Pitchmark Ltd

Computer Systems - Standards Manual

Author: Peter N. Crompton

Version: 1/00

Version date: 27-Sep-1996

Amendment History

Version	Author	Date	Description of changes
1/00	P Crompton	27-Sep-1996	First issue

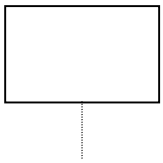
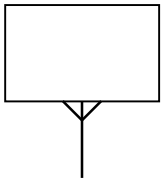
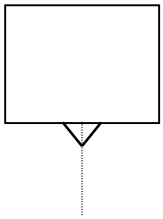
1. INTRODUCTION

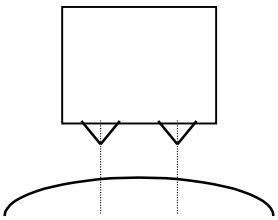
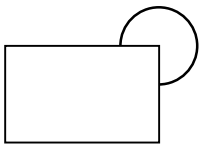
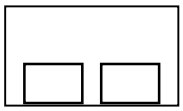
This standards manual lists the minimum that is required at each stage of the project life cycle. The phases are :- Analysis, Design, Detailed design (Module specification), Module coding,
Unit testing, System testing and implementation.

2 ANALYSIS

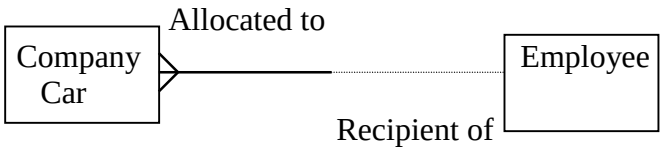
The skeleton Analysis Report should be used as a template for the output from this phase of the project. The sections below give details of diagram standards to be used in the report.

2.1 Entity model diagram standards

Symbol	Meaning
	Entity (a collection of data representing an object or thing of importance) Entities may be connected by relationships. Each relationship has two ends, each end being one of the following elements.
	Mandatory relationship. Indicates that for each occurrence of this entity, the relationship MUST be true.
	Optional relationship. Indicates that for each occurrence of this entity, the relationship MAY be true.
	Multiple relationship. Indicates that for each occurrence of the other entity, the relationship MUST be true for ONE OR MORE occurrences of this entity
	Indicates that for each occurrence of the other entity, the relationship MAY be true for ONE OR MORE occurrences of this entity.

Symbol	Meaning
	Arcs are used to indicate an "exclusive or" between two relationships. The entity may be related to one OR the other but not both.
	Relationships can be "involuted" i.e. Entities can relate to themselves.
	Entities can have sub-types

Each entity should be labelled. Both ends of the relationship should also be labelled meaningfully.
e.g.



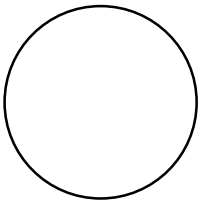
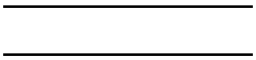
This example shows that :-

Each employee *may be* the recipient of one or more company cars

Each company car *must be* allocated to one and only one employee.

NB Entity names should be singular e.g. Company

2.2 Data flow diagrams Standards

Symbol	Meaning
	External entity. An individual, organisation, department, computer system etc., which is external to the system. Duplicates are identified by a small diagonal in the corner.
	A process. Work which is done as part of the system, this may be manual or automated.
	Data store. Any repository for information which involves some implied time delay between depositing and retrieving data.
	Data flow. Any flow of information for whatever purpose between any of the three elements above.

Each symbol should be labelled meaning fully.

3 DESIGN

The skeleton design report should be used to present the output from the design phase of the project. The sections below list standards which constrain the design and make the approach across the system more uniform.

3.1 Physical data model

3.1.1 *Table names* should follow entity names as closely as possible. The names may be shortened if required, but must provide a clear definition of the data contained in the table.

3.1.2 *Table names* should be plurals e.g. Companies.

3.1.3 *Short table names.* Every table should have a unique 3 character short name associated with it. e.g. Companies has short name cmp

3.1.4 *Column names* should reflect attribute names as much as possible, but should be as short as possible. Columns of similar types should be identified by similar names.
e.g. Start_date, End_date.
Date only fields should be post-fixed with _date. If the time is important, and to be maintained, the field should be post-fixed with _date_time.

3.1.5 *Foreign keys* should be held in the format table prefix column name.
e.g. Companies.Id is the primary in this table. In table Company_results table this field is referenced as Company_results.cmp_id

3.1.6 *View names* should reflect the definition of the data that can be retrieved from the view. When defining views careful consideration should be given to the number of rows that may be retrieved. Views which join together two or more interactive tables should be avoided in screens. Unless the view falls into the above categories the use of a view should be avoided. Wherever possible views which specify the use of group by or order by should be avoided, along with the use of the union, intersect and minus.

