UPTEC IT 09 009
Examensarbete 30 hp
Maj 2009

Database Performance and Complexity in Visual DataFlex

Håkan Johansson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Database Performance and Complexity in Visual DataFlex

Håkan Johansson

This report is meant to be used when choosing a third-party database management system in combination with Visual DataFlex. The database managers that are included in this test are Microsoft SQL Server, Pervasive.SQL, Oracle and Embedded Database. It will cover the pros and cons with each tested database manager when using applications created with Visual DataFlex. It will also cover the fault-tolerance when using clustering, the backup possibilities with minimal downtime and the complexity of installing and configuring the database manager.

Visual DataFlex is a framework for developing database applications aimed at Windows or web platforms. The framework is owned and developed by Data Access Worldwide, Inc.

Visual DataFlex provides a database manager called Embedded Database which is not suitable to be used in a multi-client environment. Therefore, in multi-client environments, a third party database management system is normally used.

Visual DataFlex is primarily used in smaller and midsized environments. Database administrators are normally not wanted in these environments and therefore it is important that the third-party database management system is easy to set up and configure.

For database independency, Visual DataFlex uses connectivity kits which are an abstraction layer between the internal interface in Visual DataFlex and the client software of the database manager in use. Through the connectivity kit, the most popular database management system can be used and database managers that are not supported directly by a connectivity kit can still be used via an Open Database Connectivity (ODBC) driver.

There are currently seven connectivity kits developed by Data Access, Inc and five developed by Mertech Data Systems, Inc.

Handledare: Anders Öhrt
Ämnesgranskare: Kjell Orsborn
Examinator: Anders Jansson
ISSN: 1401-5749, UPTEC IT09 009
Tryckt av: Reprocentralen ITC

Sammanfattning

Visual DataFlex är ett utvecklingsramverk som är specifikt riktat mot att bygga databasbaserade applikationer. Applikationerna kan vara antingen Windows eller webbaserade. Ramverket ägs och utvecklas av Data Access Worldwide.

Som databashanterare kan man använda den medföljande Embedded Database, ett filbaserat serverlöst system som medföljer utvecklingsmiljön och där varje klient hanterar filerna själv, men för fleranvändarsystem använder man normalt en databashanterare med client-server arkitektur.

Visual DataFlex används oftast i mindre och medelstora miljöer. I dessa miljöer vill företagen normalt sett inte anställa databasadministratörer och det är därför viktigt att det är enkelt att installera och konfigurera databashanteraren.

För att uppnå databasoberoende använder Visual DataFlex så kallade connectivity kits vilket är ett abstraktionslager mellan det interna gränssnittet i Visual DataFlex och klientmjukvaran för den databas som används. Det finns för närvarandet sju connectivity kits utgivna av Data Access, samt fem utgivna av Mertech Data Systems, Inc. Via dessa connectivity kit stöds de populäraste databaserna via en direkt koppling till klientmjukvaran och även att man kopplar via Open DataBase Connectivity (ODBC) vilket gör att databaser som inte stöds direkt ändå kan användas om dessa har en ODBC-klient.

Målet är att skaffa ett bra underlag för val av databashanterare, dels vid nyinstallation och dels vid eventuell migrering från nuvarande databashanterare. Databashantere som kommer att testas och utredas är Microsoft SQL Server, Pervasive.SQL, Oracle and Embedded Database. För- och nackdelar med de olika databashanterarna avseende prestanda i applikationer skapade i Visual DataFlex, möjligheten till feltolerans vid exempelvis klustring, möjlighet till säkerhetskopiering utan eller med minimala driftsavbrott samt komplexitet i installation och konfiguration behöver utredas.

Table of Contents

1 INTRODUCTION

A lot of factors are important to keep in mind when analyzing the results of a test, i.e. the hardware and software environments. The introduction is meant to clarify the different parts of the test environment.

1.1 Purpose

This report is written using a specification created by Berendsen Textile Service AB which is a company that provides textile service solutions for several different service sectors.

Berendsen owns laundries in several countries and each laundry has a database containing articles, customers, invoices, delivery notes, and more connected to that laundry. Each country controls the technical environment used at the laundries in that country and decides which database and application should be used at their laundries.

A majority of the countries are using applications created using Visual DataFlex but the database that the Visual DataFlex application is connected to differs between countries.

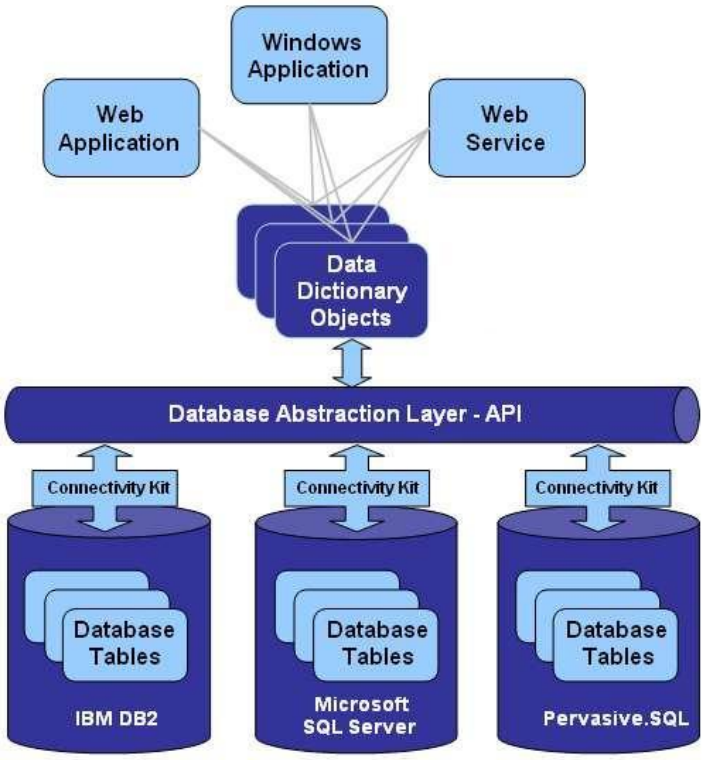
The purpose of this report is to see different aspects when using different database managers connected to a Visual DataFlex application and measure the performance, evaluate the process of setting up the database manager and more. The report is meant to provide support when choosing which database that should be used when using Visual DataFlex applications.

1.2 Visual DataFlex

Visual DataFlex (VDF) is a programming language and an environment for developing applications. The application used in the test is created using VDF which is a framework specifically aimed at creating database applications for Windows or web based platforms.

> "Visual DataFlex's architecture is designed for building Windows and web applications, web services and Service Oriented Architecture (SOA) using SQL-based DBMS servers such as Microsoft SQL Server, IBM DB2, Pervasive.SQL, Oracle and ODBC data sources. The goal of the design is to provide an efficient, easy to use, easy to maintain development environment that is largely database independent." [1]

Each database table in an application created using VDF has a Data Dictionary Object (DDO) connected to it. When an application sends or receives information from a table, the query is handled by the DDO connected to that table. Depending on which database manager is used, the table has a specific structure which is not known by the DDO. In the layer between the database tables and the DDO, a connectivity kit (CK) is used which is a driver that handles the conversions needed for the database manager in use and it makes VDF database independent.



The DDO uses a record buffer to store records that matches queries that is given by an application.

1.2.1 Record Buffer

When a database table is opened in an application, a section of memory large enough to hold one record is allocated. The allocated memory is called the record buffer and is used to handle read and write operations in VDF applications. When a record matches a query, it is stored in the record buffer.

The record buffer can only handle one record at the time compared to an SQL query that can return a set of records matching a query.

Assume that a table containing all the names of students at Uppsala University exists. The table is called Students and has a column Lastname. To find and count how many students that have the lastname Johansson, an SQL query may look like:

```
Select Count (*) from Students where (Lastname = "Johansson")
```

Select returns a set of records that matches the condition (Lastname = "Johansson") in the table Students. The query is finalized with the Count operation which counts the number of records that matches the criteria. This amount is the number of students that has lastname Johansson in the table Students.

Using VDF and the global record buffer, it looks a lot different since the record buffer only holds one record at the time:

```
Integer iCount          // Used to store the number of records
Clear Students          // Clear the record buffer
Find ge Students        // Find the first record
While (Found)
  If (Students.Lastname = "Johansson") Begin
    Increment iCount
  End
  Find gt Students      // Find next record
Loop
Showln iCount           // Displays the value on the screen
```

Found is a global variable with value true if the previously found query matched a record in the database.

