almacena-miento Indicar los siguientes parámetros por defecto del espacio de tablas [1]:

- número de extensiones iniciales, máximas, mínimas.
- porcentaje de crecimiento de siguientes extensiones (*Pct incr*).

Ficheros Nombre del fichero o ficheros que utiliza el espacio de tabla. Se formará como sigue:

- Espacio de tabla SYSTEM: *<bd>_SYST_TSDx.DBF*
- espacio de tablas de datos de aplicación: *<bd>_<alias_aplicación>_TSDxy.DBF*
- espacio de tablas de índices de aplicación: *<bd>_<alias_aplicación>_TSIxy.DBF*
- espacio de tablas de datos de uso común: *<bd>_COMU_TSDxy.DBF*
- espacio de tablas de índices de tablas de uso común: *<bd>_COMU_TSIxy.DBF*
- espacio de tablas de datos de uso común para una determinada Consejería: *<bd>_COMU_AAAA_TSDx*
- espacio de tablas de índices de tablas de uso común para una determinada Consejería: *<bd>_COMU_AAAA_TSIx*
- espacio de tablas temporales: *<bd>_TSTx.DBF*
- espacio de tablas de segmentos de rollback: *<bd>_TSRx.DBF*
- espacio de tablas por defecto de usuarios: *<bd>_USERx.DBF*

donde:

- *<bd>*: es el nombre de la base de datos.

1 Estos parámetros aplican por defecto a los objetos asignados al espacio de almacenamiento.

- X: es un número secuencial identificador de los ficheros de cada espacio de tabla (de 1 a N).
- <*alias_aplicación*>: el alias largo (cuatro caracteres) de la aplicación.
- AAAA -> Agrupación lógica que identifica la Consejería

Comentario Objeto o uso de la base de datos.
🕮

Definición de Segmentos de Rollback

La definición completa de los segmentos de rollback deberá contener los siguientes aspectos:

Sintaxis

Nombre Nombre del segmento de rollback. El nombre del segmento de rollback se formará como sigue:

RBSx

donde "x" es un número secuencial identificativo de cada uno de los segmentos de rollback creados (1 a N).

Descripción general

Público Indicación de si el segmento de rollback es público o privado (disponible para todas las instancias de Oracle8i o no). No utilizar segmentos de rollback públicos.

Base de datos Nombre de la base de datos para la que se crea el segmento de rollback.

Espacio de tabla Nombre del espacio de tabla en el que se almacena el segmento de rollback. Los segmentos de rollback específicos para ser usados con una aplicación o proceso determinado que tenga requerimientos específicos de recuperación [1]deberán estar situados en un espacio de tabla separado.

Parámetros de almacena-miento Indicar los siguientes parámetros del segmento de rollback [2]:

- número de extensiones iniciales, máximas, mínimas (número mínimo es 2)
- porcentaje de crecimiento de siguientes extensiones (*Pct incr*)
- número correspondiente a *optimal*

1 Normalmente ligados a operaciones masivas de carga, borrado o actualización en una sola transacción.

2 Estos parámetros aplican por defecto a los objetos asignados al espacio de almacenamiento.

Los segmentos rollback de una misma aplicación o proceso determinado que tenga requerimientos específicos de recuperación [1] han de tener las mismas definiciones de almacenamiento.

Número de segmentos recomendado 🕮

En función del número de transacciones concurrentes que soporta la base de datos, el número de segmentos de rollback recomendado para evitar contención es el siguiente:

Número de transacciones (n)	Número de segmentos de rollback
n < 16	4
16 <= n < 32	8
32 <= n	n/4 (sin pasar de 50)

Optimal Especificar el tamaño óptimo del segmento de rollback en kilobytes (“K”).

Comentario 🕮 Objeto o uso del segmento de recuperación.

[1] Normalmente ligados a operaciones masivas de carga, borrado o actualización en una sola transacción.

Consideraciones SQL*Net

Se tendrá un único listener, de nombre LISTENER que escuchará por dos puertos 1521 y 1526

Definición de índices

Uso de índices

Se ha de examinar la conveniencia de crear un índice en lo siguientes casos:

- Sobre las columnas que integran cada clave ajena de una tabla a fin de facilitar las uniones ("joins").
- Sobre columnas que soporten consultas frecuentes o cláusulas WHERE complejas de sentencias SQL.
- En los casos en los que sea necesario la realización de un ordenamiento de filas de tablas frecuentemente.

Evitar la creación de índices sobre columnas que forman la clave primaria o que deban almacenar valores únicos, ya que ORACLE los crea automáticamente al definir una restricción de clave primaria o clave única.

Sintaxis

Claves ajenas *<nombre_restricción_clave_ajena>_FRGN*

Ejemplo: `GE_EXP_C_DPTO_FK_FRGN`

Otros usos *<alias_aplicación>_<alias_tabla><identificador>_INX*

donde *<identificador>* hace referencia a la funcionalidad que identifica al índice.

Ejemplo: `GE_EXP_DPTO_INX`

Descripción General

Tabla o cluster Nombre de la tabla o cluster sobre la que se construye el índice.

Columnas Nombre de las columnas que forman parte del índice y su secuencia dentro del mismo.

Valores únicos Determinar si admite valores únicos o duplicados.

Parámetros de almacena-miento Indicar los parámetros de almacenamiento del índice. Ver el apartado **Parámetros de almacenamiento** en la sección relativa a tablas.

Comentario 🕮 Objeto o uso del índice.

Definición de Clusters

Uso de clusters

Se ha de examinar la conveniencia de crear un cluster en todos aquellos casos en los que concurra alguna de las siguientes circunstancias:

- tablas en relación maestro-detalle
- tablas con claves primarias o claves ajenas en común

Sintaxis

Nombre Nombre del cluster. El nombre de los clusters se formará como sigue:

<alias_aplicación >_<identificador>_CLU

El segmento <*identificador*> hará referencia a la funcionalidad que le identifique.

Ejemplo: `GE_DPT_PERS_CLU`

Descripción General

Tabla Nombre de la tabla sobre la que se construye el cluster.

Columnas Nombre, tipo de dato y secuencia de cada columna del cluster y su relación con las tablas base.

Parámetros de almacena-miento Indicar los parámetros de almacenamiento del cluster. Ver la sección **Parámetros de almacenamiento** en la sección relativa a tablas.

Comentario 🕮 Objeto o uso del índice.

Índices de clusters

Para la definición de índices de clusters aplican las normas detalladas para la definición de índices de tablas.

Definición de Sinónimos

Se estudiará la oportunidad de definir sinónimos de tablas, vistas, instantáneas, secuencias, procesos o paquetes siempre que se den las siguientes condiciones:

- Se necesita enmascarar el nombre y usuario propietario de un objeto.
- Se necesita dar transparencia a objetos remotos en bases de datos distribuidas.
- Resulta conveniente para simplificar sentencias SQL.

Sintaxis

Nombre Nombre del sinónimo. No puede coincidir con el nombre de una entidad dentro de la misma aplicación.

Se seguirán las reglas de asignación de nombres correspondientes al objeto sobre el que se crea el sinónimo.

Descripción General

Objeto Indicar el objeto al que se hace referencia: tabla, vista, secuencia, instantánea o módulo.

Comentario 🕮 Objeto o uso del sinónimo.

Sinónimos de objetos remotos A fin de facilitar el acceso a los objetos remotos y ocultar la necesidad de incluir en el nombre de los objetos el nombre del enlace se deberán definir los correspondientes sinónimos.

En caso de resultar colisión de nombres con los objetos locales se añadirá el sufijo ´_R´.

Resumen de sintaxis de objetos

Objeto	Sintaxis
Base de Datos	EEAAAANN EE -> Entorno AAAA -> Agrupación lógica que identifica la Consejería
Ficheros de Bases de Datos	• fichero de datos: <bd>_<alias_aplicación>_TSDx.DBF • fichero de control: <nombre_base_datos>_FCx • fichero de redo log: <nombre_base_datos>_FRDx donde x es un número secuencial identificador de cada uno de ellos.
Espacios de tabla	• espacio de tabla SYSTEM: <bd>_SYST_TSDx.DBF • espacio de tablas de datos de aplicación: <alias_aplicación>_TSDx • espacio de tablas de índices de aplicación: <alias_aplicación>_TSIx • espacio de tablas de datos de uso común: COMU_TSDx • espacio de tablas de índices de tablas de uso común: COMU_TSIx • espacio de tablas de datos de uso común para una determinada Consejería: *COMU_AAAA_TSDx* • espacio de tablas de índices de tablas de uso común para una determinada Consejería: *COMU_AAAA_TSIx* • espacio de tablas temporales: TEMP_TSTx • espacio de tablas de segmentos de rollback: RBCK_TSRx • espacio de tablas por defecto de usuarios: USER_TSDx donde X es un número secuencial identificador de cada uno de ellos, <alias_aplicación> el alias largo (cuatro caracteres) de la aplicación y AAAA -> Agrupación lógica que identifica la Consejería
Ficheros de Espacios de tabla	• espacio de tablas de datos de aplicación: <bd>_<alias_aplicación>_TSDx.DBF • espacio de tablas de índices de aplicación: <bd>_<alias_aplicación>_TSIx.DBF • espacio de tablas de datos de uso común: <bd>_COMU_TSDx.DBF • espacio de tablas de índices de tablas de uso común: <bd>_COMU_TSIx.DBF • espacio de tablas de datos de uso común para una determinada Consejería: *<bd>_COMU_AAAA_TSDx* • espacio de tablas de índices de tablas de uso común para una determinada Consejería: *<bd>_COMU_AAAA_TSIx* • espacio de tablas temporales: <bd>_TSTx.DBF • espacio de tablas de segmentos de rollback: <bd>_TSRx.DBF

	• espacio de tablas por defecto de usuarios: <bd>_TSFx.DBF donde: <bd>: es el nombre de la base de datos X: es un número secuencial identificador de los ficheros de cada espacio de tabla (de 1 a N). <alias_aplicación>: el alias largo (cuatro caracteres) de la aplicación. AAAA -> Agrupación lógica que identifica la Consejería.
Segmentos de Rollback	RBSx donde "x" es un número secuencial identificativo de cada uno de los segmentos de rollback creados (1 a N).
Índices de Claves ajenas	<nombre_restricción_clave_ajena>_FRGN
Otros índices	<alias_aplicación>_<alias_tabla><identificador>_INX donde <identificador> hace referencia a la funcionalidad que identifica al índice.
Clusters de tablas	<alias_aplicación >_<identificador>_CLU
Sinónimos	Se seguirán las reglas de asignación de nombres correspondientes al objeto sobre el que se crea el sinónimo.
Enlaces de bases de datos	<nodo>_<usr>_DBL

CAPÍTULO

5 Sistema de Seguridad

Se detallan a continuación las normas aplicables a la definición de los mecanismos de seguridad de la base de datos y aplicaciones. A tal efecto se detalla la definición completa de usuarios, roles de usuarios, privilegios de usuarios y cómo organizar la auditoría de transacciones mediante los mecanismos de seguridad de ORACLE.

El acceso básico se conseguirá a través del nombre de usuario y una contraseña, siendo éstos autentificados de forma única mediante el servidor de base de datos Oracle. La contraseña deberá ser alfanumérica y con un número mínimo de caracteres que se determinará.

Definición de Usuarios

Definición de usuarios ORACLE

La definición completa de usuarios de la base de datos deberá contener los siguientes aspectos:

Nombre Nombre del usuario. Será único para identificar al usuario ante ORACLE. Sólo deberá contener caracteres definidos en el conjunto de caracteres de la propia base de datos.

Los nombres de usuarios se compondrá de la siguiente forma:

<xxx><yyy><zz>

donde

<xxx> Tres primeras letras del primer apellido

<yyy> Tres primeras letras del segundo apellido

<zz> *Dos primeras letras del nombre*

Dado que pueden existir coincidencias a la hora de construir un código de usuario utilizando esta estructura, para dar de alta un usuario el CAU, el jefe de servicio correspondiente notificará dicha necesidad mediante solicitud al CAU, remitiendo obligatoriamente los siguientes datos:

Consejería de <x...x> → Se completará con el nombre completo de la misma

Dirección General de <x ... x> → Omitir "títulos" innecesarios.

Servicio de <x ... x> → Omitir "títulos" innecesarios.

Sección de <x ... x> → Omitir "títulos" innecesarios.

Cargo → Omitir "apellidos" innecesarios. Incluirá la terminación "/A" cuando su denominación admita masculino y femenino.

Nombre → Según DNI, sin abreviaturas.

Primer Apellido → Según DNI.

Segundo Apellido → Según DNI.

Despacho → Opcional.

Teléfono → 9 dígitos, sin separadores, no móviles. En caso de estar asignado a una centralita se pondrá "CENTRALITA" en el campo CARGO.

IMPORTANTE para usuarios de servicios periféricos. Dichos usuarios tendrán además que cumplir las siguientes Normas adicionales:

Dirección General → Siempre "SECRETARIA GENERAL"

Servicio → Si Consejería= "CPAT" → "DELEGACION TERRITORIAL DE XXX"

→ Si Consejería= "CSBS" → "GERENCIA TERRITORIAL DE SERVICIOS SOCIALES DE XXX"

→ Si Consejería= "CICT" y así corresponda → "OFICINA TERRITORIAL DE TRABAJO DE XXX"

→ Si Consejería= "CICT" y así corresponda → "GERENCIA PROVINCIAL DE LA ADE EN XXX"

→ Resto de Consejeríasy Unidades Periféricas → "SERVICIO TERRITORIAL DE XXX"

Siendo XXX la provincia correspondiente.

Password Se asignará a cada usuario una palabra de paso inicial. Obligatoria para todos los usuarios.

Se comprobará al cabo de un cierto tiempo que el usuario ha modificado la palabra de paso.

Espacio de tabla por defecto Se asignará un espacio de tabla por defecto a cada usuario, incluso para los que no tengan privilegios de creación de objetos (ver normas de creación de espacios de tablas en el apartado correspondiente).

Espacio de tabla temporal Se asignará obligatoriamente un espacio de tabla por defecto a cada usuario(ver normas de creación de espacios de tablas en el apartado correspondiente).

Perfil Se asignará un perfil a cada usuario de la base de datos, evitando la aplicación del perfil DEFAULT.

Comentario 🕮 Objeto o uso asignado al usuario.

Ejemplos

- Usuario de aplicaciones

```
CREATE USER            usuario
   IDENTIFY BY         palabra_de_paso
   DEFAULT TABLESPACE  tsu2
   PROFILE             administrativo

CREATE USER            usuario
   IDENTIFY            EXTERNALLY
   DEFAULT TABLESPACE  tsu2
```

- Usuario de desarrollo

```
CREATE USER              usuario
   IDENTIFY BY           palabra_de_paso
   DEFAULT TABLESPACE    tsu1
   TEMPORARY TABLESPACE  tst1
```

Login profile de usuario

Se tendrán en cuenta las siguientes normas:

Existirá un login profile para desarrollo y otro para explotación.

Deberá contener definiciones de las variables de entorno (ORACLE_SID, ORACLE_HOME, PATH, TERM) así como las variables que definan la estructura de directorios.

Definición de Perfiles

Un perfil es un conjunto de límites de recursos a los que se les asigna un nombre. Los perfiles pueden ser asignados a diferentes usuarios par controlar y limitar el uso de los recursos del sistema.

La definición de perfiles deberá contemplar los siguientes aspectos:

Nombre Se indicará el nombre del perfil creado.

Sesión por usuario Se indicará el número máximo de sesiones que puede abrir un usuario.

CPU por sesión Se expresará el máximo de centésimas de segundo de procesador por sesión de un usuario.

CPU por llamada Se expresará el máximo de centésimas de segundo de procesador por llamada.

Tiempo de Conexión Se indicará el número máximo de minutos de duración de una sesión.

Tiempo ocioso Se indicará el número máximo de minutos de tiempo ocioso de una sesión.

Lecturas por sesión Se indicará el número máximo de lecturas de bloques de datos por sesión.

Lecturas por llamadas Se indicará el número máximo de lecturas de bloques de datos por cada sentencia de SQL.

Límite compuesto Se expresará el coste total de recursos por sesión.

SGA privada Se expresará el número máximo de octetos de espacio privado de SGA por sesión.

Ejemplo

```
CREATE PROFILE profile01 LIMIT
      SESSION_PER_USER  UNLIMITED
      CPU_PER_SESSION   UNLIMITED
      CPU_PER_CALL      3000
      CONNECT_TIME      45
      LOGICAL_READS_PER_SESSION   DEFAULT
      LOGICAL_READS_PER_CALL      1000
      PRIVATE_SGA       15K
      COMPOSITE LIMIT   5000000
```

Definición de Roles

Un rol agrupa una serie de privilegios (y en caso de necesidad, otros roles), que pueden ser asignados o revocados simultáneamente a los usuarios.

Se crearán los siguientes roles:

- Base de datos de desarrollo

  Se creará un rol de desarrolladores para uso de analistas y programadores. Este rol permitirá la creación y borrado de tablas dentro del espacio de tablas asignado a su aplicación. Este rol será creado por el administrador (ver normas aplicables a espacios de tablas).

  Este rol tendrá los privilegios de RESOURCE Y CONNECT, así como un rol con privilegios de objetos.

- Base de datos de explotación

  El administrador será el responsable de la creación y borrado de objetos en esta base de datos. Tendrá asignado un rol específico de administrador.

- Cada aplicación dispondrá de tres roles:
  - Usuario responsable o propietario del esquema

    Es el usuario que realiza la entrada de datos. Tendrá privilegios de conexión, creación y borrado de tablas de trabajo de la aplicación y los privilegios necesarios para el acceso a los objetos de la aplicación.
  - Usuario consultas

    Similar al anterior, solo que los privilegios sobre los objetos diferirán, en función de las necesidades de consulta.
  - Responsable informático

    Tendrá privilegios de conexión y de exportación e importación plena de la base de datos. No se le asignará la opción de administración, a fin de que no pueda conceder sus privilegios a otros usuarios.

En ningún caso se utilizará la opción WHITH ADMIN OPTION.

La definición de roles de usuarios de la base de datos deberá contener los siguientes aspectos:

Nombre Nombre del rol. El nombre del rol se formará como sigue:

<alias_aplicación>_<funcionalidad>

El nombre del grupo tendrá un máximo de 30 caracteres de longitud.

Ejemplo: `nomi_consulta`

Password No se asignarán palabras de paso a los roles creados.

Comentario Indicar el objeto o motivo del grupo definido.

Ejemplo

```
CREATE   ROLE contable IDENTIFIED BY palabra_de_paso
```

Asignación de Privilegios

Los usuarios y roles definidos pueden tener una serie de privilegios sobre el sistema o sobre los objetos de la base de datos.

El administrador asignará a los usuarios los roles necesarios, en función de las aplicaciones que utilicen y su relación con ellas (responsable, consultas, informático).

Para la asignación de usuarios o grupos se tendrán en cuenta los siguientes aspectos:

Asignación de objetos de la base de datos

Destinatario Nombre o nombres de roles y de usuarios a los que se les conceden los privilegios.

Objeto Nombre del objeto de la base de datos (tabla, vista, secuencia o instantánea)

Columna Si este es el caso, especificar la columna de tabla o vista a la que aplica el privilegio concedido.

Tipo de privilegios Indicar los privilegios concedidos al grupo sobre dicho objeto (select, update, delete, index, alter, reference, etc.).

Tener en cuenta los privilegios realmente necesitados, ya que en un entorno de producción de una aplicación , no todos los usuarios necesitan todos los privilegios de acceso, en especial, no necesitarán los privilegios de index, alter, reference o grant option.

Grant option Especificar si el rol o usuario puede conceder a su vez los privilegios a otros usuarios.

Comentario 🕮 Indicar el objeto o motivo de la asignación..

Ejemplo

```
GRANT SELECT ON ge_poblacion TO PUBLIC
```

Asignación de privilegios del sistema

Destino Indicar si el privilegio se concede a un grupo o a un usuario y su nombre.

Ámbito Indicar si el privilegio se ha concedido sólo a un usuario, un rol o se extiende también al grupo *PUBLIC*

Tipo de privilegios Indicar los privilegios concedidos y si se conceden o no con la opción de administración.

Alcance Indicar si el grupo o usuario al que se concede el privilegio tiene a su vez autoridad de conceder a otros grupos dependientes o usuarios los privilegios concedidos.

Comentario 🕮 Indicar el objeto o motivo de la asignación.

Ejemplo

```
GRANT directivo TO direccion WITH ADMIN OPTION
```

Borrado de usuarios

En el caso de que un usuario deje de ser necesario y tenga objetos asociados en la base de datos, en vez de proceder a su borrado se le inhabilitará mediante el desactivado de la opción CREATE SESSION.

Auditoría de Transacciones

Las normas aplicables para la especificación de necesidades de auditoría son las siguientes:

Auditoría de accesos al sistema

Estudiar la necesidad de registrar los intentos de acceso a la base de datos en función de las características de los usuarios y la privacidad de los datos.

Describir las opciones de auditoría requeridas; en especial:

Opción de sentencias Indicar las sentencias específicas que se desean auditar.

Privilegios de sistema Indicar los privilegios de sistema específicos que se han de auditar para las sentencias SQL.

Usuarios Indicar, si es preciso, a qué usuario o usuarios aplica la auditoría.

Sesión o acceso Indicar si la auditoría se establece por sesión o por cada sentencia SQL.

Cuándo Indicar si se desea registrar las transacciones que tuvieron éxito o las que fracasaron.

Ejemplo:

```
AUDIT SELECT TABLE, UPDATE TABLE BY usuario
```

Auditoría de accesos a los objetos de la base de datos

Determinar las tablas, vistas y secuencias en las que se requiere activar la auditoría y en aquéllas que lo requieran, qué sentencias SQL deben ser auditadas y si la auditoría ha de hacerse al nivel de sesión o de acceso.

Describir las opciones de auditoría requeridas; en especial:

Opción Indicar si se desea registrar los intentos a nivel ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE o ALL.

Objeto Nombre del objeto al que se le aplica la auditoría.

Forma Indicar si se requiere una auditoría BY SESSION o BY ACCESS.

Cuándo Indicar si se desea registrar las transacciones que tuvieron éxito o que fracasaron.

Ejemplo:

```
AUDIT INSERT, UPDATE
ON ge_poblaciones
WHENEVER NOT SUCCESSFUL
```

Consideraciones para la auditoría

Consideraciones a tener en cuenta:

- Mover la marca de auditoría fuera del Tablespace del Sistema: Al tiempo que nuevas grabaciones se insertan en la marca de auditoría de la base de datos, la tabla AUD$ puede crecer sin límite. Dado que la tabla AUD$ crece y después empequeñece, debería ser almacenada fuera del tablespace del sistema.
- Monitorizar el crecimiento del registro de auditoría: Si la tabla de auditoría se llena, no se pueden insertar más grabaciones de auditoría y las sentencias auditadas no se ejecutarán con éxito. Se devolverán errores a todos los usuarios que realicen una sentencia auditada. Se necesitará liberar espacio en la marca de auditoría antes de que las sentencias puedan ser ejcutadas.
- Proteger la tabla de auditoría: Se debe proteger la tabla de auditoría de forma que la información auditada no pueda ser añadida, modificada o eliminada. Sólo el DBA debiera tener el rol DELETE_CATALOG_ROLE.

CAPÍTULO

6 Diseño de Módulos

En este capítulo se detallan las normas aplicables a la definición de los diferentes tipos de módulos de las aplicaciones, así como las características de elementos concretos de los módulos, tales como parámetros, estructuras de datos (aplicable a PL/SQL), disparadores y finalmente, las normas para la definición de interfaces entre sistemas.

Entendemos por módulo cualquier unidad ejecutable de forma independiente desde el entorno ORACLE, bien sea desde el gestor de bases de datos relacionales o desde las herramientas de desarrollo de aplicaciones (ORACLE Forms y ORACLE Reports).

Los módulos pueden ser de los siguientes tipos:

Pantalla Unidad ejecutable para la introducción, modificación y consulta de datos de forma interactiva.

Se compone de objetos (tales como ventanas, campos de texto, cajas de comprobación, botones, alertas, listas de valores) y de rutinas codificadas (tales como disparadores, o bloques de código PL/SQL).

Informe Unidad ejecutable que permite la salida de información almacenada en la base de datos en dispositivos tales como impresoras o pantallas.

Menú Unidad ejecutable que presenta al usuario de forma interactiva un conjunto de opciones de ejecución. Cada opción ha de llamar a un módulo determinado.

Disparador Es un bloque de PL/SQL que se ejecuta en respuesta a eventos específicos producidos en el momento de la ejecución de una aplicación.

Procedimiento Grupo de sentencias PL/SQL con un nombre definido que ejecutan una funcionalidad determinada

Función Grupo de sentencias PL/SQL con un nombre definido que ejecutan una funcionalidad determinada y que devuelven un valor al finalizar su ejecución.

Paquete Conjunto encapsulado de procedimientos, funciones y otros objetos relacionados entre sí almacenados en la base de datos.

Librería Conjunto de paquetes, procedimientos, y funciones reutilizables y referenciables almacenados en fichero o base de datos.