3.1.7 *Temporary tables* should be prefixed with 'z_temp_' followed by the table name.

3.1.8 *Reference values* and types should be held in one table called ref_values. The table consists of three columns Ref_code, Ref_descr, Domain_name. The code and the domain should not allow update by the user. If Case dictionary is in use the domain name and all of the values should be defined in case along with a description.

3.1.9 *Indexes* should be defined for all unique identifiers and foreign keys on tables. Any combination of columns which are obviously used on many occasions should also be indexed.

3.1.10 *Index names* should be formed in the following way :- The 3 character short table name plus the word prime for the primary key index OR a number for other types of indexes.
e.g. The primary key index on companies should be cmp_prime, other indexes should be cmp_1, cmp_2 etc.

3.1.11 *Audit columns* All tables should contain the audit columns listed below:-

Column name/type		***Description***
audit_user	Varchar2(30)	The oracle user id of the person that made the change.
audit_action	Varchar2(1)	Was this record I - Inserted, U - Updated, D - Deleted. Deleted is only used in history tables
audit_module	Varchar2(13)	The module Id compete with the version number that created or changed the record.
audit_date_time	Date	The date and time that the record was last created or updated.

Storing this audit data allows problems to be easily traced in the maintenance phase of the project.

3.1.12 *Database constraints* All of the benefits of constraints are available via database triggers. The form in which database triggers return error messages does not allow the use of the standard error messaging and debugging tools that form part of these standards, for this reason they should not be used.

3.1.13 *Database triggers* should only contain basic elements of logic. The main bulk of the logic should be contained in packages which can be called form elsewhere. Specifically DML should not appear in triggers. The testing of column values and the passing of values to procedures can be included.
NB Some oracle products such as forms update every column in a base table when commit is pressed. For this reason in update triggers the processing should test the new and old values of columns to make sure that a trigger is being correctly fired for column specific processing.

3.1.14 *Database procedures* must be included within a package.

3.1.15 *Database functions* must be included within a package.

3.1.16 *Database packages* should be kept to a reasonable size. Large packages are likely to be ‘aged out’ of the SGA more frequently. Packages should contain appropriately related procedures and functions, as executing a function or procedure from the package brings the whole package into the SGA.

3.1.17 *Database roles* Under investigation.

4. MODULE SPECIFICATION

The skeleton module specification, that is appropriate to the module type, should be used to present the output from this phase of the project. The skeleton specification types currently identified are :- Screen forms3.0, Non-screen, database trigger, database package.

4.1 Module naming standards

Modules should be given unique names. The module name should be used as the first part of the file name that contains the code when the module is implemented. Module names should be of the form AAA A 999.

The first three alpha characters indicate the system area e.g. An infrastructure module might be called inf.

The fourth character indicates the type of module e.g. The types identified thus far are :-

Module Type	File extensions
B - Background jobs	.pco, .cob, .pc, .c, .sql
D - DCL/ Script files	.com
E - User Exit	.pco, .cob, .pc, .c
F - Subroutines	.pco, .cob, .pc, .c
I - Include files	.txt, .h
S - Screen	.inp
R - Report	.sql, .rex
P - Database package	.sql
T - Database trigger.	.sql

999 - Is unique number within the application area.

Database triggers have a slightly different naming style that indicates their purpose.

YYY_XXX.sql will be the name of the source file and YYY_XXX will be the database trigger name.
YYY is the three character short table name. XXX indicates when the trigger is fired.
e.g. BSU - Before Statement Update, ARI - After Row Insert, BRD - Before Row Delete etc.

Data file names. Some static data tables may be populated via sql loader. The sql loader files will have names of the following format XXX.ctl where XXX is the three character short table name.

Database procedures. As procedures must be included within a package, (A standard) they do not require a source file name. However database procedure names should be of the form dp_XXXXXX where dp indicates a database procedure and XXXXXX is a meaningful name for the procedure up to twenty characters long.

Database functions must be included in a package and have names of the form df_XXXXXX where df indicates a database function and XXXXXX is a meaningful name up to twenty characters long.

Database constraints These should not be used see design.

4.2 Error processing

In general there should be 3 types of error :-

4.2.1 *System errors*. These should be used if an oracle error or operating system error occurs. Processing should be aborted if this type of error is raised.

4.2.2 *Validation Errors*. These should be used in cases where the user has entered invalid data and may correct it. Processing may continue after corrective action has been taken to alter the data. Where ever possible the following standard error messages should be used in screens:-

Cannot commit changes here
Cannot delete records here
Cannot update records here
Cannot insert records here
No previous page
No further pages
List of values is not available in this field
Function not allowed here
Open queries are not allowed in this block

4.2.3 *Warnings.* These should be used to draw the users attention to some part of the processing. They should NOT be used in commit time processing. Processing should continue after the message has been acknowledged. This type of message should be used sparingly.