Find takes a number of arguments. The first argument is the find mode. The find mode is based on the data currently in the record buffer and the index in use. When first clearing the record buffer and performing a *Find* with find mode *ge* (greater or equal to), it always finds the first record matching the query for the given index since the lowest value in the table is still greater or equal to a blank value. Each table has a default index that is used if no index is given to the Find operation as in the example above.

Students is the table to perform the find operation on. In this query, the default index is used and the above written code will search through all records in the table and try to match the lastname. When a condition is known, in this case lastname should be equal to Johansson, it is possible to use an index to make a more efficient search.

Indices can be created using tools in VDF, however, the index is still controlled by the database manager. If Oracle is used and an index is created in VDF, the connectivity kit sends a query to the database that creates the index. The index is always controlled by the database that is used and not by VDF.

Applications created using VDF are single-threaded. Threads are a way for a program to split itself into two or more simultaneously running tasks. Since applications in VDF does not support multi-threading, only one task is running at the time. This allows the record buffer to be used as a global variable since only one task at the time can use it.

Since the record buffer is global, it can be modified before the find operation to make a more efficient search using an index. Assume the table Students has an index of the column Lastname and the code can be rewritten to:

```
Integer iCount                          // Used to store the number of records
Clear Students                          // Clear the record buffer
Move "Johansson" to Students.Lastname   // Init the record buffer
Find ge Students by Index.1             // Find the first record using index 1
While (Found and Students.Lastname = "Johansson")
  Increment iCount
  Find gt Students by Index.1           // Find next record using index 1
Loop
ShowIn iCount                           // Displays the value on the screen
```

In this case "Johansson" is moved to the record buffer before the first *Find* which uses an index containing the column Lastname. If that index is the default index it does not have to be sent as a parameter to *Find*. The record that will match the query will be the first record in the table with lastname Johansson. The condition *Found* and *Students.Lastname* equal to Johansson will be false if *Find* does not find another record or if the record found does not have Student.Lastname equal to Johansson.

A *Find* query in VDF is always converted into a query that matches the current database manager in use and the connectivity kit is the layer that converts the query into the correct syntax for the database manager. This means that a database manager, when connected to a VDF application, behaves in the exact same way as it does when connected to another environment. Query optimization and other methods that are used by the database engine works in the same way when using a VDF application.

Each record in a table needs to be able to be uniquely identified. This is done with RowId which is a property that gives all records in a table a unique identifier. In earlier versions of VDF, RowId was used to create a column called recnum for all tables containing the RowId for each record. This guaranteed that tables had at least one unique key and the recnum column was required for all tables. Recnum is not required anymore since RowId can be based on a chosen column that can uniquely identity all records in a table. A lot of applications written in the earlier versions of VDF still use it. Because of that the database managers' recnum support will be evaluated.

1.2.2 Database Builder and Database Explorer

Applications written in VDF are aimed at systems using databases. VDF therefore has two powerful tools, Database Builder and Database Explorer, to handle them.

The Database Builder is used to create, modify and delete tables. A table has columns and each column has properties such as name, data type, size, etc. Actions like adding or removing columns, setting column attributes and creating indices are handled using the Database Builder.

The Database Builder creates three different files for each table. The first file contains the data dictionary class, which contains the rules and settings that the table has. Properties like default values, upper casing and range validation are defined here, as are events that trigger when a record is created, modified or deleted. The data dictionary is the layer between the application and the table. The data dictionary object (see section 1.2) is created from the data dictionary class.

The second file contains a list of definitions. Here the table numbers as well as the column numbers and types are specified. These definitions are used to bind the record buffer to the physical table.

In the third file, the columns, types, relations and indices are listed in a human readable format. This file is not compiled into the application as the other two are, but contains an overview of the table.

The Database Explorer is used to view and modify the data in a table. It also has filtering functions and it is possible to view column types, indices and relations. It can also export data from tables, either all records or using filters.

1.2.3 The Embedded Database

VDF has its own database manager which is called the Embedded Database (ED) and it uses the file system of the underlying operating system to share files between clients. For Windows, a protocol called Server Message Block (SMB) is used.

When an application is started and the ED is used, all tables are opened for read access. When a client needs to write to a table it asks the server to lock the file. If the file is unlocked, the server locks the file for that client only but if it was already locked the request is rejected. Once the file is locked, the client performs write operations on the file and then releases the lock.

When using ED as a single client, the server grants exclusive access to that client which enables the client to cache all data locally which gives fast query responses. The client can keep working locally until another client connects to the server. When that happens, all the local data is synchronized with the server and the clients share the data via the server again.

Local caching does not only provide advantages. For database managers there is a set of properties called ACID (see glossary for more information) that guarantees that database transactions are processed reliably. The ED does not have all these properties and therefore it is not a suitable database manager to use in production or critical systems.

Atomicity is a guarantee that all or nothing is committed of a transaction. ED does not handle rollbacks if the server crashes during a transaction since no write operations are logged.

Durability is a guarantee that if notified of success, a transaction will persist and not be undone. Since ED gives exclusive access for single clients this is not guaranteed. If there is a bad connection to the

client the exclusive access can get out of synch because the server can consider the client dropped and give it to another client. When the first client returns, the data that will be synchronized to the server cannot be guaranteed to be correct.

1.3 Optimizing Queries

Cursors are used to be able to process records in a result set of a select query. Cursors are database objects that points to a location within a result set and they are further used to traverse the result set.

The record buffer only handles one record at the time and there are ways to make *Find* operations more efficient using something called a block size cursor. There are two types of cursors that can be used for every table in the database, the single row cursor and the block size cursor.

The single row cursor is used for *Find equal* operations. A *Find* using the find mode *equal* finds the only record in a table that matches the data currently in the record buffer using a given unique index. For example: Assume the table Students with a unique index of the columns firstname and lastname. To find if there is a student with the name "Håkan Johansson" the single row cursor can be used and it can look like:

```
Clear Students                              // Clear the record buffer
Move "Håkan"       to Students.Firstname    // Edit the record buffer
Move "Johansson" to Students.Lastname       // Edit the record buffer
Find eq Students by Index. 2                // Find the record matching the buffer
If (Found) begin
  Send Info_Box "Exists"                    // The record is in the record buffer
End
Else Begin
 Send Info_Box "Does not exist"             // The record does not exist
End
```

The block cursor is used for all other find modes. It is used to cache a number of records that matches a query to quickly find the next record if doing another *Find*. In the example when counting how many students with the lastname Johansson exists in the table Students, a block cursor is used. When performing a *Find*, only one record can be stored in the record buffer but the connectivity kit fetches multiple records, depending on the block size, and stores the data in memory. If another *Find* is performed, the next record matching the query can be retrieved from the memory directly instead of fetching it from the database.

When using this form of caching, the risk of returning incorrect data increases. Therefore, the block cursor is controlled using a find cache timeout which has a default setting of ten milliseconds. When a *Find* is performed, a number of records are cached according to the block size property. If a new *Find* is not performed within the cache timeout value, the cache is cleared.

These cursors are handled by the connectivity kit and they will be altered in the testing session. Since it is handled by the connectivity kit, the cursors are used in different ways depending on which database manager that is connected.

1.3.1 Pervasive.SQL vs. Oracle/Microsoft SQL

Pervasive.SQL has two engines, a set based and a row based. There are differences between using the set based and the row based engine in combination with the record buffer method that VDF uses.

When looping through an entire table to update a column in a VDF application, each record in the table has to be fetched into the record buffer and only one record at the time can be fetched. A row based engine is optimized for fetching each record based on the index and the data currently loaded in the record buffer.

When performing a *Find* and using a row based engine, a record matching the query is fetched. If another *Find* is performed, that uses the same index as the first *Find*, the database manager performs the query and returns a matching record that the connectivity kit fetches into the record buffer. For each *Find*, the database manager performs a query on the database and returns a matching record if it exists. The row based engine always returns only one record that matches the query and each *Find* results in a query on the database.

A set based engine can fetch multiple records for a query. The record buffer can still only fetch one of these records for each *Find* and the connectivity kit is the layer between the application and the database engine that handles this difference between the record buffer and the set based engines result set. When a *Find* is performed and a set based manager is used, the connectivity kit creates a query that the database manager performs on the database. This can lead to a result set that matches the query. Since the record buffer only can fetch one of these records for each *Find*, performing another *Find* that uses the same index might lead to the connectivity kit fetching the next record from the result set instead of from a new query on the database. If the result set does not contain any more records, the database engine can perform a new query to return a new set of records if they exist.

