Database Model and Design

Introduction

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

In order to use and manage a database, database management systems (DBMS) are employed. Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases.

There are several well-known DBMSs like MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro.

New technology has come up with more recent database types popular known as NoSQL (Not-Only SQL). These are database software that store data as documents and not in relational format as is stored in relational DBMSs like MySQL and Microsoft SQL Server. Examples are MongoDB, Cassandra, DynamoDB and Big Data.

For the majority of Oasis Management Company projects, relational DBMSs will be used. More specifically, the company's main database software will be MySQL, Microsoft SQL Server, PostgreSQL and Oracle.

MySQL

MySQL is the world's second most widely used open-source relational database management system (RDBMS). The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. It is now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL. Other 'AMP' stacks are XAMPP, WAMP and MAMP which stand for Cross-platform, Windows and Mac respectively.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other

software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Its primary query languages are T-SQL and ANSI SQL.

Major releases are SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, SQL Server 2014.

Microsoft makes SQL Server available in multiple editions, with different feature sets and targeting different users. These editions are:

Mainstream editions

Datacenter

SQL Server 2008 R2 Datacenter is the full-featured edition of SQL Server and is designed for datacenters that need the high levels of application support and scalability. It supports 256 logical processors and virtually unlimited memory. Comes with StreamInsight Premium edition. The Datacenter edition has been retired in SQL Server 2012, all its features are available in SQL Server 2012 Enterprise Edition.

Enterprise

SQL Server Enterprise Edition includes both the core database engine and add-on services, with a range of tools for creating and managing a SQL Server cluster. It can manage databases as large as 524 petabytes and address 2 terabytes of memory and supports 8 physical processors. SQL Server 2012 Enterprise Edition supports 160 physical processors.

Standard

SQL Server Standard edition includes the core database engine, along with the stand-alone services. It differs from Enterprise edition in that it supports fewer active instances (number of nodes in a cluster) and does not include some high-availability functions such as hot-add memory (allowing memory to be added while the server is still running), and parallel indexes.

Web

SQL Server Web Edition is a low-TCO option for Web hosting.

Business Intelligence

Introduced in SQL Server 2012 and focusing on Self Service and Corporate Business Intelligence. It includes the Standard Edition capabilities and Business Intelligence tools: PowerPivot, Power View, the BI Semantic Model, Master Data Services, Data Quality Services and xVelocity in-memory analytics.

Workgroup

SQL Server Workgroup Edition includes the core database functionality but does not include the additional services. Note that this edition has been retired in SQL Server 2012.

Express

SQL Server Express Edition is a scaled down, free edition of SQL Server, which includes the core database engine. While there are no limitations on the number of databases or users supported, it is limited to using one processor, 1 GB memory and 10 GB database files (4 GB database files prior to SQL Server Express 2008 R2). It is intended as a replacement for MSDE. Two additional editions provide a superset of features not in the original Express Edition. The first is SQL Server Express with Tools, which includes SQL Server Management Studio Basic. SQL Server Express with Advanced Services adds full-text search capability and reporting services.

Specialized editions

Azure

Microsoft SQL Azure Database is the cloud-based version of Microsoft SQL Server, presented as software as a service on Azure Services Platform.

Compact (SQL CE)

The compact edition is an embedded database engine. Unlike the other editions of SQL Server, the SQL CE engine is based on SQL Mobile (initially designed for use with hand-held devices) and does not share the same binaries. Due to its small size (1 MB DLL footprint), it has a markedly reduced feature set compared to the other editions. For example, it supports a subset of the standard data types, does not support stored procedures or Views or multiple-statement batches (among other limitations). It is limited to 4 GB maximum database size and cannot be run as a Windows service, Compact Edition must be hosted by the application using it. The 3.5 version includes support for ADO.NET Synchronization Services. SQL CE does not support ODBC connectivity, unlike SQL Server proper.

Developer

SQL Server Developer Edition includes the same features as SQL Server 2012 Enterprise Edition, but is limited by the license to be only used as a development and test system, and not as production server.

This edition is available to download by students free of charge as a part of Microsoft's DreamSpark program.

Embedded (SSEE)

SQL Server 2005 Embedded Edition is a specially configured named instance of the SQL Server Express database engine which can be accessed only by certain Windows Services.

