A Feature Model of The Oracle 9i Database Server

Jang-Hwan Kwon

A Major Report

in

the Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

April 2003

Abstract

A Feature Model of the Oracle 9i Database Server

Jang-Hwan Kwon

Feature modeling is a standard method in the field of domain engineering to determine the extent of software reuse. Applying this method results in a feature diagram and an accompanying description about the mandatory and variable aspects of the software system in question. It is this classification and relationship of these relevant aspects that ultimately determine the degree of reuse.

A practical exercise in applying feature modeling would be to study a widely used, commercially available software system such as the Oracle 9i database server. The resulting feature model can give the required insight on the flexibility and capability of Oracle 9i, where it comes to evaluating it as a viable product based on client needs.

Modeling the Oracle 9i database server involves starting with the feature model of a general database model and then adding or refining features that are specific to Oracle. To do this, sources of documentation are consulted to ascertain the capabilities of each feature. The wealth of information collected from these sources is then evaluated for proper integration into the feature model.

The feature model presented here is one of many possible models that can exist for Oracle 9i. This model's intent is to show the range of its capabilities but other ones can be formulated based on a specific area of concentration or interpretation.

Acknowledgements

I wish to thank the following people for their supportive guidance, confidence, and wisdom that has helped me realize this goal in my life:

- Dr. Gregory Butler
- Irvin Dudeck
- Patricia Posius
- Larry Tansey
- Christine Mota
- Andrea Segal
- Elke D. Reissing
- Jocelyne Côté

Table of Contents

List of Figures

1.0 Introduction

Feature modeling is a product of Domain Engineering that can be applied to determining the degree of reusability within a given software system. This objective is achieved by extracting relevant features within an object of interest and then identifying the areas of commonality and variability. When this process is applied to commercial software products, it is through these similar and differing properties in the software that influence its overall reusability. The more variability a software product has been deemed to have, the more difficult it will be to reuse portions of it in another related application. Conversely, software identified with a large amount of commonality shows a stability of features, making it easy to implement them in other systems that may benefit from them.

Deriving a feature model starts by gathering as much information about the object of interest. This compilation can be achieved by consulting documentation, research literature, and relevant academic theory. The software of interest here is that of a commercially available database server, specifically the widely used Oracle database, version 9i. This type of software product has become vital in the everyday functioning of today's technological world by serving as a storage structure for the many facets of personal, commercial, and industrial data storage applications. The resulting feature model is expected to be large and extensive, as a database server is a complex software application that should not be seen as just a simple data repository.

In this report, the process of feature modeling will be applied to the Oracle 9i database server software product architecture, resulting in a feature model emphasizing the general overview of its higher-level features. The features shown in the model will be

relevant in the sense that its high-level point of view leads to a description of its overall capability.

The informational resources used to identify the features of the Oracle 9i database server are books on basic database theory, feature modeling theory, and the Oracle 9 architecture. In addition, publications originating from Oracle Corporation such as the Oracle 9i documentation set, monthly magazines, and World Wide Web pages from the Oracle Technology Network also contribute significantly towards formulating the model. A full list of the resources consulted can be found in the References section.

The process of deriving of the feature model starts with an explanation of the required theory: a description of feature modeling, its notation, and its methods. The theory is then applied by creating feature models for some simple examples. Formulation of the Oracle feature model starts by first reviewing a feature model of a generic, theoretical database server to serve as a baseline. Oracle-specific elements can then be added to this model, with some refinements to certain areas where necessary, resulting in an Oracle database feature model. It is possible that some features of the theoretical model will have to be modified or even deleted to properly accommodate and explain the feature characteristics of the Oracle feature model. Also, some features specific to Oracle that are not part of the theoretical database server may be added in as well, showing Oracle's unique capability in handling other areas indirectly related to data storage.

As Oracle 9i is an extensive database server product, it would be impractical and extremely time-consuming to extract every single feature of it; the objective here is to derive enough features that are integral to the operation and architecture of the database. Conversely, feature identification should not stop at identifying those features sufficient

for a basic setup operation. The versatility and advanced capabilities of Oracle 9i is made possible with significant enhancements, which should be captured and described in the feature model as well.

The benefit of the resulting Oracle 9i feature model is to serve as a descriptive and diagrammatic representation of the database server whose relevant functional aspects can be understood without the need to read large amounts of documentation or to become proficient with the product. The model can represent a viability guide to potential research projects that require using a database server and to see whether a mainstream product such as Oracle 9i can adequately address research-based needs.

2.0 Feature Modeling

The objective of feature modeling is to identify both the commonality and variability within a target object, leading to an indication on the overall reusability of the object being considered. A resulting feature model identified with many relevant but variable features will result in a complex implementation in terms of both the initial development process and its subsequent maintenance. This task becomes very intricate as all the cases that cause this large amount of variability have to be identified and addressed once there are changes proposed to the infrastructure; hence, reusability would not be very practical in this case. The degree of potential reuse increases with those models with features subject to lesser degrees of variability. The resulting model can serve to identify the more stable, non-variable areas of the product, representing the ideal locations of reuse.

A feature model is comprised of textual descriptions according to a list of various aspects requiring explanation. The relationship between the described features must also

be denoted through a graphical feature diagram using an established convention for notation. Although reviewing a completed model may appear to be elementary in its feature identification, the method behind collecting and assembling the information is quite extensive. An example is given at the end of this section to demonstrate how elaborate the feature modeling exercise is, even when modeling something as simple as an analog wrist watch.

2.1 Feature Model Components

A complete feature model is comprised of multiple components, namely:

- Description: A brief explanation of each feature.
- Rationale: The reason why each feature was included as part of the model. Any features that are variable should describe the conditions in which they are to be used.
- Stakeholder and client programs: Each feature should reveal any customers (stakeholders) or dependent client processes.
- Exemplar systems: Each feature should describe an actual implementation where it is used, if possible; this can be beneficial if more information is required for a given feature.
- Constraints and default dependency rules: Each variable feature should indicate any constraining conditions with respect to mutual-exclusivity or the presence of any dependent features for the variable feature to be present.
- Availability sites, binding sites, and binding modes: The availability site refers to each variable feature describing where it is available. Similarly, the binding site for each variable feature should describe the conditions (when, where, and by

whom) when it is bound; in other words, the conditions that make a feature variable. The binding mode categorizes the type of binding of a variable feature as static (always the same binding), dynamic (binds the correct way each time the feature is used in cases where multiple binding cases exist), or changeable (the binding can be changed to another binding configuration).

- Open or closed attribute: Each feature should indicate whether it is either open or closed to the addition of any new sub-features.
- Priority: Each feature can be categorized with a priority, serving as an indicator for planning when it comes to development or modification of a feature in an implemented case. This way, resources can be allocated toward those features with higher priority as they indicate the important areas of the overall feature model.
- Feature Diagram: A graphical representation of the features depicting the classification and the relative relationship between features. See section 2.2 for more details on the notation convention.

2.2 Feature Diagram Notation

To fully understand any feature model diagram, the notation used must first be explained. A comprehensive explanation of this notation can be found in reference [2]. This section will give an overview of the notation with enough explanation and examples to understand the variations that can occur in feature diagrams. The notation of feature diagrams can be broken down into features either mandatory or optional, and the relationship between features, either as Or-features or alternative features.

2.2.1 Mandatory and Optional Features

Features can be classified as either mandatory or optional. In general, a feature is denoted by a rectangle containing a descriptive label with a small dot above it. A filled dot denotes the feature as mandatory whereas a dot with a non-shaded outline denotes an optional feature. A simple example can be seen in Figure 2-1, which shows the simple feature model diagram of a men's suit, which shows two mandatory features (jacket, pants) and one optional feature (vest). In other words, the diagram depicts the facts that a suit is always comprised of a jacket and a pair of pants, which can also be complemented by an optional vest.

Because the vest is classified as optional, it is a variable feature and further explanations about the conditions of its variability can benefit the feature model description. In terms of mutually exclusive constraints, there are none as the vest is the only optional feature here. If there were more than one, any mutually exclusive conditions should be mentioned. The availability characteristic of the vest is in the form whether the manufacturer supplies a vest as part of the suit; not all suits are sold with a vest.

The binding conditions for a vest can be interpreted as to the factors that restrict its use in some way. The only use-related reason is the owner's decision not to wear the vest as part of the suit on a given outing. The binding mode classifies the nature of the owner's decision to wear (or not wear) a vest as static, changeable, or dynamic. As each owner is different in his decision to wear a vest, static refers to the same decision every time, changeable eludes to the existence of factors that may contribute to changing the

usual decision, while dynamic refers to a decision based on the state of a set of parameters.

The suit feature can be classified as closed as the suit has comprised of a jacket, pants, and vest for a very long time. Priority-wise, the more important features are the pants and jacket as these are the socially accepted elements that constitute a suit. The vest is less important today as current fashion trends do not stress its importance although it has been observed that trends have a tendency to repeat themselves in cycles. Note that a century ago, the three features of a suit would have been equally important as all three pieces were deemed to mandatory in comprising the definition of a suit. Hence, the sub-feature properties of any feature may change over time according to the present environment.

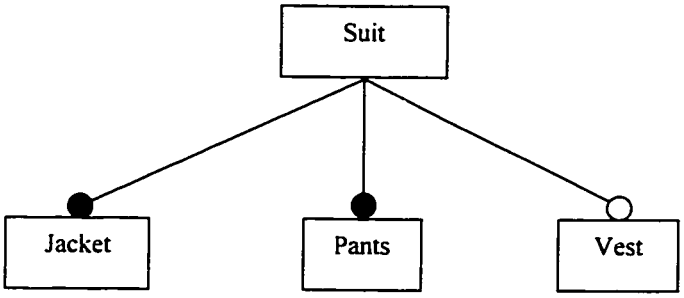

Figure 2-1. Feature diagram of a suit.

2.2.2 Alternative Features

Alternative features are used to denote multiple mandatory or optional features where the presence of only one of them is valid for any given instance. In a feature diagram, this alternative feature concept is depicted using an empty (unfilled) arc, which spans along those multiple sub-features that are affected. Figure 2-2 shows a feature diagram of a triangle, with six mandatory sub-features representing the domain of

possible triangle types (isoceles, right-angle, scalene, equilateral, obtuse, acute). The diagram depicts the fact where the presence of any one of these sub-features satisfies the definition of a triangle in any one instance.

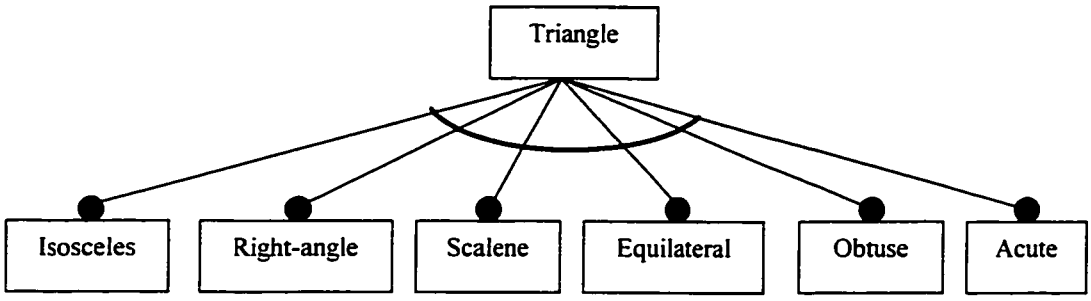

Figure 2-2. Feature diagram of a triangle and its types.

