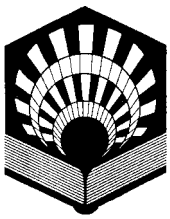


Universidad de Córdoba

NORMAS Y ESTANDARES

Desarrollo de Aplicaciones
Revisión 1.0

Servicio de Informática
Area de Sistemas
Mayo 1993

1. Introducción.

La Dirección de Informática de la Universidad de Córdoba tiene como uno de sus objetivos fundamentales la promoción de la **integración** de herramientas y procedimientos en el desarrollo de los Sistemas de Información.

Uno de los aspectos clave de esta integración es la definición y uso homogéneo de todas las herramientas de desarrollo.

Los objetivos de dicha normalización son:

- ❏ Aumentar la confianza del usuario final en las aplicaciones desarrolladas, ofreciéndole un entorno familiar, sin sorpresas ni excepciones.
- ❏ Promover la colaboración entre los diferentes grupos de desarrollo permitiendo que soluciones elaboradas para una aplicación se utilicen en el desarrollo de otras.
- ❏ Facilitar el mantenimiento de aplicaciones haciendo que que un módulo no esté ligado al programador del mismo.

Dentro de la Dirección de Informática, se identifica el Area de Sistemas como el área responsable de promover esta normalización y de verificar su cumplimiento.

Para ello, el Area de Sistemas elaborará un conjunto de documentos que sirvan como base para dicha normalización, ofreciendo un punto de referencia en la discusión sobre la misma.

Una vez aprobados, los Analistas responsables de cada aplicación cuidarán de que los estándares se lleven a la práctica y el Area de Sistemas verificará la implantación de los mismos.

Este documento contiene normas para el desarrollo de software en sus aspectos fundamentales: estándares de nomenclatura, entornos de explotación y desarrollo, uso de las herramientas,...

Los principales destinatarios son los analistas y programadores responsables del desarrollo y mantenimiento de aplicaciones de gestión. Se supone un conocimiento básico de ORACLE y de todas las herramientas comentadas.

El documento no pretende ser completo. Muchos aspectos del desarrollo de software quedarán fuera de estas especificaciones.

Por ello, resulta especialmente interesante que los responsables de las diferentes aplicaciones propongan nuevos estándares para todos aquellos aspectos no cubiertos en este documento, evitando la diversificación de criterios en las soluciones propuestas a problemas comunes.

Este documento no contiene estándares sobre todos los aspectos relacionados con el desarrollo de software. Se prepararán documentos adicionales que estandarícen aspectos como:

- Metodología de análisis.
- Desarrollo de pruebas unitarias y de integración.
- Administración de Sistemas.
- Administración de Bases de Datos.
- Documentación.
- Sistema de Gestión de Código.

Actualmente, el Area de Sistema tiene publicados dos Manuales que se incorporan a la documentación de estándares:

- *Guia de Operación*. Que normaliza el aspecto final de la aplicación desde el punto de vista del usuario.
- *Sistema de Impresión. Manual del Programador.* Que normaliza la interface usuario-listados.

2. Generalidades.

1. Conceptos.

Introducimos en este apartado algunos conceptos generales que nos serán de utilidad en la definición de estándares.

Aplicaciones

Llamaremos aplicación a cualquier desarrollo software funcionalmente independiente que, no obstante, puede interconectarse, puntualmente, a otros desarrollos. Ejemplos de aplicaciones son SIGA, SIGE y GAD.

A cada aplicación se le asocia un código único de una letra que denominamos código de aplicación. Asimismo, se le asociará un código extendido de no más de tres caracteres (*Tabla A.I, Apéndice A*).

Funciones

Las aplicaciones se dividen en funciones con poca relación entre sí cada una de las cuales cubre un aspecto del organigrama funcional de la aplicación. Una aplicación típica no tendrá más de una docena de tales funciones.

Las funciones se descomponen, a su vez, en subfunciones. La comunicación entre las diferentes subfunciones de una misma función será, en general, elevada. La subdivisión funcional puede continuar, pero solo será significativa a propósitos de estandarización un máximo de cinco niveles funcionales.

Funciones y subfunciones se codifican empleando un carácter para cada nivel hasta un máximo de cinco caracteres, formando una combinación única dentro de cada aplicación.

Módulos

El nivel más bajo de esta jerarquía está formado por el módulo. Una subfunción emplea uno o más módulos para realizar la tarea que tiene encomendada. Un módulo puede ser cualquier programa: *sell scripts*, formas, *reports*,...

Los módulos pueden ser fuentes o ejecutables, si bien, algunas clases de módulos (v.g. procedimientos SQL) pueden tratarse indistintamente como fuentes o ejecutables.

A cada módulo se le asigna un código de uno o más caracteres dependiendo de la profundidad alcanzada en la codificación de funciones: un sólo carácter si se ha llegado a cinco niveles, dos si se llegó a cuatro,...

Identificador

Por ultimo definiremos el concepto de identificador, el cual emplearemos frecuentemente en la normalización de los nombres de los diferentes objetos. Un identificador genérico se construye abreviando la descripción del objeto (campo, tabla,...) al que pretende identificar. Si la descripción consta de más de dos palabras significativas, se incluirá el símbolo "_" como separador entre las mismas. Un identificador nunca se construirá a partir de más de tres palabras. Las palabras elegidas para construir el identificador pueden abreviarse.

El identificador puede contener exclusivamente los caracteres A-Z, 0-9 y "_" debiendo comenzar por una letra.

Al identificador genérico se le añadirán códigos que aporten información sobre el objeto identificado (como se describe posteriormente en este documento) por lo que las palabras elegidas para construir el identificador no deben hacer referencia a la naturaleza del objeto identificado.

2. *Entornos.*

2.1 Definición.

Desde un punto de vista lógico se distinguirán tres entornos bien diferenciados: Desarrollo, Integración y Explotación.

Desarrollo

El Entorno de Desarrollo comprende todos los módulos sobre los cuales trabaja el grupo de programación en las primeras fases del desarrollo de una nueva aplicación: desarrollo y pruebas unitarias. Existirá un Entorno de Desarrollo por cada nueva aplicación que se empiece.

Integración

El Entorno de Integración se compone de los módulos y escenarios de datos necesarios para realizar las pruebas de integración previas a la puesta en servicio de una aplicación. Una vez entregada a los usuarios, el grupo de mantenimiento realizará su labor sobre este entorno. Cada aplicación dispondrá de su propio Entorno de Integración siendo posible disponer simultáneamente de más de una versión de la misma aplicación.

Explotación

El Entorno de Explotación consta de los datos y módulos ejecutables que emplean los usuarios finales así como de todos los módulos necesarios para reconstruir la versión actual de los ejecutables.

2.2 Responsabilidades.

Desarrollo

Cada programador dispondrá de sus propios directorios de trabajo así como de datos independientes para realizar pruebas unitarias de los módulos que vaya desarrollando. Es responsabilidad del programador la organización de su directorio de trabajo así como el mantenimiento de sus datos de prueba.

Se favorecerá el traspaso de módulos en desarrollo entre programadores con la única limitación de que un programador no pueda modificar el trabajo de otro.

Integración

Desde el primer momento que se necesite probar la integración de dos o más módulos se creará un Entorno de Integración para la aplicación correspondiente. A este entorno se pasarán los módulos que hayan superado las pruebas unitarias y sobre los que se vayan a realizar las pruebas de integración.

El Area de Sistemas creará los Entornos de Integración necesarios pasando, a continuación, la responsabilidad del mantenimiento de los escenarios de pruebas y de la coherencia del Entorno de Integración al Analista responsable de la aplicación.

Hasta la puesta en servicio de la aplicación, existirán simultáneamente los Entornos de Desarrollo e Integración para una misma aplicación. Una vez creado el Entorno de Explotación dejara de existir el de desarrollo.

Para las aplicaciones en explotación es posible mantener varias versiones de la misma en el Entorno de Integración (v.g. una para mantenimiento y otra para desarrollar nuevas funcionalidades).

Los Entornos de Integración de las diferentes aplicaciones estarán aislados. Con el acuerdo previo de los Analistas responsables, el Area de Sistemas facilitará la colaboración entre los diferentes grupos de desarrollo.

Explotación

Las aplicaciones se distribuirán en el Entorno de Explotación atendiendo a consideraciones de seguridad y de rendimiento.

La integridad y seguridad de los datos y módulos en el Entorno de Explotación serán responsabilidad exclusiva del Area de Sistemas. Se facilitaran los mecanismos adecuados para permitir a los grupos de mantenimiento de aplicaciones el traspaso de módulos modificados o nuevos al Entorno de Explotación.

El Analista responsable de cada aplicación dispondrá de acceso no restringido a los menús de su aplicación en el Entorno de Explotación. Además podrá acceder a las tablas correspondientes de la base de datos, si bien, no podrá modificar la estructura de la mismas.