In general the difference between a set based and a row based engine is that a set based engine gathers all records matching a query before updating them. A row based engine fetches one record, updates it and then fetches the next record and so on.

The connectivity kit provides the cache properties to optimize set based engines. Queries in VDF always update records one by one even if a set based engine is used. The block cursor is used between the connectivity kit and the database manager to increase the performance since the database manager can be set based. It is still only one record at the time that is handled by VDF.

When a *Find* is performed, a record is loaded into the record buffer and the block size cursor is used to cache the next ten records (depending on the block size property). This gives overhead since more than one record has to be processed and if the cache timeout is not set to an optimal value, some of these values will be discarded. The row based engine in Pervasive.SQL does not have this overhead since it is designed to fetch records one by one.

There is no setting that gives the optimal cache for all systems. It depends on how frequent the database is updated and therefore the overhead can lead to loss of performance.

The test will show the difference between using a row based and a set based engine when running VDF applications.

1.4 Database Management System

For systems where the Embedded Database is not suitable, a third party database manager can be used. When choosing which database manager to use, there are many different factors that effect the decision. These are some of the factors that will be taken into consideration when testing and evaluating the different database managers.

- Complexity when installing the server and client
- Administration and configuration of the server and client
- Performance
- Backup possibilities and support

The database managers that will be used in the test are:

- Microsoft SQL Server 2005 Enterprise Edition [2]
- Oracle 11g Enterprise Edition [3]
- Pervasive.SQL v10.10 summit [4]

Microsoft SQL Server and Oracle are two of the most commonly used database managers. Pervasive.SQL will be used because it is a commonly used database manager when using VDF. The performance of the Embedded Database will also be a part of the test. This will show which environments are suited for the Embedded Database and which are not.

The versions used will be the latest stable versions released for each database manager when this test is started which is October 2008.

1.5 Architecture

The hardware used is not important to the results as long as the same hardware was used during the entire test. It does effect the times mentioned in the results but those should only be compared to other results in the test. An individual test result does not tell you if a database manager has high performance or not.

The server and client will be connected through a Local Area Network (LAN) during the test. The reason for this is to eliminate the uncertainty of using a Wide Area Network (WAN) in order to provide as accurate tests as possible.

Figure 1.5.1. The image shows the design of the test environment. The client is connected to the server through a Local Area Network.

The application will run on a client (see figure 1.5.1). When an action is taken in the application, a database query will be sent to the server. The server processes the query and sends the results back to the client.

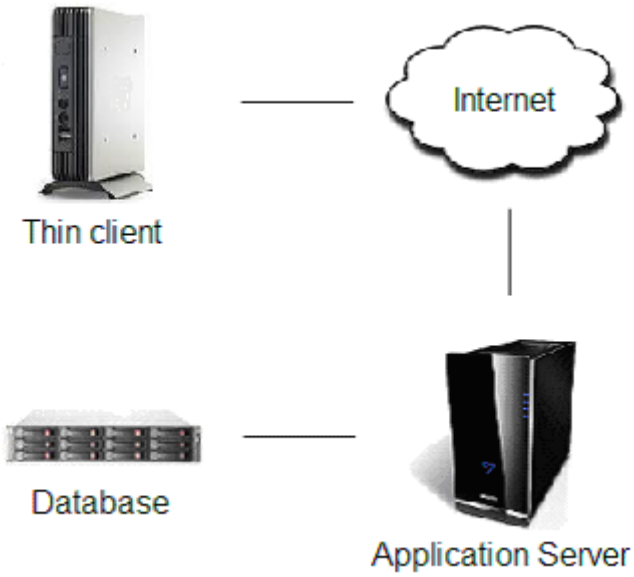


Figure 1.5.2. The image shows the connection of a thin client that uses a Wide Area Network to reach the server.

In a multi-user environment, the client is often a thin client, connected to a server through the Internet (e.g. WAN to an Application server). In this case, a user on a thin client uses the application to send queries to the application server through the WAN connection. The application server receives the query and retrieves the data from the database manager through a fast network connection (e.g. Gigabit).

A slow Internet connection in a WAN can be a bottleneck compared to the connection between the application server and the database. Therefore the WAN connection will be eliminated in the test but the interesting part of the communication is not based on the performance of the network connection between the client and the server. The interesting part to test is the amount of time the server needs to process a query before sending results back to the client. Therefore, a LAN is a good simulation for testing with more accurate results and a good measurement on how well a database manager performs.

1.5.1 Hardware

As mentioned above, the hardware is not important. To be able to take an individual result into consideration, the hardware of the server and client is as follows.

The server has the following specification:

- Intel Core 2 duo E8400 3.00 GHz
- 2.00 GB DDR2 RAM
- Intel 82566DM Gigabit Network Connection

The client has following specification:

- Intel Pentium M processor 2.13GHz
- 1.00 GB DDR2 RAM
- Broadcom NetXtreme Gigabit Ethernet

1.5.2 Software

The test will be performed using an application created in VDF. The version used to create the application is Visual DataFlex v.11.1.13.0.

To be able to run VDF applications, a runtime (see glossary) needs to be installed. The version of the runtime is Visual DataFlex 11.1.104.0.

The CK that are used are

- Pervasive.SQL Connectivity Kit 5.0.0.39
- ODBC Connectivity Kit 5.0.0.52
- Microsoft SQL Connectivity Kit 5.0.0.52
- ORAFlex 9.1.3.12

The CK that will be used in the test will be the latest CK released when this test is performed. ORAFlex and ODBC are meant to be used for Oracle. The reason for using ODBC with Oracle is that Data Access does not have a CK for Oracle. Pervasive.SQL will also be used with the ODBC driver in the performance test. This is done to be able to compare what happens when switching from a dedicated driver to the ODBC driver. The ODBC connectivity is developed by Data Access and the ORAFlex driver is provided by Mertech Data.

Both server and client are running Windows XP Professional.

1.6 Database Format and Contents

The data that will be used in the test is taken from one of Berendsens' own laundries. All tables and relations between tables are used in their environment and it is not data generated for the test only.

A laundry owns articles that are rented or sold to customers. The articles can be garments or products like carpets. The customers can be companies that are supplying their employees with work clothes or they can be renting carpets placed in the company's facilities. A company can have several addresses where the laundry delivers articles. These articles needs to be washed or if they are damaged in any way they need to be repaired and are therefore sent back to the laundry.

This example covers the larger tables that are used in the system and they are:

- Garments
  It is a table that contains about 190 000 records and 19 different columns for each garment (i.e. id, type, article, size, number of washes).

- DeliveryNotes
  It holds all delivery notes that have been sent to customers. It contains about 870 000 records and 28 columns (i.e. id, a delivery number, amount, route number, status).

- DeliveryRows
  It contains the information belonging to a delivery note. It contains about 1 900 000 records and 18 different columns (i.e. id, row number, type, amount, size, product number).

- Customers
  It contains all customers. For the data used in the test it contains 12 000 records and 72 columns with all kinds of information about each customer including name, address, contact person and status.

- DeliveryAddress
  It stores all different addresses where a customer wants articles delivered to. Companies can have several facilities and they want each article to be delivered to the correct address. DeliveryAddress contains about 20 000 records and 97 columns.

These are tables that are being used in the test. All relations can be viewed in figure 1.5.1 and how they are used will be covered in section 3.4.