In other words, any triangle can be classified as either an isoceles, right-angle, scalene, equilateral, obtuse, or acute but cannot satisfy two or more of these definitions of triangle types at the same time.

2.2.3 Or-Features

Whereas alternative features are represented by exactly one of the multiple features, Or-features represent some combination of the affected features. Graphically, Or-features are denoted with a filled arc in a feature diagram. A practical example using Or-features is modeling the various configurations of coins accepted by a self-service machine. Common machines of this type include public telephones and vending machines.

The type of coins accepted can vary from machine to machine but considering the domain of all coin types results in having an identifiable feature for each type of Canadian coin, namely the penny, nickel, dime, quarter, half-dollar, dollar, and two-dollar coins. Investigating various machines in the actual coin types accepted results in

classifying those coin types not commonly used as optional rather than mandatory features.

Most machines take in the nickel, dime, quarter, and dollar while only a minority of the machines accepts the penny and the two-dollar coins. It should be noted that although the half-dollar is considered part of the Canadian currency system, the number in circulation are small where its acceptance in vending machine is practically non-existent. Thus, the penny, half-dollar, and two-dollar can be classified as optional features, applying to only those machines programmed to accept them.

These optional features are not constrained in any way and have no default dependencies where it comes to other features that rely on them. The availability of each of the optional coin types is dictated by each self-service machine manufacturer, who programs whether each type can be taken into the machine's till or returned to the customer. An example of binding can be found with the half-dollar coin type, whose coin acceptance is bound by the number of this coin type in circulation. As the circulation level for this 50-cent piece is much lower than for other coins, the bound is static and can only be changed as a result of a policy change from the Canadian mint to increase circulation.

Purchasing a product or service from one of these machines allows one or more types of these coins to be used together whose monetary total constitute or exceed the required purchase amount. Hence, this combination of coin types accepted by self-service machines can be expressed as an Or-feature in a feature diagram as shown in Figure 2-3.

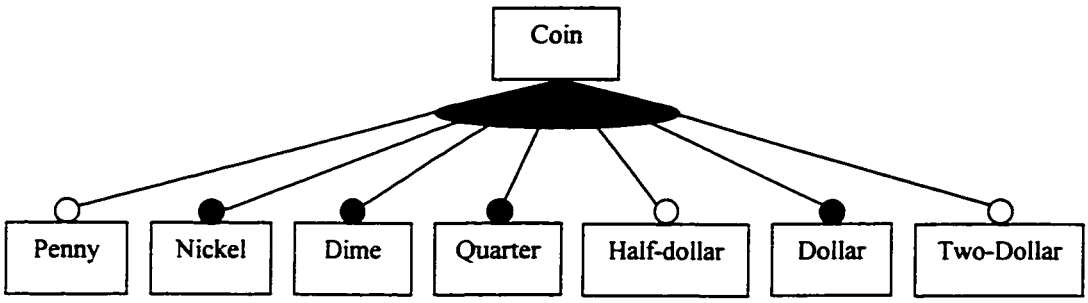

Figure 2-3. Feature diagram of Canadian coins accepted by self-service machines.

2.3 Feature Modeling Methods

Deriving a feature model is an exercise of identifying, collecting, and classifying features. To model an object of interest, its relevant features must first be identified. Feature identification is accomplished by considering the definition of a feature: a property that is reusable, configurable, and relevant. The relevancy of a feature refers to its importance with respect to something, whether it is in the form of a person (such as a user of the feature) or some other object (dependent on the feature).

Once all features have been identified, they can be collected and grouped accordingly, based on the relation of one feature to another. It is possible that a feature may turn out to be a sub-feature or super-feature of another feature. Further elaboration of a feature into sub-features may also take place if the identified sub-features are deemed to be relevant for inclusion into the model.

Classification of a feature is done in two dimensions. The first dimension is to determine whether the feature is mandatory or optional. Mandatory features are seen to represent the infrastructure of the object being modeled. Optional features refer to those properties that are considered secondary and may be considered extraneous in some implementations of the object being modeled.

The second dimension of feature classification is determining commonality or variability. A feature that is common to all known implementations can be considered to be a mandatory feature. A feature seen as variable indicates that multiple ways can exist where configuration of the feature is concerned. Those features that display alternative or or-features in the feature diagram are considered to be variant features.

Upon completion of the feature classification, the features can then be graphically arranged in a feature diagram. Annotations according to the various characteristics described in section 2.1 are also done to describe each feature. The summation of the diagram and annotations constitute the completed feature model.

Guidance in the entire process of modeling is done through researching various sources that are knowledgeable with the object being modeled. These sources can be found in the form of documentation (both user and technical), consultation with experts, and representative implementations. This source of knowledge is most helpful during the stage when establishing a feature's commonality or variability, ultimately leading to an accurate model in terms of its feature classification.

Feature modeling is seen more as an engineering activity than an exact science. The modeling process can be iterative, and more relevant features can be identified and added to an established model if appropriate. Modeling is also subjective, as two models that describe the same object may not result in identical feature models. This may occur as the focus of the modeling may differ and may lie anywhere between a generalized and a very specific point of view.

2.4 An Example Feature Model

Feature diagrams need not be constrained to the simple examples shown above. Depending on the case, a feature diagram can contain all notions such as alternative, mandatory, optional, and Or-features. Also, the features need not go down to only one level; each feature can be further explained through relevant sub-features and so on.

A basic example of incorporating all the aspects of the feature diagram notation can be achieved by deriving a feature model for an analog wrist watch. To determine its features, it is necessary to acknowledge the many different models of analog watches that are currently available. Identifying and classifying the features for an analog watch can be done by considering the vast variety of models through visiting retailers, reading manufacturer's specifications, obtaining numerous samples, as well as polling watch owners. The resulting feature diagram showing the features of an analog watch and their relationships is shown in Figure 2.4. The diagram is a result of the feature identification process and their explanations that follow.

After examining a sample of various analog watch types, the following common characteristics exist:

- All of them have a strap.
- All of them have at least an hour and minute hand to indicate the time of day.
- All of them are powered in some way.

Hence, the above three aspects can be modeled into mandatory features called Strap, Time Display, and Power respectively.

The Strap feature indicates that all watches have a strap that wraps around the wrist. For simplicity sake, the variety of strap types is not modeled here but can be shown

as alternative sub-features as only one type of strap is used on any given watch. Further categorization can be made by indicating specific strap types offered by specific manufacturers with their watches. This feature can then inform watch buyers of the choice of material available for straps. It can also allow manufacturers to consider other materials to be used for straps in the future.

The Time Display feature has been named this way as to model both the mandatory and optional sub-features in this category. The mandatory sub-features here are the hour hand and the minute hand, which constitute the minimum required information to indicate the time of day. The additional sub-feature available for time display is the second hand indicating the current number of seconds elapsed; this inclusion of this optional hand on a given analog watch is decided by the manufacturer. As the second is the smallest of the significant measures of time in terms of telling time, there is no need to expect additional sub-features here.

A binding condition exists for the second hand feature, in terms of its aesthetic quality. By convention, the hour hand is short and wide, with the minute hand being linger and slender. The second hand is usually the same length as the minute hand but even more slender in width. Hence, as this are the accepted designs for all of the watch hands, the second hand binding is static as it will not change under any circumstance.

The Power feature can be seen as mandatory for a watch as it is what drives the watch to keep going and constantly updating itself to indicate the correct time at any time of the day. The sub-features that exist here are in an alternative relation as a watch can be powered either by a battery or through winding a bezel. This feature can be considered

open as other power methods exist and may be applied to watches in the future by ambitious manufacturers.

The remaining features of an analog watch that were identified are present in some but not all watch models. The variability of these optional features encountered through surveying analog watches are:

- Some watches can be backlit for easier nighttime viewing.
- Some watches indicate the numerical day of the month, possibly being accompanied by a day of the week indicator as well.
- Those watches that have the day of the week show the day in either English or French.

Thus, the first two optional features mentioned above can be categorized into the Backlight and Date Display features respectively. The Language feature for the day of week is rather a sub-feature under the Date Display feature as that is the only characteristic of the watch where language applies.

The Backlight feature is included in the feature model since as more watches are becoming battery-powered, they may also be accompanied with a feature where the face of the background can be lighted on demand. Therefore, this feature is only available and reliant on the watch's power feature being battery-powered; there are no bezel-wind watches that can generate the power required by a light. Availability of a backlight is left up to each manufacturer who decides whether to include one in the design of each new model they develop and mass produce.

The Date Display feature has the day of the month sub-feature classified as mandatory while the day of the week sub-feature is optional. The day of the month refers

to showing a digit between 1 and 31 corresponding to the present day of the month. The day of the week sub-feature is a three-letter label corresponding to the present day of the week. In the least, those watches that can display the date do so by displaying the day of the month. It is possible to have some watches show the day of the week alongside the day of the month.

The Date Display feature can be considered closed for additional sub-features; it is always possible that another portion of the date, such as the year, could be displayed on an analog watch but that is highly unlikely due to the limited space on the face of these watches. To account for the variability with the Date Display, these two sub-features are in an Or relation to show that the day of month indicator must be present in those watches that show the day of week as well. The day of the week feature is available for those models deemed adequate by each watch manufacturer. It is bound by the presence of the day of the month indicator, as no watches have only the day of week indicator. As this is the sole rule, the binding mode is considered static.

The day of the week feature can be further extended into a language sub-feature to show the various languages available to describe the day. This feature is included to allow consumers to know that there is a choice of languages available for the day of the week. The languages described for this feature are seen as mandatory sub-features called English, French, and Other; these are in an alternative relation to depict that only one language can be used at a time to describe the day of the week on any given watch. The Other feature is used to represent the remaining languages that are available for analog watches around the world, while the English and French languages are classified as specific features as these are the two most common languages depicted for the day of

week on watches sold locally. As the English, French, and Other sub-features represent the domain of all possible languages, the day of week language feature can be considered closed.