El Area de Sistema entregará copias de los módulos fuentes actualmente en explotación siempre que le sea requerido por el grupo de mantenimiento.

Usuarios

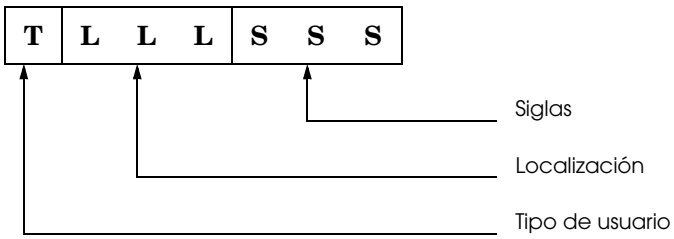
Los usuarios podrán acceder a los datos en explotación exclusivamente a través de la aplicación correspondiente. El responsable del área para la que se ha desarrollado una aplicación definirá claramente los perfiles de todos los usuarios de modo que puedan delimitarse responsabilidades.

La alteración de datos en explotación sólo será posible, en general, por medio de la aplicación, y por un usuario debidamente autorizado (con el perfil adecuado). No obstante, en casos excepcionales y con la debida autorización del responsable del área correspondiente, el Analista de la aplicación podrá modificar los datos cuando así se considere oportuno.

3. *Identificación de usuarios.*

Los usuarios de cualquier sistema serán identificados por un Código de Usuario que será asignado de forma unívoca por el Area de Sistemas. Toda persona autorizada a trabajar con cualquier aplicación de las ofertadas por el Servicio de Informática de esta Universidad dispondrá de un Código de Usuario único para acceder a todas las aplicaciones a las que esté autorizado.

El Código de Usuario consta de siete caracteres (a-z) distribuidos de la siguiente forma:



El significado de cada uno de los campos que componen el Código de Usuario es el siguiente:

Tipo de usuario (T):

> Un carácter indicando el tipo de usuario. La clasificación de usuarios atendiendo a este código se muestra en la *Tabla A.II, Apéndice A*.

Localización (LLL):

> Tres caracteres indicando la localización (edificio) o lugar de trabajo más frecuente del usuario. Las localizaciones actualmente codificadas se detallan en la *Tabla A.III, Apéndice A*.

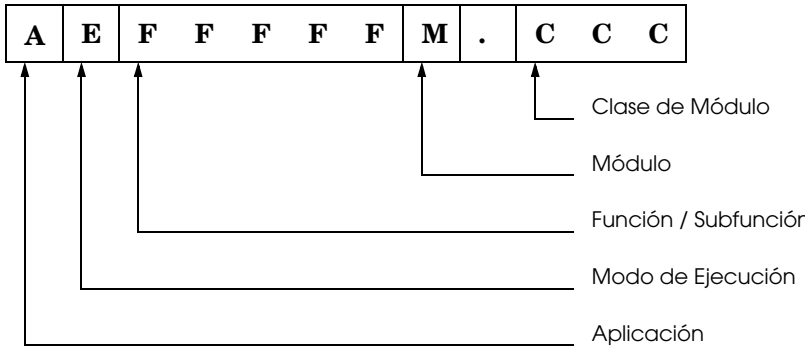
Siglas (SSS):

> Tres caracteres representando las iniciales del nombre completo del usuario. Las siglas se asignarán de un modo natural, i.e., tomando la primera letra del nombre (no se considera el segundo nombre) y apellidos. Si, de esta

forma, no es posible encontrar una combinación natural de siglas de modo que el Código de Usuario (completo) sea único, se procederá a tomar la segunda (tercera, cuarta,...) letra del ultimo (penúltimo...) apellido.

4. *Identificación de módulos.*

El nombre de un módulo consta de no más de 12 caracteres en mayúsculas (A-Z) distribuidos de la siguiente forma:



de tal forma que se le pueda identificar unívocamente.

El significado de los caracteres que componen el nombre de un módulo es el siguiente:

Aplicación (A):

> Código de la Aplicación a la que pertenece el módulo. Un listado completo de Aplicaciones y su código correspondiente se encuentra en la *Tabla A.I, Apéndice A*.

Modo de Ejecución (E):

> Los programas pueden ejecutarse en uno de las formas tabuladas en la *Tabla A.IV, Apéndice A*.

Función (F):

> Código alfanumérico de la función-subfunción a la que pertenece el módulo. La interpretación de esta codificación la define el analista, pero debe ser única dentro de toda la aplicación. Es decir, el Analista asigna uno o más caracteres a cada nivel hasta un máximo de cinco y la distribución de caracteres es única para toda la aplicación.

Módulo (M):

> Este carácter identifica un módulo dentro de una subfunción. Si la división funcional de la aplicación no cubre los cinco niveles permitidos, es posible emplear los caracteres no asignados para codificación del módulo. Por ejemplo, una codificación valida podría estructurarse del siguiente modo: un carácter para las funciones de primer nivel, dos para las subfunciones de segundo nivel, uno más para las de tercer nivel y dos caracteres para identificación del módulo.

Clases (C):

> La extensión del nombre del módulo se corresponde con la clase a la que pertenece (forms, fuente en C,...). La clase de módulo puede ser menor de tres caracteres. Un listado completo de las clases existentes puede encontrarse en la *Tabla A.V, Apéndice A*.

Por, ejemplo, un módulo interactivo (Modo de ejecución `I`) desarrollado en SQL*Forms (Clase de Módulo `inp`) perteneciente a la función `10`, subfunción `2` de la aplicación SIGE (Código de Aplicación `C`) podría nombrarse como `CI102001.inp`.

5. *Herramientas.*

La Universidad de Córdoba utiliza ORACLE como gestor de base de datos a nivel corporativo, de modo que todas las aplicaciones de gestión desarrolladas por el Servicio de Informática serán implementadas para dicho producto empleando herramientas compatibles con el mismo.

ORACLE acompaña su gestor de bases de datos con un amplio abanico de herramientas que van desde interfaces de bajo nivel (OCI) hasta utilidades de usuario final (como Oracle*Card). Algunas (como SQL*Report) han quedado obsoletas por lo que la elección de herramientas de desarrollo hay que hacerla con cuidado y no sin cierto riesgo.

Las herramientas seleccionadas para el desarrollo de aplicaciones en la Universidad son las siguientes:

- SQL*Forms 3.0 para la implementación de todos los módulos interactivos.
- SQL*Menú 5.0 para construir la estructura de menús.
- SQL*ReportWriter 1.1 para la generación de listados.
- Pro*C para la codificación de userexits y procedimientos batch.
- Herramientas CASE para la fase de análisis y diseño técnico.

Las versiones superiores de estos productos podrán irse incorporando al desarrollo de aplicaciones conforme se vaya probando su idoneidad.

El empleo para la generación de código de usuario final de herramientas diferentes a las citadas deberá consultarse previamente. Tan solo se permitirá, en principio, como apoyo al desarrollo (SQL*Plus, SQL*Loader,...) y para mantener, temporalmente, la compatibilidad de aplicaciones antiguas.

6. Objetos de base de datos.

1. Tablas, Vistas y Secuencias.

Las tablas se nombrarán anteponiendo el prefijo *A*T_ a un identificador genérico de no más de 20 caracteres, en donde *A* representa el código de aplicación correspondiente. Por ejemplo, en la aplicación SIGA, cuyo código es M, podríamos definir:

Tabla de Provincias	MT_PROVINCIAS
Tabla de Planes Estudios	MT_PLANES_ESTUDIOS

Las vistas y secuencias se nombrarán con un identificador de no más de 20 caracteres precedido por *A*V_ y *A*S_ respectivamente, siendo *A* el código de aplicación. Por ejemplo, para la aplicación SIGA:

Vista de número de alumnos por provincia	MV_ALUMNOS_PROVINCIA
Secuencia de asignación de expedientes	MS_N_EXPEDIENTE

A cada tabla, vista o secuencia se le asignará una abreviatura de no más de tres caracteres que se empleará en la cualificación de campos en sentencias SQL que involucren a más de una tabla. Por ejemplo, la abreviatura asignada a la tabla MT_PLANES_ESTUDIOS podría ser PE.

2. Indices.

Llamaremos índices primarios a aquellos que identifican unívocamente a cada fila de una tabla (clave primaria o clave candidata). El resto de los índices que se definan en una tabla (no son únicos y se añaden para optimizar algunas sentencias) diremos que son secundarios.