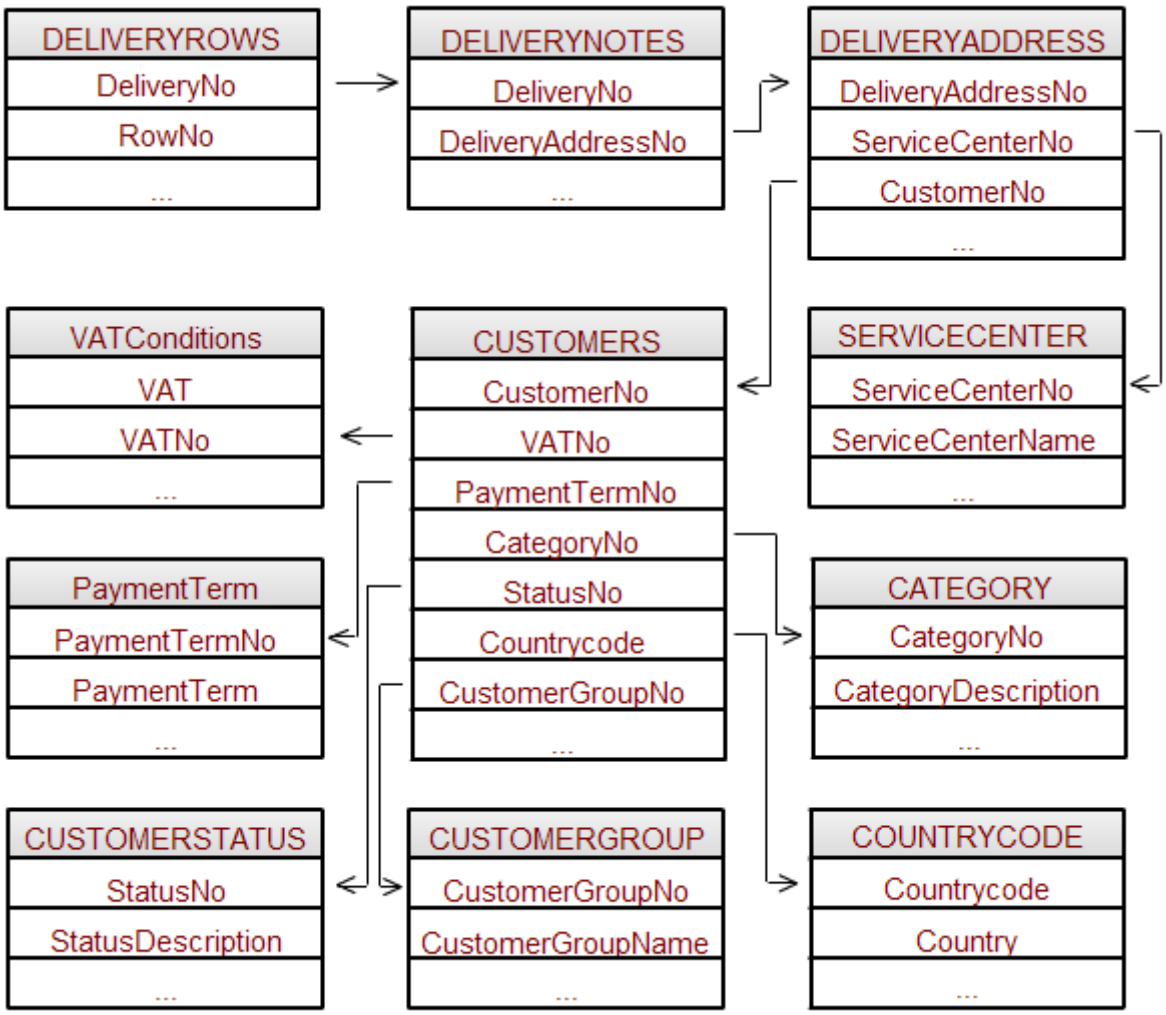


Figure 1.6.1. It shows the tables that are connected to DeliveryNotes and DeliveryRows through relations.

1.7 The Test

The test will contain several different parts. All parts of the test will be evaluated separately and the different parts are:

- Installing the server and client

The installation process should not involve any difficulties.

- Administration and configuration of the server and client

When installed, how hard is the configuration of the server and client. The configuration tools for the database managers might provide different levels of configuration options compared to each other.

- Performance

When the amount of data in the tables grows larger, it gets more and more important that the database manager handles queries efficiently. The test will show how well a dedicated CK performs compared to the ODBC CK. The performance test will be evaluated using time comparisons.

- Migration

Migrating data from one database to another will be tested. To not favour any database manager, the data will start of in the embedded format (ED) and be migrated to the different databases used in the test.

- Backup

Depending on the environment, the backup possibilities might differ. For some environments it might not be possible to do online backups. If online backups are supported, the functionality will be evaluated.

- Handling the database

A database needs a good tool to create tables and views, and to view table data. This test will compare how certain tasks are performed in the different tools. The Database Builder and the Database Explorer will be compared to the tools provided by the database managers since they are meant to be used for the same tasks.

- Transactions

When writing queries using VDF, transactions are used. A developer can create transactions explicitly but if a developer does not, VDF creates a transaction implicitly. A test will be performed to see how different sizes of transactions will affect the performance of different database managers.

The performance test will consist of several tables (see section 1.6) and several smaller tests to determine what a certain database manager is better or worse at compared to another. The performance test will cover:

- Reading an entire table

This test will compare the time it takes for each database manager to read the entire table DeliveryRows record by record into the record buffer using the default index for the table. No change will be made to the record while it is in the buffer.

- Updating an entire table

The table Garments has a column called TimesWashed. It is a numerical value and can hold seven digits. This part of the test will show how long it takes for a database manager to change the value TimesWashed for every record.

- Finding records using an indexed search

All *Find* queries in VDF uses indexed search but when searching an entire table, the index is not as important as when looking for a specific record. This test will try to find specific records using an index. The find mode *equal* will be tested using the table DeliveryNotes.

- Relations/Data Dictionaries

DeliveryNotes has relations to both child and parent tables. When a record in a table is read into the record buffer it is possible to use data dictionaries to retrieve all related records to that record. This test will show how long it takes to retrieve all related records to a record in DeliveryNotes using data dictionaries.

The first step of the testing is to use the default settings in the CK. When that is done, different ways to increase the performance through the CK will be evaluated.

1.8 Fourth Generation Language

Visual DataFlex is a Fourth-Generation Language[5]. Computer languages are currently grouped in five generations. The first generation are machine languages which consist of only binary numbers. The second generation are assembly languages which still only use notations of letters and numbers as syntax.

The third generation are high-level languages (for example C, C++ and Java). Third generation languages are closer to human language in syntax compared to earlier generations and are therefore easier to read and write. Third generation languages came into use in the 1960s and they are processed by compilers that translates the code into assembly or machine languages for a computer to be able to execute the written code.

The fourth generation are languages with syntax that even more resembles spoken languages compared to third generation languages. They also allows for a user to define what a program should do but not how it should be done.

The fifth generation are languages used for artificial intelligence and neural networks.

Examples of other 4GL languages are:

- SheerPower4GL [6]
- PowerBuilder [7]
- SQL [8]
- Matlab [9]

The term 4GL was first used by James Martin [10] in the book *Application Development Without Programmers* [11]. James Martin has written several books on the subject.

- Fourth-generation languages. Vol. I: principles [12]
- Fourth-generation languages. Vol. II: representative 4GLs [13]
- Fourth-generation languages. Vol. III: 4GLs from IBM [14]

Fourth generation languages are, as mentioned above, an even higher abstraction than third-generation languages which are already considered as high-level languages. They are often designed to allow easy ways to create applications or reports that are working towards a database. The programming syntax is often closer to human language which is a part of the higher abstraction model.

Fourth generation languages are developed with the concept to eliminate performance issues from the developer. In many third-generation languages it is important to make sure that code is written in an efficient way and the developer is in charge of the execution time in a different way compared to many 4GL languages. Using SQL it is easy to clarify the meaning of this:

When working with applications that send queries to a database, it is often important that the answer to the query is retrieved fast. In a third-generation language (like C), the developer has lots of responsibility when it comes to execution time. By using pointers in smart ways, the execution time can be decreased to make a function return results faster. This is often not an issue when using a 4GL language because the developer does not have the same possibilities.

When sending a query to a database using SQL, the developer does this with code that does not give away anything about the method of how the query will be handled. When writing code in C, it is easy to see line by line how a certain function will retrieve a result. Earlier in the report, the following SQL query where used to count how many students that have the lastname Johansson exists in the table Students.

```
Select Count (*) from Students where (Lastname = "Johansson")
```

This query does not tell a developer how the result will be retrieved. The developer only gives the database engine input about what kind of result is wanted. The developer does not affect the execution time; instead the database engine will be trusted to return the result of the query in the fastest way possible or at least within reasonable time limits. Instead of a traditional programming language where the developer has the responsibility of the execution time, the developer is given syntax that is clear what it does but not how and it is up to the developer of the database engine to optimize the query.

The article *Fourth generation problems* [17] shows the difference with code examples and also talks about consequences with loosing the ability to control the execution process. *Third-Generation Versus Fourth-Generation Software Development* [16] and *Third and fourth generation language productivity differences* [20] also talks about differences between the two generations.

Third generation languages serve no specific purpose in what kind of applications that should be developed using a certain programming language while the opposite applies to fourth generation languages. Fourth generation are often created with some sort of function in mind, like creating graphical applications (with forms and other data containers). There have been several studies conducted on the difference in using a third level and a fourth level languages for creating the same kind of application. When comparing the third and fourth generation it is often compared by counting lines of code, hours to write and the overall productivity (hours to write compared to execution of the application).