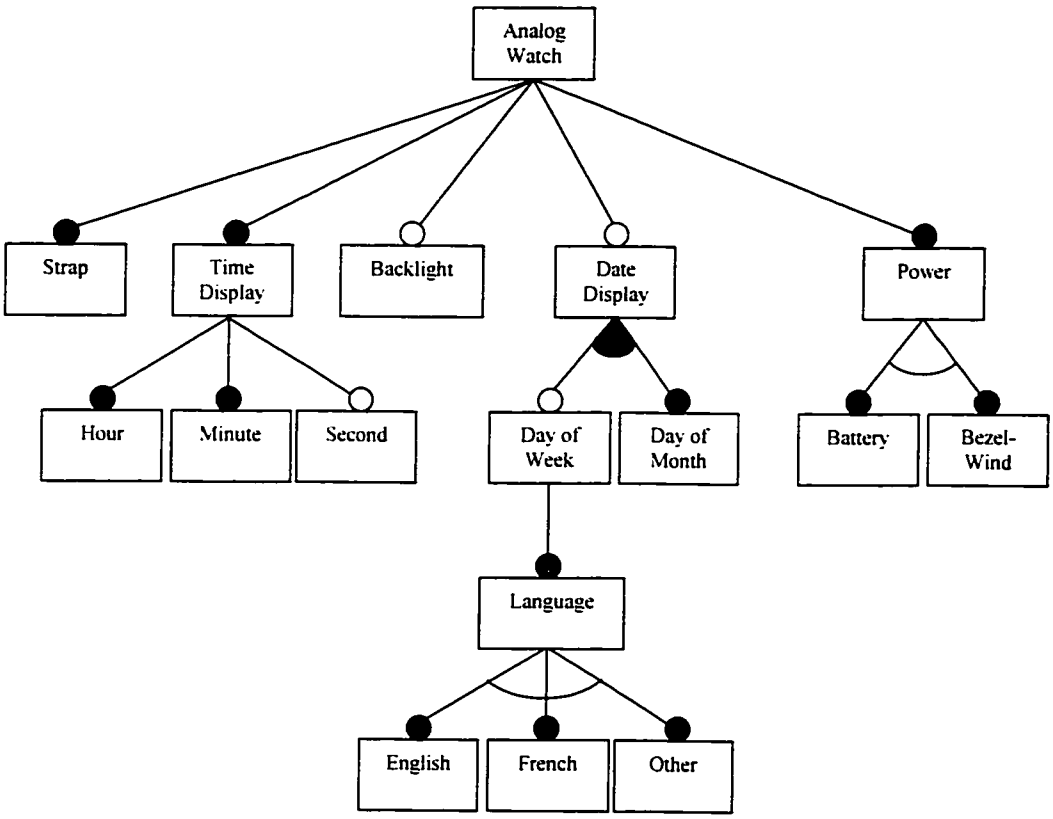

Figure 2-4. Feature diagram of an analog wrist watch.

3.0 Feature Model of the General Database Model

The first step toward formulating an Oracle feature model is to derive one for the general database model. This stage defines the starting point on those features deemed relevant for inclusion for a generic database, where Oracle-specific features can then be extended onto it. Figure 3-1 depicts the completed feature diagram for a general database model [13].

The two top-level features of a database, the data model and its storage, come from consulting references [3] and [12], where they are introduced as the initial concepts when describing a database. Both features are mandatory and considered very high priority as they are required to be present in all database implementations, while their sub-features achieve the functional requirements of a general database. The data model sub-feature describes the theoretical aspects of database organization and function while the storage sub-feature covers physical storage of the data as well as the storage of the component responsible for efficient data retrieval.

3.1 The Data Model Feature

The Data Model feature refers to the conceptual organization of components that fulfills the functional objectives of a database. It is regarded as a feature as it encompasses the required theories necessary behind operational and logical organization aspects of a database. Dependent on this feature are those components of a database responsible for the theoretical operation of processing the desired input criteria and outputting the stored data that meets it. At this level, this Data Model feature is closed from any new features as it fully addresses the modeling elements on a database.

This feature's clients are in the form of the database type, language, and theory, serving as the sub-features of the Data Model feature. The Data Model Type describes the various models developed to represent a database while the language is the two-way communication used to interact with the database. The Data Model Language refers to the theoretical concepts applied to the data representation. These two sub-features are mandatory as they are necessary components used by all existing database implementations. The Data Model Theory represents the various ways to express the desired data through standard logical methods.

3.1.1 The Data Model Type Feature

The data model type feature covers the various representations available to describe the logical organization of a database. This organization is recognized as a feature as the various types available can affect the degree of suitability for the type of data that is desired to be stored. Thus, the domain of all possible database types serve as the sub-features in this section, referring to industry-accepted standards used in at least one commercial database implementation. Depicted in the feature diagram of Figure 3-1 are the sub-features of Relational and Other in an alternative relationship as a database can be modeled only after one of the possible types.

Products such as Oracle 7 and Microsoft Access are examples of Relational database types, where the data is organized through a set of meaningful relations. Database software from Versant and Gemstone are considered as object-oriented databases, where all the database elements are categorized into a set of objects, each with a set of defined corresponding operations or methods. Oracle 9i can allow databases to be set up as a hybrid of these two types and is classified as an object-relational database,

where it makes use of a combination of selected elements from both the relational and object-oriented database types. The remaining model types classified under the Other feature include object-oriented databases as well as the older hierarchical and flat file database types.

As this feature can have many operational implications, it is of a very high priority since when other model types are developed, the resulting database and its operation will have a profound impact on planned applications that use it. As the set of sub-features account for all possible model types, the data model type is a closed feature but it is always possible for new types to be developed in the future. Should this happen, these new model types would be added to the Other sub-feature.

3.1.2 The Data Model Languages Feature

The data model languages feature refers to the communication methods used to interact between the database and the users of the data, in the objective of returning the relevant data that meets the criteria submitted by the user. This aspect is included as a feature since it refers to the communication interaction between the client and the data stored on the database. All database components that exchange information with the data store as well as all applications running off the database are considered dependents to this feature as communication is essential to normal operation.

As an example, users can either use this feature directly or indirectly depending on their role. Direct interaction is done through expert users who have the knowledge of the language, the data, and its organizational structure; these individuals can then formulate the proper syntax to achieve the desired action against the data source. Indirect interaction by other users is in the form of repeatedly executing the same scripts: written

by the expert users and containing the sequence of one or more required steps in delivering the desired data result.

To come up with the sub-features associated with this data model language feature, the languages used with various types of database systems were studied. Some examples languages are Datalog, QBE, Quel, and SQL. SQL stands for Structured Query Language, a very common implementation example of the data model language, containing a standard set of syntax and semantics that can be formulated into a set of commands to manage the database, in terms of querying the desired data, as well as maintaining the database structure.

Formulation of the data model language sub-features proceeded by establishing common categories of operations that the various languages are used for. They can be divided into three mandatory sub-features: schema definition, data querying, and data query result formats.

Schema definition is a necessary sub-feature as a database to hold data cannot exist without some way to define the data store's logical structure. An example language is SQL, with the ability of its syntax to create a database schema by defining the tables, fields, indices, and other necessary parameters. This sub-feature dictates the overall logical structure of the database, where all other logical portions of the database start and depend on this foundation. Thus, this is a very high priority sub-feature.

Data querying is a needed sub-feature as creation of the data structure and storage of data serves only as the infrastructure. There would be no way of knowing what kind of data resided within the database without having a mechanism to query the data. Depending on the type of database, some commonly used query languages include SQL,

Datalog, and Graphlog. These languages use parameters that are formulated in such a way as to find any data within the database that satisfies a particular condition. The dependents of this feature are more client-based as this is the sole component of the database to make their inquiries about the data that is stored. Thus, the query sub-feature is also of a very high priority.

The data returned from a query must be considered as a sub-feature as that is a primary purpose that a database is used for. A common form of data query results is in the form of tuples that satisfy the parameters of the query. However, depending on the database type, other formats are available as well such as tables and forms. As the answer represents some portion of each data-related query, this is a very high priority sub-feature.

An optional sub-feature here is the concept of the transaction. This sub-feature is considered optional as not all languages implement this concept. SQL is an example of a transaction-based language, where submitting a script written in this language takes the form of a transaction in its subsequent execution. This optional feature is present if the database has been designed to base its operations as transactions. It is a very high priority feature for these types of database systems. Bounds imposed by this feature are in the form of the actual implementation versus the theoretical concept of the transaction. Specifically, a transaction is a well-defined group of operations that satisfies the objectives of atomicity, consistency, isolation, and durability.

Atomicity is an all-or-nothing objective, indicating that transactions either fully execute or they do not execute at all. Consistency is an objective stating that a given transaction starts and ends with the database in the same consistent state. Isolation refers

to the concept that concurrent transactions are seen as isolated ones and are not affected by one another during their execution. Durability is a guarantee-based objective that data changes after a transaction are permanent regardless of any disruptive incidents that may threaten the data change operation.

3.1.3 The Data Model Theory Feature

The data model theory feature refers to a formal convention of syntax and semantics to represent querying of the data in the database model. This is recognized as a feature as proper formulation of this theory gives a theoretical representation of a query whose result should satisfy the data being requested. Thus, all theories that allow queries to be expressed in a logical form are dependents of this Data Model Theory feature. As databases use this theory indirectly through logical queries translated into their own querying language, the priority of this feature is low.

As this feature is optional, there are no mutually exclusive conditions to report and no default dependency features. The feature can arise when for those databases that allow for a formal logical statement to represent the query of the data desired. The feature is bound statically as the only limitations are those presented by the logical language used to define the query.

Further down, this theory can be further broken down into two sub-features: the algebra and calculus. These are features as they are accepted philosophies of formulating logical queries. There is a similarity in both features in that they both have a set of operators, each with unique definitions, which can be used together to represent the criteria of the data being queried. The main difference is that the algebra contains more operators compared to the calculus, resulting in a simpler representation of a theoretical

query. It is for this reason why the calculus sub-feature is considered optional, as the domain of possible queries can be formulated with the algebra sub-feature alone. The calculus sub-feature becomes available should users decide to formulate their query using the calculus notation set. Overall, both the algebra and calculus sub-features share the same moderate priority due to its role as a formal theoretical language.

3.2 The Storage Feature

The storage feature refers to the implementation aspects required to hold and to support the actual stored data. It is a very high priority, mandatory feature since it is responsible for all storage aspects, from the data to other necessary components that make up the database. The main dependent of this feature is the essence of the database itself; without any storage, there would be no operational database. Storage is well established into the aspects of the how to store and access it; hence, the feature is closed as all storage matters have been addressed here.

This feature can be further identified into two sub-features: the storage medium and indexing. The storage medium feature is a mandatory one as it is used by all database systems, referring to the physical implementation aspects required to house the data. Although large database systems use indexing to achieve optimal data retrieval performance requirements, the indexing sub-feature is still classified as an optional feature as not all databases may choose to use indices in referencing the data.

3.2.1 The Storage Medium Feature

The storage medium feature refers to the various components and methods used to support the physical aspects of handling the data of the database. The justification for this

aspect as a feature is that this is the area designated for database data to be stored. No other feature handles this component of necessary database operation, making this a very high priority feature with the rest of the database's features reliant on the storage medium to retrieve and store data.

The sub-features identified at this level are the memory, file, and distributed organization concepts. The grouping of these sub-features is in the form of alternative features, as a database may use one of these choices in a particular implementation. These sub-features are all mandatory as they are all viable choices to host data.

The obvious sub-features at this level are the memory and file aspects. Data is represented as a series of files on a physical storage device; once the data is requested, it can be processed through the computer memory hosting the database server. This is the way many commercial databases operate when processing the data stored within a database. As this process is fundamental in basic database operation, it is the reason why file and memory are identified as features as well as the reason why they are seen to be of very high priority.