Los índices primarios se nombrarán sustituyendo *A*T_ (siendo *A* el código de aplicación) por el prefijo *A*IP_ en el nombre de la tabla y añadiendo el sufijo _*n*, en donde *n* es un número secuencial que nos permite diferenciar los distintos índices. Por ejem-

plo, en la aplicación SIGE, cuyo código es C, si en la tabla CT_PROVEEDORES se definen dos índices primarios, estos serían:

```
CIP_PROVEEDORES_1

CIP_PROVEEDORES_2
```

Los índices secundarios se nombrarán sustituyendo *A*T_ (siendo *A* el código de aplicación) por el prefijo *A*IS_ en el nombre de la tabla y añadiendo el sufijo _*n*, en donde *n* es un número secuencial que nos permite diferenciar los distintos índices. Por ejemplo, si en la tabla CT_PROVEEDORES se definen dos índices secundarios, estos serían:

```
CIS_PROVEEDORES_1

CIS_PROVEEDORES_2
```

3. Campos.

Los campos pertenecientes a una tabla se nombran anteponiendo el prefijo *C*_ a un identificador genérico de no más de 16 caracteres, siendo *C* el código de tipo de campo: numérico, fecha,... Un listado completo de los tipos de campos aceptados y sus códigos correspondientes se muestra en la *Tabla A.VI, Apéndice A*.

Ejemplos válidos como nombres de campos son:

Fecha de firma del contrato	F_FIRMA_CONTRATO
Porcentaje de becarios	P_BECARIOS

y no serían válidos los siguientes nombres:

F_FIRMA_DEL_CONTRATO	Demasidado largo.
FIRMA_CONTRATO	No contiene código de tipo de campo.
F_FECHA_FIRMA	Referencia el tipo de campo (FECHA).
F_FRMCTO	FRMCTO no es un identificador válido.

Para más detalles de como construir un identificador ver "Generalidades.", pág. 3.

Podrán usarse abreviaturas en el identificador asociado a un campo, pero éstas deben tener un significado consistente en toda la aplicación. Por ejemplo, si en una tabla o campo se abrevia la palabra FACTURA como FAC, en el resto de tablas y campos de la aplicación debe mantenerse la misma abreviatura.

Por último, los campos no deben hacer referencia a la tabla a la que pertenecen. Por ejemplo, la fecha de grabación de la tabla facturas podrá llamarse F_GRABACION, pero no F_GRABACION_FAC. Si en una sentencia SQL se estima necesario hacer esta distinción, se realizará con el cualificador asociado a la tabla. Así, en el ejemplo anterior, escribiríamos FAC.F_GRABACION, suponiendo que FAC es la abreviatura asignada a la tabla de facturas.

7. SQL.

1. Introducción.

La tendencia actual de las herramientas ORACLE es a no emplear SQL directamente sino embebido en lenguajes procedurales como PL/SQL o C. Para trabajos que tradicionalmente se venían haciendo con SQL*Plus, las alternativas son:

- Procedimientos PL/SQL o, mejor aún, programas en PRO*C, para todos aquellos trabajos *batch* de manipulación de datos.
- SQL*ReportWriter para sustituir, con amplia ventaja, las rudimentarias capacidades de generación de listados que aporta SQL*Plus.

A pesar de que no se mantenga como lenguaje independiente, sí es oportuno fijar algunos criterios de programación en SQL dado que éste sigue siendo la base de todas las herramientas de desarrollo de ORACLE.

2. Estilo.

La codificación de cualquer sentencia DML (`SELECT`, `UPDATE`, `DELETE` o `INSERT`) debe emplear la identación para su clarificación, facilitando la búsqueda de las tablas implicadas (`FROM`) y de las condiciones impuestas (`WHERE`). Cuando están implicadas más de una tabla, deben emplearse, asimismo, las abreviaturas asignadas a cada tabla, vista o secuencia para cualificar los campos ayudando de este modo, al optimizador a realizar el *parse* de la sentencia.

Es imposible dar aquí un formato genérico para todas las sentencias que SQL nos permite construir. Damos a continuación algunos ejemplos de codificación, a modo de sugerencia, si bien, cualquier otro estilo es aceptable siempre que ayude a la legibilidad de las sentencias y sea homogéneo a través de toda la aplicación.

Select

Mostramos, en primer lugar, un SELECT simple. Si es necesario recuperar muchos campos, podrían agruparse varios por línea:

```
SELECT   C_IINGRESO,
         D_INGRESO,
         I_INGRESO
  FROM   CT_INGRESOS
  WHERE C_CIFNIF_PROV = :B_1_1.C_CIFNIF_PROV
  AND   C_TIPO_CTA    = W_C_TIPO_CTA;
```

Join

Como se ha comentado en diferentes ocasiones, se emplean las abreviaturas para cualificar los campos cuando se trata de un *join*:

```
SELECT   FAC.C_INTERNO,
         FAC.N_FACT_PROV,
         FAC.F_GRABACION,
         FAC.I_BRUTO,
         FAC.C_TIPO,
         TF.D_TIPO
  FROM   CT_FACTURAS       FAC,
         CT_TIPOS_FACTURAS TF
  WHERE F.C_TIPO = TP.C_TIPO
```

Insert

En los INSERT es importante destacar que deben especificarse los nombres de los campos para mantener la independencia lógica de la sentencia con respecto a los cambios en la definición de la tabla:

```
INSERT INTO CT_TIPOS_FACTURAS
  (C_TIPO, D_TIPO, S_TIPO)
        VALUES
  ('J', 'DOCUMENTO A JUSTIFICAR', 'A JUSTIFICAR');
```

Update

Igualmente, se especifican los nombres de los campos para mantener la independencia lógica en la sentencia UPDATE:

```
UPDATE CT_TIPOS_FACTURAS
  SET
        C_TIPO = W_C_TIPO,
        D_TIPO = W_D_TIPO,
        S_TIPO = W_S_TIPO
  WHERE C_TIPO = W_C_TIPO;
```

Subqueries

Es interesante mostrar, por último, como podrían formatearse los *subquery* anidados:

```
SELECT   C_INTERNO,
         C_TIPO
   FROM  CT_FACTURAS
   WHERE C_TIPO NOT IN (SELECT C_TIPO
                        FROM CT_TIPO_FACTURAS);
```

3. Optimización.

Es en la codificación de sentencias SQL donde el programador puede influir con mayor acierto sobre el rendimiento de la aplicación. ORACLE, Versión 6, utiliza un optimizador basado en reglas lo que implica que el camino elegido para procesar una sentencia depende, exclusivamente, de la sintaxis de la misma y de la estructura de la Base de Datos.

Conocer el comportamiento del optimizador es muy importante tanto para diseñar como para implementar un buen código. Una descripción detallada del mismo, además de muchas otras consideraciones sobre rendimiento y optimización de aplicaciones, puede encontrarse en el documento "Optimización de Aplicaciones. Guía del programador" (en preparación).

8. PL/SQL.

1. Variables de trabajo y de Entrada/Salida.

Los criterios para nombrar las variables son los mismos que los especificados para los campos de las tablas teniendo en cuenta que se antepondrán los prefijos mostrados en la *Tabla A.VII, Apéndice A*.

Si una variable es la copia local de un campo, el nombre de la variable será el del campo, anteponiéndole el prefijo `W_`. Por ejemplo, nombres de variable pueden ser:

```
W_P_BECARIOS

W_N_PAGINA
```

Siempre que una variable sea copia de un campo definido para una tabla de la Base de Datos, se empleará la cláusula `%TYPE` en lugar de declarar explícitamente el tipo y la longitud de la misma. Esto facilita el mantenimiento de los programas.

Los índices empleados para bucles (*loop, for,...*) que no tengan un significado especial se nombrarán simplemente por la letra `I`. Para bucles anidados se añadirán números que indiquen la profundidad.

2. Estilo.

Quizás sea PL/SQL el lenguaje más utilizado en el entorno ORACLE, en especial por su papel fundamental en la codificación de *triggers* para SQL*Foms 3.0.

En principio, al programar en PL/SQL se seguirán los criterios de identación, comentarios del código,... generalmente aceptados

para otros lenguajes (C, Cobol,...). En particular, damos las siguientes recomendaciones que consideramos de interés:

- No declarar procedimientos anónimos, esto es, cada secuencia de código escrito en PL/SQL debe estar incluido dentro de un bloque `BEGIN/END.`
- La identación debe emplearse para aclarar el alcance de los diferentes bloques `BEGIN/END`, así como de sentencias como `IF/ELSIF/END IF, LOOP, FOR,` etc...
- Se incluirán comentarios que clarifiquen el procesamiento. Los comentarios se especificaran con "--" en lugar de utilizar la palabra `REM`. Se añadirán comentarios en el margen derecho de la definición de variables cuya utilidad no sea evidente. Asimismo, se incluirán líneas de comentario en blanco para espaciar el código y separar los diferentes bloques lógicos de procesamiento.
- Se cuidará el ámbito de las variables, definiéndolas dentro de los bloques de código donde son utilizadas y no de modo global.
- Se buscará la consistencia en el empleo de mayúsculas y minúsculas, lo que facilitará la elaboración de herramientas de administración. En general, todo el código se escribirá con mayúsculas salvo los textos entrecomillados. Cualquier otra alternativa será considerada válida siempre que se mantenga la coherencia dentro de toda la aplicación.