Evaluation

SQL Server Evaluation Edition, also known as the Trial Edition, has all the features of the Enterprise Edition, but is limited to 180 days, after which the tools will continue to run, but the server services will stop.

Fast Track

SQL Server Fast Track is specifically for enterprise-scale data warehousing storage and business intelligence processing, and runs on reference-architecture hardware that is optimized for Fast Track.

LocalDB

Introduced in SQL Server Express 2012, LocalDB is a minimal, on-demand, version of SQL Server that is designed for application developers. It can also be used as an embedded database.

Parallel Data Warehouse (PDW)

A massively parallel processing (MPP) SQL Server appliance optimized for large-scale data warehousing such as hundreds of terabytes.

Datawarehouse Appliance Edition

Pre-installed and configured as part of an appliance in partnership with Dell & HP base on the Fast Track architecture. This edition does not include SQL Server Integration Services, Analysis Services, or Reporting Services.

PostgreSQL

PostgreSQL, often simply "Postgres", is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. As a database server, its primary function is to store data, securely and supporting best practices, and retrieve it later, as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Recent versions also provide replication of the database itself for security and scalability.

PostgreSQL implements the majority of the SQL:2011 standard, is ACID-compliant and transactional (including most DDL statements) avoiding locking issues using multiversion concurrency control (MVCC), provides immunity to dirty reads and full serializability; handles complex SQL queries using many indexing methods that are not available in other databases; has updateable views and materialized views, triggers, foreign keys; supports functions and stored procedures, and other expandability, and has a large number of extensions written by third parties. In addition to the possibility of working with the major proprietary and open source databases, PostgreSQL supports migration from them, by its extensive standard SQL support and available migration tools. And if proprietary extensions had been used, by its extensibility that can emulate many through some built-in and third-party open source compatibility extensions, such as for Oracle.

PostgreSQL is cross-platform and runs on many operating systems including Linux, FreeBSD, Solaris, and Microsoft Windows. Mac OS X, starting with OS X 10.7 Lion, has the server as its standard default database in the server edition, and PostgreSQL client tools in the desktop edition. The vast majority of Linux distributions have it available in supplied packages.

Oracle Database

The Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is an object-relational database management system produced and marketed by Oracle Corporation.

The Oracle RDBMS stores data logically in the form of tablespaces and physically in the form of data files ("datafiles"). Tablespaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. Data blocks form the basic units of data storage.

Oracle Database software comes in 63 language-versions (including regional variations such as British English and American English). Variations between versions cover the names of days and months, abbreviations, time-symbols (such as A.M. and A.D.), and sorting

Oracle Corporation has translated Oracle Database error-messages into Arabic, Catalan, Chinese, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Spanish, Swedish, Thai and Turkish.

Oracle Corporation provides database developers with tools and mechanisms for producing internationalized database applications: referred to internally as "Globalization".

Database Design

Database designs vary depending on the project currently being worked on. In other to design a sound and well structure database requirement solicitation must have been conducted extensively. In other to

avoid a project dead end as the project scope increases it is essential that several iterations of requirement gathering and analysis be conducted by the engineering team.

It is also **important** that database design be **normalized**. Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

Database normalization is independent of the project application logic and should **never** be done based the software application logic (especially on the frontend). A standard piece of database design guidance is that the designer should first create a fully normalized design; then selective denormalization can be performed for performance reasons.

Database Modeling

Data modeling in software engineering is the process of creating a data model for an information system by applying formal data modeling techniques. Database modeling involves the creation of database schemas or entity relationship diagrams (ERDs). As engineering develops, more model types can be created.

Database Schema/Entity Relationship Diagrams (ERDs)

A database schema/ERD of a database system is its structure described in a formal language supported by the database management system (DBMS) and refers to the organization of data as a blueprint of how a database is constructed (divided into database tables in case of Relational Databases). The formal definition of database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real world entities are modeled in the database.

"A database schema specifies, based on the database administrator's knowledge of possible applications, the facts that can enter the database, or those of interest to the possible end-users." The notion of a database schema plays the same role as the notion of theory in predicate calculus. A model of this “theory” closely corresponds to a database, which can be seen at any instant of time as a mathematical object. Thus a schema can contain formulas representing integrity constraints specifically for an application and the constraints specifically for a type of database, all expressed in the same database language. In a relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, database links, directories, XML schemas, and other elements.

Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In other words, schema is the structure of the database that defines the objects in the database.

For development of our database schemas, we will employ DBDesigner as the development tool. DbDesigner allows you to construct your DB in an intuitive and easy to use environment, where you have a visual representation of the tables and relations contained in your project. You can quickly see the fields in a table or how each table relates to the others. After you are finished, DbDesigner can export the schema of the database into an .sql script, or directly connect to a database backend and build it there. It can also import already existing databases from .sql scripts or db backends. It can projects into its native format (XML) so all information is kept. Due to its plugin architecture, DbDesigner is easily extensible to work with many database servers. By default it comes with 2 plugins: one for PostgreSQL and the other for MySQL.

Quality Assurance (QA)

Software quality assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. The methods by which this is accomplished are many and varied, and may include ensuring conformance to one or more standards, such as ISO 9000 or a model such as CMMI.

SQA encompasses the entire software development process, which includes processes such as requirements definition, software design, coding, source code control, code reviews, change management, configuration management, testing, release management, and product integration. SQA is organized into goals, commitments, abilities, activities, measurements, and verifications.

Quality assurance consists of key documents on a high level that are prepared during the course of a project cycle to adequately assess the readiness of a product for release. Although the level of detail in each document may vary by team and project, each is mandatory.

Product Requirement Analysis Document

The Product Requirement Analysis Document is the document prepared/reviewed by marketing, sales, and technical product managers. This document defines the requirements for the product, the "What". It is used by the developer to build his/her functional specification and used by QA as a reference for the first draft of the Test Strategy.

Functional Specification

The functional specification is the "How" of the product. The functional specification identifies how new features will be implemented. This document includes items such as what database tables a particular search will query. This document is critical to QA because it is used to build the Test Plan. QA is often involved in reviewing the functional specification for clarity and helping to define the business rules.

Test Strategy

The Test Strategy is the first document QA should prepare for any project. This is a living document that should be maintained/updated throughout the project. The first draft should be completed upon approval of the PRAD and sent to the developer and technical product manager for review.

The Test Strategy is a high-level document that details the approach QA will follow in testing the given product. This document can vary based on the project, but all strategies should include the following criteria:

1. Project Overview - What is the project.
2. Project Scope - What are the core components of the product to be tested
3. Testing - This section defines the test methodology to be used, the types of testing to be executed (GUI, Functional, etc.), how testing will be prioritized, testing that will and will not be done and the associated risks. This section should also outline the system configurations that will be tested and the tester assignments for the project.
4. Completion Criteria - These are the objective criteria upon which the team will decide the product is ready for release
5. Schedule - This should define the schedule for the project and include completion dates for the PRAD, Functional Spec, and Test Strategy etc. The schedule section should include build delivery dates, release dates and the dates for the Readiness Review, QA Process Review, and Release Board Meetings.
6. Materials Consulted - Identify the documents used to prepare the test strategy
7. Test Setup - This section should identify all hardware/software, personnel pre-requisites for testing. This section should also identify any areas that will not be tested (such as 3rd party application compatibility.)

Test Matrix (Test Plan)

The Test Matrix is the Excel template that identifies the test types (GUI, Functional etc.), the test suites within each type, and the test categories to be tested. This matrix also prioritizes test categories and provides reporting on test coverage.

1. Test Summary report
2. Test Suite Risk Coverage report

Upon completion of the functional specification and test strategy, QA begins building the master test matrix. This is a living document and can change over the course of the project as testers create new test categories or remove non-relevant areas. Ideally, a master matrix need only be adjusted to include near feature areas or enhancements from release to release on a given product line.

Test Cases

As testers build the Master Matrix, they also build their individual test cases. These are the specific functions testers must verify within each test category to qualify the feature. A test case is identified by ID number and prioritized. Each test case has the following criteria:

1. Purpose - Reason for the test case
2. Steps - A logical sequence of steps the tester must follow to execute the test case
3. Expected Results - The expected result of the test case
4. Actual Result - What actually happened when the test case was executed
5. Status - Identifies whether the test case was passed, failed, blocked or skipped.
6. Pass - Actual result matched expected result
7. Failed - Bug discovered that represents a failure of the feature
8. Blocked - Tester could not execute the test case because of bug
9. Skipped - Test case was not executed this round
10. Bug ID - If the test case was failed, identify the bug number of the resulting bug.