5..MODULE CODING

5.1 Screen forms

The bulk of user interaction with the system is via screen forms. It is very important that forms have a similar format and operate in a consistent way. This reduces the amount of training required by users and is the mark of a quality system. The following standards help to produce consistent forms.

5.1.1 *General standards*

The code skeleton for screen forms should be used as the basis for any new module.

Screen forms locking strategy When inserting / Updating / Deleting from non-base table fields shared locks should be taken out on the non-base tables in pre-commit triggers, immediately after validation checks have been performed. Non-base tables should be updated in pre-commit triggers if they are alphabetically before the base table. If a table is alphabetically after the base table it should be update in the post-commit triggers. This approach across all applications minimises the risk of dead locks. (NB alpha characters come before underscore.)

Referential Integrity. Where a parent child relationship exists between blocks the user should not be able to delete the parent record if children exist. The validation should be performed in the key-delrec trigger and the pre-delete trigger (another use may have added dependent records) The user should delete the dependent records before a delete is allowed on the parent.

The primary key function in forms should not be used, this function exists specifically for compatibility with previous versions of forms.

Coding should be placed in *procedures rather than triggers*. The trigger should then be left as a list of calls to procedures embedded in conditional statements. This applies even when the trigger contains unique code.

Comment all non-standard or difficult processing within the form. The version number of the module MUST be incremented every time a change is made. The comments at the top of the programs must also indicate the change made.

5.1.2 Procedures and functions

Procedures must have as the first line a brief comment or title that describes the functionality. This is required so that when the procedure is viewed in the spread table the purpose is instantly known.

5.1.3 Triggers

Triggers should contain the minimum of logic. Large chunks of logic should be placed in procedures when are then called from the trigger. This reduces duplication of code within the form and gives the form a consistent approach.

The Post-form trigger should be used to erase any global variables that are not being used to pass information to the calling module.

key-commit should not be used for anything other than commit form within a maintenance form.

Key-others must be used to disable keys that are not required. Any key that is required must be explicitly enabled.

5.1.4 Block level

Block synchronisation. In query only forms all blocks which are dependent on the master block should be queried if a record is returned in the master block. Blocks which are detail blocks of other detail blocks should not be queried until the user moves to that detail block. Detail blocks should be cleared on return to the master block. In forms that maintain data blocks should not be queried automatically. Detail blocks should be cleared on return to the master block.

Synchronisation. Once a master block has been populated details block are populate only when viewed. Therefore if a detail block is on the same page it is queried at the same time as the master block. If the detail block appears on a different page it is populated only when navigated to. The blocks remain populated until the master block is changed. Detail blocks are therefore do not have to be re-queried once populated.

A master record must exist prior to navigating to details.

Comments should be at the top of each block explaining its purpose.

Block names should be of the following form:-B01, B02, B12. Where the first digit indicates the page and the second digit the sequence with in the page as seen by the user.

5.1.5 Field level / Variables

Global variables must not be used where a local variable or formal parameter can be used. Globals should be reserved for passing parameters between screens.

Wild card queries must not be allowed on the unique identifier unless specified.

Field labels must be as close to the database column name as possible.

Unless specified otherwise *all fields must be in upper case.*

NBT_ should prefix any field that is non base table. The rest of the field should be the name of the source columns in the database.

When dates are compared they should have the trunc( ) function applied to them to remove the time element.

Non-base table fields MUST be commented. Indicate how the field is populated and what its for.

All fields in a control block must be prefixed with CTL_.

Fields not displayed should be ordered so that they appear after displayed fields.

List of values should be available for every field that is derived from the database.

Variables based on database columns must be declared as the same size as those in the database, except in the cases where the functionality of the form requires otherwise.
e.g. the use of '>' on NUMBER fields, or the use of #IN, or that there is insufficient room on screen to display the full length of the field.

5.2 Sqlplus v3.2

Comments. Each SQL statement must be preceded by a comment explaining in business terms its purpose.

Complex SQL statements must *state the indexes used*, as derived from using the explain plan facility. If the cost based optimiser is used then this will cease to be a standard.

Table aliases must be used. The alias should be the three character short table name . When the table is used more than once post fix the name with 1,2,3 etc.

When *inserting/updating a date field* the trunc function should be used to remove the time portion.

Code layout. Key words MUST be in upper case, all other text should be in lower case e.g. table names, columns, function, procedures etc.

Indentation should be used to clarify the structure of code along with plenty of while space aid readability.

Brackets should be used to clarify precedence rather than using defaults.

An order by clause must use the column name not its position in the select list, except where using set operations such as UNION.