3. Cursores.

La posibilidad de emplear cursores hace de PL/SQL una alternativa real a los lenguajes de tercera generación con SQL embebido (Pro*C, Pro*Cobol,...).

PL/SQL permite la declaración de cursores explícitamente para cuyo procesamiento se sigue un esquema `OPEN-FETCH-CLOSE`. Adicionalmente se pueden emplear cursores implícitos (`FOR`).

En el desarrollo de aplicaciones se evitará el uso de cursores implícitos dado que resulta muy fácil que estos queden abiertos cuando ya no se utilizan, haciendo que el número de cursores abiertos aumente innecesariamente.

Los cursores se declararán, por tanto, explícitamente asignándoles como nombre la abreviatura asociada a la tabla sobre la que se abre la consulta o la más representativa en caso de *joins*. En el supuesto de que se declare más de un cursor sobre la misma tabla, se añadirá al nombre de los mismos algún carácter extra que los diferencie.

4. Excepciones y control de errores.

Las excepciones no genéricas que se repitan con frecuencia, deberán estandarizarse dentro del ámbito de cada aplicación. El nombre de las mismas así como su procesamiento (puntos de *rollback*, mensajes,...) se fijará de modo global para toda la aplicación.

Los procedimientos comunes evitarán cualquier interacción con el usuario (mensajes, solicitud de información adicional,...) asignando un código de error a cada posible excepción y dejando que sea el módulo que realiza la llamada el que la procese. Esto aumenta la posibilidad de reutilización del procedimiento.

Por último, se vigilará el uso de la cláusula `WHEN OTHERS` dado que puede contribuir al enmascaramiento de errores. Se deben preveer todos los posibles errores que pueda provocar un determinado bloque PL/SQL y reservar el uso de la cláusula `WHEN OTHERS` para el control de errores inesperados. En el tratamiento de estos errores debería aparecerle al usuario el texto y código (`ORA-`) del mensaje producido para ayudar en la depuración del mismo. Hacemos notar que el comportamiento por defecto de PL/SQL bajo SQL*Forms 3.0 es mostrar dicho mensaje (código y texto) y realizar el *rollback* de la transacción en curso. Este tratamiento suele ser válido en la mayoría de los casos por lo que es preferible no incluir la cláusula `WHEN OTHERS` a añadirla y realizar un procesamiento inadecuado.

Una argumentación análoga nos lleva a vigilar las modificaciones del nivel de mensajes (`SYSTEM.MESSAGE_LEVEL`).

9. SQL*ReportWriter.

1. Sistema de Impresión.

Todos los listados generados por una aplicación emplearán el Sistema de Impresión (SIP, c.f. Sistema de Impresión. Manual del Programador.) por lo que deberán adaptarse a las especificaciones del mismo.

2. Consultas (queries).

Las consultas se nombrarán anteponiendo el prefijo `Q_` a un identificador de no más de 20 caracteres. El identificador se derivará de la tabla sobre la que se realiza la consulta. Para consultas complejas, esto es, *joins* de varias tablas, el identificador se derivará de la tabla más significativa. Si la elección no estuviese clara, se procurará asignar nombres de forma justificada (v.g. `Q_PRINCIPAL` y `Q_DETALLE` para nombrar dos consultas ligadas por una relación Maestro-Detalle).

3. Grupos.

Los grupos se identificarán sustituyendo `Q_` por `G_` en el nombre de la consulta asociada. Tengase en cuenta que éste es el comportamiento por defecto de SQL*ReportWriter.

4. Parámetros de entrada.

Se mantendrán todos los parámetros iniciales (*system parameters*) que incluye SQL*ReportWriter sin modificación alguna.

Los parámetros adicionales se nombrarán anteponiendo el prefijo `I_` al nombre que le correspondiera según las normas dadas para la nomenclatura de campos (ver "Campos.", pág. 13).

Si el parámetro se emplea para imponer condiciones sobre el campo de una tabla, dicho parámetro se nombrará anteponiendo el prefijo `I_` al nombre del campo.

Antes de añadir ningún parámetro, se incluirán aquellos necesarios para soportar la interface con el SIP (c.f. Sistema de Impresión. Manual del Programador.) los cuales se muestran en la *Tabla I*.

Nombre	Tipo de Dato	Longitud
I_C_LISTADO	CHAR	9
I_D_TITULO	CHAR	40
I_D_SUBTITULO	CHAR	40
I_C_EMPRESA	CHAR	14

Tabla I: Parámetros comunes de entrada.

5. *Características globales.*

La amplitud del listado se fijará en 80 o 132 caracteres En casos excepcionales, en los que sea necesario incluir mayor cantidad de información en el listado, se podrá especificar un ancho superior a 132 caracteres, pero considerando que la impresión del mismo estará ligada a las posibilidades técnicas de cada impresora.

En general, los listados tendrán 66 líneas, incluyendo cabeceras y pies de página. Esto garantiza un pequeño margen sobre papel de 12 pulgadas que es el formato de papel continuo estandarizado para las aplicaciones de gestión de la Universidad.

Los preimpresos se diseñarán considerando las especificaciones descritas anteriormente.

Los listados especiales (cheques, sobres,...) podrán crearse con un número diferente de líneas, pero se tendrá en cuenta que este cambio de formato puede no ser admitido por algunos modelos de impresoras.

6. *Atributos.*

Se emplearán atributos para realzar algunos aspectos de los listados (v.g. cabeceras) o para fijar la atención del usuario final (importes totales, nombres propios,...).

Generalmente se emplearán los atributos de negrita y subrayado, si bien, se podrán especificar otros menos frecuentes como itálica teniendo presente que el Sistema de Impresión (SIP) podrá sustituir dichos atributos por otros cuando la impresora a la que se direccione el listado no los soporte.

Para impresos de calidad (cartas, informes,...) puede considerarse la utilización de tipos de letra, proporcionalidad,... siempre que la impresora destino lo soporte (en general nos referimos a impresión láser).

7. *Cabecera de pagina.*

Se incluirá una cabecera de página como la reflejada en la *Figura 9.1.a*. La *Figura 9.1.b* muestra cómo se definiría la misma cabecera en el SQL*ReportWriter.

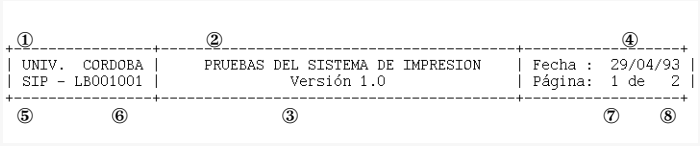


Figura 9.1.a: Cabecera de página en el listado.

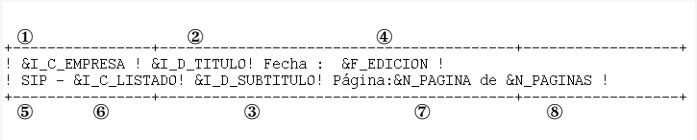


Figura 9.1.b: Definición de la cabecera de página en el SRW.

El significado de cada campo es el siguiente:

① Nombre o código de la empresa (I_C_EMPRESA).
② Título (I_C_TITULO).
③ Subtítulo (I_C_SUBTITULO).
④ Fecha de edición del informe (F_EDICION).
⑤ Código extendido de la aplicación (c.f. *Tabla A.I, Apéndice A*). Escrito en el *background*, no como campo.
⑥ Código del listado (I_C_LISTADO).
⑦ Número de página (N_PAGINA).
⑧ Número total de páginas (N_PAGINAS).

Para soportar dicha cabecera es necesario añadir algunos campos a todos los listados. La *Tabla II* muestra dichos campos así como los atributos de los mismos.

Field Name	Source	Group	Data Type	Field Width	Display Format
F_EDICION	&DATE	REPORT	DATE	9	DD/MM/YY
N_PAGINA	&PAGE	REPORT	NUM	3	
N_PAGINAS	&NUM_PAGES	REPORT	NUM	3	

Tabla II: Campos de cabecera del report.

El Area de Sistemas proporcionará cabeceras estandarizadas que sirvan de muestra para el desarrollo de nuevas aplicaciones.

8. Cabecera de listado.

Se incluirá una cabecera de listado con el mismo formato que la cabecera de página. Además se especificará la descripción de los parámetros de entrada (no genéricos o de interface con SIP) así como sus valores (*Figura 9. 2*).

Listados especiales (cartas, memorándums, preimpresos,...) no incluirán esta cabecera.

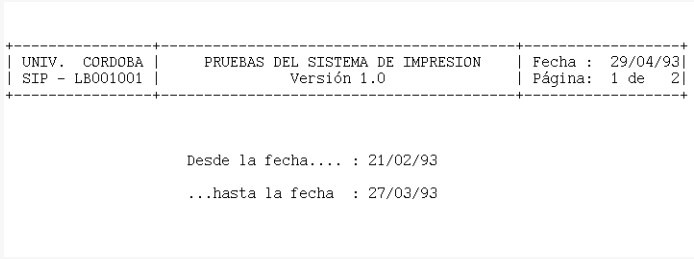


Figura 9. 2: Cabecera de listado.