Nota: Todas las características reseñadas corresponden a documentación, por lo que se omite el símbolo 🕮

Normas generales

Con carácter previo a la definición en detalle de cada módulo se realizará el diseño jerárquico de la aplicación, definiendo su estructura modular, que será reflejada en un diagrama apropiado (diagrama de estructura de módulos).

La definición general de un módulo de cualquier tipo debe cubrir como mínimo los siguientes aspectos:

Sintaxis

Nombre abreviado

Nombre abreviado del módulo. Deberá ser único dentro de todas las aplicaciones.

- Módulos de pantallas, informes y menús

  Deberán tener una longitud fija de 8 caracteres a fin de respetar la limitación impuesta por MS WINDOWS.

  El nombre del módulo se formará como sigue:

  <alias_aplicación><alias_módulo>

  donde *<alias_aplicación>*corresponde al alias corto (dos caracteres) y el *<alias_módulo>* es el nombre abreviado del módulo (seis caracteres), que hará referencia, dentro de la limitación de caracteres disponibles, a su funcionalidad. [1]

  Ejemplo: `GEMREPT`, correspondiente a la aplicación GESP (nombre corto, GE), módulo de mantenimiento de la relación de puestos de trabajo).

  En los casos en los que el número de módulos haga inviable la aplicación de la norma anterior el nombre del módulo se formará como sigue

  <alias_aplicación>_x

  donde *<alias_aplicación>* es el nombre largo (4 caracteres) y el indicador x es un número secuencial (con el fin de que sea único) de 3 posiciones, completándose las posiciones sobrantes con ceros delante de x.

  Ejemplo: GESP_015.

  El nombre abreviado se utilizará como nombre del fichero fuente y ejecutable, al que se añadirán la extensión propia de la herramienta con la que se haga el desarrollo.

- Módulos de PL/SQL

  El nombre abreviado se formará mediante la siguiente sintaxis:

[1] Ver las normas de denominación de aplicaciones en el capítulo “Diseño Lógico de Bases de Datos”, Normas Generales.

<alias_aplicación><alias_módulo>

donde *<alias_aplicación>* corresponde al alias corto, *<alias_módulo>* tendrá la longitud que se considere necesaria para su correcta identificación, pudiendo incluir subtramos separados por el carácter de subrayado "_".

Ejemplo: `GECOMPRO_LENIF`, es decir, aplicación `GE`, módulo de comprobación de la letra del NIF.

Descripción General

Nombre completo Nombre completo del módulo. Deberá ser único dentro de todas las aplicaciones.

Utilizar una descripción acorde con la funcionalidad del módulo, sin caracteres especiales de ningún tipo.

En los módulos de tipo pantalla, informe o menú, este nombre se utilizará en las pantallas de acceso y manuales de usuario de la aplicación. Por lo mismo, no será obligatorio en los módulos de tipo PL/SQL.

Objeto Objeto o descripción de la funcionalidad del módulo.

Lenguaje Lenguaje en el que está escrito el módulo. Especificar Forms, Reports o PL/SQL u otro lenguaje

El uso de un lenguaje o herramienta 4GL no habitual debe ser objeto de autorización previa justificación de su necesidad.

Tipo Tipo de módulo. Utilizar la siguiente clasificación de módulos:

P	Pantalla	módulos interactivos
I	Informe	informes y listados
M	Menú	menús de la aplicación
D	Disparador	disparadores de tablas
X	Procedimiento	procedimientos
F	Función	funciones
Q	Paquete	paquetes
L	Librería	librerías

Módulos tipo cliente (Forms y Reports)

En el caso de módulos de tipo Cliente (ejecución en el lado cliente de una arquitectura cliente-servidor) se especificará adicionalmente lo siguiente:

Formato Se especificará el formato normalizado previamente definido, si corresponde al objeto y tipo de módulo.

Utilizar los siguientes formatos:

Formato	Pantallas	Informes
Registro	X	X
Tabla	X	X
Maestro-detalle	X	X
Ruptura de bloque		X
Matriz		X
Etiquetas		X

Ubicación La ubicación de los módulos, tanto de los fuentes como de los ejecutables debe sujetarse a las normas de configuración de aplicaciones.

Comando de ejecución Definir los indicadores que han de acompañar al comando de ejecución del módulo.

Título de menú Debe consistir en una o un máximo de tres palabras descriptivas de la funcionalidad del módulo. Se ha de determinar la letra (preferiblemente la inicial) que se ha de utilizar como selector de la opción en los menús.

Se indicará la opción de menú de la que será llamado.

Descripción Debe contener una descripción de la funcionalidad del módulo en términos inteligibles por los usuarios. Este texto se utilizará para componer el Manual de Usuario.

Puede contener referencias a cálculos o validaciones realizadas, siempre y cuando se describan desde el punto de vista del usuario.

Notas Debe contener la descripción técnica del módulo. Este texto se utilizará para componer el Manual Técnico de la aplicación.

Módulos tipo servidor (PL/SQL)

En el caso de módulos de tipo Servidor (ejecución en el lado servidor de una arquitectura cliente-servidor), es decir, módulos de tipo PL/SQL se especificará adicionalmente lo siguiente:

Base de datos Path completo de la base de datos que almacena el módulo.

Ubicación La ubicación del fichero fuente. Debe sujetarse a las normas de configuración de aplicaciones.

Ámbito Puede ser de tipo Public o Private. Sólo en el caso de funciones o paquetes.

Tipo de dato de retorno Tipo de datos retornado por el módulo. Sólo para módulos de tipo función.

Notas Debe contener la descripción técnica del módulo. Este texto se utilizará para componer el Manual Técnico de la aplicación.

Definición de Parámetros

Los parámetros son datos suministrados en la interfaz del módulo, tanto de entrada como de salida (por ejemplo, parámetros de lanzamiento de un informe o datos pasados entre módulos en el momento de su llamada o retorno).

La definición de los parámetros un módulo debe cubrir como mínimo los siguientes aspectos :

Sintaxis

Nombre Especifica el nombre completo del parámetro.

Ver en el capítulo “Uso del PL/SQL” las normas de denominación de los parámetros de un bloque PL/SQL.

Descripción General

Secuencia Especifica la posición del parámetro dentro de la línea de comando. Especificar siempre la secuencia en la que se utilizarán los parámetros.

Tipo de dato Tipo de dato del parámetro. Si se especifica un tipo de datos no se definirá una estructura de datos.

Longitud Máximo número de caracteres que puede tener el parámetro. Definir sólo si se ha especificado un tipo de datos.

Decimales Número de posiciones decimales que puede tener un parámetro. Definir sólo si se ha especificado un tipo de datos.

Estructura de datos Estructura de datos que usa el parámetro en lugar de un tipo de datos. Si se especifica una estructura de datos no se definirá un tipo de datos.

Operador Operador que debe aplicarse al parámetro (=, >, =>, LIKE, etc.). Sólo aplica en parámetros de entrada de informes o llamadas entre pantallas.

Valor por defecto Valor por defecto del parámetro. Sólo aplica en parámetros de entrada de informes o llamadas entre pantallas.

Descriptor Texto que puede ser usado para solicitar un valor para este parámetro de forma interactiva. Sólo aplica en parámetros de entrada de informes.

Delimitador Carácter delimitador entre el nombre del parámetro y su valor.

Variable de sustitución Nombre de la variable de sustitución que ha de ser usada para incluir en el cuerpo del módulo el valor real del parámetro en tiempo de ejecución.

Sentido Indica el tipo de parámetro: de entrada o de salida o ambas cosas.

Descripción Breve comentario sobre el parámetro.

Definición de Datos de Programa

Se deberán especificar las variables, constantes y las excepciones de un módulo, es decir, los datos del programa.

Sintaxis

Nombre Especifica el nombre completo del objeto.

Ver en el capítulo “Uso del PL/SQL” las normas de denominación de los objetos internos de un bloque PL/SQL.

Descripción General

Tipo Tipo de la definición: Constante, Variable, Excepción.

Tipo de dato Tipo de dato del objeto.

Longitud Máximo número de caracteres que puede tener el objeto.

Obligato-riedad Indica si el dato es opcional u obligatorio.

Decimales Número de posiciones decimales que puede tener el objeto.

Valor por defecto Valor por defecto del objeto.

Ámbito Privilegios de la estructura de datos del programa.

En los paquetes, las estructuras de datos de programa de la especificación del paquete deberán ser PUBLIC y los del cuerpo del paquete PRIVATE.

Comentarios Breve descripción de la estructura de datos del programa.

Definición de Estructuras de Datos

Se deberán especificar las estructuras complejas de datos que se usan en los módulos PL/SQL, tales como registros y tablas.

Sintaxis

Nombre Nombre completo de la estructura.

Ver en el capítulo “Uso del PL/SQL” las normas de denominación de los registros y tablas de un bloque PL/SQL.

Descripción General

Tipo Tipo de la estructura: registro, tabla.

Campos Nombres de cada campo de la estructura.

Tipos de dato Tipos de dato de cada campo.

Longitud Máximo número de caracteres que puede tener el campo.

Decimales Número de posiciones decimales que puede tener el campo.

Tabla Nombre de la tabla que define el campo de la estructura.

Columna Nombre de la columna que define el campo de la estructura.

Valor por defecto Valor por defecto del objeto.

Comentarios Breve descripción de la estructura de datos del programa.

Definición de Disparadores

Para los módulos en los que se definen los disparadores asociados a tablas se definirán las siguientes características adicionales:

Sintaxis

Nombre Nombre único identificador del disparador. El nombre del disparador se formará como sigue:

TR_<alias_aplicación>_<tabla>_<funcionalidad>

donde:
<alias_aplicación>: el alias largo (cuatro caracteres) de la aplicación.

<tabla>: tabla a la que afecta el disparador. En caso de afectar a más de una se elimina.

<funcionalidad>: funcionalidad del disparador.

Descripción general

Nombre del módulo Nombre del módulo asociado al disparador.

Comentarios Breve descripción de la estructura de datos del programa.

Momento de ejecución Se indicará cuándo se ejecuta el disparador: antes o después de la sentencia del disparador

Nivel Indicar si el disparador es de tipo fila o sentencia.

Acción asociada Indicar si el disparador está asociado a una operación de tipo inserción, borrado o actualización.

Alias Alias del antiguo y nuevo valor

Condición When Texto con la condición *WHEN* aplicable al disparador.

Cuerpo Cuerpo de la acción en PL/SQL ejecutable cuando el disparador se ejecute.

Uso de Datos en Módulos

Se definirán las tablas utilizadas en cada módulo de pantalla o informe y las características específicas de su uso que difieran de las ya definidas con carácter general para la tabla.

En particular, se definirán para cada tabla los siguientes aspectos:

- Posición del bloque dentro de la página o ventana
- Tipo de ventana (normal, desplegable)
- Dimensiones y coordenadas de la página o ventana

Para cada tabla utilizada en un módulo de pantalla o informe se definirán las columnas utilizadas y las características específicas de su uso que difieran de las ya definidas con carácter general para cada columna.

En particular, se definirán para cada columna los siguientes aspectos:

- Posición de la columna y características de visualización (dimensiones, color, resalte, etc.)
- Uso de datos (inserción, actualización, borrado, anulación)
- Objetos de interfaz gráfica de usuario asociados (botón, grupo radio, caja check, caja combo, etc.)

Interfaces con otros sistemas

Minimizar el uso del comando *HOST* o cualquier otro mecanismo de acceso al sistema operativo desde los módulos de SQL*Plus, Forms y otros.

En caso de ser necesario, examinar la posibilidad de aislar el comando *HOST* en un módulo que lo contenga, haciendo que el módulo o módulos que necesiten la llamada *HOST* llamen al módulo que ejecuta el comando.

En todos estos casos hay que registrar en un apartado de la documentación técnica cuales son los módulos que utilizan llamadas de este tipo.

Definir en función de las necesidades de la aplicación el método de llamada a otros módulos (SQL*Plus, Forms, Reports, etc.) , especificando los siguientes aspectos:

- paso de parámetros
- ejecución on-line o batch
- paso de códigos de error

Asignación a grupos de derechos de acceso a módulos

Para cada módulo definido se ha de especificar el nivel de seguridad aplicado, indican los siguientes datos:

Rol Nombre del rol al que se le concede derechos de acceso a un módulo.

Módulo Nombre del módulo al que se puede acceder.

Alcance Indicar si el grupo al que se concede el privilegio tiene a su vez autoridad de conceder a otros usuarios los privilegios de acceso al módulo concedidos.

Comentario Indicar el objeto o motivo de la asignación.

Resumen de sintaxis de objetos

Objeto	Sintaxis	Longitud
Alias de la aplicación	<alias_aplicación>	largo: 4 car corto: 2 car
Módulos	• Módulos de pantallas, informes y menús <alias_aplicación><alias_módulo> <alias_aplicación>_x • Módulos de PL/SQL <alias_aplicación>_<alias_módulo>	8 caracteres <2><6> <4>_<3> <2><6>_<...>
Parámetros	PR_X_<nombre_parámetro>	PR_X_<.....>
Datos de programa	T_X_<nombre_objeto> T = {V, G, C, E}	T_X_<.....>
Datos de programa (Indices)	I_<nombre_objeto>	I_<.....>
Estructuras de datos	<nombre>	<.....>
Disparadores de tablas	TR_<alias_aplicación>_<tabla>_<funcionalidad>	TR_<4>_<…>_<…>

CAPÍTULO

7 Uso del Lenguaje SQL

El presente capítulo muestra las diversas normas aplicables, tanto para la escritura de sentencias del lenguaje SQL como recomendaciones importantes de uso de sus diferentes posibilidades y cláusulas.

Introducción

Definición del SQL

El Structured Query Language (SQL) es un conjunto de comandos o sentencias que pueden ser utilizados desde las aplicaciones para acceder a la información contenida en una base de datos ORACLE8i.

Tipos de sentencias SQL

Se distinguen los siguientes tipos de sentencias SQL:

- DML: Data Manipulation Language

  Permiten la consulta y manipulación de los datos almacenados en los objetos de la base de datos. Por ejemplo, `DELETE, INSERT, SELECT, UPDATE.`

- DDL: Data Definition Language

  Permiten la creación, modificación y borrado de objetos, la concesión y revocación de privilegios y roles, el análisis de la información sobre las tablas, índices y clusters, la definición de las opciones de auditoría y la definición de comentarios a los objetos del diccionario de datos. Por ejemplo, `ALTER DATABASE, CREATE TABLE, DROP SEQUENCE, GRANT.`

- Transaction Control Commands

  Gestionan los cambios hechos mediante los comandos DML. Por ejemplo, `COMMIT, ROLLBACK.`

- Session Control Commands

  Cambian las propiedades de una sesión de usuario de forma dinámica. Por ejemplo, `ALTER SESSION, SET ROLE.`

- System Control Commands

  Gestiona las propiedades de una instancia ORACLE8i de forma dinámica. Existe un solo comando de este tipo: `ALTER SYSTEM.`

- Embedded SQL Commands

  Sirven para utilizar los comandos de tipo DDL, DML y Transaction Control Commands dentro de programas realizados con lenguajes procedurales. Por ejemplo, `DECLARE CURSOR, EXECUTE, OPEN, FECHT.`

Dónde se usan

Los comandos SQL se pueden utilizar en:

- Programas realizados mediante lenguajes procedurales, por ejemplo, COBOL.
- Bloques de programas realizados con el lenguaje PL/SQL, lenguaje procedural de ORACLE8i
- Dentro de SQL*Plus, bien como ficheros de tipo SQL o mediante la línea de comandos de esta herramienta.

Importancia de aplicar las normas tipográficas

En el apartado 3 se definen las normas tipográficas que han de aplicarse en la redacción de las sentencias SQL. La importancia de seguir estas normas radica en los siguientes motivos:

- Legibilidad

  La aplicación de las normas garantiza una mayor legibilidad de las sentencias, lo que redunda en que su comprensión es más rápida.

- Portabilidad

  Ciertas normas definidas garantizan la portabilidad con futuras versiones del lenguaje SQL.

- Eficacia

  Se han definido ciertas normas que redundan en una mayor eficacia a la hora de ejecución de la sentencia.

Otros aspectos relevantes

El origen de este lenguaje se sitúa en el artículo “A relational Model of Data for large Shared Data Banks”, de E. F. Codd, publicado en junio de 1970 en la revista de la ACM (Association of Computer Machinery). El modelo de Codd se acepta actualmente como el modelo definitivo para la gestión de sistemas de bases de datos relacionales.

El lenguaje SEQUEL (Structured English Query Language) fue desarrollado por IBM Corporation, Inc. para usar el modelo de Codd. Posteriormente se denominó SQL.

En 1979, Relational Software, Inc. (actualmente, Oracle Corporation) introdujo la primera versión comercial del SQL.

Actualmente, el lenguaje SQL está aceptado como un estándar para la gestión de bases de datos relacionales.

El SQL de ORACLE8i cumple con los estándares aceptados por la industria informática. El estándar último publicado por ANSI (American National Standard Institute) e ISO (International Standards Organization).

El último estándar de SQL publicado por ANSI e ISO se denomina comúnmente SQL3.

Normas tipográficas

Uso de mayúsculas

Se utilizarán las mayúsculas para escribir las palabras clave tales como SELECT, SUM, mientras que se escribirán en minúsculas los nombres de tablas, columnas, alias y otras referencias a objetos de usuario.

Ejemplo:

```
SELECT  COUNT(*)        q_total_empleados
FROM    ge_empleados    emp
WHERE   emp.a_oficio = 'VENDEDOR'
;
```

Esta regla no se aplica a las cadenas de caracteres o fechas especificadas entre comillas, tales como 'ORACLE' o '23-ENE-96', contexto sensible al uso de mayúsculas.

Creación y sangrado de párrafos

Cada vez que se use una palabra clave de una nueva cláusula se empezará una nueva línea y desplazará el texto restante hasta el siguiente espacio de tabulación.

Si la cláusula es tan larga que no cabe en una línea de pantalla, insertar un salto de línea al inicio de una expresión o similar y sangrar las siguientes líneas de la cláusula.

Cada uno de los elementos de una lista de tablas, columnas o valores ha de situarse en una nueva línea.

Colocar al inicio de la línea siguiente:

- la coma que separa cada elemento de una lista
- el paréntesis que cierra una subconsulta o una lista
- el punto y coma de finalización de una sentencia SQL

Dividir una lista de elementos concatenados en varias líneas, como si se tratara de una lista de elementos.

Se escribirán con sangrado a la izquierda los componentes dependientes de la sentencia (tales como el alias de una columna o tabla).

Espacios en blanco

Se colocarán espacios o separadores antes y después de cada nombre de columna, (excepto cuando lleva un paréntesis inicial a la izquierda) y nombres de objetos o palabras reservadas.

Colocar un espacio en blanco o separador antes y después de los operadores (como el de la suma) y nombres de función.

Ejemplos:

```
SELECT  MAX (emp.i_sueldo +
             (emp.i_sueldo *
```

```
            NVL (emp.p_comis, 0) / 100 ))
,       maximo_sueldo
FROM    ge_empleados    emp
WHERE   emp.a_oficio = 'VENDEDOR'
;

INSERT INTO    ge_departamentos
    (   c_departamento
    ,   d_nombre_dpto
    ,   a_ubicacion
    )
VALUES
    (   30
    ,   'COMERCIAL'
    ,   'SEVILLA'
    )
;
```

Normas de Construcción

Uso de alias

Alias de tablas Definir un alias por cada tabla de la cláusula FROM, incluso si la consulta recupera filas de una única tabla. El alias debe ser el definido en la descripción de la tabla.

Si la tabla pertenece a otro esquema o base de datos, se deberá renombrarla mediante un sinónimo.

Si se formula una unión reflexiva [1], definir un alias de tabla para cada uno de los usos de la tabla.

Cuando se formule una subconsulta que implique el uso de una tabla que también se utiliza en la SELECT principal, definir un alias diferente.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.i_sueldo
FROM    ge_empleados  emp
WHERE   emp.i_sueldo  <=
        (SELECT AVG (emp_med.i_sueldo)
         FROM   ge_empleados    emp_med
        )
;
```

Alias de columnas Prefijar cada columna con el alias de la tabla definido en la cláusula FROM.

Ejemplo:

```
SELECT  reg.c_codigo
,       reg.a_nombre
FROM    ge_regiones       reg
;
```

Emplear un alias de columna en las líneas calculadas de una sentencia SELECT.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo * 12  sueldo_anual
FROM    ge_empleados       emp
;
```

como alternativa a:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo * 12
FROM    ge_empleados        emp
;
```

1 en inglés, *self-joint.*

Especificar un alias de columna para cada cálculo de grupo de una sentencia SELECT.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       AVG (emp.i_sueldo)      sueldo_medio
FROM    ge_empleados            emp
GROUP BY emp.a_nombre
,       emp.c_dpto
;
```

como alternativa a:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       AVG (emp.i_sueldo)
FROM    ge_empleados             emp
GROUP BY  emp.a_nombre
,         emp.c_dpto
;
```

Si dos o mas columnas de tablas diferentes tienen el mismo nombre, definir un alias para cada una de ellas.

Ejemplo:

```
SELECT  emp.a_nombre
,       dir.a_nombre
FROM    ge_empleados     emp
        ge_empleados     dir
WHERE   emp.c_director = dir.c_empleado
;
```

Comentarios

En sentencias SQL complejas deberán insertarse comentarios para aclarar en sentido de la sentencia en su conjunto o aspectos individuales

Escribir los comentarios entre las marcas "/* " y " */" (insertar un espacio después y antes de cada marca, respectivamente) o bien, si el comentario cabe en una sola línea hacerlo preceder por "-- ".

Ejemplo:

```
SELECT  emp.a_nombre /* interesa sólo nombre, sin
        apellido */
,       emp.i_sueldo
FROM    ge_empleados  emp
WHERE   emp.a_nombre LIKE 'S%' -- con índice
;
```

También se pueden utilizar comentarios para deshabilitar partes de la sentencia como medio de depurarla.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.i_sueldo
FROM    ge_empleados         emp
/* WHERE emp.a_nombre LIKE 'S%'*/
;
```

Expresiones simples

Se utilizarán las siguientes reglas par la escritura de expresiones en una cláusula WHERE:

=, <>, >, <, >=, <= Utilizar estos operadores matemáticos siempre que sea posible.

<>, !=, ^= Escoger el símbolo "<>" para designar la desigualdad y utilizarlo consistentemente.

Cláusula **IN** Escribir cada valor de la lista de comparación en una línea distinta, a fin de facilitar su lectura y mantenimiento.

Ejemplo:

```
SELECT  ped.n_numero
,       ped.i_importe
FROM    co_pedidos       ped
WHERE   ped.n_numero IN
        (      100
        ,      102
        ,      106
        ,      110
        ,      125
        )
;
```

Cláusula **OR** Escribir las expresiones OR entre paréntesis.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo
FROM    ge_empleados          emp
WHERE   (emp.c_dpto = 10
    OR  emp.c_dpto = 20)
;
```

Expresiones complejas

Se agruparán las expresiones complejas mediante el uso de paréntesis, incluso cuando la sintaxis del lenguaje no lo requiera a fin de facilitar su entendimiento y evitar errores.

En especial, especificar mediante el uso de paréntesis la precedencia de las operaciones aritméticas (suma, resta, multiplicación y división) y de los operadores lógicos (AND, OR y NOT)

Ejemplo:

```
SELECT  ((lin.n_precio * lin.q_cantidad)
          / ped.i_total) * 100 lin_porc
FROM    co_pedidos           ped
,       co_lineas_pedidos    lin
WHERE   ped.n_numero = lin.n_num_ped
    AND ped.n_numero = 100
;

SELECT  emp.a_nombre
,       emp.i_sueldo
,       emp.p_comis
,       emp.i_sueldo
FROM    ge_empleados   emp
WHERE   emp.c_dpto = 42
    OR (emp.c_dpto = 44
    AND emp.i_sueldo >= 1000000
         )
;
```

Referencias a columnas

Evitar la notación SELECT * y listar cada columna explícitamente, a fin de independizar la sentencia de cambios en la estructura de la tabla.

Utilizar excepcionalmente la cláusula SELECT * sólo en el caso de uso de la función COUNT o dentro de un cursor PL/SQL que recupere datos de una fila en una variable de registro (*rowtype*).

Cálculos de grupos

Copiar exactamente las expresiones de grupo de la cláusula SELECT en la cláusula GROUP BY para evitar errores sintácticos.

No se deben incluir expresiones de agrupación en la cláusula GROUP BY que no estén en la cláusula SELECT, para evitar resultados confusos.

Convertir mediante la función NVL el posible resultado NULL de un cálculo de grupo (cuando no se devuelven filas en la consulta) en un valor que tenga sentido.

Ejemplo:

```
SELECT  ped.n_numero
```

```
,       NVL (MAX (lpd.n_num_linea), 0) + 1
        sig_num_lin
FROM    co_pedidos          ped
,       co_lin_pedidos      lpd
WHERE   lpd.n_num_pedido = ped.n_numero
GROUP BY ped.n_numero
;
```

Ordenaciones

No fiarse en las ordenaciones implícitas realizadas como resultado del empleo de índices o de la cláusula GROUP BY porque las reglas de ordenación implícita pueden cambiar en las nuevas versiones de ORACLE.

Utilizar los nombres de las columnas en lugar de los números de orden de la cláusula SELECT siempre que se pueda. Es una forma de evitar problemas de ordenación si se modifica la cláusula SELECT. Utilizar las referencias numéricas de columnas en la cláusula ORDER BY sólo cuando se componga una sentencia SELECT compuesta con los operadores UNION, INTERSECT o MINUS.

Ejemplo:

```
SELECT  emp.f_contratacion
,       emp.a_nombre
FROM    ge_empleados            emp
ORDER BY  emp.f_contratacion    DESC
,         emp.a_nombre          ASC
;
```

como alternativa a:

```
SELECT  emp.f_contratacion
,       emp.a_nombre
FROM    ge_empleados          emp
ORDER BY  1    DESC
,         2
;
```

Consultas de múltiples tablas (uniones) [1]

Se habrá de listar las tablas de la cláusula FROM en un orden determinado. En general, se listarán las tablas según su tamaño, primero las tablas que devuelven mayor número de filas o primero las de detalle y después las maestras.

Ejemplo:

```
SELECT  reg.a_nombre          nombre_region
,       pro.a_nombre          nombre_producto
,       ped.n_numero          numero_pedido
FROM    co_productos          pro
,       co_lin_pedidos        lpd
,       co_pedidos            ped
,       co_clientes           cli
,       co_regiones           reg
```

1 en inglés, *joins*.