Indexing. For complex statements. The indexes that are to be used should be specified and other indexes should be disable regardless of whether it is used by the optimiser.
Disable date and number indexes by using + 0. Disable character indexes by appending null using ||'' to the index field.

The tables listed in the from clause should be listed so that the driving table is listed first, then in table access order.

5.3 PL/SQL v2.2

An reserved word should be in upper case and function procedure or variable name should be in lower case. This is the standard that oracle uses when generating code.

All cursor names should begin with c_

Variable names within triggers, procedures, functions and packages should be named in the following way :

Public variables - LPUBLIC_xxxxx These are available to all processes within an oracle session and are declared in the package specification.

Private variables -LPRIV_xxxxxx These are visible to all processes within the package body and are declared in the package body.

Local variables - L_xxxxxx These are available only to the process in which they are declared.

Date variables - Any program using dates should retrieve and report them in format DD-Mon-YYYY. If this format is not used alternative format must be commented.

Version numbers. Triggers and packages must define the variable LPRIV_PROG_VER this is part of the standard coding skeleton is each case. This variable contains the module id and the version number.

The first command in every object should be to call the standard infrastructure packages debugging procedure. This is part of the coding skeleton which must be used.

The exception handler in an object must call the infrastructure package to record exceptions. This is part of the standard coding skeleton.

Creating objects. Packages and triggers should be created using the create or replace option. This will make the application of the scripts simpler.

Variables. When defining the variables the %TYPE key work should be used. This allows a variable to be defined to that of a database column. This minimise the impact of a column size change on code.

In cursors try to use %ROWTYPE wherever possible this helps to reduce the number of changes required to modules should a column size need to be changed.

6 MODULE UNIT TESTING

6.1 Screen forms

This section lists the minimum set of tests that should be performed on a module.

6.1.1 Code inspection

These tests are not dependant on data being entered and are more of a quality assurance check.

Check the screen layout against the layout in the specification.

Check field sizes and attributes against the specification.

Check the base table used in each block.

Check that the comments and version number have been updated.

6.1.2 Positive testing

This stage concentrates on field and block level triggers. It makes basic checks on the functionality included within the form. It also checks that data that should be accepted is.

Check field level trigger processing to make sure that no validation has been omitted. This should be documented on the unit test plan form using the specification as the input.

Check block level triggers by trying to insert / update / delete and select from each block as per the specification.

Check that blocks are synchronised correctly.

6.1.3 Negative testing.

This stage ensures that the module prevents any data processing from occurring when the data input is invalid or an illegal operation is attempted.

Check every field that has been identified as having a validation rule against it. Checks might include validation ranges and limited responses. e.g. In a Y/N field entered invalid data such are Null, X etc.

The fields in a block which form the primary key should be checked to make sure that duplicate value errors are trapped correctly.

Each block should be tested to make sure that commit, insert, update, delete and query are allowed as specified.

In all cases, field level and block level, enter values that should be rejected and make sure that the screen indicates in a satisfactory manor that the data is invalid.

7 SYSTEM TESTING

7.1 Functional testing

The design report should be used as the main input for the testing plan. The most important business cases should be tested first.

7.2 Concurrency testing

Some form of concurrency testing should be tried using the same module multiple times over. Commit should be attempted at the same time for these modules. This should check that the locking strategy has been adhered to. This is important for modules which are the most often used.

7.3 Volume testing

An attempt should be made to volume test a new product to check that is can meet the desired level of performance.

8. IMPLEMENTATION

After each implementation there should be a *post implementation review*. The purpose of the review is to assess the effectiveness of all procedures and standards in place during the development process. The objective is to ensure that experience gained during the development is documented and any changes to procedures or standards are made to improve procedures for latter developments.

Review scheduling. The should take place no later than one moth after the successful implementation. The review should take no-longer than one hour and should not focus on details, follow up sessions focused on specific areas of details should be arranged if required. The team leader should act as the chair man for the review and should minute the meeting. Those attending the review should include all those involved with the release and developer as necessary.

Two weeks before the review The chair man should invite written submissions from all those involved with the development ho wish to express concerns about any area of the development process. The chairman should collate the submissions lengthy, or irrelevant material should be excluded. These submission should have a bearing on the agenda for the meeting.

Meeting agenda. The agenda should include the following topics :- Brief over view of the product, Major delays in business requirements, Delays, outstanding problems, actual system performance V objectives, User perception of system implementation, Actual resource expenditure V planned, Management and procedural problems,

Review minutes. Minutes should be taken and circulated to all attendees and after agreement, circulate to all those involve with the project, and those involve with areas that might need to be changed.

Lessons learnt. Any procedures which need to be changed should be progressed via the appropriate mechanism. This is usually a weekly project managers/Team leaders meeting.

End of document