9. Campos calculados.

Los campos calculados con sentencias tipo `&SQL` se nombrarán siguiendo los mismos criterios especificados para las variables de trabajo en PL/SQL, esto es, anteponiendo el prefijo `W_` a un identificador de campo.

Los campos calculados en base a funciones estadísticas (*Summary fields*) se nombrarán anteponiendo un prefijo identificativo de la función al campo sobre el que se aplica dicha función. Los prefijos admitidos se muestran en la *Tabla A.VIII, Apéndice A*. Los campos calculados a partir de una función acumulativa (*Running*) se nombrarán anteponiendo una `R` adicional.

Por ejemplo, un campo que calcule el mínimo del importe total (`I_TOTAL`) se llamaría `MIN_I_TOTAL`.

10. Consideraciones sobre rendimiento.

Es necesario destacar que SQL*ReportWriter permite anidar SELECT (*queries*) de una forma extremadamente fácil. Además,

los campos obtenidos con la sentencia `&SQL` añaden un grado más de complejidad. Como resultado, SQL*ReportWriter puede generar una consulta muy costosa de resolver para el gestor de ORACLE.

En general, cuidaremos la estructura de cada listado diseñándolo completamente antes de proceder a su programación. Improvisar la obtención de campos adicionales suele conducir a la proliferación innecesaria de sentencias `&SQL`.

Asimismo, cuidaremos que cada consulta se realice al nivel adecuado. Por ejemplo, en un listado Maestro-Detalle, evitaremos recuperar un dato por cada línea de detalle si este puede ser recuperado al nivel de la consulta maestra (si bien esta indicación parece obvia, la experiencia aconseja que no está de más recordarlo).

10. SQL*Menú 5.0

1. Aspectos generales.

SQL*Menu se empleará como única herramienta para la elaboración de menús que permitan a los usuarios finales acceder a los distintos módulos desarrollados, en general, con SQL*Forms 3.0.

Si bien SQL*Forms permite emular las funciones de generador de menús, no se empleará dicha herramienta para tales fines debido a que, en tal supuesto, se perderían las principales funcionalidades que aporta SQL*Menu: sistema de perfiles, menú de *background*, homogeneidad,...

Los menús se ejecutarán en modo *pulldown* por lo que para cada opción se incluirá:

- Una descripción detallada (*Item Text*), de un máximo de 80 caracteres, que aparecerá en la línea inferior del terminal.
- Una descripción abreviada (*Short Item Name*) que servirá para desplegar los menús.

En esta última descripción se tendrán en cuenta las siguientes consideraciones:

- Se evitará abreviar excesivamente la descripción de la función. Es preferible una sola palabra a tres o cuatro abreviadas. Tengamos presente que el usuario siempre verá una descripción completa en la línea inferior.
- Los menús desplegados para acceder a la opción actual permanecen, en general, visibles al usuario. Por tanto, cada opción sólo debería concretar lo ya especificado en los menús previamente desplegados. Por ejemplo, un listado de documentos por fecha podría estar en un menú de tercer nivel:

  ```
  LISTADOS > DE DOCUMENTOS > POR FECHA.
  ```

  En esta opción, `POR FECHA` seria la descripción abreviada y `LISTADO DE DOCUMENTOS POR FECHA`, la

descripción detallada que le aparecería al usuario en la línea inferior.

- Las descripciones se escribirán en mayúsculas excepto que se desee hacer uso de la facilidad de SQL*Menu para acceder por pulsación de inicial. En tal caso se escribirán en minúsculas todas las letras salvo la que se desee que active la opción, la cual irá en mayúscula. Así, en el ejemplo anterior, si la descripción breve se introduce como:

  ```
  por Fecha
  ```

  el usuario podrá pulsar la tecla [F] para acceder a la opción.

La distribución de menús se hará ajustándose a los siguientes criterios generales:

- Salvo raras excepciones reflejará la estructura funcional de la aplicación y no la distribución actual de responsabilidades dentro de los departamentos o servicios implicados.
- Se evitará la repetición de subestructuras funcionales en diferentes puntos del mismo menú.
- Cada usuario sólo verá aquellas opciones que sean accesibles para su perfil. Se procurará que el menú principal de cualquier usuario incluya un mínimo de tres opciones y no más de 7 u 8.
- Se buscará aquella distribución que minimice la profundidad de la mayoría de las opciones, de forma que el usuario pueda acceder a las mismas con no más de 4 o 5 pulsaciones.

2. Nomenclatura.

Tanto el menú (*Aplicacation Menu*) como el menú principal se nombrarán con el identificador *A*`M00M001`, siendo *A* el código de aplicación. Por ejemplo, el menú de la aplicación de Registro (GAD), cuyo código de aplicación es `R`, se denomina `RM00M001`.

El resto de menús de la aplicación se nombrarán anteponiendo *A*`M`, donde *A* es el código de aplicación, a cualquier codificación que el analista responsable de la aplicación considere oportuna.

Se sugiere elegir uno de los siguiente criterios para esta codificación:

- Función-subfunción, siempre que el menú se adapte a la estructura funcional de la aplicación.
- Profundidad-Número_opción, es decir, alguna codificación que haga referencia al número de opción y a la profundidad a la que se encuentra.

3. Procedimientos.

En la nomenclatura de procedimientos asociados al menú, se tendrá en cuenta que el Area de Sistemas se reserva el prefijo `ADM_` para nombrar los procedimientos necesarios para una correcta administración de las aplicaciones.

En particular, siempre existirán los siguientes procedimientos de administración:

- `ADM_ON`, con una argumento de entrada tipo `CHAR`.
- `ADM_OFF`, sin argumentos.

4. Background menú.

SQL*Menu permite agrupar hasta 10 opciones en un menú especial (*Background menu* o Menú de segundo plano) al que es posible acceder desde cualquier punto de la aplicación.

El Area de Sistemas incluirá en él aquellas opciones de propósito general como cambio de *password*, activación/desactivación de trazas,...

La aplicación moverá al *Background menu* aquellas utilidades u opciones de propósito general dentro del ámbito de la aplicación que no estén ligadas funcionalmente: selección del centro y curso academico, ejercicio económico, unidad gestora,...

5. *Perfiles.*

Se definirán tantos perfiles por aplicación como sea necesario para no comprometer la seguridad del sistema, evitando generalizaciones en la definición de los mismos. En interés del Servicio de Informática es importante definir estos perfiles y darlos a conocer a los usuarios finales, aunque los responsables del área correspondiente decidan utilizar sólo algunos de ellos diluyendo responsabilidades.

Los perfiles se nombrarán anteponiendo *APL_* a un identificador genérico de no más de 20 caracteres, donde APL es el código extendido de la aplicación (c.f. *Tabla A.I, Apéndice A*).

En todas las aplicaciones existirá un perfil denominado *APL*_PROGRAMADOR que cubrirá todas las opciones definidas en la aplicación.

6. *Enlace de opciones.*

El encadenamiento de opciones se hará respetando las consideraciones dadas en esta sección.

Menús

Para enlazar un menú simplemente se incluirá el nombre del mismo en la linea de comandos (*Command Line*) y se fijará el tipo de comando (*Command Type*) a 1.

Formas

El tipo de comando se fija a 7.

Se incluye la llamada a los procedimientos de administración previa a la invocacion de la forma:

```
ADM_ON ('Menú_item');
OS_COMMAND ('(TYPE=4) RUNFORM Form_name');
```

en donde,

Menú_item, es el nombre del menú en el que estamos e *item* es el número de opción dentro del mismo (v.g. MM131000_7).

Form_name, es el nombre de la forma invocada, esto es, el nombre del módulo.

Listados

El tipo de comando se fija a 7.

Se ha de tener en cuenta que el enlace se realiza a traves del SIP (c.f. Sistema de Impresión. Manual del Programador.) y no directamente con SQL*ReportWriter. Además se incluye la llamada a los procedimientos de administración. La línea de comandos queda de la siguiente forma:

```
ADM_ON ('Menu_item');
:GLOBAL.C_LISTADO := 'Codigo_listado';
OS_COMMAND ('(TYPE=4) RUNFORM LI201001');
```

en donde,

Menu_item, es el nombre del menú en el que estamos e *item* es el número de opción dentro del mismo (v.g. `MM131000_7`).

Codigo_listado, es el código asignado al listado en el SIP (c.f. Sistema de Impresión. Manual del Programador.).

LI201001, es la forma del SIP que interactúa directamente con el usuario.

Sistema Operativo

En general, no se incluirán opciones que llamen al sistema operativo, ni a nivel de menú ni en ningún otro punto de la aplicación. Si parece imprescindible esta llamada, el Area de Sistemas evaluará sus consecuencias antes de la implantación en el Entorno de Explotación debido a que este tipo de opciones puede comprometer la seguridad del sistema.

