**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Ver. No.*** | ***Authors*** | ***Date*** | ***Reviewers*** | ***Review Date*** | ***Release Date*** |
| 1.0 | Application Development Team | 27-Aug-2018 | QMF | 31-Aug-2018 | 03-Sep-2018 |
| 2.0 | Application Development Team | 10-Dec-2019 | QMF | 13-Dec-2019 | 16-Dec-2019 |
| 3.0 | Application Development Team | 2-Nov-2020 | QMF | 6-Nov-2020 | 10-Nov-2020 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Change History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Ver. No.*** | ***Section*** | ***Date*** | ***Change Information*** | ***RFC No.*** |
| 1.0 | All | 03-Sep-2018 | New Release | - |
| 2.0 | All | 16-Dec-2019 | New Release | - |
| 3.0 | All | 10-Nov-2020 | Annual Review | - |
|  |  |  |  |  |
|  |  |  |  |  |

**GUIDELINES FOR LOW LEVEL DESIGN**

1. **Objectives**

The objective of this document is to describe the recommended practices for creating the Low Level Design (LLD).

1. **Scope**

These guidelines are applicable to all software engineering projects.

1. **Document Structure, Glossary of Terms, etc.**
2. Physical Data Model
3. Physical Database Objects
4. Distributed-Processing Concept
5. Security Concept
6. Logging Concept
7. Migration Concept
8. **Physical Data Model**
9. Steps for preparing Data Model
10. Identify entity sets
11. Define relationships between these entity sets
12. Define cardinality (one to one, one to many, many to many) from both sides of a relationship between two entities
13. Define participation (mandatory or optional)
14. Add attributes to the entities
15. Choose identifiers for all entities
16. Perform normalization, and
17. Produce an Entity Life Cycle History diagram
18. In performing the above process, care should be taken to build into the design the performance criteria to be fulfilled
19. The above process can proceed in an evolutionary manner. That is, identify some entity sets, define relationships between them, identify few more, perhaps add few attributes, so on and so forth
20. **Physical Database Objects**
21. Define the files / tables used in the system
22. For each table, define the fields with descriptions where the fields are not self-explanatory, the number of average and maximum records
23. For each field , define the data types, length, precision, the constraints wherever applicable i.e. the range of values a field can take, whether mandatory, the validation rules (procedures) for the fields wherever applicable and client / server implementation
24. For each table, define the indexes and keys associated with the table and the field composition of the index / key
25. Define the relationships between tables identifying the tables and fields involved in the relationship, the type of relationship (one to one, one to many) and the participation (mandatory or optional)
26. Define the referential integrity rules (triggers) for all the tables
27. Ensure that in all the above definitions care is taken to build into the design the performance criteria to be fulfilled
28. Ensure that data structures that are external to the system are also defined
29. Compute the total physical storage requirement of the database objects
30. Ensure that the security concept eg. roles/privileges, if exists, is implemented for the tables/fields where applicable
31. Define query objects for views
32. For each object specify the Object type, Object Owner, and Object Privileges
33. **Distributed-Processing Concept**
34. Describe the distributed-processing concept (data distribution, function distribution)
35. Define specific requirements of the distributed system (e.g., data security, version control)
36. Evaluate the technical feasibility (e.g., performance) of the requirements
37. List the network architectures to be supported (e.g., LAN, WAN)
38. List the protocols to be used (e.g., TCP/IP, LU6.2, DDE)
39. **Security Concept**
40. Name the specific security system to be used
41. Evaluate the feasibility of the specified security requirements with the selected security system
42. Define the default security characteristics (e.g., all allowed, nothing allowed)
43. Define subjects (e.g., user, group, program)
44. Define authorization hierarchies (e.g., user, group, default)
45. List security-critical items (e.g., data fields, functions)
46. Define authorizations (e.g., grantor = owner of an object, grantee = owner of an authorization for an object)
47. Describe the security logging concept
48. Document the design of the primary security modules
49. Document the design of the authorization validation for client and server components
50. **Logging Concept**
51. Describe the mechanism for storing runtime log data for administration and problem resolution
52. What will be logged?
53. Can different logging levels be set to log different levels of detail? If so, what is logged at each level?
54. Where will the logged data be stored (plain ASCII text file, database, etc.)?
55. Document the design of the primary logging modules (e.g., for create log, write log)
56. **Migration Concept**

This section applies only to projects that develop new versions of existing products

1. Describe the functions, modules, and data required for the migration procedure :
2. Describe the prerequisites for migration
3. Describe the data to be transferred
4. Describe the processing steps (data conversion, data compression, logging, etc.)
5. Modularize and describe the migration functions (primary modules, encapsulation)
6. Define migration packets (migration for each data file, for each client)
7. Define the utilities to be used (unload program, load program). If they are new, document their design