Test Results by Build

Once QA begins testing, it is incumbent upon them to provide results on a consistent basis to developers and the technical product manager. This is done in two ways: A completed Test Matrix for each build and a Results Summary document.

For each test cycle, testers should fill in a copy of the project's Master Matrix. This will create the associated Test Coverage reports automatically (Test Coverage by Type and Test Coverage by Risk/Priority). This should be posted in a place that necessary individuals can access the information. Since the full Matrix is large and not easily read, it is also recommended that you create a short Results Summary that highlights key information. A Results Summary should include the following:

1. Build Number
2. Database Version Number
3. Install Paths (If applicable)
4. Testers · Scheduled Build Delivery Date
5. Actual Build Delivery Date
6. Test Start Date
7. Scope - What type of testing was planned for this build? For example, was it a partial build? A full-regression build? Scope should identify areas tested and areas not tested.
8. Issues - This section should identify any problems that hampered testing, represent a trend toward a specific problem area, or are causing the project to slip. For example, in this section you would note if the build was delivered late and why and what its impact was on testing.
9. Statistics - In this section, you can note things such as number of bugs found during the cycle, number of bugs closed during the cycle etc.

Release Package

The Release Package is the final document QA prepares. This is the compilation of all previous documents and a release recommendation. Each release package will vary by team and project, but they should all include the following information:

Project Overview - This is a synopsis of the project, its scope, any problems encountered during the testing cycle and QA's recommendation to release or not release. The overview should be a "response" to the test strategy and note areas where the strategy was successful, areas where the strategy had to be revised etc. The project overview is also the place for QA to call out any suggestions for process improvements in the next project cycle. Think of the Test Strategy and the Project Overview as "Project bookends".

Project PRAD - This is the Product Requirements Analysis Document, which defines what functionality was approved for inclusion in the project. If there was no PRAD for the project, it should be clearly noted in the Project Overview. The consequences of an absent PRAD should also be noted.

Functional Specification - The document that defines how functionality will be implemented. If there was no functional specification, it should be clearly noted in the Project Overview. The consequences of an absent Functional Specification should also be noted.

Test Strategy - The document outlining QA's process for testing the application.

Results Summaries - The results summaries identify the results of each round of testing. These should be accompanied in the Release Package by the corresponding reports for Test Coverage by Test Type and Test Coverage by Risk Type/Priority from the corresponding completed Test Matrix for each build. In addition, it is recommended that you include the full Test Matrix results from the test cycle designated as Full Regression.

Known Issues Document - This document is primarily for Technical Support. This document identifies workarounds, issues development is aware of but has chosen not to correct, and potential problem areas for clients.

Installation Instruction - If your product must be installed as the client site, it is recommended to include the Installation Guide and any related documentation as part of the release package.

Open Defects - The list of defects remaining in the defect tracking system with a status of Open. Technical Support has access to the system, so a report noting the defect ID, the problem area, and title should be sufficient.

Deferred Defects - The list of defects remaining in the defect tracking system with a status of deferred. Deferred means the technical product manager has decided not to address the issue with the current release.

Pending Defects - The list of defects remaining in the defect tracking system with a status of pending. Pending refers to any defect waiting on a decision from a technical product manager before a developer addresses the problem.

Fixed Defects - The list of defects waiting for verification by QA.

Closed Defects - The list of defects verified as fixed by QA during the project cycle. The Release Package is compiled in anticipation of the Readiness Review meeting. It is reviewed by the QA Process Manager during the QA Process Review Meeting and is provided to the Release Board and Technical Support.

Readiness Review Meeting

The Readiness Review meeting is a team meeting between the technical product manager, project developers and QA. This is the meeting in which the team assesses the readiness of the product for release.

This meeting should occur prior to the delivery of the Gold Candidate build. The exact timing will vary by team and project, but the discussion must be held far enough in advance of the scheduled release date so that there is sufficient time to warn executive management of a potential delay in the release.

The technical product manager or lead QA may schedule this meeting.

QA Process Review Meeting

The QA Process Review Meeting is meeting between the QA Process Manager and the QA staff on the given project. The intent of this meeting is to review how well or not well process was followed during the project cycle.