The other sub-feature associated with the storage medium is distributed organization, where the data is separated into small units and spread out among distributed areas. This is considered to be a feature as there are advantages to serving the data in this fashion, such as distributing those portions of the data store according to the geographical location of users who primarily work with only one distinct part of the database. An example would be a large organization using the same database with satellite locations with each location handling one particular aspect of the organization's

activities, such as manufacturing, sales, and accounting. As a significant number of large organizations use this kind of data distribution, the priority of this sub-feature is high.

These three sub-features are most beneficial when determining the ways to store the data in order to come up with a configuration that meets specified performance requirements. Future additional sub-features are possible for this open Storage Medium feature as new storage methods are researched and accepted as viable alternatives.

3.2.2 The Indexing Feature

Indexing is the concept of marking off the data stores of the database in order to improve processing times when locating the target data of a submitted query. It is a feature as the improvements gained by using indexing are substantial, especially for large quantities of data. Indexing comes with a cost as a secondary structure is required to keep track of the rapid access points of the data source. Hence, indexing would not be warranted for small amounts of data, making it an optional feature as it use caters to specific situations. However, for the significant number of large databases in existence, it is more of an essential feature as indexing is relied upon to meet certain minimum performance requirements, making it a high priority optional feature.

As it is optional, indexing does not have any mutually exclusive constraints and does not serve to affect any other features in this model. The availability of indexing is made possible through issuing commands by technical administrators enabling this feature on various portions of the database that require it. Thus, the feature is bound by the configuration established by these administrators, yet it is changeable as the overall indexing settings may require continual updating as the data in the database grows.

Indexing can be further elaborated into the sub-features of the data structure and the dimensionality. The data structure sub-feature describes the various implementations of known data structures that can support the indexing information. The dimensionality aspect refers to the types of indices that can exist in terms of marking points in data represented along one or more dimensions.

3.2.2.1 Data Structures used for Indexing

Storage of the information representing the indices used in a database is accomplished through commonly used data structure implementations. The reasoning for this feature is that these structures form the basis of the indices defined in a database, serving as the data source for the indexing information. Dependent on this is the querying mechanism that must interact with this feature in order to get the required information when trying to access the required data quickly from the database. As the structures serve as the shortcut pathways to the database data in use by many indexed databases, the feature can be considered to be high in priority.

The choices available for indexing can be categorized into the B-Tree, Hashing, and Other measures, all of which can be considered mandatory sub-features. The B-Tree sub-feature refers to locating index markers in the form of traversing a balanced tree structure. The hashing sub-feature uses a mathematical hash function to locate the index values. The remaining types of data structures that can be used for indexing fall under the remaining sub-feature called 'Other'. This sub-feature is the only one of the indexing data structure feature that can be considered open to additional data structures as the other sub-features describe all other specific data structures suitable for implementing indices.

As the sub-features cover all possible data structure types, this feature is closed with any new data structures classified as part of the Other sub-feature or perhaps being identified as a sub-feature of its own if it is significant to be noted independently. As usually only one type of structure is used to accommodate all the indices of a database, the sub-features are presented in an alternative relationship.

3.2.2.2 Dimensionality Available with Indexing

The dimensionality of indices is considered a feature as it reveals the capability of whether data can be marked across more than one dimension. Conventionally, data stored in a database is linear, having only one dimension to it. However, there can exist data within a database with more than one dimension such as geographical or biological data; this multi-dimensional data is better known as spatial data, requiring specialized indices to handle spatial queries. The dimensionality feature can be beneficial in those cases where the stored data is more elaborate, but these cases make up a specific group of instances, labeling the feature priority as moderate.

Categorization of sub-features of this dimensionality aspect is in the form of single and multiple dimensions. These two sub-features allow this dimensionality feature to be considered closed as it encompasses the domain of all possible dimensions. The Single dimension sub-feature is identified on its own since a majority of database implementations use one-dimensional data. It is this large proportion of databases using one-dimensional indices that make the sub-feature a very high priority one. Feature classification for those indices handling spatial data is in the form of a Multiple dimension sub-feature. There is no need to explicitly classify the dimensions as two-

dimension, three-dimensions, etc. as the indexing strategy used is consistent, regardless of the degree of the dimensions.

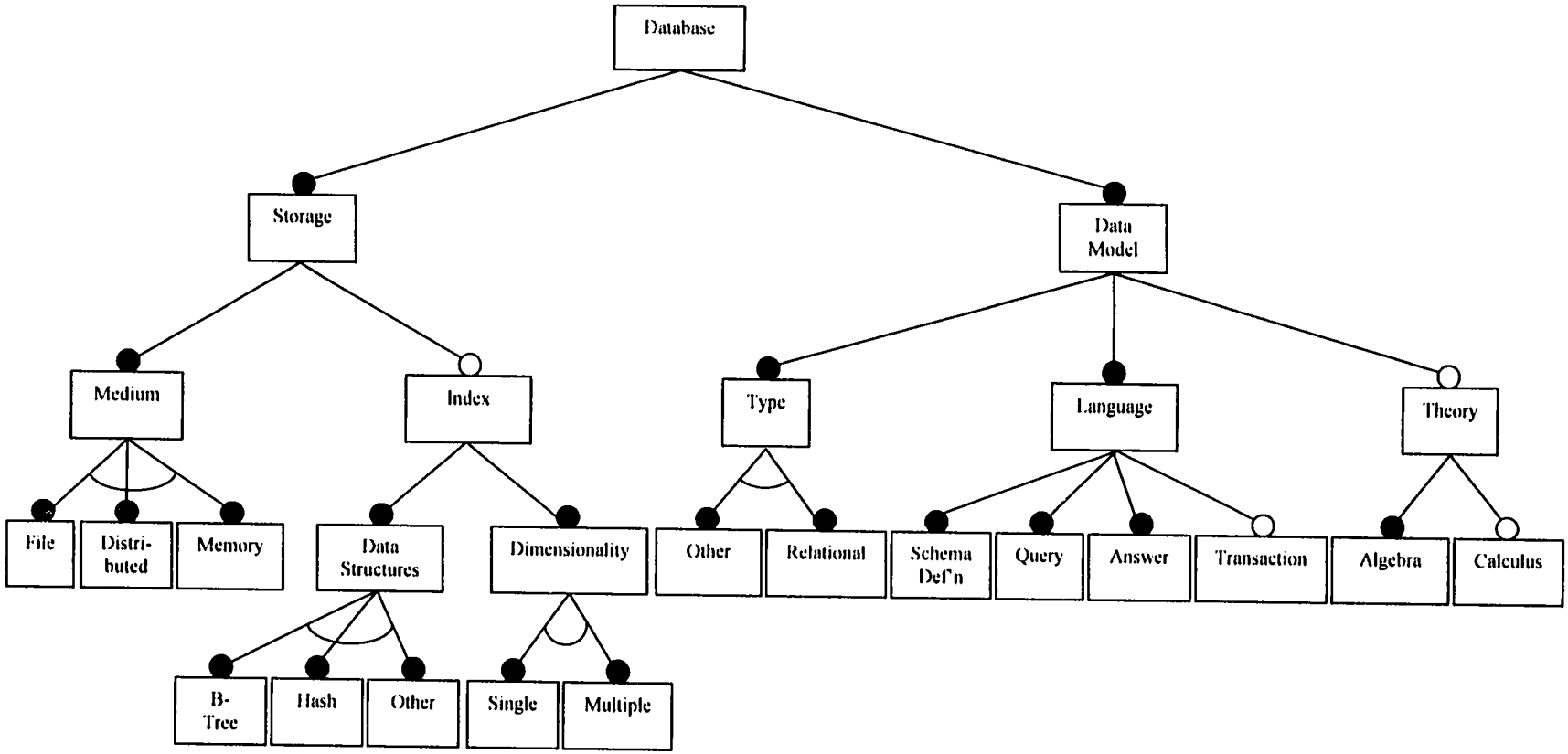

Figure 3-1: Feature diagram for a general database model.

4.0 Formulating the Oracle 9i Feature Model

The general database feature model presented in the previous section serves as the baseline toward constructing the feature model of the Oracle 9i database server. As this is an extensive product where the database is merely just one component of the entire software system, formulation of the entire model is done in the following two sections, 4.1 and 4.2.

The first section describes the feature model of the database elements seen in the previous section and how the Oracle-specific counterparts fit into this structure. The second section gives a brief high-level description of the associated functionalities that interact with but are not part of the database structure, comprising the rest of the Oracle 9i database server.

4.1 The Oracle 9 Database

Derivation of the Oracle database feature model is the result of determining which Oracle-specific aspects differ with the general database feature model. This section is devoted to explaining these differences. The differences are in the form of more specific but less sub-features with some reclassification, with the elaboration of the remaining features as they relate to the particular implementation of an Oracle database feature model. However, at this level, the Oracle 9 database can still be considered to comprise of the data model and storage sub-features.

4.1.1 The Oracle 9 Data Model

Oracle is very particular in its implementation of the data model characteristic of a database, with emphasis on continued support of longtime features (such as SQL, the

relational database model, and relational algebra) as well as the introduction of a new feature (object-relational technology) in this area. Some of the features found in the general database model are not implemented in Oracle, while other features change their classification from an optional to a mandatory feature type. The following three subsections explain just how Oracle implements the data model aspects of a database in terms of type, language, and theory of a database.

4.1.1.1 Oracle Data Model Types

Until the release of version 8, all Oracle databases were based on only one data model type: the relational data model. From Oracle 8 onwards, a choice of two data models exists when creating Oracle databases: the relational and object-relational model. This object-relational technology allows the use of constructs found in the object-oriented paradigm while allowing the use of relational rules to manipulate the data stored in the objects. Note that it is not currently possible to have an Oracle database that is fully and solely object-oriented: there must exist at least some relational interaction.

To model these type possibilities, the only two sub-features identified at this Oracle Data Model level are defined as Relational and Object-Relational. The Relational sub-feature follows from the description explained in the general database model and is still a mandatory feature as all databases, relational or object-relational, require the relational type in its operation. The Object-Relational sub-feature refers to the use of only object type constructs to model a database. As not all databases define objects in their schema design, this feature is considered optional. Hence, the Oracle Data Type model consists of component Relational and Object-Relational sub-features.

An example of creating an Object-Relational data source is where working with objects is more natural than defining a set of tables such as e-commerce applications. The objects can have its own attributes and methods while taking advantage of Oracle's established and reliable relational functionality, such as relational queries. Dependencies of this Object-Relational feature are the portions of the client applications that interact with the various objects. This feature is open to further refinements, especially if there is an evolution toward support for a fully object-oriented design with Oracle.

The optional categorization of the Object-Relational sub-feature introduces other aspects. There are no constraints, no default dependencies, and no binding conditions imposed found with instances that enable this optional feature. The feature becomes available during the definition step of a new database through the declaration of object constructs.

As database models continue to develop and considering the addition of the Object model in a recent additional feature as an example, the Oracle Data Model type feature is open to the addition of further data models. It may eventually introduce a fully object-oriented database without any relational dependencies.