One study conducted at Massey University[22] shows that when using the fourth generation it requires fewer lines of code compared to the third generation and the total amount of hours to write the code also decreased. However, the overall productivity still decreased when taking the number of lines of code, the actual hours for writing the code and the efficiency of the application. Less lines

of code but still more work needed to be done per line compared to the third generation language into consideration.

Another study conducted by *UCLA Graduate School of MIS* [23] also showed fewer lines of code, fewer working hours but in the end, roughly equivalent productivity.

These studies are not easy to interpret when making a decision of using a third or a fourth generation language. When comparing lines of code between the generations, one must keep in mind that a general idea with fourth generation languages is to let the compiler optimize the code which automatically leads to the same functionality requiring fewer lines of code in a fourth generation language. There have been discussions that one line of fourth generation code is equivalent to six lines of third generation code but no such definition have been accepted [22].

Basically there is no easy way to determine whether to use a third or a fourth generation language for a certain task. A lot of factors have to be taken into consideration and that is the resources to put into a project, the knowledge, the aim of the application, the need of controlling the application on all levels and much more. Since the third and fourth generation are not meant for the same task, it is hard to conduct studies with very specific results.

For further reading on the subject:

- In praise of 4GLS [18]
- The impact of fourth generation languages on systems development [19]

As mentioned earlier, there is also a fifth generation and to read about the differences between the fourth- and fifth generation and the possibilities it offers, see *Fourth and fifth generation programming languages. Vol.1* [15]

1.9 Summary

The introduction section has been meant to clarify the different components that will be used in the test. It explains how Visual DataFlex is designed and shows code examples with explanations. A lot of the source code can be found in the appendix and the introduction should make it easier to understand the code for readers without Visual DataFlex experience.

2 DATABASE MANAGERS AND CONFIGURATION TOOLS

The starting point of the test is installing the database managers. This process is expected to be easy and therefore all steps in the process will not be covered. If anything unexpected appears in the process it will be mentioned. This applies to installing the database managers and the connectivity kits (CK).

When using a database it is important that it provides useful tools for administration of the database. Creating tables, adding rows, adding columns, etc. should be easy to do. Since this is assumed to be easy to do with the tools that are provided with the database managers, a test will be performed to make sure that they provide the necessary functionality in an easy way.

Situations may appear where a migration between different databases is required. Therefore a test of migrating data between the different database managers will be performed.

2.0.1 Order Entry

VDF has an example application included when installing the development environment. The example is called Order Entry and will be used when testing migrating data and the functionality of the configuration tools. The reason for using this example is to have a starting point that does not favour any of the database managers in the test since the Order Entry system is in the Embedded Database format.

The tables and the relations in the Order Entry system are:

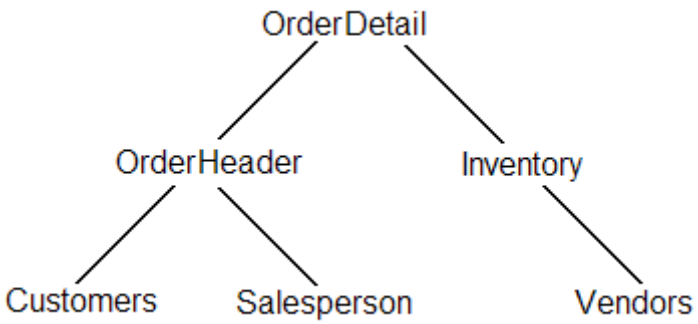


Figure 2.0.1. This shows the tables and relations of the Order Entry example.

The Order Entry is an application that involves vendors who delivers items that are put in the inventory and customers can order items through a salesperson. A company has customers who buy items and employees that sell them. They are put in the tables Customer and Salesperson. The company has an inventory where the items are stored and the items are bought from a vendor. When items are bought by a customer, an order is created and stored in the table OrderHeader. OrderHeader stores who made the order and who sold it, the date the sale went through, shipping information and the total cost of the order. Details of each item that belongs to the order is stored in the table OrderDetail such as which item was ordered, quantity and the price of the item.

The first step is to convert the data from the embedded format into the format of each database manager. This will be done using the Database Builder and the CK installed for the database manager.

Before converting the data a descending index will be created. A descending index is sorted in inverse order compared to a normal index (see glossary for more information). The reason for doing this is that VDF supports descending indices instead of emulating a descending search by returning a result set in inversed order that normally is the case with SQL engines. The descending index will be created in the Inventory table.

Using tools provided for each database manager, a few simple tasks will be performed. The tables OrderHeader and Inventory will be used for this test.

The first test is to find out how many records OrderHeader has. OrderHeader has a column called Order_Date which stores the date an order has been made. Test number two will be to put a filter on the table and find out how many orders that where made 2004-05-10. The date is chosen based on the data that already exists in the table.

The last step of this test is to add an index. The index will be added to the table Inventory and will consist of three columns and one of the columns will be set to sort in descending order.

These examples will provide testing of:

- Viewing table properties
- Searching a table
- Modifying a table

These are tasks that should be easy to perform using graphical tools.

2.1 Pervasive.SQL v10.10

The installation process starts of by letting the user decide whether to install the server or client of Pervasive.SQL and whether to use the 32 or 64 bit version. In this case the 32 bit server version is installed. No surprises appear when installing the server or the CK.

To handle databases and the data in the tables, the application Pervasive Control Center (PCC) is used and it is installed automatically when the server is installed.

Converting from the ED to Pervasive.SQL did not present any problems. The descending index did not give any problems with the default settings of the CK and the recnum columns are working as intended after the migration. If recnum support is not needed, converting from recnum tables to standard tables is easily done.

When opening a table, a text area and a grid appears. The grid contains the data in the table and the text area is an editor for performing SQL queries on the table. Using SQL queries, the answer to how many orders where made on a certain date can be answered. The number of records in a table is easily found by viewing properties on a table or by a select query in the text area but no graphical filtering function exists.

Viewing indices in the PCC is meant to be done by right-clicking on a table and viewing properties and choosing the index tab page. In the table properties section, it says that the table Inventory has four indices which are correct but when viewing the index page it shows an empty grid. It is the same for all other tables as well so this is a bug.

Creating a new index can be done using the query editor or by choosing the task *Edit Tabledesign*. That can also be used to view indices since the index tab page does not work as it should.

2.2 Microsoft SQL Server 2005 Enterprise Edition

The installation process begins with a system configuration analysis to evaluate if all necessary components to run Microsoft SQL Server are correctly installed. If the configuration fails, either the installer will install the missing components automatically or it will tell the user which components are missing and how to fix it.

With the exception of the system configuration analysis, the only thing to mention about the installation process is choosing which type of security to use for the server. A user can choose to use Windows Authentication, SQL authentication or both. Windows authentication means logging in with the Windows account currently logged in and SQL authentication means username and password. Choosing to log in using Windows Authentication is meant to be combined with Active Directory, which is a way to manage users and user groups. [21]

The configuration tools that are installed provide the necessary basic configurations to handle a database. Some of these settings are setting the data path, configuring error handling and log messages and setting domain name. The configuration tool lets the user edit the basic settings of the server. To be able to set up databases and handle the data in the database (like tables, views and indices) the SQL Server Management Studio will be installed. The version of the Management Studio is 9.00.1099.00. The CK was installed without issues.

Converting the Order Entry system did not give any problems. Recnum is supported and handled after converting the data. Converting recnum tables to standard tables is an option that is given when converting and it removes the recnum columns from converted tables.

The Management Studio is used to test the basic functionality and the results are as expected except for the possibility to create a filter on a table. It is possible to count the number of orders on a certain date using a query editor but not through a graphical tool.

2.3 Oracle 11g Enterprise Edition

As mentioned in the introduction, Oracle will be tested with both the ODBC driver and the dedicated driver (ORAFlex).

The installation process of Oracle 11g went as expected. The process could be performed using a simple or an advanced installation. The advanced installation lets the user set memory management, security (enabled or disabled), character settings, automatic backup options. Installing the CK also went as expected.

2.3.1 ODBC

The migration of the Order Entry system from the ED to Oracle passed successfully according to the Database Builder although a few problems appeared when viewing the data in the tables.

All tables with decimal columns are migrated successfully if all values in the column are integers. All values in the Order Entry system are integers so converting all records worked. However if any record in a numerical column is set to a non integer value, the record will be dropped after the migration. Dropping a record cannot be accepted when converting data. The log file does not give the user the information needed to be able to recreate the records.