This is the opportunity for QA to discuss any problems encountered during the cycle that impacted their ability to test effectively. This is also the opportunity to review the process as whole and discuss areas for improvement.

After this meeting, the QA Process Manager will give a recommendation as to whether enough of the process was followed to ensure a quality product and thus allow a release.

This meeting should take place after the Readiness Review meeting. It should be scheduled by the lead QA on the project.

Release Board Meeting

This meeting is for the technical product manager and senior executives to discuss the status of the product and the teams release recommendations. If the results of the Readiness meeting and QA Process Review meeting are positive, this meeting may be waived.

The technical product manager is responsible for scheduling this meeting.

This meeting is the final check before a product is released. Due to rapid product development cycles, it is rare that QA receives completed PRADs and Functional Specifications before they begin working on the Test Strategy, Test Matrix, and Test Cases. This work is usually done in parallel.

Testers may begin working on the Test Strategy based on partial PRADs or confirmation from the technical product manager as to what is expected to be in the next release. This is usually enough to draft out a high -level strategy outlining immediate resource needs, potential problem areas, and a tentative schedule.

The Test Strategy is then updated once the PRAD is approved, and again when the functional specifications are complete enough to provide management with a committed schedule. All drafts of the test strategy should be provided to the technical product manager and it is QA's responsibility to ensure that information provided in the document (such as potential resource problems) is clearly understood.

If the anticipated release does not represent a new product line, testers can begin the Master Test Matrix and test cases at the same time the project's PRAD is being finalized. Testers can build and/or refine test cases for the new functionality as the functional specification is defined. Testers often contribute to and are expected to be involved in reviewing the functional specification.

The results summary document should be prepared at the end of each test cycle and distributed to developers and the technical product manager. It is designed more to inform interested parties on the status of testing and possible impact to the overall project cycle.

The release package is prepared during the last test cycle for the readiness review meeting.

Maintenance and Support

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that the majority, over 80%, of the maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system. More recent studies put the bug-fixing proportion closer to 21%.

The key software maintenance issues are both managerial and technical. Key management issues are: alignment with customer priorities, staffing, which organization does maintenance, estimating costs. Key technical issues are: limited understanding, impact analysis, testing, maintainability measurement.

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanisms must be developed for evaluation, controlling and making modifications.

So any work done to change the software after it is in operation is considered to be maintenance work. The purpose is to preserve the value of software over the time. The value can be enhanced by expanding the customer base, meeting additional requirements, becoming easier to use, more efficient and employing newer technology. Maintenance may span for 20 years, whereas development may be 1-2 years.

Software Maintenance Planning

An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resource and cost estimates. A new decision should be addressed for the developing of every new system feature and its quality objectives. The software maintenance, which can last for 5–6 years (or even decades) after the development process, calls for an effective plan which can address the scope of software maintenance, the tailoring of the post-delivery/deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs. The selection of proper enforcement of standards is the challenging task right from early stage of software engineering which has not got definite importance by the concerned stakeholders.

Software Maintenance Processes

1. The implementation process contains software preparation and transition activities, such as the conception and creation of the maintenance plan; the preparation for handling problems identified during development; and the follow-up on product configuration management.
2. The problem and modification analysis process, which is executed once the application has become the responsibility of the maintenance group. The maintenance programmer must analyze each request, confirm it (by reproducing the situation) and check its validity, investigate it and propose a solution, document the request and the solution proposal, and finally, obtain all the required authorizations to apply the modifications.
3. The process considering the implementation of the modification itself.
4. The process acceptance of the modification, by confirming the modified work with the individual who submitted the request in order to make sure the modification provided a solution.
5. The migration process (platform migration, for example) is exceptional, and is not part of daily maintenance tasks. If the software must be ported to another platform without any change in functionality, this process will be used and a maintenance project team is likely to be assigned to this task.
6. Finally, the last maintenance process, also an event which does not occur on a daily basis, is the retirement of a piece of software.

There are a number of processes, activities and practices that are unique to maintainers, for example:

1. Transition: a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer;
2. Service Level Agreements (SLAs) and specialized (domain-specific) maintenance contracts negotiated by maintainers;
3. Modification Request and Problem Report Help Desk: a problem-handling process used by maintainers to prioritize, documents and route the requests they receive