4.1.1.2 Oracle Data Model Languages

Whereas the general database model accounted for the various different types available for the sub-features (schema definition, query, and transaction), the Oracle database model implements these sub-features in the form of a mandatory sub-feature entitled Oracle SQL, whose capabilities encompass schema definitions, querying, and transactions. This sub-feature is both mandatory and of very high priority as SQL represents the essence of all database creation and data manipulation with an Oracle data

source. Use of this sub-feature is necessary in all instances when establishing new or maintaining existing databases.

Within this sub-feature, it should be noted that that transactions in Oracle are mandatory, compared to the optional classification it got in the general database model. This change in the feature classification is due to Oracle designing all of its SQL commands to be treated as transactions; hence, Oracle cannot have a SQL command that does not go through its transaction processing procedure.

The remaining sub-feature for the data model language feature, Answer, does not change too much with the particularities of Oracle. Oracle databases can return results in the form of tuples satisfying the input query criteria; the results of a query can be also be established into a new database table for more elaborate processing. However, an Oracle database cannot support query results in any of the other remaining answer formats mentioned in the general database model, such as the form sub-feature. Hence, the data model languages sub-features supported by Oracle are tuple and table but not form.

Overall, the Oracle Data Model Language feature is of very high priority as it represents the source of all data interaction and database structure implications. It is closed, just as the general database model counterpart, as the sub-features fully describe the necessary interactions of this feature.

4.1.1.3 Oracle Data Model Theory

The data model theory aspect in Oracle is a more specialized version of the one presented in the general database model. The sub-features can be more specifically renamed to relational algebra and relational calculus as they cater to only the relational data model type offered by Oracle. They are also both mandatory sub-features as these

two aspects are prevalent by way of its implementation and execution of Oracle SQL. This is because Oracle SQL is considered to be relationally complete, where a query can be interpreted regardless of whether the theoretical query is formulated in relational algebra or relational calculus.

The Oracle Data Model Theory feature is currently closed, with the relational sub-features representing its relational infrastructure. However, as the object-oriented side of Oracle evolves, the theory may also grow with it, classifying it as an open feature in the future. The feature's priority is moderate as it serves as the essence of modeling SQL queries but the SQL constructs have been designed to go along the rules presented by relational algebra and calculus.

4.1.2 Oracle 9 Data Storage

Data storage in Oracle is equivalent to that of the general database model: it consists of a medium to store the data and indexing for efficient access to the data. The priority is very high as the Oracle Data Storage feature is integral with hosting the data, containing all the necessary elements required to keep data away in a safe place, ready for its perusal on demand. Also contributing to the priority level is the feature's representation of the various ways to efficiently access vast amounts of data contained in the database.

Just as was seen with the general database model, the Oracle Data Storage feature is closed as the description of storage is sufficient with the medium and index sub-features. Specifically, Oracle Data Storage contains a mandatory medium sub-feature, as well as an optional indexing sub-feature. The following two sections explain each of these aspects in further detail.

4.1.2.1 Oracle Data Storage Medium

Storage of data in any Oracle database is primarily through the use of files. Thus, storage media features identified in the general database model but not used by Oracle is the memory, flat file, and distributed features. There is more than one type of file to accommodate the multiple purposes required for database function. In terms of the location of the files, they can be all within a single area or part of a distributed database.

From an Oracle standpoint, the Data Storage Medium feature comprises of only the mandatory File sub-feature as Oracle has long been designed to operate on a series of files. Hence, the priority is very high as it contains the sole building block required to constitute any portion of the database. With no new advents on storing any part of this set of files any differently appear forthcoming, the feature can also be considered closed in terms of the ways to store data.

Note that distributed databases can exist in Oracle but in the form of distributed files of the data store. The term distributed, as described as a storage medium in the general database model of this feature, is different as it preludes to the capability that the components of the database can exist in a distributed fashion. This is not possible with Oracle as each site of a distributed setup is a fully functional database hosting a portion of the total data representing the logical data store.

4.1.2.2 Oracle Data Storage Index

As with the general data model, Oracle's indexing feature can be separated into the mandatory sub-features of the index data structure and dimensionality aspects of its implementation details. The result is a selection of features that cater toward common commercial database implementation types, implying that using Oracle for more

research-based databases with complex querying may not be adequate in terms of implementation or performance.

The Oracle Data Storage Index feature remains optional just like its general database counterpart but there are no identifiable constraints. As a default dependency, these are in the form of applications or query scripts that rely on indices being established for reasonable performance of the data referred to in the database in order to carry out its programmed task. The availability of the index is made possible through its initial declaration to order a set of one or more attributes. The index can be bound through its parameter settings; it need not always be in some form of an increasing order. The binding of a particular index is changeable as the configurable settings of any index can be changed at any time in the hopes of improved performance.

4.1.2.2.1 Oracle Data Structures for Indices

Indices in Oracle are implemented in four of the ways mentioned in section 3.2.2.1. Specifically, an Oracle index can be one of the following types [4]: B*-tree, reverse key, bitmap, and function-based. Like in most other commercially available database products, Oracle recognizes the B*-tree type as the default index type. This would be the index structure of choice when no distinct patterns about the attribute being indexed appear prevalent.

The reverse key type is offered as an alternative to the B*-tree in cases where the B*-tree becomes skewed in its structure due to the index values; the reverse key would reverse the order of the index values, trying to normalize all the values for a more balanced tree structure. Using the reverse key index is most beneficial when establishing a table whose index attribute values are increasing with the addition of each new record.

Bitmap indices use a bitmap grid to locate values within a data table and can be a more efficient index type in cases where the indexed quantities repeat often, resulting in a very compact index that can be searched more quickly than a B*-tree index. Hence, the indexed attribute should also not vary much due to repetitive values, resulting in a small domain of distinct values.

The function-based index type in Oracle 9i allows for quick queries retrieval of data where the selection criteria are formulated as a mathematical function. By setting up the index based on the same function, the particular evaluation of the function also points to the location of the desired query result. This type of structure is most desirable when the function is known beforehand with assurances that the function will not change in the future.

These four index types can be seen as sub-features in an alternative relation as any given index can be implemented using only one of these types at a time. The sub-features themselves are classified as mandatory as they always exist as choices when declaring an index. The priority of these four index structures is almost equal as a moderate priority with a more high priority weighing given to the B*-tree structure as that is the most common structure used for indices.

The Index Data Structure feature is closed as the indexing choices are limited to these four index types. Other index types may ultimately be included but its effectiveness should match those of the sub-features currently being offered. The priority of this feature is high as these data structure represent the implementation and ultimate existence of all defined indices of an Oracle database.

4.1.2.2.2 Oracle Index Dimensionality

Defining indices in Oracle can only be done along a single dimension where points of data contained with a database table can only be marked off along a linear path. Thus, the sub-features under the Index Dimensionality sub-feature reduces to only the one called Single. Since the single dimensionality has long been established and as there are no plans to accommodate indices with multiple dimensions, the feature is both closed and of low priority.

Note that although multiple one-dimensional indices can be defined for a given data table, it should not be confused with Oracle supporting the concept of multiple-dimensional indices, where a single index can mark the data along more than one aspect at the same time. Hence, the absence of this feature results in the loss of the advantages of querying complex data efficiently along multiple, simultaneous pathways. As an alternative, [4] suggests using bitmap indices for performance where there are multi-dimensional queries on extensive data sources such as data warehouses.

4.2 Associated functionalities of Oracle 9i

The Oracle 9i database server is much more than an elaborate data repository structure. Within it is a vast collection of supporting components that define its advanced architecture in delivering a complete database software system solution. Identifying these various functionalities offered by Oracle 9i allows the feature model to highlight the full range of capabilities that can be used along with the actual data storage component. This section will describe these features from a high-level point of view. The feature from this level is entitled "Associated Functionality", as a way to group significant additional aspects from those features that are confined strictly to the Oracle database.

All of the sub-features described in this section can be considered as mandatory as they make up part of Oracle 9i's overall architecture. However, these are considered to be Or-features to account for the variability in their use from one database implementation to another. All of these features are considered worthy of mention as they are indicators of how extensive and elaborate the capabilities can be when interacting with the stored data of the Oracle database. The clients of these features are highly variable as it will depend on which of the features are put to use in a particular implementation. Consult each feature description in this section to see the kind of dependency of each particular type of stakeholder.

As Associated Functionality is a very general grouping and feature catering to the useful features outside of the database, this feature can be considered open to those future developed sub-features that are outside of the database architecture but part of the Oracle database release. As these sub-features describe the span of the extensiveness of other

useful and potentially integral aspects of the system, the priority of this Associated Functionality feature is considered high.

4.2.1 Internet Functionality

Versions 8 and 9 of the Oracle database server are better known as Oracle 8i and 9i respectively. The letter "i" suffix at the end of the version number refers to built-in internet capability in addition to the primary data storage functionality of Oracle databases.

Internet functionality is established as a feature to highlight those aspects of Oracle that use the internet, primarily as a communication protocol between a client and the database. This feature is the product of grouping Oracle's internet-related features together: Java programming, the Oracle Internet File System (iFS), and Extensible Markup Language (XML) support. As their use can vary with the implementation, these three sub-features can be seen as Or-features, as up to all three of them may be used in a particular setup. Each of these three sub-features is explained in more depth in the sections that follow.

As internet capability is built-in as part of Oracle 9i, this is seen as a mandatory feature. It is included and recognized as a feature as the internet has become a widely accepted communication standard, with user expectations of database communication going along this route as well. In response to this changing trend, dependent on this internet capability feature are newer applications and utilities designed with the internet in mind while accessing data from an Oracle 9i source. A good example of this feature is the Oracle Technology Network web site (http://otn.oracle.com), where a huge collection

of technical documents available via the internet is stored and made possible through use of this feature.

4.2.1.1 Support for the Java Programming Language

The Java programming language is part of Oracle 9i's built-in internet functionality. It allows applications to be developed with a programming language that is widely associated and accepted with the internet. To help accomplish the development task, Oracle provides its own Java development environment, Java development models, and supplemental Java-related utilities.

Java language support is considered to be a mandatory feature as the complete Java environment is supplied by Oracle 9i. This is a significant sub-feature representing internet capability because of the amount of development resources available to build Java applications as well as the extent of the capability of these applications that may be developed to interact with data in an Oracle 9i database.

The priority of this feature is high and will stay high as long as the internet and programming are considered to be the primary method of interaction between a client and an Oracle data store. The stakeholder of this feature is in the form of all developed Java applications that rely on this Java language support for any required future modifications in their operation. There are many examples applications operating off of an Oracle database; here, the only difference between these applications is the choice of the Java as the implemented programming language.

This Java language feature can be considered closed as Oracle provides established and multi-faceted support for the Java programming language. However, the feature may ultimately evolve with future releases made available by Oracle Corporation;

any upcoming changes should be viewed as enhancements in the form of upgrades to the component modules of this feature.