The problem with converting numerical values is that the settings for decimal separators are not the same in the ODBC driver and Oracle. The CK has a setting for decimal commas that is based on the

regional settings of the operating system. Oracle is by default set to use U.S. comma settings which is not the regional settings for the operating system. A solution is to set the ODBC to also use U.S. settings. Before changing this setting it is only possible to add new records that have integer values.

Each table in Order Entry has a recnum column and it is converted successfully from the embedded format to Oracle but Oracle does not have recnum support which leads to problems when creating new records. Since Oracle does not have recnum support, each table needs to be modified for it to work. Each table needs a trigger and a sequence (see glossary for more information about triggers and sequences).

Creating the trigger and the sequence is done using either an SQL editor or the graphical tools provided with Oracle. The trigger is five lines of code and through the Control Center it is possible to use a graphical interface to create all rows except for the select statement that is the action to perform when the trigger is triggered.

All tables have a property in the CK called *Generate Record ID Method* and it is default set to *None*. When it is set to *None*, the trigger is not allowed to perform the task it is meant to perform and the recnum support is still not functional. The property has to be set to *External* to allow the trigger to handle the recnum column. Creating the trigger, the sequence and changing the property has to be done for all tables that are converted or later created.

Before the data is migrated, the option is given to convert recnum tables into standard tables. This removes the column recnum from all tables and when tested it was successful so the migration process provides an easy way to eliminate recnum when converting.

The descending index failed during migration and the problem is also the default settings in the CK. The default setting is that descending index is not supported and it has to be changed in order for it to work.

2.3.2 ORAFlex

The ORAFlex driver is provided with its own application for connecting to databases and handling migrations. The application is called FLEX2SQL and since it is a different application it also provides a different configuration setup.

A lot of the configuration settings that can be used in this application can also be set for the ODBC driver but in FLEX2SQL it is possible to change settings using graphical tools. FLEX2SQL still has more settings and the most valuable when converting to Oracle is the recnum support and it also handles the decimal comma setting by default. The application also offers a conversion from recnum tables to standard tables.

As mentioned earlier, a user has to give each table recnum support by using a sequence and a trigger. When using FLEX2SQL and enabling the setting *Use recnum*, this is handled automatically for each table that is converted. It is also supported when creating new tables.

Descending index also works after converting the data although it is a setting that has to be set and it is not handled by default.

2.3.3 Configuration Tools

Oracle has several tools provided when installed. For the basic test two of them will be used. The first one is called Database Control. It is a web interface that allows for normal users or administrators to log in and view or edit information about a database, set user privileges, create and delete users, etc. It is possible to view status and statistics of a database to make sure that everything runs as intended. The interface also allows a user to view tables created in different schemas (see glossary) belonging to the database and can view columns, tables, triggers and procedures that are created. Database Control is a good tool for monitoring a database and handling users but for creating tables, triggers, and adding data to tables, the utility SQL Developer will be used.

SQL Developer is a utility which is a part of the installation of Oracle. It is a Java application and requires that a Java Software Development Kit (SDK) is installed. Database Control can perform all tasks that the SQL Developer can but it is easier to use SQL Developer for the basic tasks that are being tested. The basic functionality is easily performed with the SQL Developer.

Oracle showed interesting behaviour in two ways. The first thing was that it took a noticeable amount of time to open a table without cache files. All database managers create cache files for all tables to open them faster later but when using Oracle it takes an unreasonable amount of time to create the cache files.

The second behaviour lead to Oracle having to be reinstalled. The problem was that the server had no ip-address during the installation which lead into Oracle making localhost the target for the database created in the installation process. All configuration files made localhost the target for the database instance created. When the server later got an ip-address the structure of the files changed without setting all configuration settings to the new ip-address.

A summary of this problem is that the server is expected not to be changed. The ip-address that the server has when creating the first database is assumed to be a static ip. This problem is probably not only related to the ip-address but also the name of the network. It is assumed not to be changed. It also takes a long time to launch the Oracle database compared to Pervasive.SQL and Microsoft SQL. Servers are rarely rebooted but when they are, Oracle is a lot slower to start than Pervasive.SQL and Microsoft SQL.

2.4 Visual DataFlex Tools

The Database Builder and Database Explorer (see section 1.2.2) are tested in the same way as each database manager.

The Database Explorer is used to count the number of records in the table OrderHeader. To find how many orders were made on a certain date, a graphical filtering interface is used. Putting the filter on the table is very easy although the filtering interface turned out to have bugs. When switching between columns to put a filter on, the search area needs to accept different input lengths of text depending on the data type of the column. Switching between the data types does not properly set the input data length and a user might end up with a date type allowing a maximum input of one character to give an example. Recreating the filter solves the problem.

Adding and removing indices is done using the Database Builder and is performed using a helpful graphical view. The tools provided with VDF handles the basic tasks impressively easy but the filtering interface needs an upgrade.

2.5 Summary

Before being able to run a performance test, all database managers have to be installed and configured properly. The data to use in the performance test have to be converted for each database manager. This has been the first part of the test.

The installation process of the database managers did not involve any problems for any of the database managers. It is running an executable file and following instructions. Installing the CK is basically the same process. After installed, Oracle had to be reinstalled later because the configuration files changed when the ip-address changed.

Configuring the databases to be able to convert data from the embedded format to each database managers own format turned out to involve a few problems when converting to Oracle using ODBC. Since Oracle does not have recnum support, a trigger and a sequence have to be manually created for each table. Converting to Pervasive.SQL and Microsoft SQL gave no problems.

The default decimal comma settings did not handle the data converted either. When using the ORAFlex driver to convert the data to Oracle the recnum columns are handled by the driver.

When converting to Oracle using either the ODBC or the ORAFlex driver, the user has to make sure that all settings are correctly set before converting. This is because of default settings being set to disable for example recnum and descending index.

The tools provided with each database manager did not live up to the expectations that performing basic tasks should be easy. It is easy if the user is comfortable with SQL-query editors but graphical tools should be provided for tasks that are performed often. One example is putting filters on tables. All three managers have tools that lets a user perform SQL queries manually on tables but Oracle was the only of the three that had graphical tools for each basic task that where tested.

The biggest surprise of the first test is that the Database Builder and Database Explorer are better tools than those that the database managers provide. It is easier to put filters on tables and there are graphical tools for everything that they can handle.

3 PERFORMANCE TEST

The test will produce a lot of measuring data. All tests belonging to this section will be evaluated using milliseconds or seconds. All results are presented in the appendix in milliseconds but some are showed in seconds in the performance test depending on the time it took to perform the test. All data that where used to create the diagrams or in other ways evaluate a test are presented in the appendix.

All database managers have their own directory where the data is stored. To avoid performance loss due to fragmentation, the disc where defragmented before the test.

The database managers are optimized for making efficient searches in the data. The database managers optimize queries differently and might also use cache to find certain data faster. Therefore, each record in the tables used in the test will be read before performing the test queries to give the database managers an opportunity to use cache and query optimization.

Each test will run either seven or twelve times depending on the test. The highest and lowest values will be discarded and the remaining either five or ten values will be used when evaluating the results. The reason for discarding the highest and the lowest values is to even out the measurements that might deviate from the rest.

The code that is written for each test can be found in the appendix.

3.1 Looping a Table

The first test is to read all records in a table. This will show how efficient each database manager is when a VDF application and the record buffer are used. The table that will be used is DeliveryRows and contains around 1.9 million records.

This test uses only a single client reading from the database and with the default settings for each connectivity kit (CK).

This test uses the find mode *greater than* and that means that a block size cursor is used (see section 1.3). The block size cursor is meant to make a more efficient search when using set based engines since they are not using a record buffer in the same way that a row based engine does. All CK except for the ones used with Pervasive.SQL and the ED have a setting to change properties for the block cursor and they will be tested to see if the default setting is the most efficient setting or if it can be altered to get more efficient queries.