```
WHERE   pro.c_codigo     = lpd.c_pro_codigo
    AND lpd.n_num_ped    = ped.n_numero
    AND ped.c_cliente    = cli.c_codigo
    AND cli.c_region     = reg.c_codigo
    AND pro.a_nombre     = 'Bango'
    AND reg.a_nombre     = 'Europa'
;
```

Escribir las condiciones de la unión antes de cualquier otra condición de la cláusula WHERE. Listar las condiciones de la unión en el orden en que aparecen las tablas en la cláusula FROM.

Situar las condiciones de una unión externa [1] en el nivel más alto de la cláusula WHERE. Evitar en ellos el uso de operadores lógicos diferentes de AND, ya que las uniones externas con el operador OR son muy confusas.

Ejemplo:

```
SELECT  emp.a_nombre      nombre_empleado
,       dpt.a_nombre      nombre_dpto
FROM    ge_empleados      emp
,       ge_departamentos  dpt
WHERE   emp.c_dpto    =   dpt.c_codigo (+)
    AND dpt.c_codigo  IS NULL
UNION
SELECT  emp.a_nombre      nombre_empleado
,       dpt.a_nombre      nombre_dpto
FROM    ge_empleados      emp
,       ge_departamentos  dpt
WHERE   emp.c_dpto (+) =  dpt.c_codigo
    AND emp.c_dpto     IS NULL
;
```

Subconsultas

Para la escritura de subconsultas de tablas se utilizarán las siguientes normas:

Sangrar cada subconsulta e iniciar nueva línea.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.i_sueldo
FROM    ge_empleados  emp
WHERE   emp.i_sueldo <=
        (SELECT   AVG (emp_med.i_sueldo)
         FROM     ge_empleados     emp_med
        )
;
```

Sentencias INSERT

1 en inglés, *outer joint.*

Especificar la lista completa de columnas que se van a insertar, a fin de independizar el orden de definición de columnas en la base de datos de su orden en la sentencia.

Especificar una lista de valores completa en la cláusula VALUES o una relación completa de columnas a insertar en la construcción SELECT del INSERT.

Insertar valores nulos explícitos mediante la palabra reservada NULL, a fin de distinguir la inserción de valores nulos de posibles olvidos.

Cada nombre de columna de la cláusula INSERT o cada valor de VALUES debe ser escrito en una línea separada, a fin de facilitar la lectura y el mantenimiento.

Ejemplo:

```
INSERT INTO ge_departamentos
   (   c_codigo
   ,   a_nombre
   ,   c_region
   )
   VALUES
   (   31
   ,   'Comercial'
   ,   NULL
   )
;
```

Sentencias UPDATE o DELETE

Si se modifican o borran todas las filas de una tabla se ha de incluir un comentario que indique que la cláusula WHERE se omite, par distinguirlo de un posible olvido.

Ejemplo:

```
DELETE
FROM    ge_empleados  emp    /* cláusula WHERE
                           omitida */
;
```

Colocar cada modificación de la cláusula SET de la sentencia UPDATE en una línea distinta, par facilitar la lectura y edición.

Ejemplo:

```
UPDATE  ge_empleados  emp
SET     c_empledo     = 60
,       emp.i_sueldo  = 2550
WHERE   emp.c_codigo  = 1
;
```

Recomendaciones de optimización

Expresiones simples

Se utilizarán las siguientes reglas para la escritura de expresiones en una cláusula WHERE en función de su optimización en tiempo de ejecución:

Cláusula BETWEEN

Convertir este operador en dos comparaciones que utilicen operadores en los límites de la comparación, si la formulación mediante el operador BETWEEN resulta compleja de entender.

Ejemplo:

```
SELECT  ped.n_numero
,       ped.f_pedido
,       ped.f_entrega
FROM    co_pedidos      ped
WHERE   ped.f_pedido    <= '01-SEP-96'
    AND ped.f_entrega   >= '01-SEP-96'
;
```

como alternativa a:

```
WHERE   '01-SEP-96' ped.f_pedido
   AND  ped.f_entrega
```

Cláusula LIKE

Evitar el uso del símbolo comodín "%" en la primera posición de la cadena de comparación a fin de evitar la desactivación del índice.

Evitar igualmente el uso de este operador con números y fechas a fin de no forzar una conversión implícita de datos a una cadena de caracteres.

Ejemplo:

```
SELECT  pro.c_codigo
,       pro.a_nombre
,       pro.n_precio
FROM    co_productos       pro
WHERE   pro.c_codigo >= '20000'
    AND pro.c_codigo <= '20999'
;
```

como alternativa a:

```
WHERE  pro.c_codigo LIKE '20___'
```

Si existiese un índice sobre la columna `pro.c_codigo` la primera sentencia lo utilizaría, mientras que la misma sentencia utilizando la cláusula LIKE, lo deshabilitaría.

Si la optimización se basa en costes, el optimizador podría escoger en el primer caso el uso del índice o no, dependiendo de las estadísticas generadas para la tabla.

Cláusula IS [NOT] NULL

Convertir este operador en una condición sencilla con operadores matemáticos si la columna contiene pocos valores a fin de no desactivar el uso de índices.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.p_comision
,       emp.i_sueldo
FROM    co_empleados      emp
WHERE   emp.p_comision    >= 0
;
```

como alternativa a:

```
WHERE  emp.p_comision IS NOT NULL
```

Si existe un índice para la columna la primera sentencia lo utilizaría, mientras que la misma sentencia utilizando la cláusula IS NOT NULL, lo deshabilitaría.

Si la optimización se basa en costes, el optimizador podría escoger en el primer caso el uso del índice o no, dependiendo de las estadísticas generadas para la tabla.

Cláusula AND

Reducir la condición compleja a una condición simple usando las funciones MIN, MAX o el operador BETWEEN siempre que sea posible.

Ejemplo:

```
SELECT  lin.n_num_pedido
,       lin.n_num_linea
,       lin.c_producto
,       lin.q_cantidad_pedida,
,       lin.q_cantidad_servida
FROM    co_lin_pedidos         lin
WHERE   lin.q_cantidad_servida <
        MIN (100,
        lin.q_cantidad_pedida)
;
```

como alternativa a:

```
WHERE     lin.q_cantidad_servida < 100
      AND lin.q_cantidad_servida <
          lin.q_cantidad_pedida
```

Cláusula OR

Escribir las expresiones OR entre paréntesis; si es posible, reducirla a una condición simple mediante el operador IN.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo
FROM    ge_empleados          emp
WHERE   emp.c_dpto IN (10,20)
;
```

como alternativa a:

```
WHERE   emp.c_dpto = 10
     OR emp.c_dpto = 20
```

Cláusula NOT

Escribir la expresión sin el operador NOT siempre que sea posible.

Ejemplo:

```
SELECT  inv.c_prod
,       inv.c_almacen
```

```
,       inv.q_cantidad
,       inv.q_punto_pedido
,       inv.q_max_stock
FROM    co_inventario      inv
WHERE   inv.q_cantidad >=
        inv.q_punto_pedido
    AND inv.q_cantidad <=
        inv.q_max_stock
;
```

como alternativa a:

```
WHERE   NOT (inv.q_cantidad <
        inv.q_punto_pedido
   OR   inv.q_cantidad >
        inv.q_max_stock)
```

Escritura de cálculos

Realizar los cálculos de la cláusula WHERE con constantes en lugar de utilizar columnas, siempre que sea posible, para mejorar el rendimiento de la consulta.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo
FROM    ge_empleados        emp
WHERE   emp.i_sueldo * 12 > 3500000
;
```

como alternativa a:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       emp.i_sueldo
FROM    ge_empleados        emp
WHERE   emp.i_sueldo > 3500000 / 12
;
```

Convertir explícitamente los tipos de datos de un cálculo, mediante las oportunas funciones de conversión, evitando conversiones implícitas de tipos. Las conversiones implícitas pueden degradar el rendimiento o desactivar índices y están sujetas a posibles modificaciones en las nuevas versiones del SGBDR ORACLE.

Ejemplo:

```
SELECT  TO_CHAR (pro.c_codigo)|| pro.a_nombre
        nombre
FROM    co_productos       pro
;
```

como alternativa a:

```
SELECT  pro.c_codigo || pro.a_nombre
         nombre
FROM    co_productos          pro
;
```

El empleo de TO_CHAR maximiza el rendimiento al convertir explícitamente el tipo de datos de la columna `pro.c_codigo.`

Ejemplo:

```
SELECT  MAX (emp.i_sueldo) * 1,15  sueldo_nuevo
FROM    ge_empleados               emp
;
```

como alternativa a:

```
SELECT  MAX (emp.i_sueldo * 1.15)  sueldo_nuevo
FROM    ge_empleados               emp
;
```

La primera sentencia es más eficiente que la segunda porque la operación de multiplicación sólo se realiza una vez.

Cálculos de grupos

Transformar la cláusula HAVING en una cláusula WHERE siempre que sea posible para mejorar la eficiencia de la consulta.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       SUM (emp.i_sueldo)      total_sueldos
FROM    ge_empleados            emp
WHERE   emp.c_dpto > 10
GROUP BY emp.c_dpto
;
```

como alternativa a:

```
SELECT  emp.a_nombre
,       emp.c_dpto
,       SUM (emp.i_sueldo)     total_sueldos
FROM    ge_empleados            emp
GROUP BY  emp.c_dpto
HAVING    emp.c_dpto > 10
;
```

La primera sentencia es más eficiente que la primera, ya que el servidor ORACLE elimina las filas no necesarias antes de realizar la agrupación. No es posible convertir una expresión HAVING en una expresión WHERE si en la condición se utiliza una función de grupo que requiera el proceso de agrupación.

Ordenaciones

Evitar combinaciones de ordenaciones ascendentes y descendentes en lo posible, por razones de rendimiento. Las combinaciones de ordenaciones inversas pueden inhibir la utilización de índices concatenados. Si una combinación de este tipo es necesaria, indicar las palabras clave ASC y DESC en todas las columnas a fin de mejorar la legibilidad.

Ejemplo:

```
SELECT  emp.f_contratacion
,       emp.a_nombre
FROM    ge_empleados            emp
ORDER BY  emp.f_contratacion    DESC
,         emp.a_nombre          ASC
;
```

como alternativa a:

```
SELECT  emp.f_contratacion
,       emp.a_nombre
FROM    ge_empleados          emp
ORDER BY 1                    DESC
,        2
;
```

Subconsultas

Convertir una subconsulta en una unión siempre que se pueda. Aunque las subconsultas ofrecen mayor claridad de lectura que las uniones, por razones de rendimiento es mejor evitar su uso o convertirlas en uniones siempre que se pueda. Si no es posible evitarlas, siempre son preferibles las subconsultas no sincronizadas, porque se ejecutan una sola vez.

Ejemplo:

```
SELECT  emp.a_nombre
,       emp.c_dpto
FROM    ge_empleados    emp
,       ge_empleados    emp_garcia
WHERE   emp1.c_dpto     = emp_garcia.c_dpto
    AND emp_garcia.a_nombre = 'Garcia'
;
```

como alternativa a:

```
SELECT  emp.a_nombre
,       emp.c_dpto
FROM    ge_empleados  emp
WHERE   emp.c_dpto IN
        (SELECT       emp_garcia.c_dpto
         FROM         ge_empleados     emp_garcia
          WHERE       emp_garcia.a_nombre = 'Garcia'
        )
;
```

Gestión de transacciones

Las sentencias COMMIT o ROLLBACK explícitas son aquéllas realizadas expresamente por el programa, a diferencia de las implícitas, que son emitidas por el SGBDR ORACLE8i de forma automática (por ejemplo, cuando se termina una transacción).

Aplicar las siguientes normas:

Ejecutar una sentencia COMMIT explícita cada vez que se realiza un cambio o grupo de cambios permanentes inmediatamente después de la última operación DML de la transacción.

Evitar la visualización en pantallas de datos temporales que se han de deshechar por haberse interrumpido la transacción por causas ajenas al usuario mediante la ejecución de una sentencia ROLLBACK explícita.

Ejecutar una sentencia COMMIT o ROLLBACK explícita antes de cada sentencia DDL.

Terminar cada bloque PL/SQL, fichero SQL*Plus o módulo de aplicación con la ejecución de una sentencia COMMIT o ROLLBACK, según corresponda.

CAPÍTULO

8 Uso del Lenguaje PL/SQL

En el presente capítulo se definen las normas aplicables para el uso del lenguaje PL/SQL. Se abordan aspectos tales como normas tipográficas y uso de comentarios para una mejor legibilidad de los fuentes, normas de denominación y declaración de variables y las normas y recomendaciones aplicables a cada una de las secciones de un módulo PL/SQL: Sección de Declaración, Sección de Ejecución y Sección de Definición de Excepciones.

Se examina en último lugar las técnicas de procesamiento de entradas y salidas y gestión de bloqueos de tablas.

Introducción

Definición del PL/SQL

PL/SQL es la extensión de lenguaje procedural de Oracle Corporation al SQL, el lenguaje estándar para el acceso a base de datos relacionales.

PL/SQL contiene características de ingeniería de software modernas, tales como encapsulado de datos, ocultamiento de la información, sobrecarga [1] y gestión de excepciones.

Dónde se usa

Un bloque PL/SQL puede ser utilizado en diversas instancias:

- En las herramientas no procedurales de Oracle

  PL/SQL añade funcionalidad procedural a las herramientas no procedurales tales como Oracle Forms y Oracle Reports. Mediante el uso de PL/SQL en estas herramientas se pueden utilizar construcciones sintácticas tradicionales para el desarrollo de aplicaciones. Por ejemplo, se puede hacer un bloque PL/SQL en un disparador de Oracle Forms.

- En lenguajes tradicionales

  Un bloque PL/SQL puede colocarse en programas fuente escritos en lenguajes tradicionales de tipo procedural (COBOL, C, etc.). Durante la ejecución del programa una vez compilado el bloque permitirá el paso de datos en ambos sentidos, mediante la ejecución de las oportunas sentencias SQL y su procesamiento secuencial, mediante las sentencias y construcciones lógicas del PL/SQL.

- En SQL*Plus

  SQL*Plus permite también la inserción de bloques PL/SQL en los ficheros de comandos ejecutables en su entorno, ampliando de esta manera la propia funcionalidad ofrecida por esta herramienta.

- En disparadores de tablas

  Un disparador de base de datos es un bloque PL/SQL asociado a una tabla. Contiene adicionalmente información sobre cuándo se dispara la lógica del bloque (al insertar, actualizar o borrar datos de la tabla y si es antes o después de dichas operaciones), así como si la lógica se aplica a todas las filas afectadas por la operación o una sola vez al producirse el evento.

Los bloques PL/SQL pueden ser de dos tipos:

- Bloque anónimos

  Un bloque PL/SQL anónimo es un bloque sin nombre determinado, por lo que en tiempo de ejecución, el bloque es enviado al Oracle Server, donde se compila y ejecuta.

- Subprogramas almacenados

[1] En inglés, *overloading*.

Los subprogramas almacenados son bloques PL/SQL con un nombre determinado que se compilan y se almacenan en la base de datos Oracle de forma permanente, dispuestos para su ejecución. Es un objeto más de la base de datos y como tal puede ser llamado desde cualquiera de las herramientas de desarrollo anteriormente mencionadas.

Los programas almacenados pueden ser:

- Funciones

  Bloques PL/SQL que devuelven un valor, vía el mecanismo *RETURN*.

- Procedimientos

  Bloques PL/SQL que ejecutan una serie de acciones. Pueden devolver uno o más valores, vía el mecanismo de parámetros de salida.

- Paquetes

  Incluyen funciones y procedimientos, así como declaraciones de datos propios.

Relación entre el SQL y el PL/SQL

El lenguaje SQL es el lenguaje estándar de acceso a base de datos relacionales y puede utilizarse desde programas escritos en lenguajes de tercera generación tradicionales (COBOL, C, etc.) o desde aplicaciones que soporten su interpretación (Ejemplo, SQL*Plus).

Por su parte, el lenguaje PL/SQL, además de las construcciones sintácticas típicas de los lenguajes procedurales (declaración de variables, bloques de tipo IF, WHILE, REPEAT, etc.) contiene asimismo las sentencias del lenguaje SQL en toda su extensión, lo que le da una flexibilidad mayor para la construcción de aplicaciones de acceso a bases de datos relacionales Oracle.

Normas tipográficas

Uso de mayúsculas

En la escritura de bloques PL/SQL se utilizarán las siguientes normas en cuanto al uso de mayúsculas y minúsculas:

Aplicable a	Tipo de letra	Ejemplo
Palabras clave	mayúsculas	DECLARE, BEGIN, IF
Tipos de datos	mayúsculas	VARCHAR2, BOOLEAN
Identificadores y parámetros	minúsculas	v_i_sueldo
Tablas y columnas de la base de datos	minúsculas	ge_empleados, f_pedido

Ejemplo 1:

```
DECLARE
        v_i_sueldo    ge_emp.i_sueldo%TYPE
        ;
BEGIN
        SELECT emp.i_sueldo
        INTO   v_i_sueldo
        FROM   ge_empleados  emp
        WHERE  emp.c_dpto = 10
        ;
        IF     v_i_sueldo  = 13.5
        THEN
               UPDATE ge_empleados    emp
               SET    emp.d_tipo_sueldo := 'Alto'
               WHERE  emp.c_dpto     = 10
               ;
        END IF
        ;
EXCEPTION
        WHEN   NO_DATA_FOUND
        THEN   ROLLBACK WORK
        ;
END;
/
```

Los bloques PL/SQL embebidos en un fichero de comandos SQL*Plus han de terminarse con "/" para distinguirlos de otras sentencias SQL y comandos SQL*Plus y para poder ejecutarlos.

---

1 El ejemplo sólo pretende mostrar las normas de formato de bloques PL/SQL y el uso de sentencias dentro de sentencias. El uso de las dos sentencias SELECT y UPDATE se debería sustituir por razones de eficiencia por la sentencia UPDATE ... SET ... WHERE ...

Para escribir las sentencias SQL en un bloque PL/SQL utilizar los estándares de escritura del lenguaje SQL definidas.

Uso de espacios

En la escritura de bloques PL/SQL se utilizarán las siguientes normas en cuanto al uso de espacios:

Tipo de espacio	Uso
Nueva línea	Cláusulas principales (como DECLARE, BEGIN o SELECT) o la siguiente sentencia del bloque PL/SQL.
Sangrado	Variables dentro de la sección DECLARE y bloques condicionales o iterativos.
Espacios en blanco	Poner un espacio en blanco en los siguientes casos: • entre palabras reservadas (por ejemplo, entre END e IF), • entre símbolos (como el de "+") o signos de puntuación (como el de ";"), • antes y después de nombres de variables, tablas, etc., • antes del paréntesis inicial y después del paréntesis final, • después de las comas.

Si la sentencia es demasiado larga como para que quepa en una sola línea, dividirla antes del inicio de una expresión.

Ejemplo:

```
IF      :v_i_sueldo  = 13.5
THEN    :v_i_sueldo := 15.0;
END IF
;
```

Las sentencias incluidas en un bloque han de sangrarse. Si una sentencia PL/SQL es demasiado larga como para que quepa en una sola línea, sangrar la línea siguiente a fin de distinguir claramente que es continuación de la anterior.

Las sentencias incluidas en otras sentencias han de ser sangradas.

Reglas de construcción

Comentarios

En la escritura de bloques PL/SQL se utilizarán las siguientes normas en cuanto a la escritura de comentarios:

Si el comentario ocupa una sola línea hacerlo preceder de dos guiones y un espacio: ("-- ").

Si los comentarios ocupan más de una línea encerrarlos entre las marcas "/* " y " */" (insertar un espacio después y antes de cada marca, respectivamente).

Escribir los comentarios al principio de los bloques o secciones o antes del código comentado. Sangrar los comentarios a la misma posición que el bloque o sentencia.

Escribir los comentarios que afecten a una línea concreta a su derecha, si el comentario cabe en esa línea. Si no cabe, escribir el comentario debajo de la línea, sangrándolo a la misma posición.

Se debe repetir el nombre del procedimiento o función tras la palabra reservada END a fin de dar mayor legibilidad a procedimientos o funciones largas.

Ejemplo:

```
DECLARE
              i_contador     NUMBER (") .= 1; -- Num línea pedido
BEGIN
/*
            Este bucle incrementa el contador de i a 10 y para cada
            número inserta una línea de pedido en la tabla
            LINEAS_PEDIDO
*/
            WHILE i_contador <= 10 LOOP
              INSERT INTO ge_lineas_pedido -- Inserta una fila
                 (   n_ped_numero
                 ,   n_num_linea
                 ,   c_producto
              VALUES
                 (   10
                 ,   i_contador
                 ,   100860
                 );
              i_contador := i_contador + 1;
              /* Incrementa el contador */
          END LOOP;
END ejemplo;
/
```

Cabecera

- Disparadores de Oracle Forms

  Los programas PL/SQL que se escriban en los disparadores de Oracle Forms deberán llevar la siguiente cabecera:

  `/* Descripción: breve descripción del programa */`

- Subprogramas (funciones, procedimientos y paquetes)

  Se deberá insertar la siguiente cabecera:

```
/*
 * Autor :            Nombre del creador del programa.
```

```
 *
 * Fecha de creación   dd-mm-aaaa.
 *
 * Descripción:        Breve descripción de la funcionalidad del subprograma.
 *
 * Parámetros:         Relación de los parámetros que recibe y su descripción.
 *
 * Variables globales: Relación de variables globales utilizadas y descripción.
 *
 * Valores devueltos:  Descripción de los posibles valores que se devuelven y
                       su significado.
 *
 ********************************************************************************
 *
 * Control de modificaciones:
 *
 *  Fecha     Efectuados por     Motivo de modificaciones
 * -------  -----------------  ---------------------------------------------
 *
*/
```

Nombres de objetos

En la denominación de objetos utilizados en bloques PL/SQL se utilizarán las siguientes normas:

Objeto	Denominación
Tabla PL/SQL [1]	*T_<nombre>* Si la tabla corresponde a una tabla de la base de datos <*nombre* > utilizará el nombre de dicha tabla. Ejemplo: `t_ge_empleados`
Registro PL/SQL [2]	*R_<nombre>* Si el registro corresponde a una fila de una tabla <*nombre* > utilizará el nombre de la tabla. Ejemplo: `r_ge_personas`
Variable local [3]	*V_X_<nombre>* X es un carácter indicativo del tipo de objeto [4]. Si la variable corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `v_a_nom_empleado`
Variable de indexación [5]	*I_<nombre>* Ejemplo: `i_contador`

1 Se trata de un vector (*array*) de valores en memoria de trabajo, no de una tabla de la base de datos.

2 Se trata del tipo de datos registro, no de un registro de la base de datos

3 Con respecto al procedimiento o función en el que se define.

4 Ver las normas de denominación de columnas de tablas en el capítulo “Diseño Lógico de Bases de Datos”.

5 Variable utilizada como contador de bucles.

Variable global [1]	*G_X_<nombre>* X es un carácter indicativo del tipo de objeto [2]. Si la variable global corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `g_i_sueldo_total`
Constante	*C_X_<nombre>* X es un carácter indicativo del tipo de objeto [3]. Si el dato corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `c_f_hoy`
Cursor	*CUR_<nombre>* Ejemplo: `cur_empleados`
Excepción	*E_<nombre>* Ejemplo: `e_pro_incorrecto`
Etiqueta	<*nombre*> Ejemplo: `<<principal>>`
Procedimiento	*P_<nombre>* Ejemplo: `p_calc_sueldo_base`
Función	*F_<nombre>* Ejemplo: `f_calc_descuentos`

1 Con respecto al paquete en el que se define.

2 Ver las normas de denominación de columnas de tablas en el capítulo “Diseño Lógico de Bases de Datos”.

3 Ver las normas de denominación de columnas de tablas en el capítulo “Diseño Lógico de Bases de Datos”.

Parámetro de entrada o salida	*PR_X_<nombre>* X es un carácter indicativo del tipo de parámetro[1]. Si el parámetro corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `pr_f_inicio`
Objetos Oracle	*OBJ_<nombre>* Si el parámetro corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `obj_coche`
Variables Objeto Oracle	*V_OBJ_<nombre>* Si el parámetro corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `v_obj_coche`
Types Oracle	*TY_<nombre>* Si el parámetro corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `ty_sociedad`
Varray	*ARR_<nombre>* Si el parámetro corresponde a una columna de una tabla <*nombre*> utilizará el nombre de la columna. Ejemplo: `arr_clase`

[1] Ver las normas de denominación de columnas de tablas en el capítulo “Diseño Lógico de Bases de Datos”

Variables de contenido Ficheros de texto	*X_tipofichero_<nombre>* Ejemplo: `x_doc_manual`
Variables de contenido Ficheros gráfico	*G_tipofichero_<nombre>* Ejemplo: `g_jpg_foto`
Variables de contenido Ficheros video	*V_tipofichero_<nombre>* Ejemplo: `v_avi_video`
Variables de contenido Ficheros sonido	*S_tipofichero_<nombre>* Ejemplo: `s_wav_sonido`
Variables de contenido Fichero	*FIC_tipofichero_<nombre>* Ejemplo: `fic_exe_ejecutable`
Directorio	*DIR_<nombre>* Ejemplo: `dir_directorio`
Variable Directorio	*V_DIR_<nombre>* Ejemplo: `v_dir_directorio`

Declaración de variables

Utilizar como nombres de variables los propios nombres de las columnas de la base de datos, si procede, u otros nombres que tengan un sentido claro, en los casos en los que la variable no recoge valores de columnas (precedidos del sufijo correspondiente).

Aplicar siempre el criterio de que los nombres de los objetos es un aspecto importante de la documentación de una unidad de programa.

Utilizar como nombre para los cursores el prefijo "cur_" seguido del nombre abreviado de la tabla más significativa de la sentencia SELECT. Si el alias de dicha tabla se usa repetidas veces en un segmento de código PL/SQL, añadir un número secuencial como sufijo, a fin de generar nombres de cursores únicos.

Si se utilizan múltiples cursores como fuente para una variable de tipo registro nombrar la variable de registro con el nombre o alias de la tabla más significativa que mantenga columnas en la variable de registro.

Siempre que sea posible se inicializarán las variables declaradas con un valor.

Evitar en la declaración de variables el uso de tipo de datos diferentes de VARCHAR2, NUMBER, DATE o BOOLEAN.

Si una variable debe tener un valor, declararla como NOT NULL e inicializarla en la declaración.

Declarar una variable como constante si su valor no ha de cambiar durante la ejecución del programa.

Declarar los tipos de datos de las variables, siempre que sea posible con el atributo %TYPE, por ejemplo, cuando una variable PL/SQL tiene que ser del mismo tipo que una columna de la base de datos.

Si se produce una conversión de tipo de datos en una función de formateo, añadir un sufijo al nombre de la variable de destino, indicativo del nuevo tipo de dato, según se indica:

`_c`	`CHAR`
`_d`	`DATE`
`_n`	`NUMBER`

Valorar la necesidad de declarar las variables con una longitud mayor (por ejemplo, de diez posiciones) con respecto a la declarada en las propias columnas de tablas, a fin de poder absorber cambios en su longitud sin cambiar los programas PL/SQL.

Sección de Declaración

- **Declaración de cursores**

  Declarar cursores explícitos para todas las sentencias SQL que devuelven más de una fila.

Si una sentencia SELECT requiere variables de enlace 1, declararlas como parámetros del cursor.

Declarar una estructura de tipo registro por cada cursor definido explícitamente utilizando el atributo %ROWTYPE a fin de que se acople a los campos y tipos de datos de las columnas de la tabla correspondiente.

Declarar el tipo de dato de las variables de enlace con el atributo %TYPE.

- **Declaración de excepciones de usuario**

  Colocar juntas todas las excepciones definidas por el usuario en la parte declarativa del bloque PL/SQL.

  Colocar primero las excepciones definidas por el usuario del servidor ORACLE y a continuación las excepciones no relacionadas con el servidor ORACLE.

  Declarar explícitamente las excepciones para cada una de las condiciones de error del servidor ORACLE que se puedan producir durante la ejecución normal del programa.

  Asignar los errores del servidor ORACLE a excepciones definidas por el usuario utilizando PRAGMA EXCEPTION_INIT().

  Definir excepciones relacionadas con errores de tipo funcional o condiciones de advertencia (*warning*).

1 en inglés, *bind variables.*

Sección de Ejecución

- **Control de la lógica de procedimiento**

  Evitar la posibilidad de que la aparición de valores nulos en la evaluación de la condición de una sentencia LOOP o IF ... THEN ... ELSE ... provoque que segmentos de código PL/SQL se salten incorrectamente utilizando la función NVL a ambos lados del operador de comparación.

  Ejemplo:

```
DECLARE
          v_x NUMBER := NULL;
          v_y NUMBER := NULL;
          i_cont      NUMBER;
BEGIN
          IF  NVL    (v_x, 0) = NVL (v_y, 0)
          THEN
              SELECT  COUNT (emp.c_codigo)
              INTO    i_cont
              FROM    ge_empleados    emp
              ;
          END IF;
END ejemplo_nvl;
/
```

  Utilizar la construcción ELSIF para ejecutar condicionalmente un único segmento de código de entre un grupo de segmentos de código PL/SQL.

  Escoger de entre los tres tipos de constructores de bucles (LOOP universal, WHILE y FOR) el más adecuado en cada caso:

LOOP básico	Bucle infinito que requiere una sentencia EXIT de salida para su finalización o la aparición de una excepción definida por el usuario. La sentencia EXIT puede ubicarse en cualquier punto. Utilizar preferentemente cuando no existe ninguna condición de entrada en el bucle.
WHILE	Utilizar preferentemente cuando exista una condición de entrada en el bucle.
FOR	Utilizar cuando al entrar en el bucle se sabe el número exacto de veces que se va a ejecutar.

  Al establecer las condiciones de entrada o salida de un bucle usar preferentemente los operadores ">=" o "<=" en vez del operador "=", ya que el operador de igualdad puede hacer fracasar al comparar dos valores con pequeñas diferencias de redondeo.

  Incluir la cláusula WHEN OTHERS para tratar condiciones inesperadas siempre que sea posible.

  Asegurarse de que en los bucles la condición de finalización ha de ocurrir siempre. Cuidar que no aparezcan valores nulos en la condición de control del bucle que puedan originar su terminación antes de lo previsto.

  Asignar un nombre a cada bucle mediante el uso de una etiqueta.

  Ejemplo:

```
ACCEPT pr_c_dpto PROMPT 'Entre el código de Departamento: '
```

```
DECLARE
     v_c_dpto ge_departamentos.c_codigo%TYPE := &pr_c_dpto;
BEGIN
<<principal>>
    LOOP
       INSERT INTO  ge_departamentos
            (  c_codigo
               ,  a_nombre
               ,  c_region
               )
               VALUES
               (  v_c_dpto
               ,  'XXX'
               ,  10
               );
       v_c_dpto := v_c_dpto + 1;
       EXIT WHEN v_c_dpto = v_c_dpto +9;
    END LOOP principal;
END;
/
```

Evitar la sentencia GOTO cuando ésta enmascara el uso de un bucle o una sentencia IF completa.

No utilizar la sentencia GOTO para que la ejecución salga fuera de un bucle ni dentro de una estructura IF ... THEN ...ELSE o similar.

- **Gestión de cursores**

  Simplificar el código PL/SQL generado utilizando cursores gestionados por un bucle FOR en sentencias SELECT que recuperen más de una fila.

  El uso de este bucle en un cursor realiza lo siguiente:

  - Declara una estructura registro basada en las columnas de la sentencia SELECT.
  - Abre el cursor al entrar en el bucle.
  - Repite la iteración para cada fila recuperada.
  - Asigna los datos a la estructura de tipo registro.
  - Finaliza la ejecución de la iteración cuando no encuentra más filas.
  - Cierra el cursor cuando la iteración acaba, incluso en los casos en los que se ejecuta una excepción.

Ejemplo:

```
DECLARE
    v_n_total_dias           NUMBER (5);
    CURSOR     cur_empl IS
    SELECT    emp.c_codigo
    ,         emp.f_contrato
    FROM      ge_empleados      emp
    ;
BEGIN
    <<principal>>
       FOR r_empl IN cur_empl
          LOOP
             v_n_total_dias := SYSDATE-r_empl.f_contrato;
             INSERT INTO      temp
                (  c_id
                ,  n_total_dias
                )
             VALUES
                (  r_empl.c_codigo
                ,  v_n_total_dias
                );
          END LOOP principal;
END;
/
```

- **Llamadas a procedimientos y funciones**

  Invocar siempre un procedimiento o función de un paquete cualificando su nombre con el nombre del paquete.

  Colocar cada parámetro de un procedimiento o función en una línea distinta

  Ejemplo:

```
ACCEPT pr_c_empleado          PROMPT 'Entrar código empleado: '
ACCEPT pr_poc_com             PROMPT 'Entrar porc. comisión: '
DECLARE
          v_codigo    ge_emp.c_codigo%TYPE := &p_c_empleado;
          v_porc_com  NUMBER := &p_porc_comision;
          v_valido    BOOLEAN;
      BEGIN
          <<principal>>
          v_valido := pk_comision.validar_porc_com
             (       v_porc_com
             ,       v_codigo
             );
          IF v_valido = FALSE
             THEN      v_porc_com := 0;
             END IF;
      END;
      /
```

  Utilizar las normas de denominación de objetos o bien definir sinónimos para invocar un procedimiento o función de un esquema o base de datos distinta.

  Ejemplos:

```
EXECUTE exp.pk_comision.asignar_porc_com (0,15);
EXECUTE exp.pk_comision.asignar_porc_com@bd (0,15);
```

Sección de Definición de Excepciones

- **Normas de codificación**

  Situar el bloque de manejo de excepciones en la última parte del bloque PL/SQL.

  Evitar la definición de bloques de manejo de excepciones excesivamente largos, manteniendo esta sección corta y eficaz.

  Evitar utilizar esta sección como mecanismo de control del flujo de ejecución. Construir un bucle que contenga la excepción para continuar el procesamiento del bloque después de que se origine una excepción.

  Evitar el uso de SQLCODE para bifurcar al bloque de manejo de excepciones.

- **Uso de la tabla STDERR**

  Ejecutar un rollback de los cambios no deseados antes de insertar el mensaje de error en la tabla STDERR 1. A continuación se deberá ejecutar *commit* del mensaje de error en la tabla STDERR.

  Ejemplo:

```
EXCEPTION
      ROLLBACK;
      WHEN NO_DATA_FOUND THEN
            INSERT INTO STDERR
              (       error_id
              ,       timestamp
              ,       message
              )
              VALUES
              (       error_seq.NEXTVAL
              ,       SYSDATE
              ,       'Código de cliente inexistente.'
                      );
      COMMIT;
      END;
      /
```

[1] Esta tabla permite almacenar la fecha y hora y el mensaje de error.

Procesamiento de entradas y salidas

Los datos de entrada necesarios para la ejecución de un bloque PL/SQL desde SQL*Plus se obtendrán mediante el uso de parámetros de sustitución y los comandos DEFINE y ACCEPT.

Los datos de salida obtenidos en un bloque PL/SQL ejecutado desde SQL*Plus se llevarán a una tabla temporal.

Técnicas de bloqueo de tablas

No aplicar inmediatamente los cambios. Almacenar los cambios WRITE en el buffer y comprometerlos (commit) sin que interaccionen con el usuario.

En las operaciones de actualización y borrado se deberá bloquear la fila existente para evitar que otros usuarios realicen cambios en ella antes de la actualización deseada.

Se deberá poner la tabla en modo SHARE UPDATE a fin de que varios usuarios puedan hacer un bloqueo de fila de forma concurrente. Se asegurará que todos los usuarios acceden a la tabla en el mismo modo (EXCLUSIVE o SHARE UPDATE), a fin de que ningún usuario haga esperar al resto.

A fin de evitar situaciones de *deadlock* hay que asegurar que el orden en el que se modifican las tablas dentro de una unidad de *commit* sea constante para todas las aplicaciones. Para ello se asignará a cada tabla un número único indicativo del orden de bloque respecto al resto y se asegurará que todas las aplicaciones que modifiquen más de una tabla en una única unidad de *commit* realicen las modificaciones en el orden de secuencia de la tabla.

En las aplicaciones Forms con bloques múltiples y tablas base, los números de secuencia de las tablas deberán estar de acuerdo con el orden que estas tablas tienen como tablas base dentro de las aplicaciones Forms. Por ello, las tablas de detalle deberán tener un número de secuencia mayor que las tablas maestras correspondientes.

Si una aplicación Forms no admite una adecuada ordenación de tablas, los bloques de la aplicación habrán de resecuenciarse para que acepten dicha ordenación, cambiando las teclas de función a fin de hacer este cambio de orden transparente al usuario.

Desarrollo con PL/SQL para IAS

Se recogen las consideraciones a tener en cuenta para el desarrollo de una aplicación con PL/SQL orientado a Web y con IAS como servidor de aplicaciones.

Esta tecnología permite la publicación de contenidos dinámicos en el web mediante el acceso a bases de datos, y su principal ventaja reside en que cualquier persona que conozca el lenguaje de base de datos Oracle (***PL/SQL***) puede, en un corto periodo de tiempo, desarrollar una aplicación para la visualización de datos procedentes de una base de datos haciendo uso de un navegador Internet.

Cabe resaltar el concepto "publicación", ya que aunque esta tecnología permite realizar aplicaciones en las que intervengan actualizaciones de datos, la programación de elementos transaccionales no es trivial.

En cuanto al nombrado de objetos, el estándar de nombrado será el mismo empleado en el desarrollo sobre Oracle tradicional, con las salvedades introducidas en virtud del empleo como herramienta de análisis y diseño de Designer 6i.

Paquetes de la Aplicación

En estas aplicaciones, el código ***PL/SQL*** estará embebido en la propia base de datos en forma de paquete. Los paquetes resultantes se nombrarán de la forma ***WEBxxxx_yyyyzz***, donde '***xxxx***' es la Consejería, '***yyyy***' la aplicación y ***zz*** el tipo de paquete; por ejemplo, para la aplicación de consultas sobre Incendios, de la consejería de Medio Ambiente, los paquetes serían:

- ***WEBCMA_INCEGE*** para el paquete general o común, con la definición de cabeceras, pies, etc...
- ***WEBCMA_INCEPR*** para el paquete propiamente dicho de la aplicación, que incluiría el formateo de las páginas devueltas por la búsqueda y los detalles.
- ***WEBCMA_INCEBQ*** para las funciones de las ventanas de búsqueda.
- y en general, si fueran necesarios más paquetes, se variaría el índice 'zz' para indicar la utilidad de los procedimientos y funciones contenidas.

Sin embargo, y dentro de lo posible, habría que tratar de tener básicamente tres: uno general con las funciones y procedimientos comunes a distintas aplicaciones (cabeceras, pies, etc...); otro particular, para cada una de las funcionalidades incluidas en la aplicación – normalmente asociados al formateo de las páginas y que se llaman desde dicha página -; y por último, otro con las funciones y procedimientos comunes a distintas páginas, que son llamados desde varias (búsquedas, etc...).

Hay una consideración más. Por limitaciones de algunas herramientas -Sql*Plus, por ejemplo - hay problemas si el fichero tiene más de 64 Kbytes. En realidad, si pensamos un poco en una división modular, jamás un fichero de procedimientos sql debería llegar a dicho tamaño. El problema es que si llegara, se produciría un efecto de truncamiento al ejecutarlo. Por tanto, es un tamaño a evitar.

Hasta la existencia de un tablespace ***COMU*** general para todas las Consejerías, aquellas aplicaciones que residan en el servidor Web externo incluirán en su propio tablespace las tablas comunes que necesiten. Se montará un tablespace por Consejería con la denominación ***COMU_XXXX_TSD1***, donde ***XXXX*** identifica la consejería.

El estándar de visualización debe ser 800 x 600.

El código debería estructurarse en la misma forma en que se estructura una página HTML, con las indentaciones correspondientes a la estructura de dicho documento. No deben olvidarse las consideraciones relativas a la documentación, modularidad, etc...

Por ejemplo, tendría el siguiente aspecto:

```
BEGIN

  htp.htmlopen;

    htp.headopen;

          htp.title ('JCyL – CICT – Detalle de Monumentos');

          webcict_turige.p_color_links;

    htp.headclose;

    htp.bodyopen (null,'BGCOLOR=" ' || webcict_turige.c_color_bg_pag
|| '"');

                                .............

    htp.bodyclose;

  htp.htmlclose;

EXCEPTION

          .............

END;
```

En cuanto a la disyuntiva sobre si utilizar el ***PLSQL Toolkit*** o directamente la función "htp.p" para imprimir los tags de Html, resulta mejor utilizar el paquete por la validación a las sentencias que se realiza en tiempo de compilación, que añade una garantía grande al desarrollo. Sólo si el tiempo de ejecución resultara crítico y se observaran tiempos de generación dinámica de las páginas elevados debería recurrirse a la otra técnica.

Se recomienda la utilización de hojas de estilos, para minimizar el tamaño de las páginas y homogeneizarlas de cara a cada navegador.

Hay que forzar que la utilización de ***javascript*** para conseguir efectos de HTML dinámico si así se quisiera, sea válida tanto para Internet Explorer como para Netscape, por ser los navegadores más extensamente utilizados. Nunca sólo para uno de ellos.

Asimismo, en ***javascript***, debe existir el código necesario para hacer las validaciones pertinentes en los campos de entrada para que los valores que llegarán al servidor sean válidos. Igualmente, en los paquetes se darán valores por defecto para cuando llegan en blanco.

Por otro lado, el código ***PL/SQL*** contendrá el tratamiento de excepciones pertinente a cada caso, para evitar la no generación de la página.

La aplicación, de cara al ***IAS***, tendrá también un nombre de cuatro letras – a ser posible el mismo – y deberá ser suficientemente específico para evitar el problema de coincidir el nombre con el de otra aplicación, no ya de la misma Consejería, sino de cualquier otra.

El camino de acceso en el 'link' inicial de la aplicación será de la forma:

http://www.jcyl.es:8080/ora_xxxx/plsql/yyyy.zzzz

donde **xxxx** es el nombre de la aplicación, **yyyy** es el nombre del paquete y **zzzz** el nombre del procedimiento.

A partir de este punto, las referencias a cualquier elemento de la página serán relativas, de la forma "yyyy.zzzz".

El único enlace absoluto es el que se corresponde con el organismo coordinador de la información, que siempre deberá apuntar a http://www.jcyl.es/bin/correo, que es un cgi que en función de la página permitirá enviar un correo a quien corresponda.

Tanto para el ***IAS*** como para Oracle, cada aplicación tendrá al menos un usuario con el mismo nombre que la aplicación y que será el propietario del esquema. Todos los objetos se referenciarán anteponiendo a su nombre el del propietario del esquema.

Para consultas se creará un segundo usuario del mismo nombre seguido de '***WRO***' – web read only – y si se permiten actualizaciones en la misma web, se emplearán los sufijos '***WRW***' – web read write – y '***WWO***' – web write only – aunque éste último será más extraño.

Para la instalación del esquema en el servidor externo será necesario conocer los nombres de los usuarios y el tamaño del tablespace.

En la fase de desarrollo, para emplear el ***IAS*** instalado en la DGTT será necesario conocer la entrada del fichero ***TNSNAMES.ORA*** y se creará la ***DAD*** correspondiente apuntando a la base de datos. Se suponen creados en la base de datos local los usuarios correspondientes a la aplicación más los propietarios de los paquetes del ***IAS***; a saber, ***WEBSYS*** y ***OAS_PUBLIC***. Ambos tendrán como tablespace por defecto uno de nombre ***OWS_TSD1*** y nunca el tablespace ***SYSTEM***. Tendrán los roles ***CONNECT*** y ***RESOURCE***, aunque después se le quitará el privilegio ***UNLIMITED_TABLESPACE*** y lo que se hará será darle cuota "unlimited" en su tablespace y en el temporal.

Referencia a Objetos del Servidor Web

Todos los servidores que poseen servicios ***IAS*** disponen de ***2*** servidores web, uno que escucha en el puerto ***80*** y que sirve de punto de entrada para todas las páginas estáticas, y otro que escucha en el puerto ***8080***. Es este segundo servidor web el que contiene el ***ORB*** necesario para ejecutar las aplicaciones hechas con ***IAS***, y una vez que ha tomado el control sirve tanto las páginas y elementos estáticos como los contenidos dinámicos, para lo cual posee el mismo árbol de páginas que el servidor primario.

Es por esto que los usuarios navegan por el servidor principal, y habrá una página que les permitirá acceder a los contenidos dinámicos. Esto, que lo llamaremos ***punto de entrada***, se realiza poniendo un enlace al servidor web secundario con los parámetros adecuados que llaman a la aplicación, como ya hemos visto antes:

```
<A HREF=
"http://www.jcyl.es:8080/ora_xxxx/plsql/yyyy.zzzz">Aplicación</A>
```

Cuando el usuario pincha en el enlace se activan los mecanismos de ***IAS*** que arrancan la aplicación correspondiente, y el servidor web del puerto ***8080*** toma el control, no siendo necesaria ninguna referencia explícita al mismo a partir de ese momento. Esto significa lo siguiente:

- Los paquetes y procedimientos se deben referenciar unos a otros de forma directa, sin indicar ***URL***alguno delante de ellos. Si el procedimiento a invocar se encuentra en otro paquete diferente al que estamos ejecutando en un momento dado, sólo hay que anteponer el nombre del paquete al del procedimiento, separados por un punto (esto no tiene nada que ver con ***IAS***, es un requisito del lenguaje ***PL/SQL***.)

- Las páginas, imágenes y documentos estáticos se deben referenciar de forma relativa a la raíz de las páginas web en el servidor, sin anteponer ***URL*** alguno delante de ellos. De esta forma no nos importa cómo se llame el servidor donde están alojadas las páginas.

El servidor Web de la Junta de Castilla y León se encuentra organizado de forma jerárquica según la estructura orgánica de la misma y llegando hasta el nivel de Dirección General, que es donde residen los diferentes temas que dotan de contenido al sitio. Por ejemplo, el tema dedicado a los Incendios Forestales cuelga de la D.G. del Medio Natural de la Consejería de Medio Ambiente, y por ello su ruta en el Web de la Junta es:

http://www.jcyl.es/jcyl/cma/dgmn/ince/

Luego si desde una aplicación dinámica queremos hacer referencia a un documento *ince_1.doc*, por ejemplo, en el código irá embebida la siguiente sentencia:

```
<A HREF=
"/jcyl/cma/dgmn/ince/ince_1.doc">Posibles causas de incendios</A>
```

Como vemos, no se hace referencia alguna al servidor web, sino que se indica el camino desde la raíz de documentos del servidor.

Por razones de acceso rápido y de comodidad, se puede solicitar a los administradores del servidor web un alias para la ruta. Siguiendo el ejemplo anterior, el administrador puede crear un alias de la forma:

```
/incendios/  ==  /jcyl/cma/dgmn/ince/
```

con lo cual, la sentencia anterior quedaría de la siguiente forma:

<A HREF="/incendios/ince_1.doc">Posibles causas de incendios</A>

Herramientas de Desarrollo

- Las referenciadas como estándares en la página WEB del Servicio de Informática Corporativa de la Junta de Castilla y León (S.I.C).

CAPÍTULO

9 Uso del Lenguaje SQL*PLUS

Se incluyen a continuación las normas de escritura y uso del lenguaje SQL*Plus. Estas normas abarcan tanto la estructura formal de un módulo SQL*Plus (cabeceras de programa, comentarios, normas tipográficas) como el uso del lenguaje para la emisión de informes, el paso de parámetros en tiempo de ejecución y el uso de las sentencias DML contenidas en un fichero de ejecución.

Introducción

SQL*Plus es un sistema de software de Oracle Corporation que permite la ejecución de comandos SQL y PL/SQL introducidos desde el terminal o bien desde un fichero.

Adicionalmente, SQL*Plus ofrece una serie de comandos propios para la realización de tareas tales como:

- Edición y ejecución de comandos SQL y bloques PL/SQL.
- Formateo, ejecución, almacenamiento e impresión de informes y consultas a la base de datos.
- Listado de información de objetos de la base de datos.
- Conexión a otras bases de datos locales o remotas y extracción de información.
- Envío de mensajes y recogida de respuestas del usuario de forma interactiva.

SQL*Plus puede ser llamado desde Oracle Forms u otras herramientas que soporten el comando HOST o similar, aunque su uso normal se relaciona con la ejecución de comandos y bloques PL/SQL de forma interactiva.

Reglas de construcción

Comentarios

Se comentará cada sección del fichero de comandos y cada sentencia SQL para documentar su propósito mediante el uso del comando "REM".

Las líneas adicionales a la primera deberán iniciarse igualmente mediante la palabra reservada "REM".

Cuando los comandos de formato sean específicos para un entorno particular (por ejemplo, "papersize", "printer", "terminal") deberán comentarse.

Cabecera

El formato de la cabecera será la siguiente:

```
REM
REM Autor :             Nombre del creador del programa.
REM
REM Fecha de creación   dd-mm-aaaa.
REM
REM Descripción:        Breve descripción de la funcionalidad del programa.
REM
REM Parámetros:         Relación de los parámetros que recibe y su descripción.
REM
REM ************************************************************
REM
REM Control de modificaciones:
REM
REM  Fecha     Efectuados por     Motivo de modificaciones
REM -------  -----------------  ----------------------------------
REM
REM
```

El nombre del informe y el del fichero ejecutable de tipo .SQL serán idénticos.

Estructura del programa

Los ficheros SQL*Plus deberán estructurarse en los siguientes segmentos:

- Cabecera (ver formato a continuación)
- Comandos de entorno tipo SET y WHENEVER (por ejemplo, "pagesize", "termout")
- Comandos para convertir argumentos de paso de valores en línea de comando a variables de sustitución.

  Por ejemplo:
  ```
  DEFINE v_c_departamento = &1
  ```

- Comandos de definición de cabecera y pie de página (por ejemplo, "ttitle", "btitle")
- Comandos de formato de columna (por ejemplo, "column", "format", "heading")
- Comandos de roturas de grupos (BREAK ON ...)
- Definición de COMPUTE ...
- Comando de definición de SPOOL
- Documentación de comandos SQL y bloques PL/SQL
- Comandos de SQL y bloques PL/SQL
- Comando de cierre de spool
- Comandos para limpiar opciones (por ejemplo, "clear breaks", "clear computes")

Escritura de comandos

Para la escritura de los comandos propios de SQL y PL/SQL son de aplicación las normas descritas en el uso de estos lenguajes. En cuanto a la escritura de comandos específicos de SQL*Plus, se aplicarán las siguientes normas:

Los comandos específicos de SQL*Plus así como sus valores compuestos por palabras reservadas se escriben en mayúsculas.

Ejemplo:

```
BREAK         ON    c_departamento
COMPUTE SUM   OF    i_sueldo
              ON    c_departamento
```

Insertar un tabulador inmediatamente después del nombre del comando SQL*Plus, así como antes y después de cada palabra clave de las cláusulas de un comando.

Dado que los comandos de SQL*Plus se terminan por un fin de línea, para estructurar comandos a través de varias líneas se utilizará el indicador de continuación del comando en la línea siguiente ("-").

Ejemplo:

```
COPY FROM   usr_1/clave_1
TO          usr_2/clave_2          -
REPLACE     ge_empleados           -
USING       SELECT *               -
            FROM    ge_empleados
```

Los comandos que se extiendan a lo largo de varias líneas se sangrarán de forma que se identifique claramente la totalidad del comando.

Ejemplo:

```
COLUMNA   apellido     HEADING     'Nombre del Empleado' -      JUSTIFY
                       CENTER
COLUMN    usuario      HEADING     'Ident. de Usuario'   -
                       JUSTIFY     RIGHT                 -      FORMAT
                       999999
```

Ejemplo:

```
     TTITLE     RIGHT    'Pág.: '    FORMAT     sql.pno SKIP 1   -
                LEFT     'Análisis de Rentabilidad'            -
                CENTER   '26-ENE-95'           SKIP 1   -
                CENTER   'Importes en Ptas.'   SKIP 2
```

Insertar una línea en blanco antes y después de cada uno de los segmentos de un fichero SQL*Plus, según se ha definido en Estructura del programa.

Emisión de informes con SQL*Plus

Cabecera de informe

La cabecera de los informes de SQL*Plus se realizará aplicando las normas establecidas para Oracle Reports (epígrafe “Cabecera de página”).

Formato de columnas del informe

Declarar el formato y características de cada columna del informe mediante el comando COLUMN.

Evitar los nombres y formatos suministrados por defecto por SQL*Plus.

Para cambiar el formato de columnas de tipo DATE utilizar la función TO_CHAR en la sentencia SQL que recupera los datos.

Comandos BREAK

Especificar claramente los comandos BREAK del informe, así como el número de líneas de salto.

Comandos COMPUTE

Antes de un comando COMPUTE ha de insertarse el comando BREAK correspondiente, que especifique la expresión, columna o alias de columna referenciada en la cláusula ON.

Verificar que la expresión, columna o alias de columna referenciada en la cláusula ON aparece también en la sentencia SELECT.

Escribir el comando COMPUTE de forma clara, mediante el alineamiento de sus cláusulas y el uso del símbolo "-" al final de la línea.

Ejemplo:

```
BREAK              ON      c_departamento
COMPUTE SUM  OF    salario -
                   ON      c_departamento
```

Comando SPOOL

El resultado de un informe ejecutado con SQL*Plus deberá almacenarse en el fichero definido en el propio ejecutable mediante el comando SPOOL.

La extensión a utilizar será ".LIS".

Al final del fichero de comandos deberá insertarse el comando "SPOOL OFF".

Paso de parámetros

Las variables utilizadas para volcar los valores de parámetros tendrán nombres significativos para que su uso en el programa sea claro y no ambiguo. En general, su nombre se forma con el prefijo "v_" y el nombre de la columna con la que se relaciona.

Ejemplo:

```
v_c_departamento
```

donde c_departamento es el nombre de la columna correspondiente

Uso del comando ACCEPT

La recogida de valores de los parámetros de ejecución del fichero se realizará mediante el comando ACCEPT y la cláusula "PROMT".

El texto ("prompt") especificado en el comando ACCEPT estará redactado de forma clara y no ambigua, desde el punto de vista de las necesidades del usuario que ejecutará el programa.

Ejemplo:

```
ACCEPT    v_c_dpto         CHAR
          PROMPT 'Código del departamento: '

SELECT    emp.a_nombre
,         emp.i_sueldo
FROM      ge_empleados     emp
WHERE     emp.c_dpto = '&v_c_dpto'
;
```

Uso del comando DEFINE

Se utilizará el comando DEFINE para pasar a una variable el valor del argumento introducido por el usuario al invocar el fichero de ejecución.

Ejemplo:

```
DEFINE v_c_dpto = &1

SELECT    emp.a_nombre
,         emp.i_sueldo
FROM      ge_empleados     emp
WHERE     emp.c_dpto = 'v_c_dpto'
;
```

Los parámetros correspondientes a columnas de tablas que almacenan valores en mayúsculas deberán convertirse a mayúsculas (mediante la función UPPER) antes de utilizarlos como sustitución de variables en una cláusula WHERE.

Para una mayor flexibilidad de ejecución se usarán caracteres comodines "%", "_") dentro de las cláusulas WHERE en las que se encuentren variables de sustitución que se vayan a sustituir por los valores de los parámetros, en combinación con el operador LIKE.

Uso de sentencias DML

Si un programa SQL*Plus realiza modificaciones en la base de datos, se preverá una sentencia "WHENEVER SQLERROR ...", para impedir que la ejecución continúe y siga actualizando la base de datos erróneamente al ocurrir un error en una de estas sentencias.

CAPÍTULO

10 ORACLE Forms 6i

Se describen a continuación las normas aplicables para el desarrollo de aplicaciones en Oracle Forms.

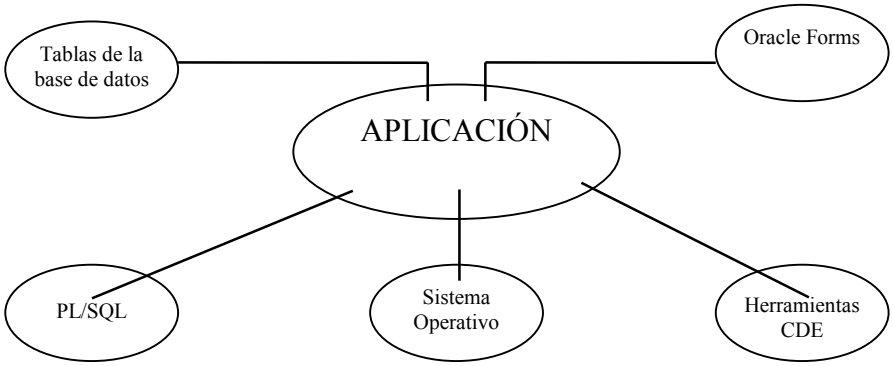
Las normas se organizan en las áreas de especificación de la interfaz gráfica de usuario consistente a lo largo de toda la aplicación y entre aplicaciones, el entorno de desarrollo, donde se especifican los estándares que se seguirán durante el desarrollo de módulos de pantallas y las normas relativas a la definición de menús de aplicaciones codificación de mensajes.

Introducción

Conceptos generales

Una forma es una aplicación que permite a los usuarios el acceso a la información almacenada en una base de datos. Una forma es más que una simple ventana, ya que puede incluir elementos de tablas de la base de datos situados en múltiples ventanas.

El desarrollo de aplicaciones con *Oracle Forms* se realiza en un entorno en el que se sitúan los siguientes elementos:



El desarrollo de aplicaciones con *Oracle Forms* se realiza mediante los siguientes componentes:

Oracle Forms Builder Forms Builder es un ambiente de desarrollo de aplicaciones que permite la elaboración de tres tipos diferentes de módulos Oracle: formas, menús y bibliotecas.

Forms Builder comprende un conjunto de herramientas visuales para la creación de objetos, determinar sus propiedades y escribir el código de la aplicación.

Oracle Forms Compiler El generador se utiliza para generar los ficheros ejecutables de las aplicaciones. La generación de un módulo permite la compilación de todos los objetos y crea el fichero de ejecución .FMX.

Oracle Forms RunTime Este componente es el motor de ejecución que los usuarios de las formas utilizan para ejecutar las aplicaciones de Oracle Forms.

Una aplicación Oracle Forms puede incluir los siguientes tipos de módulos:

Forms Son colecciones de objetos y su código, tales como ventanas, elementos o campos de texto, cajas de comprobación, botones, disparadores y procedimientos.

Menús Son una serie de objetos relacionados con la gestión de los menús de la aplicación (menú principal,

menús desplegables, opciones de menú) y el código de ejecución del sistema de menús.

Bibliotecas Son una serie de procedimientos, funciones y paquetes programados en PL/SQL que pueden ser llamados desde cualquier módulo de la aplicación.

Una aplicación Oracle Forms puede incorporar también otros módulos de otras herramientas CDE, tales como Oracle Reports y Oracle Graphics.

Finalmente, una aplicación Oracle Forms puede contener objetos visuales tales como controles VBX y objetos OLE. Además, puede llamar a cualquier ejecutable desarrollado en otros lenguajes y comandos del sistema operativo.

Programación orientada a eventos

Oracle Forms se basa en un modelo de programación orientado a eventos. Una vez definida la estructura básica y la funcionalidad de la aplicación mediante la oportuna creación de objetos y el establecimiento de sus propiedades, es posible ampliar y profundizar esta funcionalidad ofrecida por la herramienta por defecto mediante la escritura del código adecuado.

Este código se escribe en PL/SQL, extensión procedural realizada por ORACLE al lenguaje SQL.

La forma básica de añadir este código a las aplicaciones es mediante la escritura de disparadores [1]. Un disparador es un bloque de PL/SQL que se anexa a un objeto específico y que se ejecuta como respuesta a un evento específico. Por ejemplo, para crear un botón que ejecute una acción determinada cuando sea pulsado, hay que diseñar el botón y a continuación, asociar el disparador *WHEN-BUTTON-PRESSED* que ejecute las acciones deseadas.

[1] En inglés, *triggers*.

Arquitectura Forms Server (3 capas)

Oracle Forms Server es un servidor de aplicaciones que sirve para ejecutar aplicaciones Forms en Internet. Oracle Forms Server proporciona la infraestructura necesaria para ejecutar aplicaciones en Internet. Tiene tres componentes fundamentales:

- Cliente Java: Es un applet que se descarga en tiempo de ejecución desde el servidor de aplicaciones al navegador web del cliente. El cliente Java visualiza el interface de usuario del forms y gestiona la interacción entre el usuario final y el Forms Server.
- Forms Listener: Arranca el runtime de Forms Server y establece la conexión entre el cliente Forms y el Forms Server Runtime Engine.
- Forms Runtime Engine: Gestiona la lógica y el procesamiento de la aplicación. Mantiene la conexión entre el cliente Java y la base de datos. El código ejecutado por el Forms Server Runtime Engine es el mismo código (Forms, Reports, Menús) que el que se ejecuta para cliente-servidor sobre la misma plataforma. No es necesario realizar modificaciones en el código para ejecutar las aplicaciones en Internet.

Cuando un usuario ejecuta una sesión de forms sobre la Web, un applet ligero de Forms basado en Java se descarga dinámicamente desde el servidor de aplicaciones y se carga automáticamente en el caché de la máquina cliente Java. El mismo applet Java se utiliza para cualquier form, independiente de su tamaño o complejidad.

Principios de diseño de un IGU

Un entorno IGU debe respetar los siguientes criterios básicos de diseño de una aplicación:

Intuitivo para el usuario

El diseño y disposición de los elementos en las pantallas de la aplicación y el propio diseño de los menús deberá ser lo suficientemente autoexplicativo como para permitir que los usuarios puedan manejarlo de forma intuitiva.

Consistente

El diseño de las pantallas y menús deberá ser consistente a lo largo de toda la aplicación de manera que un mismo símbolo, descripción u opción haga las mismas cosas o lo más parecido posible a lo largo de toda la aplicación.

Paradigma objeto/acción

El diseño deberá poner el énfasis en los objetos de la aplicación ofreciendo para cada uno de ellos las posibles acciones a realizar sobre los objetos, frente a la orientación clásica mediante la cual las aplicaciones ofrecen una serie de funcionalidades aplicables a objetos.

Ofrecer retro-alimentación

Se debe proveer al usuario con mecanismos que le permitan la opción de retroceder a un punto anterior, si ello es posible.

Basado en el reconoci-miento

Reversibi-lidad

Las aplicaciones deben poseer un cierto grado de reversibilidad, en el sentido de que han de permitir al usuario la posibilidad de deshacer acciones una vez ejecutadas.

Apariencia estética

El diseño de pantallas debe respetar un mínimo de sentido estético, orientado a su vez a hacer más manejable e intuitivo su manejo.

Por ejemplo, el uso de colores deberá no sólo combinar los colores de forma estética, sino que se buscará a través de su uso facilitar el manejo de la aplicación.

Adaptación a los usuarios

Una aplicación normalmente deberá ser utilizada por usuarios de distinto grado de experiencia y con diferentes responsabilidades.

Por ello, se deberá diseñar de manera que no dificulte la comprensión básica a los menos expertos y que los que ejecuten determinados procesos específicos no tengan que ver las opciones de menús no aplicables en su trabajo.

Sencillez

Se aplicará en todo momento el principio de conservar la mayor sencillez en la presentación de la información y en el flujo de trabajo a ejecutar por el usuario, de forma que los aspectos complicados o repetitivos queden enmascarados internamente.

Recomendaciones generales de trabajo y diseño

Se aplicarán las siguientes recomendaciones generales en el diseño de las pantallas de las aplicaciones:

Control de la interacción

El control de la interacción entre la aplicación y el usuario debe residir siempre en el usuario, de forma que todas las acciones ejecutadas por la aplicación deben suministrar información al usuario, antes, durante y después de su ejecución:

Información antes de la ejecución

Cualquier decisión o acción a tomar por la aplicación debe ser antes confirmada por el usuario, mediante el oportuno mensaje de confirmación.

En los casos en los que una confirmación solicitada al usuario suponga múltiples pasos de la aplicación, debe informarse al usuario de las implicaciones últimas de su decisión, sobre todos cuando las acciones son de carácter irreversible.

Información durante la ejecución

Durante la ejecución de procesos de finalización no inmediata debe suministrarse al usuario suficiente información de progreso de la actividad en curso.

La entrada en modos de funcionamiento tales como inserción, modificación o consulta debe ser señalada al usuario mediante indicadores específicos.

Información posterior a la ejecución

Con posterioridad a la ejecución de la operación el usuario debe ser informado de su finalización correcta o errónea. En caso de finalización errónea de la operación, la aplicación debe dar información suficiente al usuario de cómo recuperar el error.

Se debe informar igualmente del número de elementos que han sido afectados por la operación (número de registros borrados, modificados, etc.).

Uso de colores y mecanismos de ayuda

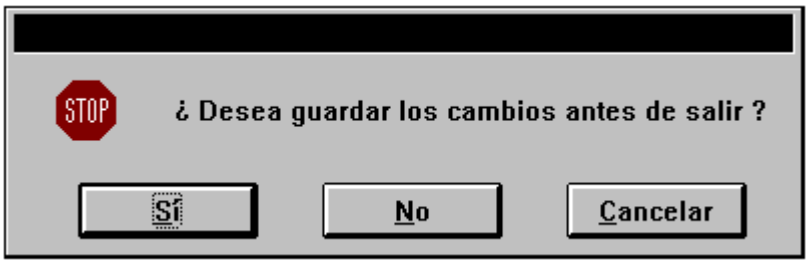
Mecanismos de información previa	Insertar allí donde sea posible y útil para el usuario mecanismo que suministren información por defecto o basada en anteriores sesiones o ejecuciones.
Valores posibles	Diseñar las ventanas de introducción de datos de forma que incorporen ayudas al usuario tales como listas de valores, valores por defecto, valores supuestos o calculados en función de datos introducidos anteriormente y desplazamiento automático.
Colores y tipos de letra significativos	Utilizar los colores y el tipo y tamaño de letra de forma significativa para el usuario. Por ejemplo, utilizar un color diferente para los mensajes de error y los mensajes de advertencia. Mantener el número de colores utilizados en una misma ventana a un mínimo (no más de tres colores) y seleccionar adecuadamente los contrastes y luminosidad.
Vocabulario técnico	Evitar la inclusión en la interfaz de usuario de palabras técnicas de uso no generalizado entre los usuarios.
Consistencia de interfaz	Las aplicaciones deben mantener una consistencia de tipo conceptual, lingüística, visual y operativa tanto dentro de ellas mismas como en relación con el resto de las aplicaciones. Se deben aplicar en la medida de lo posible las normas establecidas de interfaz gráfica de usuario del propio entorno operativo nativo (Windows, Motif u otro).
Ayuda en línea	Las aplicaciones deben suministrar ayuda en línea a nivel de campo en forma de texto de pantalla plena, así como una línea de ayuda cambiante para cada campo.
Salida de las aplicaciones	El usuario debe poder salir de la aplicación en cualquier punto en que se encuentre mediante procedimientos claros y sencillos.

Modos de trabajo de una pantalla

Como norma general, se podrán realizar cuatro operaciones básicas en cualquier pantalla de una aplicación (altas, bajas, modificaciones y consultas), aunque se considerará en cada caso la idoneidad de este modo de trabajo estándar:

Altas	Captura de datos en una o más líneas e inserción en la base de datos previa la ejecución de las acciones previstas por la lógica asociada al bloque o tablas afectadas.
Bajas	Borrado de filas completas de la tabla o tablas asociadas al bloque.
Modificaciones	Actualizaciones de uno o más campos de una o varias filas del bloque.
Consultas	Acceso a información de las tablas asociadas en función de los valores restrictivos establecido por los usuarios.

Las operaciones de alta, baja y modificación modifican los valores de los campos de un registro, por lo que si el usuario desea cambiar de registro, realizar una "acción de búsqueda" o salir de la pantalla, se presentará la siguiente pantalla :



Efectos resultantes:

Sí	Almacena los cambios en la base de datos y continúa la ejecución.
No	No almacena los cambios en la base de datos y continua la ejecución.
Cancelar	No almacena los cambios y vuelve al estado anterior.

Definición de Objetos de Diseño

Oracle Forms utiliza un serie de objetos diversos para el diseño de formas, que se pasa a definir y caracterizar a continuación.

Al final del apartado se muestra un ejemplo de pantalla de tipo maestro-detalle.

Elementos de la interface gráfica de usuario

Aplicación Conjunto de unidades ejecutables (Forms, Reports, otros) pertenecientes a una misma funcionalidad.

Módulo Programa o unidad ejecutable que contiene una forma, menú o biblioteca dispuesta para acceder y operar con información almacenada en la base de datos.

Dentro de una aplicación de Oracle Forms puede haber tres tipos de módulos:

Form Es una colección de objetos tales como bloques, elementos, ventanas, etc.

Menu Es una colección de objetos tales como menús, elementos de menús, etc.

Library Es una colección de procedimientos y funciones PL/SQL que son compartidas por otros módulos

Ventana Las ventanas son los marcos en los que se presenta la información.

Una forma puede tener varias ventanas asociadas.

Lienzo Superficie sobre la que se visualizan los objetos de una ventana.

Cada ventana puede estar compuesta por uno o más lienzos. Sin embargo, un lienzo sólo se puede asignar a una sola ventana.

Cada lienzo dispone de una propiedad denominada *viewport*o *view* (vista) que indica el área del lienzo que se visualiza en la ventana en tiempo de ejecución y las coordenadas en las que se posiciona el ángulo superior izquierdo de la vista.

Los lienzos definidos con el mismo tamaño que la ventana a la que pertenecen y con las coordenadas (0,0) se acoplan exactamente al área interior de la ventana.

Las vistas de lienzos pueden ser de uno de los dos tipos siguientes:

- Content view

  Es la vista básica que ocupa todo el espacio disponible de la ventana a la que está asignada.

  Puede haber varias *content view* en una misma ventana, pero en tiempo de ejecución sólo se podrá visualizar una de ellas a la vez.

- Stacked view

  No es una vista básica de la ventana, por lo que puede visualizarse

en las coordenadas (x,y) deseadas dentro de la ventana al mismo tiempo que la content view y otra u otras staked views.

Cada staked view oculta parcialmente las vistas sobre las que se superpone durante todo el tiempo que permanece abierta.

Bloque Un bloque es la referencia que une una forma o parte de una forma con una o más tablas o vistas de la base de datos.

Una forma se compone de uno o más bloques.

Hay dos tipos de bloques:

- Bloques de tabla base

  Contiene elementos provenientes de una sola tabla o vista de la base de datos. Estos elementos pueden ser elementos de la tabla base o elementos de control, tales como botones.

- Bloques de control

  Contiene elementos no relacionados con la base de datos o elementos de control.

Un bloque puede visualizar la información de una de las dos maneras siguientes:

- bloque monoregistro:

  Presenta la información de una sola fila de la tabla en una o más líneas de campos.

- bloque multiregistro:

  Conjunto de filas de una misma tabla presentados en pantalla en líneas dentro de un bloque.

Además, un bloque puede ser:

- Bloque maestro:

  Visualiza un registro maestro asociado con un registro de detalle. Suele adoptar la forma de bloque monoregistro.

- Bloque de detalle:

  Visualiza los registros asociados con un bloque maestro. Suele adoptar la forma de bloque multiregistro.

Registro Un registro de un bloque visualiza una fila de una tabla o vista de la base de datos y permite la consulta, modificación o inserción de sus valores.

Elementos (*items)* Es un objeto visual que visualiza datos de una columna de una tabla o vista de la base de datos.

Todos los elementos han de estar asociados a un bloque, provengan de la base de datos o no. De aquí en adelante se empleará el término *campo* como sinónimo de *elemento*.

Existen los siguientes tipos de elementos o campos:

Campo de texto (*text item)*	Campos de una o mas líneas donde se visualiza texto, fechas o números en una

variedad de formatos.

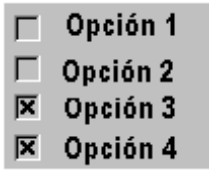
Un campo de texto puede visualizarse mediante una o más líneas.

Campo de consulta (*display item*)	En un campo de texto de sólo lectura. 
Campo de gráfico *(chart item)*	Es un campo que visualiza gráficos generados en Oracle Graphics.
Campo de imagen (*image item*)	Campo en el que se visualizan imágenes almacenadas en la base de datos o en un fichero de sistema operativo.
Campo de sonido (*sound item*)	Este campo permite al usuario cargar, grabar, editar, guardar y reproducir sonidos. Los datos de sonido se pueden almacenar en la base de datos o en un sistema de ficheros.
Control ActiveX (*control ActiveX*)	Es un contenedor para controles ActiveX
Árbol jerarquico (hierarchical tree)	Es un área que muestra los datos agrupados en forma de navegador estándar.

Elementos de tipo IGU

Son elementos complejos que presentan una interfaz gráfica de usuario específica. Normalmente están asociados a columnas de la base de datos.

Se distinguen los siguientes tipos:

Grupos radio (*Radio groups)*	Son campos donde se selecciona de forma excluyente una de las varias opciones presentadas (puede estar asociado a una columna de la tabla y sólo a una). 
Cajas de Comprobación (*Check Boxes*)	Campos utilizados para indicar un valor que puede tener dos estados (de forma no excluyente). 

Botones

Permiten realizar un proceso determinado sobre los datos de la pantalla actual o bien mostrar otras pantallas relacionadas con la pantalla actual.

Los botones pueden ser:

- Botones de texto:

  Contiene un texto descriptivo de su acción.

- Botones icónicos:

  Contienen una imagen descriptiva de su acción.

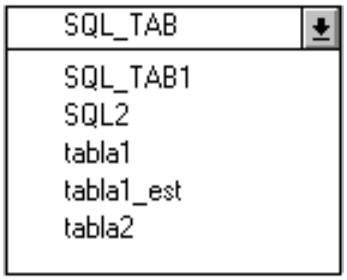
Elementos de listas

Son campos que visualizan listas de valores disponibles.

Se distinguen los siguientes tipos:

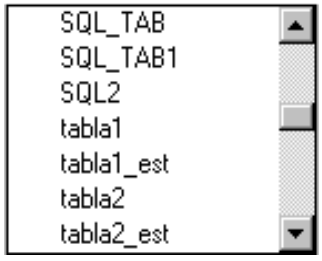
- Listas desplegables (poplists)

  Aparecen inicialmente como un campo normal. Cuando el usuario selecciona el icono de listas, se despliega una lista de valores disponibles.



- Cajas de listas (T-list)

  Aparece como un rectángulo que muestra los elementos de lista (cuando el area de visualización no lo suficientemente grande como para que aparezcan todos los elementos de la lista, aparece automáticamente una barra de desplazamiento a la derecha para consultar los elementos de lista restantes



- Cajas combo (combo *box*)

  Combina las características de las listas desplegables y de las cajas de listas, pero a diferencia de éstas, acepta no sólo valores seleccionados de la lista sino valores introducidos por el usuario.

  Aparece como una caja vacía con un icono específico a su derecha.

El usuario puede entrar el texto directamente en la caja o pulsar el icono y acceder a la lista de valores asociada a ese campo.

Listas de valores (LOV)

Son formas que visualizan valores y permiten a los usuarios la selección de un valor determinado. Son similares a las listas desplegables pero con un número de elementos muy superior.

Se despliegan cuando el usuario pulsa el botón de Lista de Valores de la barra de botones:

Las listas de valores tienen el siguiente aspecto :

Alertas

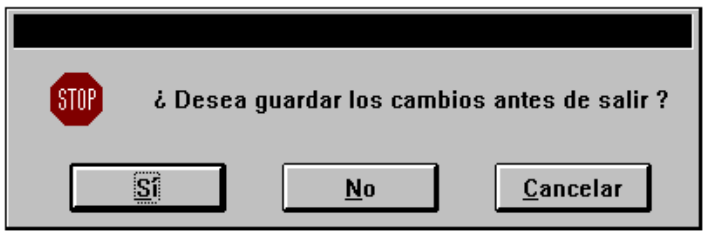
Ventanas que solicitan confirmación de acciones de los usuarios, les advierten de posibles problemas o les piden que tomen una decisión determinada.

Hay tres tipos de alertas:

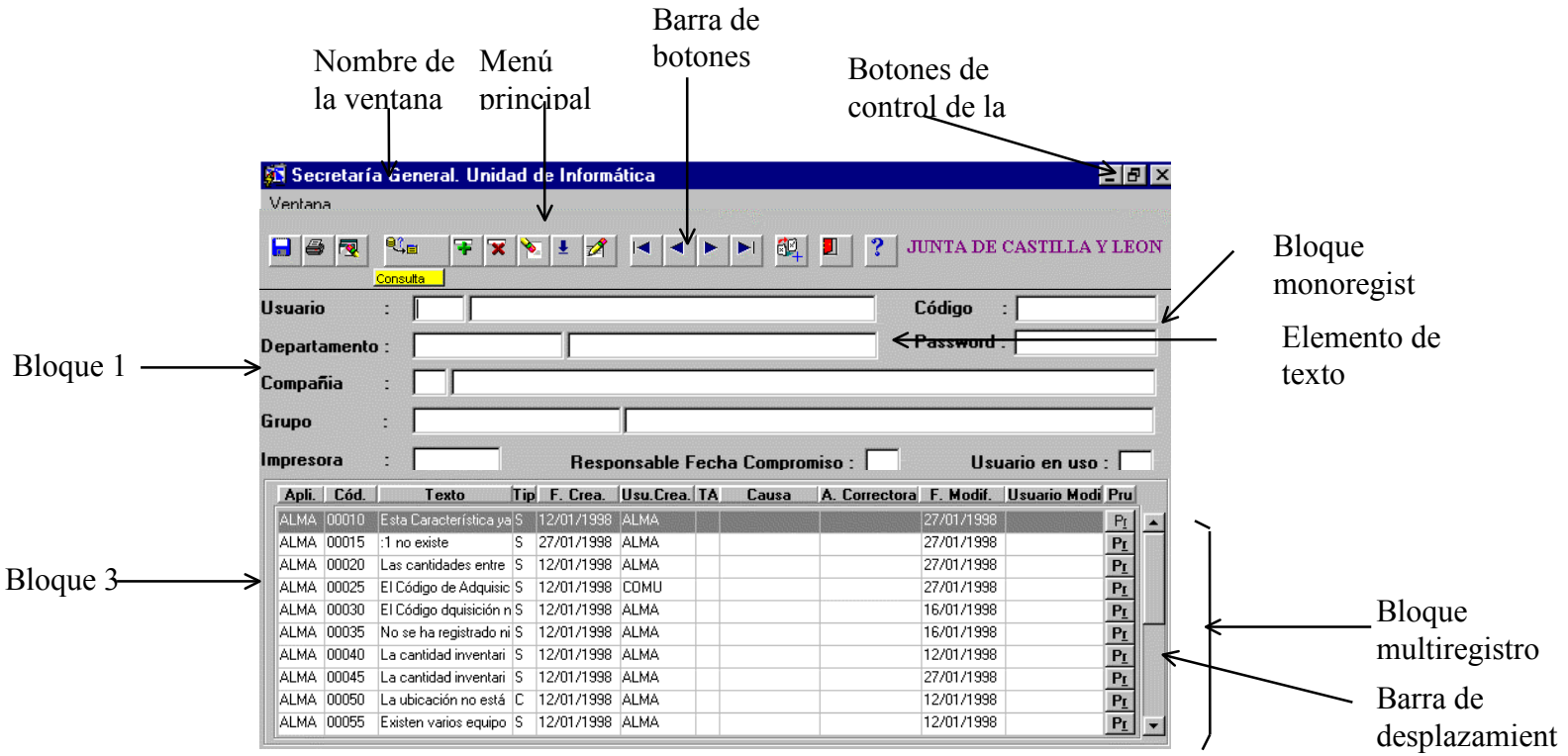
- Stop
- Caution
- Note

Cada una de ellas difiere de la anterior en el nivel de severidad del mensaje desplegado.

A continuación se muestra un ejemplo del tipo *Stop*:



Ejemplo de pantalla maestro-detalle



Elementos del entorno de desarrollo

Además de los elementos descritos anteriormente existe una serie de objetos no directamente relacionados con la interfaz gráfica de usuario, sino con el entorno de desarrollo proporcionado por Oracle Forms, que se describen a continuación:

Clases de propiedades

Es un objeto que contiene una lista de propiedades y sus valores. Una vez creada puede ser asignada a diferentes objetos de una aplicación, los cuales heredan todas las propiedades de la clase de propiedades asignada que son compatibles con la propia naturaleza del objeto al cual ha sido asignada.

Una clase de propiedades puede estar definida en base a otra clase de propiedades. Por ejemplo, se puede definir una clase de propiedades para los campos de texto y otra clase de propiedades basada en ésta que gobierne los campos de texto para fechas.

Atributos visuales

Son conjuntos de propiedades de tipos de letra, colores y patrones que pueden ser asignados a los objetos de una aplicación.

Hay tres tipos de atributos visuales:

- Custom

  Atributos visuales aplicados a un objeto individual.

- Default

  Atributos visuales predefinidos en Oracle Forms.

- Named

  Atributos visuales definidos en tiempo de desarrollo que pueden ser asignados a diferentes objetos de una aplicación.

Grupos de objetos

Permiten compartir objetos comunes entre diferentes módulos. Un grupo de objetos se define dando un nombre a una serie de objetos relacionados entre sí por tener la misma funcionalidad.

Por ejemplo, una funcionalidad determinada puede incluir una ventana, un lienzo, varios bloques y campos, así como determinados disparadores.

Se puede crear un grupo de objetos con estos elementos e incluirle en todos las formas que lo necesiten mediante una simple operación de arrastrar y soltar.

Grupos de registros

Es una estructura de datos interna de Oracle Forms similar a la estructura de columnas de una tabla.

Sirven como áreas de memoria de dos dimensiones.

Parámetros

Es un mecanismo que permite el paso de valores al iniciar la ejecución de una forma. Pueden adoptar el tipo de dato CHAR, NUMBER o DATE.

Disparadores

Son bloques de código PL/SQL que permiten completar la funcionalidad de las aplicaciones.

Pueden implementarse en los siguientes niveles:

- Forma
- Bloque
- Elemento

Esquema de relación entre elementos

La siguiente tabla muestra la relación que puede haber entre los principales elementos de una forma:

	Lienzos	**Bloques**	**Elementos**	**Disparadores**
Forma	X	X		X
Bloque			X	X
Elementos				X

Nomenclatura de los objetos de diseño

En general, los nombres de los objetos de un módulo de Oracle Forms se generarán añadiendo a una palabra descriptiva de la funcionalidad del objeto un sufijo indicativo del tipo de objeto, tal y como se detalla a continuación:

Formas Evitar nombres crípticos o codificados. Elegir nombres de formas aplicando las normas establecidas en el capítulo de *Diseño de Módulos*.

Ventanas El nombre de cada ventana se formará mediante el cualificador "WIN_" y un nombre específico, en función del tipo de ventana.

En el caso de la ventana principal de la aplicación (aquélla que contiene la botonera y otros objetos genéricos) deberá identificarse mediante la siguiente sintaxis:

WIN_<alias_aplicación> (alias largo)

En las ventanas en las que se pueda asociar claramente un bloque principal, el nombre de la ventana se derivará del nombre de la tabla base del bloque principal de la ventana.

Si la ventana no corresponde a un bloque de tabla base, en lugar del nombre abreviado de la tabla base se tomará un nombre que exprese la funcionalidad de la ventana.

Lienzos (*canvas)* El nombre de cada lienzo se formará mediante el cualificador "CVS_" y un nombre específico, en función del tipo de lienzo.

Los nombres específicos de los lienzos deberán estar formados por el nombre específico de la ventana.

En el caso de que una ventana disponga de varios lienzos el nombre así formado se completará con un sufijo numérico que exprese la secuencialidad de cada lienzo dentro de la misma ventana.

Bloques Los nombres de los bloques se formarán como sigue:

Bloques de tablas base	Se utilizará el nombre abreviado de la tabla base prefijado por "BLK_".
Bloques de control	El bloque de control de una ventana se denominará "BLK_ CONTROL ".

Elementos (*items*) El nombre se formará mediante una palabra que exprese la funcionalidad del elemento.

¡mirar designer! Si el elemento no pertenece a una tabla base el nombre reflejará lo más fielmente posible su funcionalidad.

Se le añadirá el siguiente prefijo, indicando el tipo de elemento :

TXT_	Text item
DIS_	Display item
IMG_	Image
BTN_	Button

CHK_	Check Item
RGR_	Radio Group
LST_	List item
OLE_	OLE container
VBX_	VBX control
USR_	User area

Grupos de registros

Se utiliza el prefijo "RGP_" seguido de un texto descriptivo de su funcionalidad.

Si el grupo de registros corresponde a una LOV determinada el texto descriptivo será lo más similar posible al utilizado en la lista de valores.

Botones que llaman a otras formas

Se denominarán mediante la sintaxis:

BTN_<*nombre_módulo*>

Otros elementos

Su nombre se determina mediante uno de los prefijos reseñados a continuación seguido de un texto descriptivo de su funcionalidad.

Botones	BTN_
Listas de valores	LOV_
Grupos de objetos	GOB_
Parámetros	PRM_
Clases de propiedades	CLS_
Alertas	ALR_
Atributos visuales	AVS_

En caso de objetos del tipo anterior basados en columnas de tablas de la base de datos se aplicará la sintaxis:

<*prefijo*>_<*nombre_columna*>

Normas de Diseño del IGU

Las normas relativas a la interfaz gráfica de usuario definidas a continuación deben interpretarse como una guía, ya que en último término, es la propia funcionalidad de cada elemento de la aplicación la que debe determinar la forma de presentación al usuario más adecuada.

Navegación

Para la navegación en las pantallas se deberán aplicar las siguientes normas (además de las definidas en cada apartado específico):

Uniformidad

El funcionamiento de las pantallas deberá ser lo más similar en todas ellas.

Navegación por ratón y teclado

Se deberá asegurar en cada pantalla la posibilidad de una navegación suficientemente operativa mediante el uso exclusivo del teclado (ratón deshabilitado), mediante las teclas de movimiento de cursor, tabulador, retorno y teclas de función que permita el acceso al menú de la pantalla.

Se podrá navegar con el ratón sobre todos los campos o bien a través de la barra de botones.

Se estudiará cuidadosamente en cada una de las formas desarrolladas los efectos de una navegación con ratón no secuencial o campo a campo, restringiéndose si es necesario tal navegación si se prevé una lógica complicada o efectos no deseados.

Aplicaciones

Entorno de desarrollo y explotación

Definición del entorno de desarrollo y prueba

Antes de acometer un proyecto en Oracle Forms se establecerá el ambiente o entorno de desarrollo y prueba.

Estructura de directorios de desarrollo

Se dispondrá en un directorio accesible el conjunto de bibliotecas 1 (plantillas de Forms, PL/SQL, etc.) y módulos reutilizables.

La **estructura del directorio** de una aplicación en desarrollo será la siguiente:

`Analisis`	Documentos de análisis de la aplicación.
`Ayuda`	Ficheros .necesarios para la creación de las ayudas
`CargaInicial`	Contiene los ficheros .CTL, .LOG, .BAD y de datos desarrollados para la conversión de datos mediante SQL Loader.
`Dbscript`	Contiene los ficheros de comandos usados para la creación de la base de datos y todos sus objetos (base de datos, tablas, restricciones, privilegios, roles, etc.).
`Discoverer`	.dis o libros de trabajo a desarrollar por el usuario.
`Documentacion`	Documentación aportada por el usuario (decretos, órdenes, etc.) en los que se basa la aplicación.
`Fuentes`	Contiene todos los siguientes ficheros: • ficheros .FMB con el código binario de las formas utilizadas en la aplicación. • ficheros .FMX con los ejecutables de las formas de la aplicación. • ficheros .MMX con los ejecutables de menús de la aplicación. • ficheros .RDF con el fuente de los informes de la aplicación. • Librerías fuentes .PLL y compiladas .PLX de procedimientos y funciones PL/SQL comunes y compartidos a tarvés de las aplicaciones generadas. • Ficheros .REP con los ejecutables de los informes de la aplicación. • Ficheros .ICO con los iconos que pueden ser utilizados en las formas.

1 En inglés, *libraries.*

	• Para IAS los iconos deben ser Ficheros .GIF.
Impresos	Documentos de Word que va a emitir la aplicación.
Manuales	Manuales de la aplicación.
Portal	Los ficheros que se van a ver en el portal de desarrollo.
Procedimientos	Contiene el código fuente de los procedimientos, paquetes y funciones almacenados.
SQL	Consultas para la aplicación.
Triggers	Contiene el código fuente de los disparadores de la base de datos.
Vistas	Vistas de la aplicación.

Uso de subdirectorios en aplicaciones complejas

En aquellas aplicaciones en las que por su complejidad, gran número de módulos o modularidad lo requiera, se utilizarán subdirectorios para organizar las diferentes áreas de la aplicación.

En este caso, se habrá de prever el uso de directorios globales a toda la aplicación o específicos a áreas determinadas de la misma.

Estructura de directorios de explotación

Se dispondrá en un directorio accesible el conjunto de bibliotecas 1 (plantillas de Forms, PL/SQL, etc.) y módulos reutilizables.

La **estructura del directorio** de una aplicación en explotación será la siguiente:

Discoverer	Ficheros generados en Discoverer para el usuario.
fuentes	Contiene todos los siguientes ficheros: • ficheros .FMB con el código binario de las formas utilizadas en la aplicación. • ficheros .MMB con los ejecutables de menús de la aplicación. • ficheros .RDF con el código binario y fuente de los informes de la aplicación.
ejecuta	• ficheros .FMX con los ejecutables de las formas de la aplicación. • ficheros .MMX con los ejecutables de menús de la aplicación. • ficheros .REP con los ejecutables de los informes de la aplicación. • Ficheros de ayuda • Iconos específicos de la aplicación. • Ficheros .PLL con las bibliotecas PL/SQL de

[1] En inglés, *libraries.*

	procedimientos y funciones comunes y compartidos a través de las aplicaciones generadas
Impresos	Documentos de Word que va a emitir la aplicación.
Modelos	Impresos Word que va a emitir la aplicación.

Paso de desarrollo a explotación

Queda pendiente de definir.

El módulo básico de una aplicación

Elaboración del módulo básico de la aplicación

Al inicio del desarrollo de cada aplicación se establecerá un módulo de forma denominado COMUN.FMB que estará basado en el módulo básico o plantilla de todas las aplicaciones. Este módulo actuará como repositorio para el diseño de la interfaz estándar de la aplicación y características adicionales

Compilación de la forma básica

Esta forma ha de compilarse cada vez que se modifique, ya que actuará como biblioteca maestra de estándares para objetos compartidos tales como ventanas, alertas, botones de listas de valores, elementos de texto y otros.

Cada vez que se produzca un cambio en la forma COMUN.FMB de objetos que ya han sido previamente utilizados en otras formas se deberán compilar las formas de la aplicación que hacen referencia a los objetos modificados, a fin de producir nuevos ejecutables .FMX que reflejen los cambios.

Elementos reutilizables

Los elementos reutilizables de cualquier tipo se insertarán en esta forma y se clasificarán según se deban copiar o referenciar.

Clases de propiedades de la forma COMUN

- Ver el punto 2.1 del documento estandares199202.doc

Definición de grupos de objetos

Definición de grupos de objetos

Es importante que todos los objetos creados se organicen en grupos de objetos que realizan una tarea similar o que se relacionan entre sí, a fin de facilitar el proceso de copia o de referencia en las formas a construir.

Plantillas de ventanas comunes

Por ello, se ha de establecer al inicio del proyecto las plantillas comunes para las ventanas tipo de la aplicación. Partimos siempre del módulo PLANTILLA.FMB.

Estas plantillas deberán incluir objetos que referencien a los objetos y propiedades de la forma COMUN.FMB a fin de conservar a lo largo de todas las ventanas de la aplicación un aspecto y funcionamiento básico común.

Definición de clases de propiedades

Se definirán clases de propiedades para ser asignadas a los diferentes tipos de objetos. Estas clases de propiedades serán la clave para el mantenimiento de la consistencia de tipos de letra, tamaños y otras características de visualización a lo largo de todas la aplicación.

Biblioteca de funciones y procedimientos comunes

Se utilizará la biblioteca `lib` que actuará como repositorio de trabajo para almacenar las unidades de programa PL/SQL compartidas de la aplicación.

Referencias a elementos de disparadores o bloques PL/SQL

En todos los disparadores y bloques PL/SQL hay que mencionar de forma no ambigua el bloque y el elemento al que se hace referencia desde el programa, mediante la sintaxis:

:<*bloque*>**.**<*elemento*>

Modificación de plantilla básica

La introducción de nuevos objetos en esta forma durante el desarrollo del proyecto deberá ser autorizada por el administrador técnico o responsable del proyecto.

Atributos visuales

Definición de atributos visuales

Se definirán diferentes atributos visuales en función del tipo de objetos que se desea generar en la aplicación para facilitar las modificaciones y la estandarización de la apariencia (tipo de letra, patrón y color) de los objetos a lo largo de una misma aplicación.

Antes de definir los atributos visuales de un objeto, hay que examinar si aplica alguno de los ya creados, o si se ha de proceder a la modificación de alguno ya existente o a la creación de uno nuevo.

Atributos visuales predefinidos

Evitar la asignación a objetos de la aplicación de los atributos visuales establecidos por defecto en Oracle Forms o atributos visuales a medida (tipo *custom*).

Utilizar un atributo visual con nombre de entre los que están creados, sobre todo si se desean crear aplicaciones portables entre diferentes entornos.

Lista de atributos visuales aplicables a elementos de Forms:

Nombre	Elemento	Tipo	Estilo	Tamaño (Puntos)	Color
AVS_ALERT	Alertas utilizadas para mostrar mensajes	Ms Sans Serif	Negrita	8	Negro sobre Darkgray
AVS_NORMAL = CG$ITEM	Elementos de texto que no sean de visualización.	Ms Sans Serif	Normal	8	Negro sobre blanco
AVS_REGISTRO_ACTUAL = CG$CURRENT_RECORD	Registro actual de un bloque multiregistro	Ms Sans Serif	Normal	8	Blanco sobre r88g100b88
AVS_LOV = CG$LOV	Listas de Valores	Ms Sans Serif	Normal	8	Blanco sobre gray
AVS_CANVA = CG$PUSH_BUTTON	Botones, checks y radio groups	Ms Sans Serif	Normal	8	Negro sobre gray
AVS_VERDE	Títulos de las partes que constituyan una pantalla	Ms Sans Serif	Negrita	8	Blanco sobre darkcyan
AVS_PROTEGIDO=CG$DISPLAY_ITEM	Elementos de texto de visualización.	Ms Sans Serif	Normal	8	Darkgray sobre blanco