Although not considered part of this feature model but to better understand the extensiveness of Java language support, Oracle's development suite is more than a run-time module to execute Java programming code; first, it offers JServer, their version of the Java virtual machine (JVM), containing support for standard JVM components such as Java Database Connectivity (JDBC), and an Embedded SQL in Java (SQLJ) translator.

Second, the Java development models supplied by Oracle feature standardized models such as Enterprise JavaBeans (EJB) and the Common Object Request Broker Architecture (CORBA). As an alternative, Oracle allows customized Java coded routines in the form of stored procedures, similar to the development done with Oracle's PL/SQL development tool with the exception that the coded logic is written in Java rather than Oracle's own Procedural Language.

Third, Oracle provides Java development utilities. These tools include JDeveloper, a complete Java integrated development environment, and JPublisher, a utility the produces Java wrapper classes. The duo of simple tools named loadjava and dropjava are database server interfacing utilities that install or remove prepared java code respectively.

4.2.1.2 Oracle iFS – The Internet File System

Another internet-based aspect of Oracle is its Internet File System (iFS). To the client, iFS appears as a directory of documents of one or various types allowing authorized users to retrieve, view, and edit them. In reality, the storage of these document files is not in a directory but rather in the Oracle database itself. The iFS feature translates

the stored documents from a database to a directory viewpoint as well as ultimately converting all operations performed on a document back to the database.

As iFS is part of Oracle 9i, it is a mandatory feature. It is considered to be a feature as the database can now double as a virtual file server in implementations where there is storage of large amounts of documents, of one or more types. The clients for this feature are business users who do not need to know SQL to get their documents; they can merely access the directory and concentrate on manipulating the documents. The task of migrating the documents is left with iFS.

As iFS caters to a very specific yet significant pool of people, the priority of this feature can be considered moderate. The iFS was developed as a feature of Oracle 8i and has evolved with release 9i; it is expected to evolve with subsequent release as the acceptance and use of iFS grows. Hence, iFS can be considered an open feature, as more components are developed to constitute this feature.

4.2.1.3 Extensible Markup Language (XML) Support

Oracle 9i offers built-in storage of world wide web-based documents rendered in Extensible Markup Language (XML) without the need for any pre-processing of the documents required for storing them. As this is a part of Oracle 9i, this feature is a mandatory one and is recognized as a feature due to rising popularity of XML as a web document standard. The more people adhere and accept this standard, the more Oracle will be used as a means to store these XML documents in addition to conventional database data.

Clients who benefit from this feature are those who produce and track XML documents as the foundation of their business. Examples of using Oracle as an XML

repository are those organizations that use standardized documents, such as purchase orders, where the various fields eventually translate to fields stored in the database. Currently, the priority of the XML feature is moderate as using Oracle for XML storage is not widely used yet; eventually, the priority will change to high as more people use XML as the web document standard and store them in Oracle databases. The XML feature can be classified as open, as its use and support is still new with further enhancements upcoming, to be followed by a corresponding evolution in Oracle to accommodate these changes.

4.2.2. Performance

Performance is the resulting feature of combining critical performance sub-features of On-Line Transaction Processing (OLTP) and High Availability. It is a mandatory and high priority feature as Oracle databases are available on-line and their response time must be reasonable. Any aspects relating to on-line activities and availability are the dependents of this feature. It is also an open feature as the set of sub-features are not limited to just these two where on-line issues are concerned. OLTP deals with those issues related to communicating with the data source in an on-line mode while High Availability refers to the level in which data is available, minimizing the amount of downtime.

4.2.2.1 High Availability

High Availability refers to the high commitment level that a software system is made available to its users, taking into account the potential mishaps that may occur causing the system to be unavailable. Examples of systems using high availability are

those that require data constantly throughout the twenty-four hour day such as global banking and commerce. Overall, as availability of these types of systems is very important, recovery utilities are commonplace and offered by commercial database server vendors as a necessary contingency mechanism in the event of failure.

Oracle 9i supports the feature of high availability, allowing an Oracle data source to be made available constantly at all times. This feature is identified and included as Oracle has gained the reputation as a reliable database in mission-critical applications where the amount of downtime prevalent to the clients must be practically non-existent. Hence, the priority of this feature is high as the database must perform to guarantee this level of undisrupted service.

High availability is made possible through such recovery mechanisms as instance recovery, the Oracle Parallel Server (OPS) module, and the Transparent Application Failover (TAF). Instance recovery refers to the procedures that are run to restore operation of a database instance after a crash. OPS is a comprehensive utility that allows for multiple hardware servers to represent a single database in a parallel fashion; this parallelism can be beneficial in the event that if one of the machines in the group fails, one or more other machines can take its place in a temporary role during the crashed server's downtime. TAF is a utility that works in conjunction with OPS, but serves as a parallel option at the database instance level. Should an instance fail, TAF will dispatch users of the failed instance to another instance on another server and re-run all database transactions outstanding at the time of the original server's failure.

Throughout the years, the Oracle database server architecture has seen the advent of OPS and more recently TAF as a response to contributing toward high availability.

These developments serve as indicators that theories on making database systems highly available are continually being developed and forthcoming, making this feature an open one.

4.2.2.2 On-line Transaction Processing (OLTP)

On-line Transaction Processing (OLTP) refers to an interactive way of working with the data of a database, as opposed to using a batch mode for submitting database transactions. OLTP applications generate a large number of concurrent and repeated transactions, each of which operate on small amounts of data simple enough to require only low amounts of CPU and I/O times. However, these transactions characteristically include a significant proportion of inserts and updates to the database.

OLTP can be regarded as an important feature as it covers one of the two ways to submit transactions to the database. As opposed to the other option, the off-line batch mode, clients of OLTP systems have expectations that the data be readily available with hardly any downtime, as well as a prompt response time to their interactive tasks through an on-line application. There are many examples of these on-line systems such as those developed by Oracle Corporation: Oracle Sales Online, Oracle Financials, and Oracle E-business suite. The extensiveness of the types of applications that use on-line processing and the preference of users to make on-line transactions over batch ones make OLTP a very high priority feature.

A strong dependency exists for this feature as a data warehouse OLAP data source (section 4.2.4) is derived from data copied, manipulated, or filtered from existing OLTP databases. This is done for the main purpose of having the OLAP database serve

many ad-hoc queries. Should no databases be defined or exist to feed the data warehouse, the defined OLAP data source reduces to being an independent OLTP database.

To satisfy user expectations, Oracle databases using OLTP must possess the characteristics of concurrency, performance, availability, scalability, and reliability. The full list of measures contributing to OLTP performance is quite extensive, to the point where further sub-features are not expected, classifying OLTP to be a closed feature. Reference [4] describes a complete list of all aspects that can affect OLTP. Four compelling concepts that help meet these OLTP requirements are shared SQL, stored outlines, network connection management, and the Database Resource Manager.

Shared SQL is a feature of Oracle that can store frequently used and identical queries in a shareable cache of SQL queries and not spend additional time having to parse and optimize the same submitted query continually. Depending on the quality of the writing of the query, performance can be adversely affected. The objective is to write all queries in a structured fashion such that they can be viewed as identical once they are sent for processing. Non-uniformity in the query will result in no time saving as each query will have to be parsed and optimized first.

Stored outlines refer to specific execution plans configured by administrators to be used when processing a particular query. Having a declared execution plan as opposed to one formulated by the database server upon reception of a query guarantees stability and predictability in the performance of a query. This stability is crucial in delivering an acceptable response time and allows robustness from any changes to the server setup that may affect performance had the execution plan been left up to the database to formulate.

Network connection management encompasses connection pooling and connection multiplexing. Both of these concepts help toward maximizing the number of users that can access the database as a method to support the required bandwidth inherent with OLTP. Connection pooling allows multiple users access the database through a pool of available connections, thereby eliminating any monopolizing of a connection by a single user. Connection multiplexing involves coordinating the multiple users with using a smaller number of available connections.

The Database Resource Manager (DRM) is a utility that establishes the amount of CPU and parallel processing resources in order to satisfy particular trends of database transactions. The configurable aspect of this feature is that the proportion of allocated resources can be adjusted so as to cater to the large concurrent number of simple transactions that accompany OLTP applications. As the proportion of the allocated resources may be re-adjusted over time to improve performance, this feature is considered changeable. The feature can be considered closed as only the CPU and the parallel processing aspects can be adjusted, with no allowance of configuring other parameters.

4.2.3 Elaborate Data Handling

Elaborate data handling is a feature created from grouping those features in those cases where more than just standard data is handled. This feature exists to show that Oracle can handle storage and processing of complex data in terms of its size, function, or type. The mandatory sub-features under here are Data Warehousing, Very Large Databases, and Content Management. Data Warehousing is a supplementary database using data copied from one or more on-line databases to serve as a data source for

analytical queries. Very Large Databases are self-explanatory – they are databases of a much larger scale, which require specialized needs compared to normal Oracle databases. Content Management addresses storage of data whose data types are not part of the standard list of data types.

Elaborate Data Handling is an optional feature as not all Oracle databases use these avenues. There are no mutually exclusive conditions and no features are dependent on it. Availability is based on the configuration choices made for each database with no bounds imposed on enabling this feature.

4.2.3.1 Data Warehousing and On-Line Analytical Processing (OLAP)

A data warehouse can be set-up as an additional Oracle database to complement existing OLTP databases, where the objective is to serve as a read-only data repository for business analysis uses through On-Line Analytical Processing (OLAP). This is a significant feature in terms of feature priority and rationale as most organizations with extensive databases do define a separate data warehouse database devoted for their business trend needs so as to not disrupt the daily processing performance of the OLTP database. An example of data warehousing is where a commercial business records their daily sales transactions on OLTP databases while data trend analysis functions such as sales forecasts and quarterly results are confined to OLAP data warehouse databases.

Currently, the sub-features associated with this data warehouse feature are the business functionality modules built-in to the Oracle database architecture. These modules are in the form of extended SQL and Data Mining. The extended SQL feature in Oracle supports an extended SQL vocabulary in the form of additional aggregation functions catering to business-based queries. Data Mining is a devoted analytical module

performing more stringent data analysis such as sophisticated statistical analysis functions. As data warehousing addresses a specific analytical need, this feature can be seen as open to further advances in research, allowing the addition of more tools that enhance data warehouse performance or functionality in the future.

4.2.3.2 Content Management

Oracle content management is a feature that allows for various types of data to be stored in an Oracle 9i database. This feature is known as the interMedia, Spatial, Time Series, and Visual Information suite. The interMedia portion of the suite serves as a multimedia data storage format. Spatial handles data that is in the geographical format of spatial metrics such as location coordinates and descriptions of spatial areas. The Time Series is meant to process time-stamped data more efficiently within the database. Visual Information Retrieval portion of Oracle 9i involves being able to query a database of images such that the result returned matches a starting sample where image qualities such as color, size, pattern, etc. serve as the query parameters.

This data content management module can be considered a feature as it allows for non-text based data to be stored alongside textual data within an Oracle database. This variation of data types allows for the potential of configuring elaborate data stores to be used by specialized industries that deal with data types that are native to them but not native to previous database server architectures. It also give the opportunity to this population of users that Oracle 9i is a viable, less costly alternative to managing specialized data as opposed to procuring specially designed databases that cater to this kind of data exclusively.