11. SQL*Forms 3.0

1. Introducción.

SQL*Forms es la herramienta clave en las tareas de desarrollo dentro del entorno ORACLE. Al soportar la práctica totalidad de la interacción con el usuario final, una completa estandarización del desarrollo con SQL*Forms permitirá obtener un producto homogéneo desde la perspectiva del usuario. Asímismo, puede lograrse uniformidad interna que facilite la colaboración y el traspaso de personal informático entre aplicaciones.

2. Formato de página.

Cada forma se compone de una o varias páginas que ayudan a recoger información o a mostrarla por pantalla. Si la información necesaria implica la utilización de varias páginas, éstas se programarán sobre la misma forma en lugar de hacer una forma para cada página. Esto evita llamadas dinámicas y uso de variables globales en tiempo de ejecución.

Cada página se compone de una cabecera y de uno o más bloques lógicos. Definimos un bloque lógico como un grupo de campos lógicamente relacionados desde el punto de vista del usuario y que se piden o muestran conjuntamente. En general, un bloque lógico estará asociado a un bloque de SQL*Forms, pero esta relación no es necesariamente unívoca. Cada bloque se recuadrará para ayudar al usuario a identificarlo.

3. *Cabecera estandar.*

Todas las páginas visibles por el usuario una cabecera estandarizada como la mostrada por la *Figura 11. 1*.

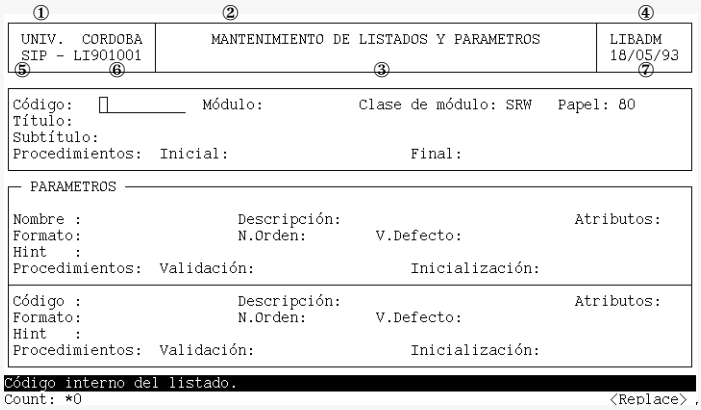


Figura 11. 1: Cabecera de forma.

El significado de cada campo es el siguiente:

① Nombre o código de la empresa (C_EMPRESA).
② Título.
③ Subtítulo (opcional; no aparece en la figura).
④ Código de usuario (SUBSTR(USER,5)).
⑤ Código extendido de la aplicación (c.f. *Tabla A.I, Apéndice A*).
⑥ Nombre del módulo (SYSTEM.CURRENT_FORM).
⑦ Fecha actual (SYSDATE).

El título, subtítulo y código extendido de la aplicación podrán estar en el *background* de cada forma en lugar de tomarse como variables.

Las páginas tipo *pop-up* no llevarán esta cabecera.

Todos los componentes de esta cabecera, salvo las líneas de *background*, se copiarán por referencia de un módulo común que contendrá:

- La definición de un bloque llamado `CAB` con los siguientes campos:

Nombre	Tipo de Dato	Longitud
C_EMPRESA	CHAR	14
F_ACTUAL	DATE	8
C_USUARIO	CHAR	7
C_MODULO	CHAR	8

Tabla III: Campos del bloque de cabecera.

- Un procedimiento de inicialización de dichos campos que tomara `C_EMPRESA` de una variable global, `F_ACTUAL` de la variable `SYSDATE`, `C_MODULO` de la variable `CURRENT_FORM` y `C_USUARIO` de la variable `USER` (eliminando el prefijo `OPS$ con SUBSTR)`.

El Area de Sistemas facilitará un modelo estandarizado de cabecera para que sirva de muestra en el desarrollo de nuevas aplicaciones.

4. Bloques.

Los bloques se nombrarán con un identificador de no más de tres caracteres, al cual se le añadirá un sufijo *_P* indicando el número de página en el que se ha definido. Si el bloque está asociado a una tabla, se procurará tomar la abreviatura estandarizada de la misma como nombre del bloque.

Si el bloque emplea más de una página, se indicará la página de comienzo en el sufijo *_P*. Si hay más de un bloque en la misma página, se añadirá un segundo sufijo *_n* indicando el numero de orden en que dichos bloques aparecen.

Por ejemplo, si dos bloques, uno con datos personales y otro con datos económicos, se definen sobre la página tres de una forma, estos podrían llamarse:

```
DP_3_1

DE_3_2
```

En general, también es aceptable nombrar los bloques con un identificador del tipo `B_`*P*`_`*n*. En el ejemplo anterior:

```
B_3_1

B_3_2
```

5. Campos.

Los campos no asociados a tablas se nombrarán siguiendo los mismos criterios que las variables de trabajo en PL/SQL, esto es, anteponiendo `W_` a un identificador de campo. Si el campo es copia de uno asociado a tabla, se nombrará como éste anteponiendo el prefijo `W_`.

Para variables locales se emplearán, preferentemente, campos de trabajo en la página cero, en vez de declarar variables globales (las cuales ocupan memoria hasta el final de la sesión o hasta que se borran explícitamente).

6. Transacciones.

Durante la fase interactiva de una forma, todas las transacciones serán de sólo lectura. La única transacción que puede actualizar la Base de Datos estará centralizada en la función `ACEPTAR` y no permitirá la interacción con el usuario antes de finalizar por completo la transacción (`COMMIT`).

En cualquier momento durante la interacción con una forma de actualización el usuario tendrá dos funciones disponibles:

`ACEPTAR`. La forma validará los datos y, si todo es correcto, se pedirá confirmación. Si el usuario confirma la actualización, se disparará la única transacción de escritura en la Base de Datos que puede definirse en una forma. Una vez terminada la

transacción (COMMIT), la forma finalizará o devolverá el control al usuario para iniciar una nueva introducción de datos.

CANCELAR — El usuario abandona la forma (sin validación de los datos) y la Base de Datos queda en el estado posterior a la última confirmación aceptada. Si previamente no se había aceptado ninguna modificación, la Base de Datos quedará en el estado anterior a la invocación de la forma.

Esta filosofía de cancelación/confirmación de transacciones:

- Permite centralizar el proceso transaccional evitando bloqueos y problemas de integridad e inconsistencia.
- Ofrece al usuario un mayor control sobre las modificaciones a la Base de Datos clarificándole en que momento se realizan las mismas y dándole la seguridad de cancelar la operación en cualquier momento.

La función CANCELAR no validará los datos introducidos y hará un *rollback* de todas las transacciones pendientes (que, según lo expuesto previamente, siempre serán de sólo lectura) sin pedir ningún tipo de confirmación.

La función ACEPTAR mostrará una ventana de confirmación (*Figura 11. 2*) y se posisionará en el campo NO de dicha ventana. El usuario podrá elegir entre aceptar o no las modificaciones.

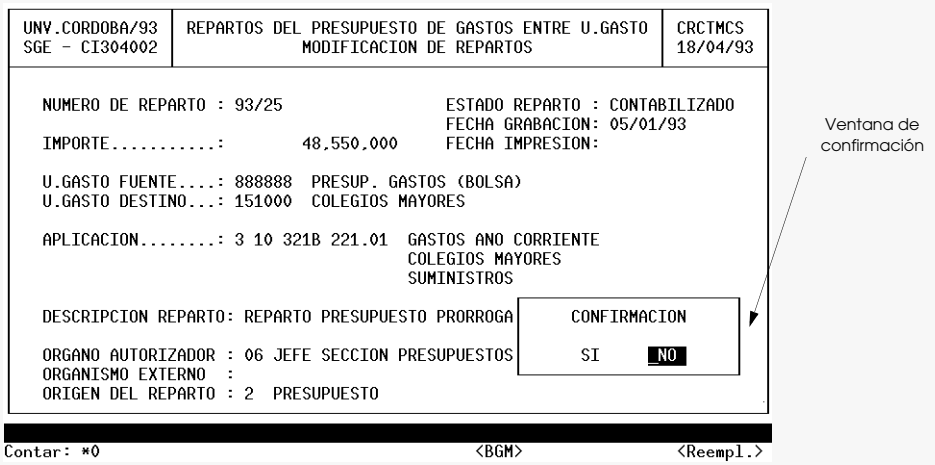


Figura 11. 2: Ventana de confirmación.

El Area de Sistemas facilitará un ejemplo de implementación de la función ACEPTAR que sirva de muestra para el desarrollo de nuevas aplicaciones.