AVS_TEXT	Etiquetas que quieran ser tratadas como campos de Forms.	Ms Sans Serif	Negrita	8	Negro sobre Gray

Clases de Propiedades

Definición de clases de propiedades

Se deberá hacer un amplio uso de clases de propiedades definidas en función de los diferentes tipos de objetos que se manejan en la aplicación.

Definir una jerarquía de clases de propiedades

Establecer una adecuada jerarquía de clases de propiedades, definiendo clases genéricas en las que se apoyen las clases específicas que se vayan a utilizar para asignar a objetos de cada tipo. Evitar sin embargo, jerarquías complejas de clases de propiedades.

Cómo definir clases de propiedades

Basar la definición de atributos visuales de las clases de propiedades creadas en atributos visuales estándar previamente definidos.

Al definir cada clase de propiedad determinar aquellas propiedades que no se deben incluir dado que se desea modificarlas dinámicamente en tiempo de ejecución.

Si tras la asignación de una clase de propiedad a un objeto se observa la necesidad de modificar alguna de las propiedades heredadas, se deberá decidir la conveniencia de crear una nueva clase de propiedad o modificar las propiedades en el propio objeto que las recibe.

Reutilización de módulos

Copiado o referenciado de objetos

Se determinará la conveniencia de reutilizar objetos ya definidos en otros módulos de la aplicación, o de otras aplicaciones mediante las funcionalidades de copiado o referencia de objetos.

Copiado de objetos

En el caso de necesitar una copia se deberán estudiar las propiedades que pueden o no pueden ser cambiadas tras ejecutar la copia a fin de determinar si procede.

Referenciado de objetos

Utilizar la opción de referenciar objetos, en vez de copiarlos, siempre que se pueda, a fin de facilitar la aplicación de estándares y el mantenimiento de las aplicaciones.

A fin de evitar posibles problemas de borrado de objetos utilizados en otros módulos, el método de referencia se utilizará sólo referenciando objetos en la forma base de la aplicación.

En caso contrario, se llevará un cuidadoso control de qué objetos se referencian en cada módulo y a qué módulos apuntan los objetos.

Al referenciar objetos se deberá tener el mismo sistema de coordenadas en el módulo fuente y destino, ya que Oracle Forms no convierte automáticamente las coordenadas de posición y tamaño.

Se deberá eliminar la referencia al camino (*path*) del módulo fuente en el caso de que éste no esté almacenado en la base de datos ORACLE y definir dicho camino como parámetro de Forms en el fichero ORACLE.INI de la aplicación.

Creación de grupos de objetos

Se estudiará la conveniencia de crear grupos de objetos (*Object Groups*) que empaqueten objetos relacionados que se deseen copiar o referenciar en otros módulos de una forma simple.

Funciones y procedimientos de usuario

Las funciones y procedimientos comunes a la aplicación definidos por el usuario se deberán colocar en una biblioteca objetos de tipo PL/SQL que se pueda adjuntar a los módulos a desarrollar.

Las referencias a esta librería en el módulo no deberán incluir el camino (*path*) de los objetos de la biblioteca.

Esta biblioteca se denominará <*alias_aplicación*>_GEN.PLL y ha de ser común o al menos consistente con las correspondientes al resto de las aplicaciones.

Portabilidad de pantallas

El diseño de pantallas compatibles con el modo carácter, terminales monocromo y multilenguaje exige tener en cuenta las siguientes normas adicionales:

Tipo de letra y colores

Utilizar exclusivamente el tipo de carácter `Courier New`.

El uso de colores para significar estados de información (por ejemplo, saldos negativos) debe tener un distintivo secundario reconocible en pantallas monocromas.

Ubicación de objetos en la rejilla

Todos los objetos de la pantalla deben colocarse de forma que coincidan con la rejilla para caracteres, incluidas la descripciones de campos, líneas de separación y elementos IGU (check boxes, texto de fondo de pantalla (*boilerplate*), etc.)

Todas las pantallas deben caber en una rejilla de 80 x 24 caracteres

Tamaño de los objetos

Todos los objetos de la pantalla deben dimensionarse de forma que ocupen un carácter por posición.

Navegabilidad por teclado

La pantalla en su totalidad ha de ser navegable exclusivamente mediante el teclado (teclas de movimiento de cursor, tabulador, retorno, borrado, etc.).

Toda la funcionalidad ofrecida por los botones debe encontrarse también en los menús de cada pantalla.

Zonas de desplaza-miento

Limitar el uso de zonas de desplazamiento.

Espacio para traducción

Todos los elementos susceptibles de ser traducidos han de prever un 15% a un 20% de espacio disponible para su posible expansión al ser traducidos a otros lenguajes.

Módulos de pantallas

Las propiedades de los módulos establecen el marco general para la apariencia de cada forma.

Propiedades básicas

Información de coordenadas	Sistema de coordenadas	Real
Atributos funcionales	Mouse Navigation Limit	Form
	Validation level	Default

Cabecera y pie de pantalla

Las pantallas no llevarán cabeceras ni pies. Se utilizará la barra de menú para escribir el título de la aplicación y el de la pantalla en curso.

Información *Acerca de ..*

Se habilitará una opción de menú denominada *Acerca de ...* para dar información de identificación de la pantalla u otra información útil.

Títulos de elementos de módulos

Los títulos de pantallas, bloques y descriptores de campo deberán llevar en mayúscula la primera letra de todas las palabras, excepto artículos, conjunciones, preposiciones y restantes palabras exigidas por la sintaxis gramatical.

Ventanas

Información a visualizar

Cada ventana mostrará el título de la ventana desde la que se abre (o el título de la aplicación en caso de ser la primera ventana) y su propio título.

Barras de desplaza-miento

Las ventanas no dispondrán de barras de desplazamiento propias; sin embargo, los bloques u otros elementos de la ventana si podrán tenerlos.

Tamaño por defecto

Se habilitará un procedimiento común para entrar en la aplicación, de forma que al acceder a la ventana MDI [1]se abra y se maximice al tamaño prefijado, así como las restantes ventanas que el usuario vaya abriendo durante la ejecución de la aplicación.

Todas las ventanas se harán plenamente visibles al ser abiertas y podrán ser movidas de lugar, aunque las posiciones a las que se muevan no serán guardadas al salir de la forma.

Tamaño

	Ancho	Alto
Máximo	7,8”	5,0’’
Mínimo	2’’	2’’

Una ventana podrá tener cualquier tamaño comprendido entre ambos límites, aunque deberá procurarse que su tamaño se conforme a unos pocos tipos idénticos o parecidos.

Diseño de ventanas

La disposición de la información en las ventanas de la aplicación deberá seguir la visión lógica que el usuario tiene de la misma, evitando un diseño demasiado cercano al diseño físico de la base de datos.

Ventanas con muchos elementos

Cuando se necesite visualizar en la pantalla un gran número de campos pertenecientes a una tabla se deberán agrupar mediante el mecanismo de grupos de elementos, a fin de guiar al usuario en la búsqueda de un campo determinado.

Bloque de control

Cada ventana tendrá su propio bloque de control.

Todos los botones se colocarán en el bloque de control.

Lógica de

La ubicación de los componentes de una pantalla se deberá orientar

1 Cuando una aplicación se ejecuta bajo MS Windows, el usuario interactua con “documentos” visualizados en una ventana marco. La ventana marco puede ser de tipo SDI (*single document interface*) y MDI (*multiple document interface*). En ambos casos, el sistema operativo se encarga de la mayor parte de la gestión de la ventana (mover, cambiar de tamaño, cerrar, etc.)

diseño siguiendo una secuencia de tareas de arriba hacia abajo.

Los bloques, grupos de elementos y campos han de disponerse en lo posible, por orden de precedencia, de izquierda a derecha y a continuación, de arriba abajo. y de izquierda a derecha.

Espaciado de elementos No se ha de desperdiciar el espacio de la pantalla. El tamaño de las ventanas será el estrictamente necesario.

Utilizar espacios vacíos a fin de agrupar la información y facilitar al usuario el reconocimiento de la información.

Alineación de elementos La alineación de objetos entre sí deberá adecuarse a la naturaleza de cada pantalla pero deberá ser consistente en los niveles de ventana, aplicación y entre aplicaciones.

Márgenes Por lo menos, se dejarán un espacio equivalente a un carácter entre el marco de la ventana y los elementos del lienzo. En la medida de lo posible se dejará también una línea en blanco en la parte superior e inferior de la ventana.

Recuadros En caso de ser necesario para clarificar la situación de los elementos de la venta al usuario, se utilizarán líneas y rectángulos que los separen o agrupen.

Lienzos *(canvasses)*

Lienzo principal

Cada ventana tendrá su propio lienzo principal, que ocupará todo el tamaño de la ventana.

El lienzo básico se marcará como visualizable al entrar en la ventana.

Lienzos apilados

Podrán visualizarse lienzos apilados (*staked canvasses*) sobre el lienzo principal.

Los lienzos apilados tendrán justo las medidas necesarias para contener los elementos que han de mostrar.

Sólo se marcará como visualizable el lienzo que se ha de mostrar cuando su ventana se abre inicialmente.

Bloques

Presentación

Cada bloque mostrará un título indicativo de su función.

Ubicación de los elementos

En cada bloque se situarán los campos referentes a una misma tabla base y los campos adicionales que se requiera.

Bloque tipo registro

En los bloques de tipo registro los campos estarán alineados a la izquierda, en la medida de lo posible.

Bloque multiregistro

En los bloques multiregistro, los campos estarán encolumnados y alineados con respecto a su cabecera.

De esta colección de líneas la fila actual es la que tiene el foco, lo cual se indica por tener el fondo del texto de diferente color que el resto.

En los bloques multiregistro el registro actual se distinguirá del resto por tener el fondo del texto de otro color (color por defecto: verde).

Navegación

La navegación entre campos de un mismo bloque se realizará con las teclas <TAB> o <ENTER> o con el ratón.

Para acceder a los datos de otro bloque será necesario moverse con el cursor al bloque deseado, o bien usar los botones <BLOQUE ANTERIOR> O <BLOQUE SIGUIENTE> de la barra de botones.

Descripciones de elementos

Colocar los títulos y las descripciones arriba o a la izquierda del elemento que están describiendo.

Los descriptores de campos de bloques tipo registro estarán colocados a la izquierda del campo, mientras que en los bloques de tipo multiregistro los descriptores se colocarán sobre cada columna de campo y alineados de igual forma que lo están los datos a los que describen.

Señalización del contexto de trabajo

El usuario debe tener siempre claro el registro en cuyo contexto está trabajando en cada momento.

Para ello se utilizarán los siguientes mecanismos:

Bloques de detalle	Si el bloque de detalle está situado en una ventana diferente en la que se encuentra el bloque maestro, se mostrará información del contexto en el bloque de detalle.
Bloques multiregistro	Si el bloque maestro es multiregistro, al pasar al bloque de detalle el registro actual del bloque maestro debe quedar claramente indicado.

Desactivación de elementos

En el caso de que el cursor se sitúe sobre un bloque, registro o campo que tenga opciones de menú o botones no disponibles, las correspondientes opciones del menú o botones han de mostrarse desactivadas.

Campos

Atributos visuales según tipos de campos

Se utilizarán atributos visuales diferentes según los campos sean modificables o de mera consulta:

Campos de texto	Fondo:	blanco
	Primer plano:	negro
Campos de consulta	Fondo:	blanco
	Primer plano:	azul

Navegación en campos de consulta

Los campos de consulta no serán navegables (excepto si el usuario puede establecer valores en modo de consultas).

Desactivación de campos dependientes de otros campos

Los campos cuya introducción de valores sólo es posible si se han introducido valores en algún otro campo (por ejemplo, un campo que sólo puede ser introducido tras la introducción en el correspondiente campo maestro) deberán adoptar el color de fondo del lienzo hasta que el campo del que dependen no tenga un valor válido.

Cuando se desactive un campo determinado se deberán desactivar también los campos asociados o vinculados a éste (por ejemplo, los campos que contiene su descripción o campos calculados en función del campo desactivado).

Campos con lista de valores

Los campos con lista de valores disponible deben mostrar esta posibilidad, por lo menos cuando el cursor entra en su zona.

Indicador de campo con editor

Los campos de tipo texto deben mostrar siempre un indicador de editor disponible, por lo menos cuando el cursor entra en su zona.

Campos de texto

Todos los campos de texto deberán llevar descriptores asociados.

Los descriptores no deberán llevar separadores tales como ":" sino uno o mas espacios.

Los campos con idéntica funcionalidad han de tener idéntico descriptor en todos los bloques en los que se utilicen. Los descriptores de campos de funcionalidad parecida deberán estar redactados de forma coherente.

Grupos de campos

Estudiar la conveniencia de agrupar los campos con dependencia funcional en grupos separados por cajas, de forma que permita al usuario una más fácil identificación.

Pantallas de introducción de datos

En las pantallas de introducción masiva de datos evitar cualquier mecanismo que no pueda ser operado desde el teclado y sin mirar a la pantalla.

Por ejemplo, un *check box* puede ser operado desde el teclado pero el

usuario tendrá que mirar a la pantalla obligatoriamente para ver el valor que presenta.

Uso de mayúsculas y minúsculas

Los campos de texto permitirán la introducción de valores en mayúsculas o minúsculas indistintamente, recogiendo así los valores que el propio usuario determine, excepto cuando funcionalmente no sea aconsejable o posible.

Campos de texto multilínea

Los campos de texto de varias líneas deberán disponer de una barra de deslizamiento y se tendrán la propiedad *Wordwrap* [1] asignada con el valor *Word*.

La longitud asignada a este tipo de campos deberá ser dos o tres caracteres superior a la longitud de almacenamiento para permitir en todos los caso la lectura completa del texto.

Pantallas de consulta

En las pantallas de consulta todos los campos serán navegables.

[1] Propidad que permite que las líneas de un campo de texto se plieguen al final de línea coincidiendo con palabras completas, evitando de este modo cortes a mitad de palabras.

Listas de valores

Cuándo se utilizan

Se creará una lista de valores en todos los campos correspondientes a claves primarias y ajenas y se dispondrá un botón icónico en la barra de botones que permita su apertura mediante el ratón.

Campos opcionales

Las listas de valores de campos no obligatorios deben incluir una fila en blanco para permitir que el usuario no se vea obligado a admitir un valor de entre los presentados.

Listas restringidas

Las listas de valores se abrirán restringiendo sus valores en función de los valores actuales del contexto en el que se abran, si procede.

Selección de valores

Para seleccionar un valor de la lista de valores e insertarlo en el campo actual será necesario pulsar <ACEPTAR>. Pulsando <CANCELAR> se cerrará la lista de valores sin hacer nada.

Opción de búsqueda

En el caso de que la tabla sobre la que se construye la lista de valores presente un elevado número de filas y con el fin de restringir la lista y agilizar la respuesta del sistema se podrán introducir valores en el campo "Buscar" para restringir la búsqueda.

Grupos radio (*Radio groups)*

Presentación Se deberán disponer dentro de un grupo de elementos, con el nombre del grupo en la esquina superior izquierda y rodeado por una caja. Las diversas opciones se dispondrán preferentemente de forma vertical.

Otros valores En los casos en los que se prevea que una consulta pueda arrojar para ese campo valores diferentes de los señalados dentro de las opciones estáticas se implementará una opción adicional denominada *Otros* o similar, que se deshabitará para evitar que el usuario lo seleccione fuera del modo de consulta.

En bloques multiregistro No se utilizará cuando el campo forma parte de un bloque multiregistro.

Cajas de Comprobación (*Check boxes)*

Uso Utilizarlo únicamente cuando sólo aplica un valor en una situación del tipo S*i/No* y es obligatoria su asignación, o bien cuando pueda admitir un sólo valor y se permita la asignación de un valor nulo.

Presentación El descriptor se colocará a la derecha de la caja y ésta se alineará en vertical respecto a los campos de texto superiores, excepto si el descriptor de la caja de comprobación es muy largo, en cuyo caso la caja se alineará verticalmente respecto al descriptor de los campos de texto superiores.

Botones

Tecla asociada

Todos los botones deberán tener una tecla rápida asociada (hot-key) que será aquella que aparezca subrayada en el nombre del botón.

Botones inactivos

No todos los botones son operativos desde todas las pantallas en todo momento.

Los botones inactivos aparecerán en la pantalla con los colores "nublados". Al pulsarlos no realizarán ninguna función.

Botones activos →  ← Botones inactivo

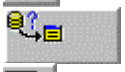
Ayuda de burbuja

En caso de que el usuario no recuerde la función realizada por cada icono podrá visualizar una ayuda de burbuja deteniendo unos segundos el puntero del ratón sobre el icono del botón deseado.

Barra de botones

Todas las pantallas dispondrán de una barra de botones estándar para la realización de las operaciones normales:

Funciones de cada botón :

	Guardar últimas operaciones realizadas: Modificaciones, altas y bajas de datos y registros.
	Imprimir pantalla u otros documentos referidos al registro que se muestra en pantalla.
	Limpiar todos los bloques de pantalla
	Consultar
	Añadir registro nuevo
	Borrar registro actual
	Limpiar datos del registro
	Lista desplegable de valores posibles
	Editar campo
	Primer registro
	Registro anterior
	Registro siguiente
	Ultimo registro
	Duplicar registro
	Salir
	Ayuda

Tras pulsar el botón <CONSULTAR> se pasa al modo de consulta, apareciendo los siguientes botones disponibles:

	Ejecutar consulta	Ejecutar la búsqueda, una vez seleccionado el filtro sobre la consulta.
	Cancelar consulta	Cancelar la búsqueda, si el usuario no desea realizar la consulta en curso.

Ubicación de la barra de

Se usará preferentemente una sola barra de botones en cada ventana, en la que se colocarán, además de los botones estándar, aquéllos necesarios para

botones cada ventana específica.

Botones de ventanas modales [1] En las ventanas modales se dispondrá como mínimo del botón de *Salir,* que se situará en la esquina inferior derecha de la ventana.

[1] Una ventana modal necesita siempre alguna respuesta del operador para poder ser cerrada y continuar el proceso.

Orientación a objetos

Aunque Oracle Forms no es una herramienta orientada a objetos, dispone de una serie de características de desarrollo de aplicaciones orientado a objetos.

Cualquier aproximación al desarrollo orientado a objetos supone menores tiempos de desarrollo y una mayor calidad del producto obtenido.

Se incluye a continuación una recapitulación de las **normas** relacionadas con la orientación a objetos:

Uso de clases de propiedades

Crear clases de propiedades aplicables a los campos de texto, botones, cajas de comprobación, grupos radio, lienzos y otros objetos de la aplicación, a fin de evitar su definición uno a uno y asegurar que todos los objetos tienen el mismo esquema de color, tipo de letra, dimensiones y aspecto visual.

Disparadores en clases de propiedades

Incluir disparadores en las clases de propiedades creadas, a fin de extender los beneficios de la orientación de objetos a la lógica de los objetos definidos mediante dichas clases.

Jerarquías de clases de propiedades

Crear una jerarquía de clases de propiedades y subclases que permita beneficiarse de los mecanismos de herencia de propiedades multinivel.

Las clases genéricas se deben combinar con las subclases específicas para crear una jerarquía de propiedades consistente.

Por ejemplo, se puede crear una clase de propiedad denominada *cls_boton_basico* con las características visuales básicas aplicables a todos los botones y a continuación crear una clase denominada *cls_boton_salir* que contenga información sobre la posición, fichero de icono y el disparador asociado al dicho botón.

Características de herencia de propiedades

Aprovechar la característica de Oracle Forms mediante la cual sólo se heredan las propiedades que tiene sentido en el contexto del tipo objeto que recibe la clase de forma que con un mínimo de clase se obtenga el máximo de resultados.

Característica de arrastrar y soltar

Aprovechar la posibilidad de arrastrar y soltar objetos para crear prototipos.

Antes de crear una clase de propiedades se puede crear un objeto y probar y modificar sus características hasta obtener un resultado satisfactorio.

En ese momento, crear la clase a partir de la copia de las propiedades del objeto prototipo creado.

Por ejemplo, se puede crear un botón de *Salir*, probar y modificar sus características y finalmente, crear una clase de propiedades a partir de las propiedades del botón creado.

Encapsulamiento de objetos reutilizables

Encapsular los objetos reutilizables en un módulo genérico y copiar o referenciarlos en otros módulos o aplicaciones mediante la característica de seleccionar el objeto en el módulo genérico y arrastrar y soltar en el módulo de destino.

De esta forma, los objetos pueden ser desarrollados y probados una vez y ser utilizados múltiples veces sin esfuerzo adicional.

Uso de los grupos de objetos

Encapsular múltiples objetos y disparadores relacionados entre sí en un grupo de objetos (*object group*) que pueda ser heredado por otros módulos mediante una simple operación de arrastrar y soltar.

Creación de bibliotecas de módulos PL/SQL

Crear bibliotecas de módulos PL/SQL que contengan rutinas que las aplicaciones utilizan en diferentes lugares.

Una biblioteca PL/SQL es una colección de paquetes, procedimientos y funciones que puede ser anexada (*attached*) a cualquier módulo de una aplicación Forms, de forma que automáticamente disponga de toda la funcionalidad de la biblioteca.

De esta forma, el personal de desarrollo no necesita conocer los detalles de la implementación, sino sólo la funcionalidad que realiza y los parámetros de entrada y salida con los que se comunica cada pieza de la biblioteca. Con ello se consigue en mayor o menor medida programar y probar una vez y utilizar lo creado múltiples veces sin esfuerzo adicional.

Uso de plantillas de bloques genéricos

Definir para cada tabla o vista básica un bloque genérico o bloque plantilla que podrá ser utilizado como base de trabajo cada vez que se necesite un bloque en una pantalla.

Estos bloques genéricos implementarán toda la funcionalidad ligada a los datos de la tabla o vista base, de forma que pueda ser utilizada múltiples veces por los programadores sin esfuerzo adicional y sin necesidad de conocer todos los detalles de los bloques que estén en cada momento utilizando.

Uso de disparadores genéricos multinivel

Los disparadores de Forms pueden definirse en el nivel deseado de una aplicación: módulo, bloque o campo. Estos disparadores se ejecutan para cualquier objeto dentro del ámbito en el que están definidos.

Por ejemplo, se podrán definir disparadores que realicen validaciones básicas para los objetos de un bloque de datos. Pero al mismo tiempo, determinados campos necesitan validaciones específicas.

En vez de tener que reescribir el código de validación genérico para todos los campos en el nivel de campo, el disparador asociado al campo puede incluir sólo la validación específica para ese campo y heredar el disparador genérico situado en el nivel del bloque.

Se puede especificar además si el disparador de nivel de campo debe ejecutarse antes o después del disparador de nivel de bloque.

La herencia de disparadores permite escribir más código genérico, ubicado en niveles superiores, que puede luego ser aumentado o restringido cuando se tenga que aplicar en un nivel más reducido.

Menús de aplicaciones

Normas generales

Ubicación Los menús se asociarán a cada forma y se desplegarán dentro de la ventana de la forma que los contiene.

Menú inicial Todas las aplicaciones presentarán un menú inicial como respuesta a la activación de la aplicación por el usuario.

Número de niveles Se utilizará el menor número de niveles posible compatible con la claridad de la estructura de la aplicación, a fin de abreviar al máximo el tiempo de acceso a las opciones de la aplicación.

Diseño de opciones Se procurará que las opciones más frecuentemente utilizadas por los usuarios queden en los niveles superiores de los menús, mientras que las menos utilizadas no deberán aparecer en las primeras instancias de los menús.

Ordenar las opciones de los menús según su secuencia de utilización o bien por orden de frecuencia de uso.

Menú del administra-dor Se dotará a la aplicación de un menú de acceso a la totalidad de las opciones de la aplicación, de uso exclusivo del administrador o superusuario de la instalación.

Denominación de opciones

Nombres de opciones

Los nombres de las opciones del menú serán claras, concisas y podrán ser asociados sin dificultad a la función que realizan.

Normas de denominación

Para la denominación de opciones de menús se utilizarán en especial las siguientes normas:

- Usar palabras simples, siempre que sea posible.
- Evitar el uso de abreviaturas, excepto en los casos en los que el propio usuario los utiliza.
- No utilizar signos de puntuación
- Evitar el uso del mismo nombre de opción en diferentes niveles del menú si no existe una correspondencia funcional.
- Poner en mayúscula la primera letra del nombre de la opción, pero no poner punto final.
- El nombre de la opción del menú no debe ser muy diferente del título de la pantalla a la que da acceso

Seguridad

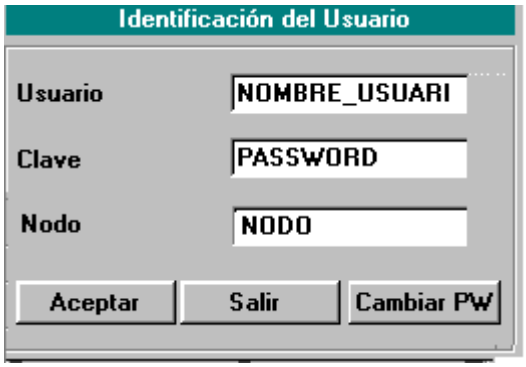
Control de acceso

El control de seguridad de acceso a las aplicaciones se controlará mediante la creación de roles de usuario ligados a las diferentes opciones del menú principal.

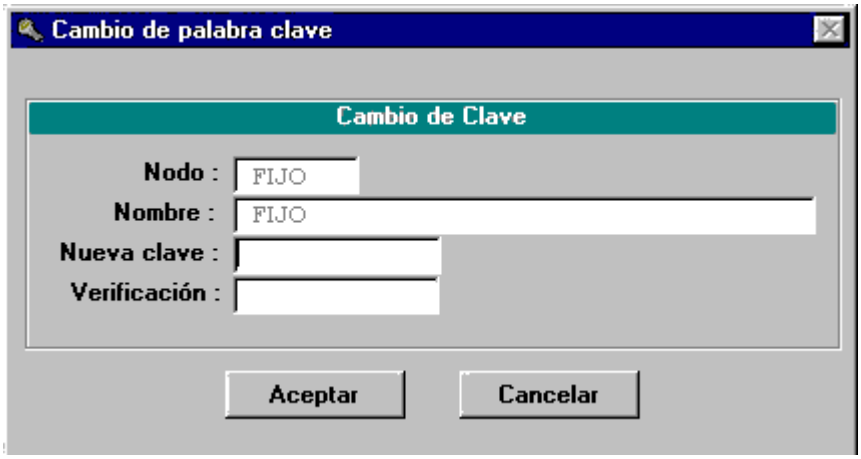
Interfaz de Conexión

Se establecen los siguientes tipos de mensajes y tipos de solución:
Acceso de **un usuario** a una la aplicación:

- El usuario dispondría de un icono de aplicación que ejecutaría un form de entrada a la aplicación.
- Solicita la identificación



- La forma de validar la información queda pendiente de definir hasta decidir sobre la normativa de directorio electrónico.
- Se ejecutará un procedimiento en el form que se encarga de mostrar sólo aquellas opciones del menú a las cuales el usuario tenga acceso (ya sea en modo modificación o bien consulta). Dependiendo de sus permisos de acceso, a la hora de ejecutar un form del menú se realizará con un procedimiento que le permita el modo normal o solo modo consulta.
- En el menú principal del usuario existirá la opción *Cambiar Clave* dentro de la opción *Herramientas*, que permitirá modificar la clave de acceso del usuario.



Mensajes de la Aplicación

Tipos de mensajes

Se establecen los siguientes tipos de mensajes y tipos de solución:

Mensaje de error	Informa de un acción o condición que puede impedir u originar efectos inmediatos o futuros no deseados.	Fuerza al usuario a corregir la causa del error.
Mensaje de advertencia	Informa de la ejecución de una acción que dará origen a una condición inconsistente o anormal.	Dar información al cliente para gestionar la condición.
Mensaje de información	Clarifica acciones, informa al usuario del estado presente del procesamiento o presenta información sensible al contexto	Presenta información al usuario que le permita seguir, modificar o suspender operaciones a realizar.

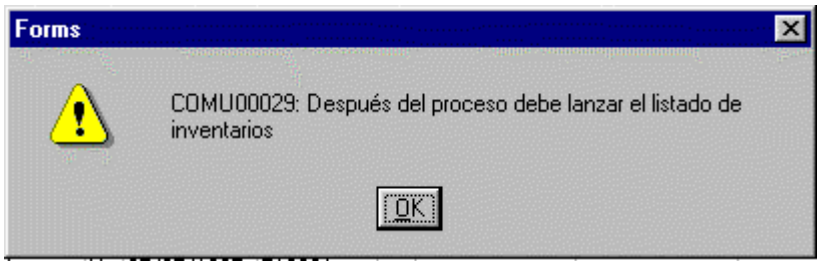
Los mensajes en tiempo de ejecución han de diferenciarse claramente de acuerdo con su importancia y determinar la forma adecuada de manejar cada situación en función de la causa indicada en el mensaje.

Mostrado de mensajes

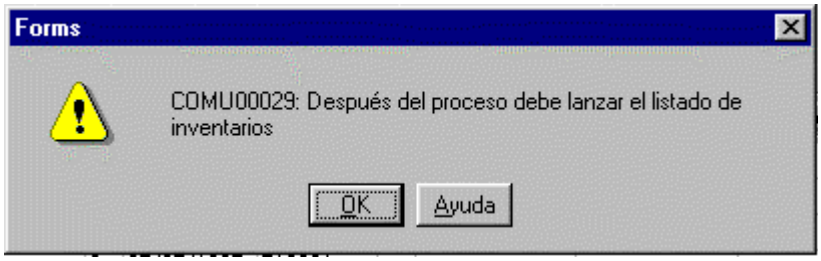
Los **tipos de mensajes** que vamos a poder utilizar serán los siguientes :

- **A (Aviso):** Al mostrar mensajes de este tipo simplemente le aparecería el mismo en la barra de estado del form (barra inferior).

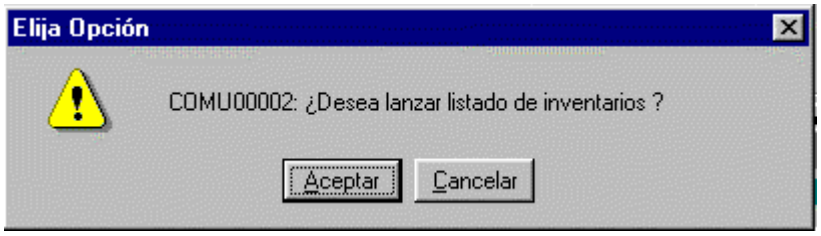
- **C :** Este tipo lo utilizaremos para mostrar mensajes de precaución o de aviso de los cuales queremos que tenga constancia el usuario antes de que se produzca una determinada acción.



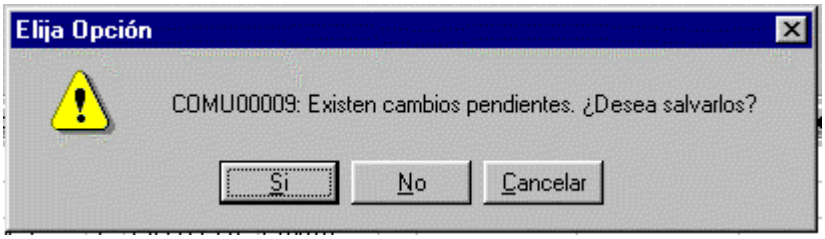
En el caso de que deseemos mostrar ayuda acerca del mensaje en cuestión, deberíamos introducir el mensaje de ayuda en el campo diseñado a tal efecto así mismo podríamos querer proponer determinada acción con respecto a la situación que se está produciendo y la indicaríamos en el campo correspondiente (por ejemplo si fuese un error, en la ayuda ampliaríamos información sobre el error y en la posible solución propondríamos alguna acción con vistas a corregirlo). Si hubiese ayuda, automáticamente el mensaje incorporaría un botón para invocar a la ayuda.



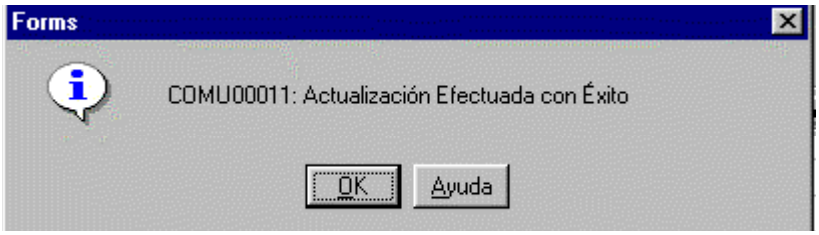
- **D (Decisión):** Estos mensajes los utilizaremos para que un usuario decida realizar o no una determinada acción.



- **P(Pregunta):** Mensajes que utilizaremos para preguntar al usuario sobre la posibilidad de realizar una determinada acción.

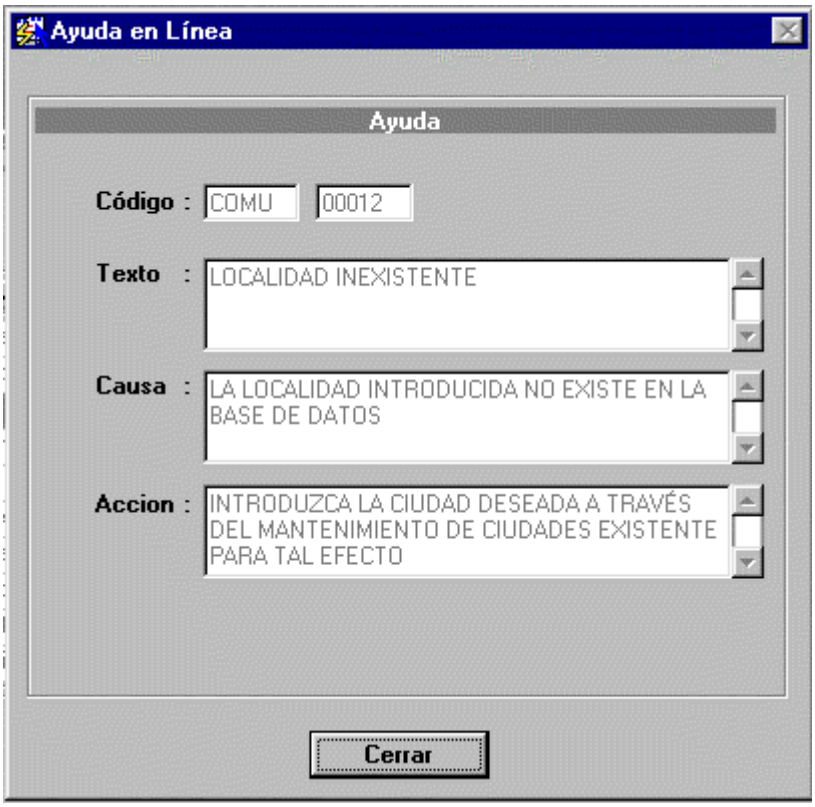


- **N (Nota):** Estos mensajes los utilizaremos para mostrar información al usuario. Igualmente estos mensajes tienen la posibilidad de incorporar ayuda y acciones a realizar para hacer más explicativo y amplio el contenido de la información.



- **S (Stop):**Mensajes utilizados para mostrar un determinado error que se ha producido, ante el cual lo único que se pretende es informar al respecto al usuario. También estos mensajes disponen la posibilidad de mostrar información acerca del error producido y las posibles acciones de ayuda a la hora de solucionarlo.

- **Ayuda mostrada en los mensajes:** Hemos visto que hay tipos de mensajes (‘C’,’N’,’S’) en los cuales se puede mostrar ayuda indicando más información referente al mensaje y acciones propuestas para solucionar aquello que ha dado lugar al mensaje. Estos mensajes llevan un botón de ayuda que iría a un form encargado de mostrarnos esta información:



En este form se nos mostraría el código (nombre aplicación, código mensaje) y texto del mensaje junto con la causa y las posibles acciones a llevar a cabo.

Codificación de mensajes

Formato según tipo

Los diferentes tipos de mensajes se distinguirán entre sí mediante la aplicación de un formato consistente y una codificación adecuada.

Mantenimiento de mensajes

Se establecerá quién es la persona responsable del mantenimiento de los mensajes de la aplicación.

Estructura de un mensaje

Cada mensaje ha de estar codificado de la siguiente manera:

Prefijo de mensaje

Se empleará el alias de la aplicación.

Ejemplo: `SG` - Sistema de Gestión de Expedientes

Se evitarán prefijos ya utilizados por otros sistemas utilizados concurrentemente con la aplicación, por ejemplo, sistema operativo, ORACLE, etc.

Número de mensaje

Se tomarán tres dígitos, que serán prefijados según el tipo de mensaje (error, advertencia, información), separando el prefijo de mensaje del número de mensaje mediante un guión.

Ejemplo:

`SG-E000`	en adelante	mensajes de error
`SG-A000`	en adelante	mensajes de advertencia
`SG-I000`	en adelante	mensajes de información

Texto del mensaje

El texto del mensaje ha de estar redactado de forma clara, concisa y específica.

En particular, se aplicarán las siguientes normas:

- Redactar mensajes cortos, de forma positiva y de acuerdo con la gravedad o importancia de la condición que lo desencadena.
- Utilizar términos descriptivos para describir el área o problema específico a solventar.
- Poner en mayúscula inicial los términos que deben ser resaltados para una mejor comprensión.
- Evitar el uso de términos técnicos, no utilizados normalmente por los usuarios que han de manejar el sistema.
- Indicar la operación a realizar, si es necesario, pero evitar dar indicaciones que el usuario pueda interpretar de forma diferente.
- Evitar la mención a fallos o errores del usuario de forma directa excepto si es estrictamente necesario.

Ejemplos:

Incorrecto:	Correcto:
`Ha introducido un valor incorrecto`	

un nuevo número

Ha cometido un error al seleccionar la opción; repita operación

- Evitar crear alarma en el usuario mediante una redacción que parezca más grave de lo que se deriva de la condición desencadenante.

Ejemplos:

Incorrecto:	Correcto:
Imposible insertar registro; valor de Primary Key no encontrado	Imposible añadir Empleado; falta el Número de Empleado
La tabla se ha borrado	No se han encontrado Líneas de Pedido para este Pedido
Ha introducido un **Número** de **Departamento** erróneo	Número de Departamento inexistente; volver a introducir el **Número** de Departamento

Causa Causa o causas que originan el error.

Acción Descripción de las actividades necesarias para la recuperación o solución del error detectado.

Ejemplo:

Código:	SG-E152
Texto:	No fue posible insertar en <tabla>. No se encontró <columna> .
Causa:	El sistema no permitió insertar una nueva fila en la tabla <tabla> porque el valor especificado en la columna <columna> no existe en la tabla que contiene los valores posibles para esta columna.
Acción:	Compruebe el valor entrado en el campo <columna>. Si el valor no es correcto, dar un valor correcto y relanzar el proceso. Si el valor es correcto, compruebe en la tabla de valores posibles la existencia de ese valor; si no existe, lo deberá dar de alta y volver a relanzar el proceso. Si aún así se recibe este error, contacte con soporte de aplicaciones.

Parametrización de mensajes

Parametrización de mensajes

Se considerará la posibilidad de parametrizar los mensajes, de manera que incluyan referencias dinámicas a la tabla, columna u otra información que aclare el mensaje y pueda ser captada en tiempo de ejecución.

Normas de parametrización

Para la parametrización de mensajes se tendrán en cuenta las siguientes normas:

- Determinar el formato apropiado para codificar el parámetro.
- Formatear el parámetro en función de la variable especificada para cada tipo de aplicación.
- Encerrar la variable entre delimitadores apropiados según la herramienta que hace la llamada; por ejemplo, "<" y ">".
- Determinar el método correcto para descodificar el parámetro en función del tipo de aplicación, por ejemplo, SQL*Plus, ORACLE Forms, disparador de base de datos, procedimientos o funciones, etc.
- Considerar la posibilidad de casos en los que la variable no tenga ningún valor asignado mediante la programación de mensajes en la propia aplicación.
- Asegurarse de que la longitud del mensaje una vez que se sustituya el valor de la variable no excede el área de visualización de mensajes establecida en la aplicación.

Ejemplos:

```
SGE-0384: No fue posible añadir el registro; ha de dar un
valor para '&p_nomcol'.

RAISE_APPLICATION_ERROR

(-21700, 'SGE-01700: No se puede acceder a ' ||

v_nom_tabla || ';' || v_nom_nodo ||'no accesible.');
```

Almacenamiento de mensajes

Almacenamiento en la base de datos

Los mensajes codificados se guardarán en tablas de la base de datos. Para ello se definirá una tabla para almacenar los mensajes con las columnas apropiadas.

Indicativo del lenguaje

Si se prevé el uso de aplicaciones multilenguaje, se incluirá en la clave primaria una columna indicativa del lenguaje.

Ejemplo:

```
CREATE TABLE ge_mensajes
    (    c_prefijo          VARCHAR2 (3)
    ,    n_numero           NUMBER   (5)
    ,    a_text             VARCHAR2 (65) NOT NULL
    ,    f_actual DATE
    ,    CONSTRAINT msj_pk        PRIMARY KEY
         (         prefijo
         ,         sufijo
         )
    )
;
```

Consolidación con el Manual de Usuario

Reflejo en el Manual de Usuario

El sistema de mensajes implantado en cada aplicación tendrá su reflejo adecuado en el Manual de Usuario.

Si las tablas utilizadas para el almacenamiento de mensajes contiene toda la información, se reconsiderará la necesidad de reflejarlos por completo en el Manual de Usuario.

Sistema de Ayuda en Línea

La necesidad de elaborar una ayuda en línea para las aplicaciones viene determinada por los siguientes aspectos:

- Uso poco frecuente de determinados módulos o funcionalidades para los que el usuario necesita recordar el procedimiento de uso.
- Reducción del tiempo de entrenamiento en nuevos usuarios.
- Cubre un posible amplio abanico de experiencia de los usuarios, que abarca desde usuarios inexpertos a usuarios con buenos conocimientos.
- Más fácil de consultar y mantener que un manual en papel

Mecanismos de ayuda

Este sistema de ayuda en línea deberá tener los siguientes mecanismos esenciales:

Descriptores de elementos de pantallas	Los títulos de pantallas, bloques y descriptores de todos los elementos o campos de la pantalla, así como de los botones (ayuda de burbuja), deberán estar diseñados de manera que proporcionen una ayuda de primer nivel inmediata al usuario.
Línea de ayuda (*hint*)	La línea de ayuda visualiza un corto mensaje explicativo del campo o elemento sobre el que se sitúa el cursor en cada momento, debiendo jugar un papel fundamental en el sistema de ayuda de la aplicación.
Botón de ayuda	La barra de botones de cada ventana deberá incluir un botón de ayuda que ha de suministrar al usuario una ayuda de pantalla plena sensible al contexto en el que está situado el cursor en el momento de pulsar el botón. 

Niveles de ayuda

La Ayuda suministrada por el botón de ayuda deberá estar estructurada en los siguientes niveles:

- Ayuda de aplicación
- Ayuda de módulo
- Ayuda de bloque
- Ayuda de campo

Prototipo de Pantallas

CAPÍTULO

11 ORACLE Reports 6i

En el presente capítulo se definen las normas aplicables para el uso de ORACLE Reports versión 6i. Las normas definidas se organizan en las áreas de componentes de un listado, formato estándar de listados, formato estándar de la ventana de lanzamiento de listados y normas relacionadas con el entorno de desarrollo de listados e informes.

Normas Generales

La amplia casuística de formatos de salidas impresas que puede contemplar una aplicación se resume como sigue:

- Listados específicos

  La información a imprimir debe ajustarse a un formato predeterminado, bien porque se deba utilizar papel preimpreso de imprenta o bien porque se utiliza papel en blanco pero el módulo de la aplicación genera el propio preimpreso.

- Listados comunes

  Se utiliza papel en blanco y no se ajustan a un formato específico predeterminado.

Las normas que se refieran al formato de un listado, contempladas en el presente documento son aplicables a estos últimos.

Componentes del listado

Todas los listados de la aplicación tendrán una apariencia similar y se compondrán básicamente por los mismos componentes.

Los listados están formados por los siguientes componentes:

Cabecera	Datos identificativos de la totalidad del informe.
Cuerpo	Datos propios del listado.
Parámetros	Pantalla previa al lanzamiento del listado o informe en la que el usuario introduce los parámetros del listado.

Ejemplo de formato de listado

Consejería de xxxxxxx **Informe de Expedientes** **xxlxxxx**
Servicio de xxxxxxx **FECHA : 03/12**/95
PÁGINA: x

Núm Expediente	Destino	Fecha Alta	Asunto
XXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX	XXXXXXXXX	XXXXXXXXXX
XXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX	XXXXXXXXX	XXXXXXXXXX
XXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX	XXXXXXXXX	XXXXXXXXXX
XXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX	XXXXXXXXX	XXXXXXXXXX

Pantalla de petición de parámetros

Pantalla de petición de parámetros

Partimos de un form donde establecemos los criterios de selección y le pasamos como parámetros al report:

1 parámetro.: Toda la condición where de la select.

2 parámetro.: El orden que se va a establecer en la select para el listado (si es que se pide en el form). Ejem.: nº expte., apellidos,etc.

3 parámetro.: La unidad administrativa del usuario, hasta el nivel que quieran.

En el Report. :

1. El usuario no lo sacamos.
2. módulo. El nombre del form de parámetros es igual que el del report. Ponemos el nombre del listado: <alias corto>Lnombre (limitado a 8 pero se puede ampliar).
3. página y fecha.

Los documentos los hacemos con word. Mediante DDE, fusionamos con word. Seleccionamos un fichero con los datos desde **Forms**, se fusiona con el modelo, abrimos el resultado y el usuario imprime si quiere.

Opciones tras la introducción

Tras la introducción de los valores de parámetros el usuario dispondrá de las siguientes opciones:

Cancelar	Cancela la acción y volvemos al menú inicial.
Imprimir	Visualiza y Activa el lienzo de control de impresoras de MS-Windows si lo desea.

Normas de diseño de informes

Se hace referencia a continuación a todas aquellas características externas al contenido del informe como títulos, página inicial, descriptores de campos o columnas, encabezados y otros.

Formato

Usar siempre tamaños de página estándar, bien sean de tipo apaisado o peraltado aunque se ha de usar de preferencia la orientación apaisada.

Limitar los caracteres de resaltado (negrita, subrayado, etc. a aquéllos soportados por todas las impresoras de la organización.

Utilizar la cabecera para mostrar el título del informe y los parámetros de lanzamiento.

Deberá tenerse en cuenta la necesidad de los usuarios de archivar o encuadernar el informe en el diseño de los márgenes.

El espacio es proporcional. Utilizar tipo de letra ARIAL porque es más clara y se pueden incluir más caracteres en una misma fila.

Utilizar colores sólo si se desea resaltar aspectos determinados de los informes en modo de previsualización (modo *Preview*).

Cuando las líneas son demasiado largas o con abundante número de campos que pueden aparecer en blanco, se utilizarán alternativamente líneas en blanco y color.

También para totales (línea de color o negrita o ambas.)

Cabecera de página

La cabecera de página deberá mostrar la siguiente información:

- Margen superior izquierdo:
  - Consejería de XXXXXXX
  - Unidad administrativa
- Parte central:
  - Título del informe
- Ángulo superior derecho
  - Nombre del informe (nombre del módulo)
  - "FECHA: " formato DD/MM/AAAA
  - "PÁGINA: " formato nº página

Como subcabecera puede aparecer el nombre del elemento que permita romper la página.

Por ejemplo, en un listado que rompa por provincias, aparecerá el literal "PROVINCIA: " pegado a la izquierda de la página y a continuación, el nombre de la provincia.

Si hubiera más rupturas, aparecerá debajo de la primera.

El título de un informe reflejará su uso primario. En caso de necesidad, sobre todo para evitar títulos largos o poco claros, utilizar un subtítulo.

Si aplica, se deberá indicar la fecha o período de tiempo al que aplica la información, tipo de unidades físicas o monetarias y otros datos de identificación de la información emitida.

Los informes confidenciales irán marcados con la palabra "CONFIDENCIAL" en la cabecera del informe.

Después de la cabecera de página se dejarán dos líneas en blanco.

Cuerpo del informe

Campos Mostrar sólo los campos directamente relacionados con el objetivo del informe.

Minimizar el número de campos del informe.

Utilizar textos y longitudes de descriptores de campos y columnas consistentes a lo largo de todos los informes de la aplicación.

Cuando en un listado aparezca un código, si hubiera espacio suficiente, se incluirá a continuación su descripción. En caso de existir una descripción larga y otra corta, se utilizará de preferencia esta última.

Los literales se deberán escribir en mayúsculas siempre que sea posible.

Formatos Los formatos de los diferentes campos se ajustarán como sigue:

- Fechas: DD/MM/AAAA
- Hora: HH:MM:SS
- Números: 9.999,99 (eliminar ceros no significativos)

Descriptores Los descriptores de campos y columnas deberán llamarse igual o lo más parecido a los descriptores equivalentes de las pantallas.

Las cabeceras de líneas de detalle aparecerán subrayadas con línea continua, alcanzando la longitud máxima que pueda tener la columna.

Los literales de dichos descriptores de columnas y los datos a presentar en las respectivas columnas tendrán la siguiente alineación:

- Texto: centrado
- Número: centrado

Los descriptores de columnas deberán mostrar por debajo una línea a lo largo de su longitud.

```
          Código
          Empleado   Nombre
 ------------   --------------------
```

Los campos alfabéticos se alinearán a la izquierda y los numéricos a la derecha.

```
Identificador       Descripción          Existencia    Pedidos
------------  ----------------------  -------------  ---------
xxxxxxxxxxxxx   hhhhhhhhhhhhhhhh           999999        999
xxxxxxxxxxxxx   hhhhhhhhhhhhhhhh           999999        999
xxxxxxxxxxxxx   hhhhhhhhhhhhhhhh           999999        999
```

Los descriptores de campos individuales deberán ir seguidos de ":" y espacio. Estos descriptores irán alineados a la derecha del carácter ":".

```
Código del Empleado: 8987
             Nombre: xxxxxxxx xxxxxxxxxxxxxx
          Dirección: xxxxxxx xxxx xxxxxxxxx xx
```

Incluir una columna de unidades a continuación de una columna de cantidades si el tipo de unidad cambia de una fila a otra de forma no obvia.

```
    Referencia           Cantidad        Unidades
-------------------   ---------------   -------------

xxxxxxxxxxxxxxx                9999999    xxxxxxxxx
xxxxxxxxxxxxxxx                9999999    xxxxxxxxx
xxxxxxxxxxxxxxx                9999999    xxxxxxxxx
```

Normas sobre totales

Documentar adecuadamente el empleo de subtotales y totales, de forma que sean claramente identificados.

```
Cliente           Factura         Importe
----------------  --------------  ---------------
 xxxxxxxxxxxxxx       999999999      99999999999
                      999999999      99999999999
                      999999999      99999999999
                                    ---------------
Total cliente:                       99999999999

 xxxxxxxxxxxxxx       999999999      99999999999
                      999999999      99999999999
                                    ---------------
          Total cliente               99999999999

 xxxxxxxxxxxxxx       999999999      99999999999
                      999999999      99999999999
                      999999999      99999999999
                                    ---------------
          Total cliente               99999999999

                                    ---------------
          Total general               99999999999
                                    ---------------
```

Los datos correspondientes a SUBTOTAL y TOTAL estarán alineados a la derecha y debajo de la columna correspondiente y sus literales respectivos se imprimirán sin sangrado:

```
     Subtotal:              9.999,99
     Total:                 9.999,99
```

Se mostrarán los códigos de clave primaria sólo cuando sea necesario (quita espacio) como sus columnas descriptoras (por ejemplo, C*ódigo del Empleado* y *Nombre del Empleado*.

```
Código
 Empleado      Nombre
------------  --------------------
 xxxxxx        hhhhhhhhhhhhhhhh
 xxxxxx        hhhhhhhhhhhhhhhh
 xxxxxx        hhhhhhhhhhhhhhhh
 xxxxxx        hhhhhhhhhhhhhhhh
```

Los campos numéricos de longitud mayor de 30 caracteres deberán mostrarse plegados sobre un campo de 30 caracteres. ??

```
Comentarios
------------------------------
Xxxxx xxxx xxxx x xxxxxx xx
Xx x xxxxx xxx xxxx x x xxxxx
X xxx x xx xxxx x xxx xxxxx
```

Normas sobre el final del informe

Final del Informe

El final del informe se indicará mediante la siguiente línea:

-- FIN DE INFORME --

Entorno de desarrollo

Normas generales

Almacena-miento de módulos

En un servicio de ficheros.

Iniciar en C:\WINDOWS\TEMP

En REGEDIT ORACLE en misc_reg_entries indicamos el directorio de la aplicación. Por ejemplo: “Z:\COAD_DES”.

En este último caso, el responsable del proyecto se asegurará de que existen los adecuados procedimientos de copia de seguridad y de control de acceso al sistema operativo.

Definición de formatos

Al inicio de la aplicación se definirá, para cada tipo de informe, los aspectos tales como formatos de cabeceras, cuerpo y contenidos de los informes.

Componentes reutilizables

Se estudiará la posibilidad de elaborar funciones o procedimientos reutilizables que serán referenciados desde las hojas de propiedades de elementos del informe (filtros de grupos, fórmulas de columnas, contenedores de columnas 1, marcos, marcos de repetición, campos, y disparadores) como alternativa a definir hojas de propiedades específicas a elementos concretos de cada informe.

1 En inglés, *column placeholders.*

Denominación de componentes

El nombre de cada uno de los tipos de objetos de un informe se compondrá de la siguiente manera:

Consultas	Q_<*identificador*>	donde <*identificador*> hace referencia al nombre de la tabla o funcionalidad asociada a la consulta.
Grupos	G_<*identificador*>	donde <*identificador*> hace referencia a la funcionalidad asociada al grupo.
Parámetros	P_<*identificador*>	donde <*identificador*> hace referencia a la funcionalidad asociada al parámetro.
Marcos de repetición (*Repeting Frames)*	R_<*identificador*>	donde <*identificador*> hace referencia a la funcionalidad asociada al marco.
Marcos genéricos	M_<*identificador*>	donde <*identificador*> hace referencia a la funcionalidad asociada al marco.
Campos	F_<*identificador*>	donde <*identificador*> hace referencia al nombre de la columna de la tabla o a la funcionalidad asociada al campo.
Botones	B_<*identificador*>	donde <*identificador*> hace referencia a la funcionalidad asociada al botón.