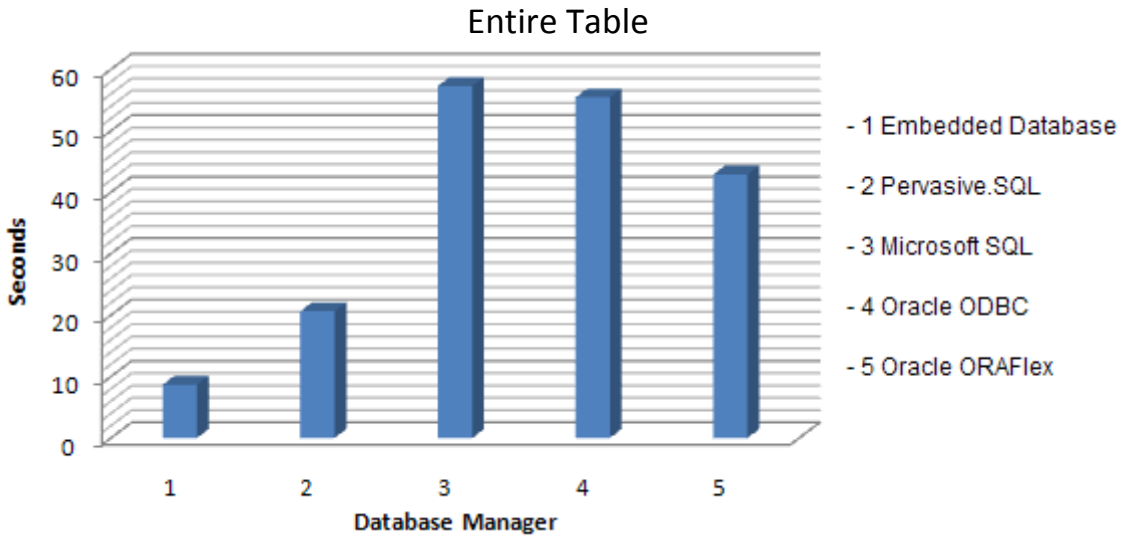


Figure 3.1.1. This is the result of reading the table DeliveryRows and reading all records into the record buffer for each database manager.

The diagram shows big differences between the database managers. Pervasive.SQL and the Embedded Database results shows the difference in working record based against working set based. The Embedded Database is a lot faster than Pervasive.SQL since the Embedded Database is very fast when only a single client is used. The structure of the Embedded Database allows for this to be executed fast (see section 1.2.3).

The set based managers differ a bit from each other but the difference clearly shows the advantage in using a row based engine when a VDF application is used. This is with the default settings of the CK. There are settings that can help to make the set based managers to perform this task faster and that is covered in the next section.

3.1.1 Connectivity Kit Configuration

Microsoft SQL will be the first database manager evaluated when trying to alter the settings in the CK. There are two settings in the Microsoft SQL CK that are interesting to test and they are the cache timeout and the block size properties.

When altering these settings it is important to first analyze what they are doing and what this can lead to. The block size is used to set how many records that should be fetched for each access to the database. Since Microsoft SQL is set based, the CK can fetch more than one record each time it accesses the database (see section 1.3). This property can be set for each table individually.

The cache timeout property is used to clear the cache of the block cursor after the given number of milliseconds and force a new database access for the next find operation. This property is the same for all tables.

Using a longer cache timeout enables for using a larger block size but if the data in a table is updated frequently, a large cache size might lead to not using the newest added or modified data. Setting a short cache timeout means that data will be fetched more often from the database instead of from memory but it also leads to the block size property having to be reduced. Otherwise there might be loss in performance due to reading records into the memory that are discarded and have to be fetched from the database again.

The goal is to find a balance between these two properties. Here are some examples that should show how the timeout and the block size are dependent of each other.

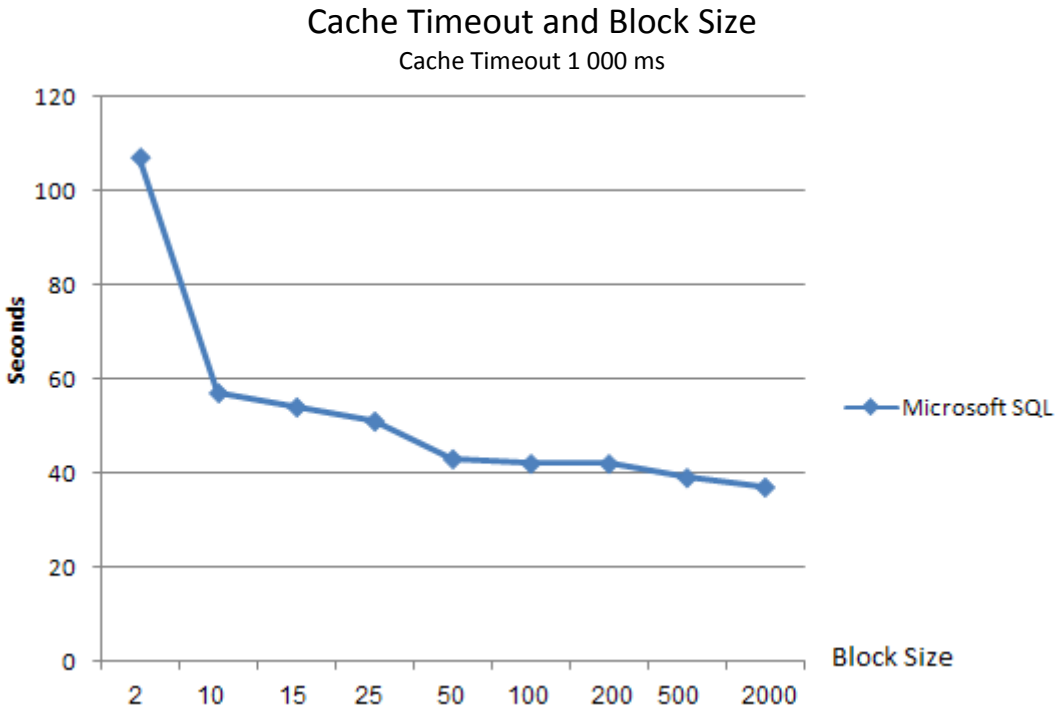


Figure 3.1.2. The diagram shows the difference in reading the entire table DeliveryRows for different block sizes and with a cache timeout of 1 000 ms.

In figure 3.1.2 the block size is set to values that are small enough for records to be processed without the cache timeout being reached. This is to show that editing the block size can increase the

performance significantly as long as the timeout value is long enough for all records to be processed before having to retrieve new records from the database.

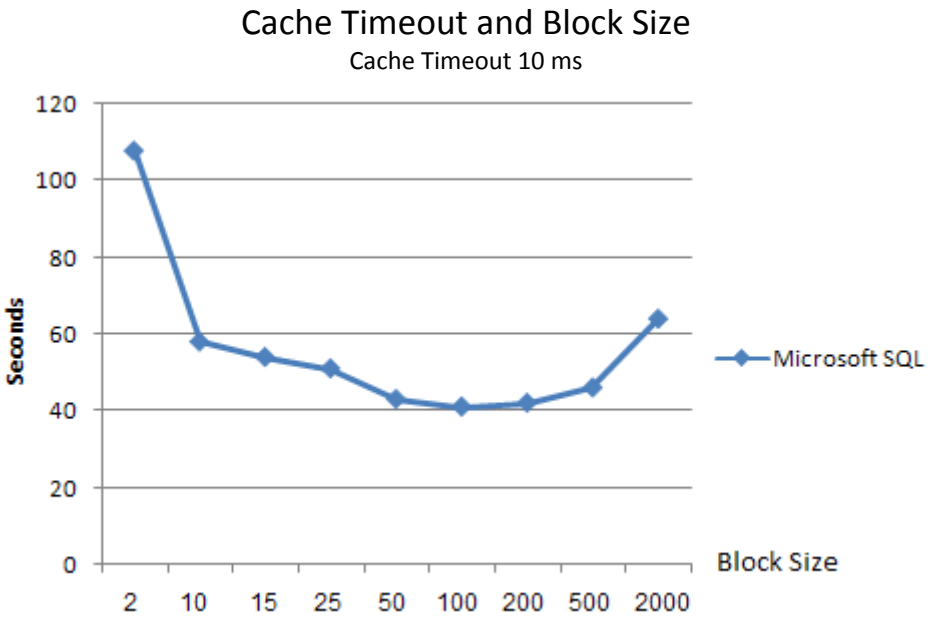


Figure 3.1.3. The diagram shows the difference in reading the entire table DeliveryRows for different block sizes and with a cache timeout of 10 ms.

Figure 3.1.3 shows what happens if the block size and the timeout are not balanced. The diagram shows that with a timeout of ten milliseconds and a block size of between 2 and 100, the performance increases with the block size since the records that are fetched are processed before the cache is discarded. When the block size is set to 200 and above, all records that are fetched from the database are not processed before the timeout is reached and therefore they are discarded before being used. Reading records more than once leads to overhead and that is what happens if the cache timeout and the block size are not in balance.

Finding the balance is hard since the same timeout value is applied to all tables. For a table that is not updated frequently, a longer timeout value could be accepted but since the same timeout is used by all other tables as well, the shortest acceptable timeout have to be applied to all tables.