7. Funciones disponibles.

Si bien SQL*Forms permite la definición de múltiples funcionalidades, sólo serán aceptadas las mostradas en la *Tabla IV*. Dicha tabla recoge asímismo el mapeado estandarizado de cada función sobre un teclado VT220. Para un mapeado sobre PC consulte la Guia de Operación.

Función	Tecla
Mostrar teclas†	^K
Ayuda	[AYUDA]
Refrescar pantalla†	^R
Abajo	↓
Arriba	↑
Derecha†	→
Izquierda†	←
Siguiente campo	[RETURN]
Campo anterior	[F12]
Siguiente bloque	[PANT_SIG]
Bloque anterior	[PANT_ANT]
Aceptar	[EJECUTAR]
Cancelar	[PF4]
Lista de valores	[F11]
Introducir consulta†	[SELECCIONAR]
Ejecutar consulta†	[BUSCAR]
Inicializar campo†	[F6]
Borrar carácter anterior†	<X]

Tabla IV: Listado completo de teclas y funciones.

Función	Tecla
Borrar carácter actual†	^D
Borrar registro	[BORRAR]
Insertar / Sobreescribir†	^A
Primera línea†	[PF1] ↑
Ultima línea†	[PF1] ↓
Editar	[PF3]
Imprimir	^P
Teclas de aplicación	[F7], [F8], [F9],[F10]
Mostrar menú de utilidades†	[PF1] 0
Menú de utilidades: opciones 1 a 9†	[PF1] 1 a [PF1] 9

Tabla IV: Listado completo de teclas y funciones.

Las funciones marcadas con una daga (†) no son redefinibles por *trigger* desde SQL*Forms.

El resto de funciones se inhibirán, por defecto, en todas las formas y se irán definiendo conforme vaya siendo necesario. Cada forma incluirá un trigger `KEY-OTHERS` anulando dichas funciones. Este trigger se implementará incluyendo el siguiente código:

```
BEGIN
  NULL;
END;
```

Esto anula todas la teclas por lo que la forma deberá ir programando todas aquellas funciones que necesite de entre la funciones permitidas.

En esta sección se explica en detalle cada una de las funciones incluyendo el nombre del *trigger* con el que se programarán y la ayuda que le aparecera al operador cuando pulse Mostrar Teclas (en la pantalla *Trigger Definition* de SQL*Forms se marcará una `X` en `Show Keys` y se incluirá esta descripción en el campo `Descrip`).

Ayuda.

Trigger: KEY-HELP

Ayuda: Ayuda.

Accede al Sistema de Ayuda Interactiva (SAI). Consulte el Manual del Programador.

Abajo.

Trigger: KEY-DOWN

Ayuda: Abajo - Siguiente página.

En un bloque multiregistro, mueve el cursor al campo actual del siguiente registro. La ayuda al operador será “Abajo”.

En un bloque monoregistro sobre el que se haya hecho un *fetch* de más de un registro, mueve el cursor al campo actual del siguiente registro. Esto aparece ante el operador como un cambio de página por lo que la ayuda al mismo será “Siguiente página”.

Antes de abandonar el registro actual, deben dispararse todas las validaciones necesarias para garantizar la integridad del mismo.

Arriba.

Trigger: KEY-UP

Ayuda: Arriba- Página anterior.

En un bloque multiregistro, mueve el cursor al campo actual delregistro anterior. La ayuda al operador será “Arriba”.

En un bloque monoregistro sobre el que se haya hecho un *fetch* de más de un registro, mueve el cursor al campo actual del registro anterior. Esto aparece ante el operador como un cambio de página por lo que la ayuda al mismo será “Página anterior.”.

Antes de abandonar el registro actual, deben dispararse todas las validaciones necesarias para garantizar la integridad del mismo.

Siguiente campo.

Trigger: KEY-NXTFLD

Ayuda: Siguiente campo.

Avanza, previa validación, al siguiente campo del bloque. Si no hay más campos en el bloque actual se avanza al primero del siguiente. En este caso se dispararán las validaciones cruzadas necesarias para garantizar la integridad del bloque antes de abandonarlo.

Si no hay más bloques se le indicará al operador que pulse [EJECUTAR] para confirmar las modificaciones.

Campo anterior.

Trigger: KEY-PRVFLD

Ayuda: Campo anterior.

Retrocede, previa validación, al campo anterior del bloque. Si no hay más campos en el bloque actual se avanza al último del anterior. En este caso se dispararán las validaciones cruzadas necesarias para garantizar la integridad del bloque antes de abandonarlo.

Si no hay más bloques se le indicará al operador que pulse [EJECUTAR] para confirmar las modificaciones.

Si el campo es obligatorio pero se deja a nulo, se permitirá retroceder al campo anterior. Esto permite al usuario corregir errores en campos relacionados. La validación a nivel de bloque garantizará que el campo no quede nulo.

Siguiente bloque.

Trigger: `KEY-NXTBLK`

Ayuda: Siguiente bloque

Avanza al siguiente bloque lógico sobre el que se puedan introducir datos (campos con *input allowed*).

Antes de abandonar el bloque actual, deben dispararse todas las validaciones necesarias para garantizar la integridad del mismo.

Bloque anterior.

Trigger: `KEY-PRVBLK`

Ayuda: Bloque anterior.

Retrocede al bloque lógico anterior.

Antes de abandonar el bloque actual, deben dispararse todas las validaciones necesarias para garantizar la integridad del mismo.

Aceptar.

Trigger: `KEY-COMMIT`

Ayuda: Aceptar.

En las formas de selección, esto es, aquellas que simulan una lista de valores, acepta el registro actual como el seleccionado y devuelve el control y el `ROWID` de dicho registro a la forma que hizo la llamada.

En los bloques que actúan como maestros de otros (relación *Master-detail)*, se dispararán todas las validaciones del bloque maestro y, una vez superadas, se accederá al bloque detalle, posiblemente disparando una *query* sobre el mismo.

En formas de introducción y/o modificación de datos, se realizan todas las validaciones cruzadas necesarias y, si todo es correcto, se muestra la ventana de confirmación (*Figura 11. 2*). Si se con-

firma la operación, se llamará a un procedimiento en el cual se centraliza toda la transacción. Al final del mismo se devolverá el control al menú, `exit_form(no_validate)`, o bien se permitirá que el usuario continúe trabajando sobre la misma forma (`clear_form(no_validate)` e inicialización identica al *trigger* `KEY-STARTUP`).

Cancelar.

Trigger: `KEY-EXIT`

Ayuda: Cancelar.

En una forma de selección, esto es, una forma que simula una lista de valores, devuelve el control a la forma que hizo la llamada sin realizar ninguna selección.

En la ventana de confirmación (*Figura 11. 2*) vuelve al punto dónde se pulsó [EJECUTAR].

En cualquier otra situación, devuelve el control al menú de la aplicación. Los cambios realizados hasta ese momento desde la última confirmación o desde la activación de la forma serán desestimados (*rollback*).

Lista de valores .

Trigger: `KEY-LISTVAL`

Ayuda: Lista de valores.

Muestra la ventana de lista de valores para el campo actual. En general se ejecutará en modo *restrict* salvo que la codificación no lo aconseje.

Borrar registro.

Trigger: KEY-DELREC

Ayuda: Borrar.

Marca el registro actual para ser borrado eliminándolo de la pantalla. El usuario confirmará la operación con la función Aceptar. Si abandona la forma con la función Cancelar el registro no será borrado.

Editar.

Trigger: KEY-EDIT

Ayuda: Editar.

En los campos con descripciones largas, se incluirá esta función para ayudar al operador a introducir el texto.

Incluir esta facilidad es opcional y se deja al criterio del Analista responsable de la aplicación.

Imprimir.

Trigger: KEY-PRINT

Ayuda: Imprimir.

Se accede al Sistema de Impresión (SIP). Consulte el Manual del Programador.

Teclas de aplicación.

Trigger: `KEY-Fn`

Ayuda: *Variable.*

Se podrán asignar a las teclas [F7], [F8], [F9] y [F10] (trigger `KEY-F7`, `KEY-F8`, `KEY-F9` y `KEY-F0`) funciones propias de cada aplicación.

Se incluirá una ayuda al operador específica de la función que se defina.

Una misma tecla de aplicación no se redefinirá con diferentes funciones dentro de la misma forma. Se procurará, asímismo, guardar la coherencia interna dentro de una aplicación asignando siempre las mismas funciones a las teclas de aplicación.

Si una función no definida aquí se considerase de interés general, se añadirá a este Documento en vez de programarla como tecla de aplicación liberando éstas para otros usos.

8. Consultas.

Las consultas (`ENTER_QUERY` / `EXECUTE_QUERY`) se restringirán a aquellos campos que formen parte de índices. Además, los Manuales de Usuario de las diferentes aplicaciones limitarán la explicación del modo *enter query* a las siguientes posibilidades:

Igualdad: Condiciones del tipo `campo=valor`.

Proximidad: Condiciones del tipo `campo LIKE valor%`. No se permitirán búsquedas del tipo `%valor`.