This feature is of moderate priority as it currently serves a very specialized population of users. As it gains acceptance, this priority may change in the future as Oracle is relied upon more to store all types of data. The current set of dependencies is the various data types that are presently accommodated. An example of a content management database can be found with on-line shopping of music where the titles for sale showcase audio file snippets of the content. As content management is a relatively new development in Oracle, there is the possibility of having other specialized types to be added in subsequent versions classifying this feature as open.

4.2.3.3 Parallel Execution (VLDB)

Parallel execution is a feature that improves performance in primarily large Oracle databases, such as data warehouses and very large databases (VLDB). This is identified as a feature as there are a significant amount of vast Oracle databases in use, with most implementations requiring specific software and hardware requirements such as partitioned tables and using multiple processors. An example of VLDB and parallelism are global financial organizations such as banks where a large amount of financial data is kept. This is a moderate priority feature, as it caters to specific database implementations in terms of the size aspect.

Any Oracle database is ready to be set up for parallel execution if necessary. Most parallel operations offered by this feature are SQL-based, specifically in the categories of querying, DDL statements, DML statements, and user-defined functions. The remaining operations offered by an Oracle database that can advantage from parallelism are recovery, replication, and data loading (using the SQL*Loader application).

Somewhat dependent on this feature are those configurations using multiple processors. It is a parallel execution setup where multiple processing is most beneficial so although having a single processor with parallel processing enabled will still work, it will not yield the performance compared to having devoted processors working in parallel.

The future may bring some enhancements to each of these major categories supported by parallel operation but the feature itself can be considered closed, as no further significant areas of the database architecture can benefit from using parallelism.

4.2.4 Manageability Utilities

The operational utilities that come with an Oracle database can be grouped into a feature called Manageability Utilities. The set includes such tools as Enterprise Manager, Recovery Manager, Workflow, Database Resource Manager, and Legato Storage Manager. This is considered to be a feature as the use of these utilities is essential to the proper and efficient configuration and operation of any Oracle database. Hence, the priority of this feature is very high.

These utilities are designed, not for users, but for technical professionals required to maintain the database in an operational sense such as database administrators. Dependency-wise, the operation and performance of the database is reliant on this feature. If all of the configurable parameters of the database are not set properly for the environment and its intended use, performance of the data source will be less than exemplary. As the definition of effectively managing a database can change with time, this feature is an open one, accepting new innovative utilities that complement or replace the current set of utilities in the future.

4.2.5 Data Access

The Data Access feature is the compilation of the Networking and Security features. It addresses any features that apply to the flow of the data from the database to the requesting clients. Since this communication is reliant on the connectivity (the Networking feature) and the access privileges of the data (the Security feature), no other sub-features are expected here so the Data Access feature is a closed one. As access is necessary at all times for the database to fulfill its role, this is a mandatory feature.

4.2.5.1 Networking

Networking is identified as a feature under associated functionality as it refers to all aspects of communicating the data back and forth between the database server and requesting client within a networked environment. This feature pertains to more than just the networking convention that Oracle uses for communication; it also refers to dedicated connection options, multiplexing, access control, network protocol support, and multi-tier communication. Information on all these network-related aspects can be found in reference [7].

The stakeholder of this feature is the database itself as the networking settings establish the ways in which the data is accessed such as with dedicated connection pooling. Similarly, the clients accessing this data is affected as well, as the actual routing to the data source may not be a direct one that the clients visualize but rather a specific pre-defined one dictated by the current network settings.

Essentially all database implementations of Oracle use some aspect of this networking feature making it very high in feature priority; without it, all access to the data store would have to be made directly from the server, which is not very practical.

Whether clients to a database are situated locally or globally, a network connection must exist between them and the data source. The networking aspect of Oracle is quite in-depth to the point that any further development of features seems unlikely. Hence, this feature can be considered closed to any future sub-features.

4.2.5.2 Security

Although data is centralized in a database for availability to its users, security must be applied to ensure that the intended data is accessed only by the set of users that it is intended for. This data access regimen gains more importance if the database has a multiple role of serving internal users as well as external users via the world wide web, for instance. Thus, security is considered to be a feature as its exclusion would mean that everybody could access any aspect of the data, leading to an easily corruptible data source. Hence, it is a very high priority feature, as it is imperative to disallow access to sensitive data from unauthorized individuals.

This lack of security may apply to those environments where very small populations of users have full access rights to the data, including data access, modification (such as data deletion), and data source configuration. However, in almost all implementations, a security policy must be established before the data can be used. The client programs using this feature are all those interfaces that must access the data. For example, user screens, SQL scripts, and programmed interfaces must go through a security check to see if the data is allowed to be accessed by the requesting client. This checking procedure goes through verifying the user profile of permitted access privileges, resource quotas, and user roles. These values are constrained and dependent on the settings established by organization and entered by an administrator of the database.

When implemented, security is enforced at all times when the data source is on-line and available to users. Binding of this security feature can take place in the form of changing the settings for a given user. This is done manually through an individual responsible for the database security. The priority assigned to this feature is high as it is important in most instances to protect the data from misuse. Any enhancements to the Oracle database server should definitely consider that the existing security mechanisms are either maintained or enhanced.

The security feature can be considered open to new sub-features, dependent on whether Oracle chooses to develop additional ways to implement data security. Currently, security of the data can be implemented in two forms: standard discretionary access control and advanced security. Discretionary access control establishes parameters for data access whereas the advanced security is as an optional module that can extend the security, through methods such as encryption, when data is made available in a larger environment. Implementation details of these two security methods can be found in references [8] and [10].

4.2.6 Programming Interfaces

A useful feature available in Oracle is the numerous interfacing capabilities available to access the data through some form of programming. This can be justified as a feature since it may be required that an organization's data may need to be further processed in such ways that can only be accomplished through the more powerful constructs and logic offered by programming languages rather than scripting manageable SQL queries.

This feature offers a multitude of language choices and methods to instill the required programming to process the data in the desired fashion. Usually, the programs are written in a particular language and need to be pre-compiled with the appropriate SQL statements required to achieve the data querying requirements of the data source. Supported languages in Oracle 9i include C++, COBOL, and Java. The Java programming language is described in depth in section 4.2.1.1 as it is internet-related.

Dependencies found with this feature are the set of written programs that access the database and produce a result based on the data that is queried and then manipulated. As the number of these programs can be vast, this feature is of a high priority as these programs represent functionality desired by the users of the data source. Most commercially sold applications are examples of a pool of files representing programs written in one or more supported programming languages.

This feature is not limited to the list of pre-compilers of supported programming languages. The sub-features also include the Oracle Call Interface, Open Data Base Connectivity (ODBC), Common Object Models (COM), and Object Linking and Embedding (OLE). These are the current programming choices that are available in Oracle but as new languages and methods to access data are developed and accepted, this set of supported languages will ultimately change, making this feature open.

4.2.7 Distributed Databases

Oracle allows a database to be set up as a distributed database: a series of data stores in multiple locations that are grouped together to appear as a single logical database to its users and its application. This particular configuration method is recognized as a feature as it is a commonly used, efficient, and cost-effective

implementation approach for large databases that serve many decentralized clusters of users and distributed applications. An example of a distributed database is for those organizations with geographical clusters whose combined data represent the global database for the entire organization.

This feature is readily available, as setting up a distributed database is straightforward as long as the databases to be grouped together exist. The process to link the databases together represents the dependency of this feature: the distributed database cannot exist until the linking commands have been entered for each of the databases that comprise the distributed database.

The primary sub-features offered by Oracle distributed databases are replication, the two-phase commit mechanism, advanced queuing, and the capability of defining a distributed database as a set of heterogeneous databases.

Replication allows a copy of a distributed database component to exist and serve data, either immediately or as a disaster contingency. The copy of the data may reside in the same location as its original counterpart or can be copied to another location within the distributed network of data stores.

The two-phase commit mechanism applies to ensuring that a transaction is successful along all of the multiple physical locations in which the data that needs to be committed applies. Essentially, the first phase is to send a message to check to see if all the locations are ready to take on the portion of the transaction that applies to each of them. If a response is received from all pertinent locations, the transaction goes ahead; if not, the transaction aborts and the data is rolled back to its prior state.

Advanced queuing is used in a distributed database environment but more for distributed applications to communicate to the data store or other parts of the application by means of passing messages through a queue. This queue is located directly in the database, offering a reliable location to store these important messages.

Oracle has the ability to configure distributed databases comprising not only of Oracle-defined databases but databases from other software manufacturers as well. Communication to these non-Oracle, heterogeneous portions of the distributed database is done either through ODBC or OLE DB connections or with an Oracle Transparent Gateway designed for the specific brand of non-Oracle database.

This Distributed Database feature can be classified as open based on the various sub-features that can be ultimately added to further improving the operation of distributed databases. Past examples of these sub-feature additions made by Oracle are advanced queuing (released with version 9i), and Oracle Streams (another messaging facility released with Oracle 9i). Future research into distributed databases will undoubtedly create further sub-features here.