Comparación: Condiciones del tipo `campo > valor`.

Se evitará hacer referencia a las posibilidades extendidas del modo *enter query* (aquellas a las que se accede introduciendo variables en los campos antes de ejecutar la consulta).

9. Procedimientos.

Los procedimientos se programarán siguiendo las directrices dadas para PL/SQL.

No se emplearán trigger de usuario dado que estos existen sólo por compatibilidad con la versión 2.3 de SQL*Forms. Por la misma razón, no se empleará la posiblidad de SQL*Forms de generar relaciones maestro-detalle automáticamente.

Se potenciará el uso de procedimientos para modularizar el código y facilitar la lectura del mismo.

Los procedimientos comunes a más de una forma se declararán en una forma diferente y serán llamados por referencia. Cada procedimiento se incluirá en una forma independiente que no tendrá ningún procesamiento adicional.

El Area de Sistema proporcionará una muestra de este tipo de formas para ayudar en el desarrollo de nuevas aplicaciones.

12. Pro*C

1. Introducción.

Se cuidará que la mayor proporción de código de una aplicación esté desarrollada en PL/SQL. Sin embargo, se pueden emplear lenguajes de tercera generación en una de las siguientes situaciones:

- Cuando el rendimiento del procedimiento a programar sea crítico y se haya comprobado que es imposible obtener buenos resultados desde PL/SQL.
- Cuando sea necesario utilizar recursos del sistema no accesibles desde PL/SQL.
- En la programación de validaciones e inicializaciones para el Sistema de Impresión (c.f. Manual del Programador).

Como lenguaje *host* se empleará Pro*C, estando supeditada la utilización de otros lenguajes a su disponibilidad y a la conveniencia de los mismos.

2. Estilo.

No existen normas de estilo universalmente aceptadas para el lenguaje C y tampoco se darán aquí. Se incluyen, no obstante, las siguiente recomendaciones:

- Seguir las sugerencias dadas en este Documento para otros lenguajes (PL/SQL, SQL, ...).
- Adoptar una norma común dentro de cada aplicación que unifique el aspecto global de todos los módulos en Pro*C (esto no es dificil si se tiene en cuenta que el número de

módulos en Pro*C dentro de una aplicación es muy limitado).

- Incluir cabeceras con comentarios sobre el módulo: propósito, argumentos, notas de uso,...
- Cuidar la definición de variables externas dado que pueden existir problemas de interferencias con otros módulos.
- Cuidar el uso de punteros. Dado que el unix no permite proteger el código de forma satisfactoria, un uso inadecuado de punteros podría tener efectos colaterales dificilmente detectables.

A continuación damos un ejemplo de *userexit* desarrollada en Pro*C para el Sistema de Impresión (c.f. Manual del Programador):

```
/*
-------------------------------------------------------------
 Modulo: MPS00001.pc - Inicializa el nombre del secretario.
-------------------------------------------------------------

 Argumentos
 de entrada :

        I_C_CENTRO NUMBER ( 3) Codigo de centro.

 Descripcion :

     Recupera el nombre del secretario de la tabla MT_CENTROS
     para el centro especificado como argumento de entrada.

-------------------------------------------------------------
*/

#include <stdio.h>
#include "usrxit.h"


EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;

/* Parametros de entrada */

VARCHAR i_c_centro[3];

/* Variables de trabajo */

VARCHAR w_d_centro[81];

EXEC SQL END DECLARE SECTION;
```

```
extf int MPS00001 ();

int MPS00001 (param, paramlen, erm, ermlen, query)

 register char *param;
 int      *paramlen;
 char     *erm;
 int      *ermlen;
 int      *query;

{
 char w_t_spc = ' ';
 char w_t_dummy[8];


/* Paso del string de entrada a variables independientes */

 strcpy (w_t_dummy, strtok(param,&w_t_spc));
 strcpy (i_c_centro.arr, strtok(NULL,&w_t_spc));
 i_c_centro.len = strlen(i_c_centro.arr);
 i_c_centro.arr[i_c_centro.len] = '\0';


/* Seleccion del nombre del secretario para el centro
especificado */

 w_d_centro.arr[0]='\0';

EXEC SQL  SELECT  D_CENTRO
           INTO   :w_d_centro
           FROM   MT_CENTROS
           WHERE  C_CENTRO = :i_c_centro;

w_d_centro.arr[w_d_centro.len]='\0';

/* Inicializacion del parametro */

 PUT_VAL(w_d_centro.arr);

 return(IAPSUCC);
}
```

Los argumentos de entrada no se han adaptado a los estándares por respetar la interface tradicional de *userexits* con SQL*Forms.

A. Tablas de Referencia.

Código de Aplicación	Código extendido de Aplicación	Aplicación
A	CMN	Administración y Utilidades comunes
S	SGC	Sistema de Gestión de Código
L	SIP	Sistema de Impresión
H	SAI	Sistema de Ayuda interactiva
D	SSD	Sistema de Seguridad de Datos
M	SGA	Sistema de Gestión Académica
C	SGE	Sistema de Gestión Económica
R	GAD	Sistema de Gestión Administrativa (Registro)
P	SGP	Sistema de Gestión de Personal
U	GAU	Gestión de Acceso a la Universidad

Tabla A.I: Aplicaciones

Tipo de Usuario	Descripción
d	Desarrollo
m	Usuario SGA
c	Usuario SGE
r	Usuario GAD
p	Usuario SGP
i	Usuario Intervención

Tabla A.II: Tipos de Usuarios.

Tipo de Usuario	Descripción
s	Usuario Dirección

Tabla A.II: Tipos de Usuarios.

Código de Localización	Descripción
agr	E.T.SI.A.M.
cen	Secretaría Central
cie	Facultad de Ciencias
cri	Instituto Andalúz Interuniversitario de Criminología
dep	Ciencias Morfofuncionales del Deporte
der	Facultad de Derecho
egb	E.U. del Profesorado de E.G.B.
enf	E.U. de Enfermería
fil	Facultad de Filosofía y Letras
med	Facultad de Medicina
min	E.U. de Ingeniería Técnica Minera
pol	E.U. Politécnica
rct	Rectorado
soc	E.U. de Estudios Sociales
vet	Facultad de Veterinaria

Tabla A.III: Localizaciones.

Código	Modo	Descripción
I	Interactivo	El usuario interactúa directamente con el módulo.
B	Batch	La activación del módulo y las condiciones iniciales las fija el usuario por medio de un módulo interactivo (I). Una vez activado, la ejecución del módulo se realiza bajo el control del sistema operativo.
A	Automático	El usuario no interviene directamente en la activación del módulo, sino que esta es automática siempre que se cumpla una condición prefijada en el diseño (v.g. periódicamente). Una vez activado, la ejecución del módulo se realiza bajo el control del sistema operativo.
P	Procedimiento	El módulo se ejecuta cuando es llamado por otro módulo y no contiene elementos interactivos. Una vez finalizada la tarea, el control es devuelto al módulo que hizo la llamada.
D	Definición	Módulos empleados por el Area de Sistemas para definir la estructura de las bases de datos.
M	Menú	Módulos que intervienen en la construcción de menús.

Tabla A.IV: Modos de Ejecución.

Clase de Módulo	Descripción
sh	Procedimiento de comandos Bourne shell.
sql	Procedimiento SQL genérico.
inp	Fuente SQL*Forms
frm	Ejecutable SQL*Forms
rpt	Fuente SQL*Report.
pc	Fuente Pro*C
dmp	Datos corporativos de inicialización. Formato EXPORT.
ins	Procedimiento SQL para definición de objetos de la base de datos.
h	Include file para Proc*C
rex	Fuente SQL*ReportWriter
sip	Inicialización de tablas del SIP para cada listado.

Tabla A.V: Clases de Módulos.

Código	Descripción
C	Código. Número identificativo. Opciones.
D	Descripción larga (más de 12 caracteres).
F	Fecha (fipo *Date*).
I	Importe.
L	Lógico (dos valores): verdadero/falso, si/no,...
N	Número: real, entero,...
P	Porcentaje.
S	Descripción corta. Sólo tiene setido cuando exite una descripción larga.
T	Texto genérico.

Tabla A.VI: Codificación de campos.

Prefijo	Descripción
I_	Dentro de un procedimiento, para todos los argumentos de entrada.
O_	Dentro de un procedimiento, para todos los argumentos de salida.
B_	Dentro de un procedimiento, para todos los argumentos de entrada/salida.
W_	Para todas las variables de trabajo que no pertenezcan a ninguna de las categorías antes mencionadas.

Tabla A.VII: Variables de trabajo.

Función	Prefijo
SUM	S_
MIN	MIN_
MAX	MAX_
COUNT	N_
AVERAGE	M_
%TOTAL	P_
FIRST	F_
LAST	L_

Tabla A.VIII: Nomenclatura de campos calculados.