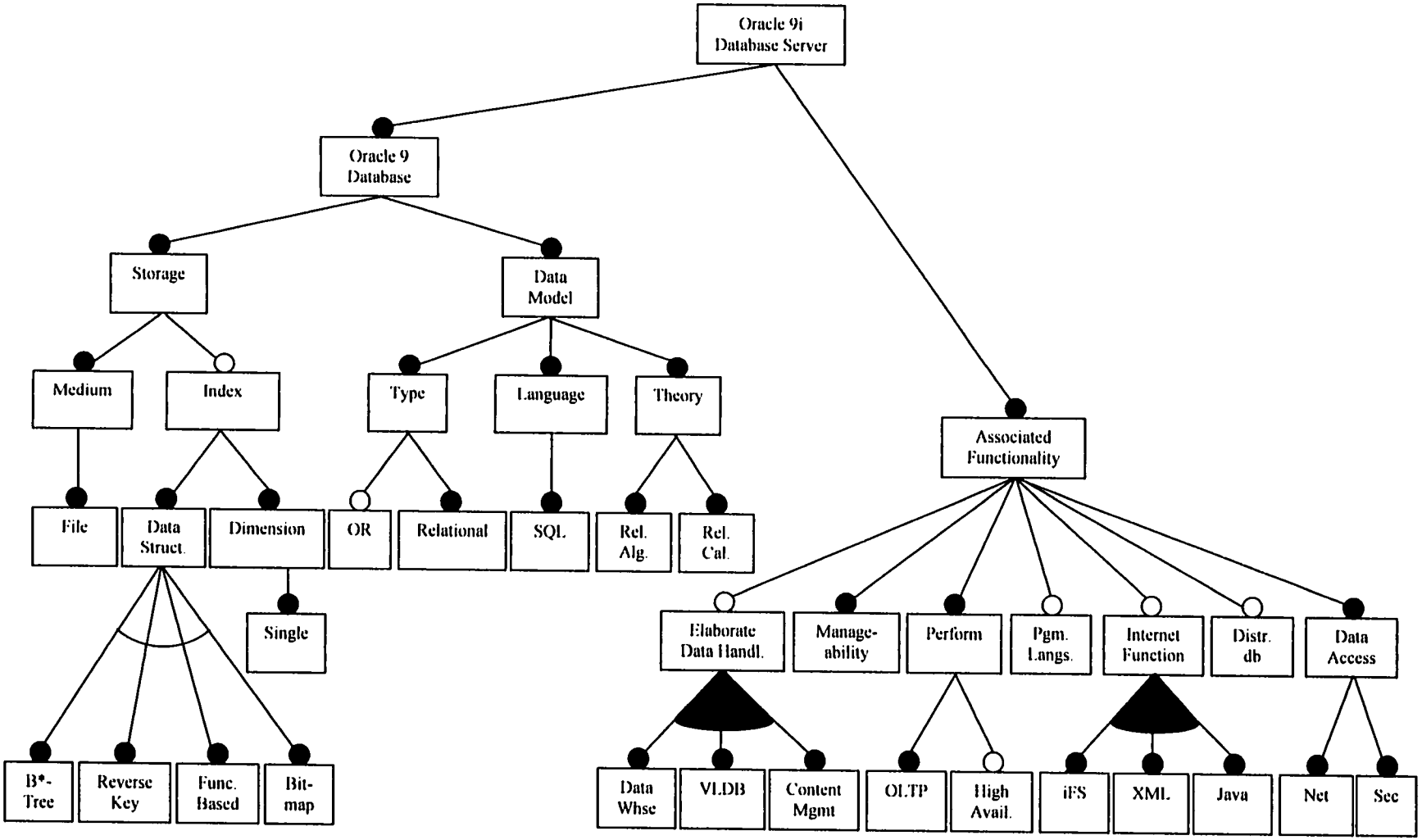

Figure 4-1: Feature diagram for the Oracle 9i database server.

5.0 Conclusions & Recommendations

Feature modeling is a method used in Domain Engineering that can be applied to identify all commonality and variability within a software system of interest, ultimately revealing the degree of reusability. As was seen in this report, arriving at the completed feature model involved a series of in-depth steps, even though the final result appears to be elementary in its presentation and explanation of identified features.

Based on this experience of modeling the Oracle 9i database server, formulating any feature model should begin with a good proficiency of the modeling procedures, diagram notation, and the necessary descriptive components. Simple examples should be modeled by the person doing the modeling in order to get an idea of how detailed the feature identification and description task is; preferably, the examples chosen should be objects where the modeler has vast knowledge about them in order to derive a very comprehensive model. This practice task will show the various difficulties that can be encountered along the way of constructing a feature model.

Once the modeling methods have been learned and practiced, the collection of information about the object being modeled need to be collected. It is recommended that as much can be found out about the subject object through the use of literature, experts, and similar implementations. The more information that is available, the more accurate the commonality and variability results will be, giving way to a more accurate feature model. At this point, it is important for the impending design of the feature model to be made flexible enough to accommodate any new relevant information that may arise during or after the modeling process.

As the available information is compiled and reviewed, the features that are identified and described can then be assembled into the feature diagram, relative to the other features that have already been found to be relevant. The description of the features should be comprehensive, with accompanying explanations of as many aspects of each feature as possible, such as rationale, stakeholder, and priority information. The end of this step should result in a completed feature model, but it is worthwhile taking advantage of the fact that the modeling procedure is iterative. The whole process can be repeated to see if any more relevant features can be identified and integrated into the model.

Modeling the Oracle 9i database server was an example of trying to model a vast and widely used commercial software product. As the feature diagram denoted, the database portion is merely one component of Oracle 9i, complemented by a series of additional modules that enhance its main data storage function.

Deriving a feature model as large as Oracle 9i helped by starting with a top down approach of using the feature model of a general database as a starting baseline. This model served as an effective structure, identifying which aspects of Oracle had to be further studied to get the feature information specific to Oracle. Initially, a bottom-up approach was used in trying to derive this feature model but the initial results revealed an inconsistent level of detail between various aspects of the system. Some components were explained in the finest details while others were superficial in its feature identification. Although there are merits to using a bottom-up approach, such as identifying as many features about the subject as possible, it is recommended that the top-down approach be used first, in order to establish the focus required in describing the features that need further elaboration. A top-down approach is also beneficial as it is

more helpful in defining the relationships between identified features by determining the sub-features of a given feature. Establishing the feature relationships using a bottom-up approach required finding what the super-features of a given feature are. Difficulties can arise with this super-feature approach as other identified features must be checked to see if they have the same super-feature. Even if super-features can be matched, a top-down approach is still required at this point as to verify whether there are other features associated with this super-feature.

The comprehensive feature model of the Oracle 9i database server described in this report gives readers an indication of its overall range of capabilities and configurable aspects. This model shows just how different and extensive the Oracle instantiation is compared with the general database model. It also serves as a guide to differentiate Oracle 9i with other competing commercial database server products. Hence, a broad and general approach was taken in extracting those features that accompany the database without going into too much detail or extracting a hierarchy of sub-features that would not contribute in further differentiating Oracle. More in-depth information about the features shown can always be consulted in the various sources of documentation listed in the References section.

The model presented here is not the only feature model possible for Oracle 9i. Many more models can be derived, some of which looking totally different from this one but still retaining the objective of demonstrating the product's flexibility and reusability. The determining factor is the general focus and on which areas of the database should be concentrated on for any potential models. The particular emphasis of this model was on the database product's capability characteristics. Similarly, other models may focus

efforts in identifying all features pertaining to other, specific components of Oracle 9i, such as those features considered as part of the associated functionality (section 4.2). Some of these features are very complex and a feature model devoted to solely one of them would be beneficial in understanding the full realm of the features it has to offer. For example, the built-in XML capability of Oracle 9i could benefit from this kind of analysis to reveal its component features as well as its suitability in specific implementations.

Further detailed elaboration of the feature model presented here of one or more of its high-level features is one idea for future work and research. Another idea is to derive similar feature models for other database products (such as SQL Server, Informix, IBM DB2, and Sybase) and then to compare them to see what the features of all products have in common and which are limited to one single product. A master feature model could be derived of all commercial databases in this respect. This kind of comparison can be most helpful when it comes to choosing a database product for a proposed research or new design implementation. By finding out the full domain of the capabilities for all candidate products, one or more can be identified in terms of addressing the needs and requirements of future research projects.

Oracle 9i was released in 2001, which will most likely not be the final version of an Oracle database server. Constructing a feature model of subsequent version of Oracle is an additional idea for future work in an effort to show the evolution of Oracle features between two consecutive versions. This kind of in-depth comparison would be beneficial, not only in research but in industry as well, serving as a high-level visual guide to the direction of the software.

Feature modeling can be applied to any application to help identify relevant features that can ultimately help in determining its viability and reusability. The feature model presented here shows just how elaborate a particular commercially-available database server can be, even after examining just the high-level features. If the time and resources are available, feature modeling is a worthwhile exercise as the results can be beneficial to those evaluating or working with the product in question. Consultation of a feature model can save time, money, and resources by indicating the degree of configuration of the aspects identified to be relevant to the object in question.

6.0 References

1. Bobrowski, S., *Oracle 8 Architecture*, McGraw-Hill, Toronto, Ontario, 1998.

2. Czarnecki, K. and U.W. Eisenecker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, Montreal, Quebec, 2000.

3. Desai, B. *An Introduction to Database Systems*, West Publishing Co., Saint Paul, Minnesota, 1990.

4. Greenwald, R., R. Stackowiak, and J. Stern, *Oracle Essentials: Oracle 9i, Oracle 8i, and Oracle 8*, Second Edition, O'Reilly & Associates, Inc., Sebastopol, California, 2001.

5. Lee, K., K.C. Wang, W. Chae, and B.W. Choi, *Feature-Based Approach to Object-Oriented Engineering of Applications for Reuse*, Software: Practice and Experience, Volume 30, Issue 9, 2000.

6. Oracle Corporation, *Getting to Know Oracle 8i*, Part No. A68020-01, Belmont, California, 1999.

7. Oracle Corporation, *Net8 Administrator's Guide*, Part No. A67440-01, Belmont, California, 1999.

8. Oracle Corporation, *Oracle 8i Administrator's Guide*, Part No. A67772-01, Belmont, California, 1999.

9. Oracle Corporation, *Oracle 8i Concepts*, Part No. A67781-01, Belmont, California, 1999.

10. Oracle Corporation, *Oracle 8i Reference*, Part No. A67790-01, Belmont, California, 1999.

11. Oracle Corporation, *Oracle 8i Tuning*, Part No. A67785-01, Belmont, California, 1999.

12. Oracle Corporation, *Oracle 8i Utilities*, Part No. A67792-01, Belmont, California, 1999.

13. Silberschatz, A., H.F. Korth, and S. Sudarshan, *Database System Concepts*, Third Edition, McGraw-Hill, New York, New York, 1997.

14. Xu, L. *Framework: Advanced Technology in Object-Oriented Field*, Ph.D. Research, Concordia University, Montreal, Quebec, 1999.

15. Oracle Corporation, *Oracle 9i Concepts*, Part No. A96524-01, Belmont, California, 2002.

16. Oracle Corporation, *Oracle 9i Database Generic Documentation Master Index*, Part No. A96625-01, Belmont, California, 2002.

17. Oracle Corporation, *Oracle 9i Database New Features*, Part No. A96531-01, Belmont, California, 2002.

18. Oracle Corporation, *Oracle 9i Database Reference*, Part No. A96536-01, Belmont, California, 2002.

19. Oracle Corporation, *Oracle 9i Database Utilities*, Part No. A96652-01, Belmont, California, 2002.

20. Oracle Corporation, *Oracle 9i Data Warehousing Guide*, Part No. A96520-01, Belmont, California, 2002.

21. Oracle Corporation, *Oracle Internet Directory Administrator's Guide, Release 9.2*, Part No. A96574-01, Belmont, California, 2002.

22. Oracle Corporation, *Oracle 9i Security Overview*, Part No. A96582-01, Belmont, California, 2002.

23. Oracle Corporation, *Oracle 9i Streams*, Part No. A96571-01, Belmont, California, 2002.

24. Banerjee, S., *Oracle XML DB*, Oracle Corporation, Belmont, California, 2002.

25. Cheevers, S., *Oracle 9i Database Summary*, Oracle Corporation, Belmont, California, 2002.

26. Dawson, D., *Managing Data in the Age of the Internet: Oracle Internet File System (9iFS)*, Oracle Corporation, Belmont, California, 2002.

27. Lee, G., *Simple Strategies for Complex Data: Oracle 9i Object-Relational Technology*, Oracle Corporation, Belmont, California, 2002.

28. Lejeune, H., *Scalability and Performance with Oracle 9i Database*, Oracle Corporation, Belmont, California, 2002.

29. Scardina, M.V., *XML Support in Oracle 9i*, Oracle Corporation, Belmont, California, 2002.