When using a cache timeout of 30 seconds and setting the block size for the table DeliveryRows to 100 000, Microsoft SQL reads the entire table in about 15 seconds. This is just to show what is possible to do when using the Microsoft SQL CK. Using a timeout of 30 seconds is not realistic in most applications but it can definitely be increased from the default setting of ten milliseconds.

3.1.2 Oracle with the ODBC Driver

Oracle with the ODBC driver uses the timeout value and the block size in the same way that Microsoft SQL does. It is a lot harder to find the same pattern that Microsoft SQL showed with a timeout of one second. A timeout of neither one nor five seconds gives the performance boost that was expected. Since Oracle and the ODBC driver does not perform as well as the other database managers, it needs a longer timeout value to process the same amount of data. Since five seconds is not enough to get the expected behaviour, the ODBC driver is not suitable for this kind of optimization of the block size. A timeout of five seconds or more is unlikely to be acceptable in most applications.

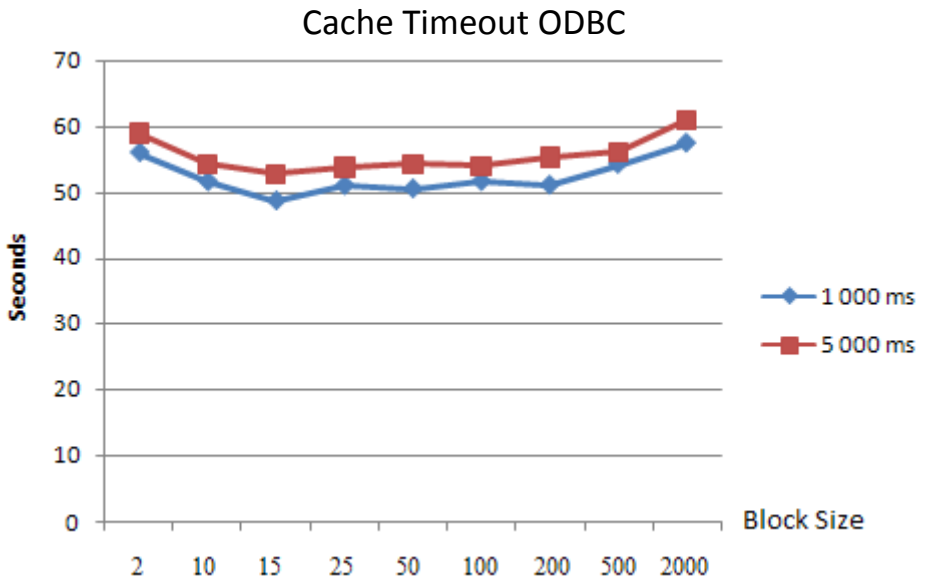


Figure 3.1.4. The diagram shows the difference in reading the entire table DeliveryRows for different block sizes.

It does not seem to be possible to get the same performance boost with the ODBC driver that Microsoft SQL got by changing the timeout and the block size.

3.1.3 Oracle with the Dedicated Driver

The test above shows the results of using the default settings of the CK and altering of the timeout and block size for Microsoft SQL and Oracle with the ODBC driver. Oracles dedicated driver (ORAFlex) has options that the other CK do not have. It has the option to use the block cursor to fetch multiple records but the cache has no timeout setting. When a find is performed, a number of records are fetched from the database and stored in memory. When another find is performed the cached section will be searched first and if the record is not there, it will be fetched from the database. A query can be set to bypass the cache if it is important that the most recent data has to be used. Not having the ability to control the timeout can be dangerous and one way to avoid this is to always fetch records from the database instead of using the cache which decreases the performance

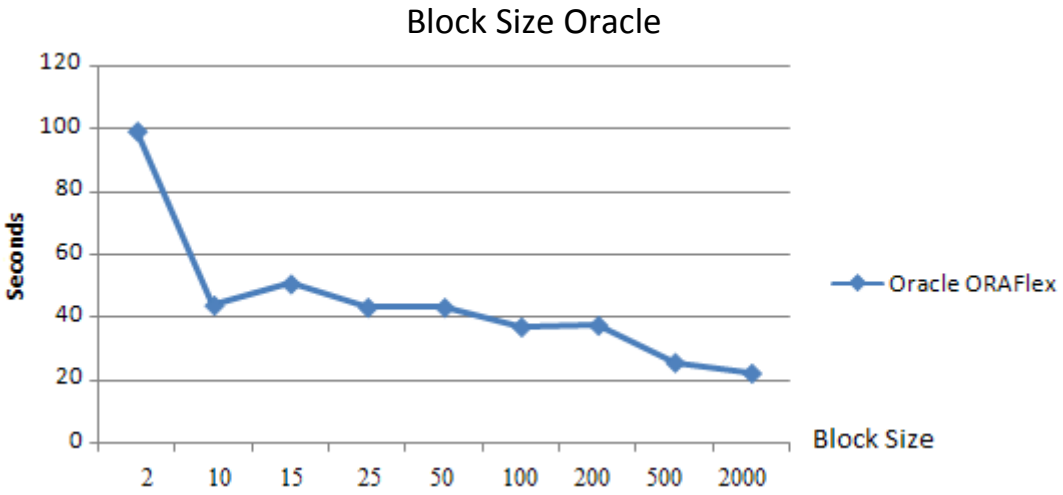


Figure 3.1.5. The diagram shows the difference in reading the entire table DeliveryRows for different block sizes.

Oracle reads the entire table DeliveryRows in about 20 seconds with a block size of 2 000. This shows that it is possible to increase the performance but a large block size without having a timeout of the cache is unacceptable.

The ORAFlex driver offers an interesting option that the other CK do not. Since VDF queries are row based, it is possible to set individual tables to be optimized for either row or set based queries. It can be set before migrating tables but it can also be dynamically changed during execution of a program. Since it is possible to change it dynamically, some functions can be set to use it and others can use the default set based behaviour.

This is just a setting in the CK; it is still Oracles set based engine that executes the queries. It cannot be compared with the row based engine that Pervasive.SQL uses. It will however eliminate the problem with overhead data when using block cursors.

Setting the driver to use a row based find for reading the entire table DeliveryRows does not increase the performance compared to reading the entire table with a block size of ten records. Compared to using the block size cursor this guarantees that the most recently added data is fetched from the database and it does this in the same amount of time.

3.2 Indexed Search

Indexed search can be done in many different ways. The idea here is to test the efficiency when using the find mode *equal*. *Equal* does not use the block size cursor, instead it uses the single row cursor (see section 1.3) and the find mode only works for unique indices.

The table that will be used for this is DeliveryNotes. The table has a column with a unique id for each record and an index consisting of that column. That index will be used and the test will measure how long it takes for each database manager to perform 10 000 *Find equal* operations to determine if a record exists or not. Of 10 000 Finds, only about 100 searches will not be found so most of the *equal* operations will find a record and read it into the record buffer.

Figure 3.2.1. The diagram shows the performance when performing 10 000 *Find equal* operations.

This test also shows big differences between the different environments. The ED and Pervasive.SQL have the advantage of working record based which clearly is an advantage with the find mode *equal*.

Oracle with the ORAFlex driver, configured as set based, and Microsoft SQL performs equally well. Just finding a specific record using a unique index is probably performed equally fast by all database managers but converting between the set based and row based environment leads to performance loss for Oracle and Microsoft SQL.

Oracle with the ODBC driver looses even more performance since it has a more general interface, compared to the ORAFlex and Microsoft SQL drivers that are optimized for converting for each database.

When using ORAFlex and setting DeliveryNotes to find by row instead of by set, the performance increased with about ten percent.

The single row cursor is used for this test which means that there are no cursor settings to test to make the performance improve.

3.3 Transactions

Transaction management is an important part of a database manager and when using applications written in VDF transactions are required. This means that if the user is not explicitly creating transactions when updating tables, VDF creates them implicitly.

This test will show how different transaction sizes affect the time it takes to change values in a table. The table is Garments and contains about 190 000 records. A numerical column will be updated for each record in the table with different transaction sizes.

Creating transaction blocks with the ODBC driver will not be evaluated in this test. The reason is that when using a transaction for every record it works but when increasing the size of the transactions, an error saying fetched out of sequence appears and the table is not updated. No simple workaround was found and therefore Oracle with the ODBC driver will not be evaluated. Updating the table Garments with transactions for each record with the ODBC driver takes around 14 minutes to be able to compare with the other database managers.

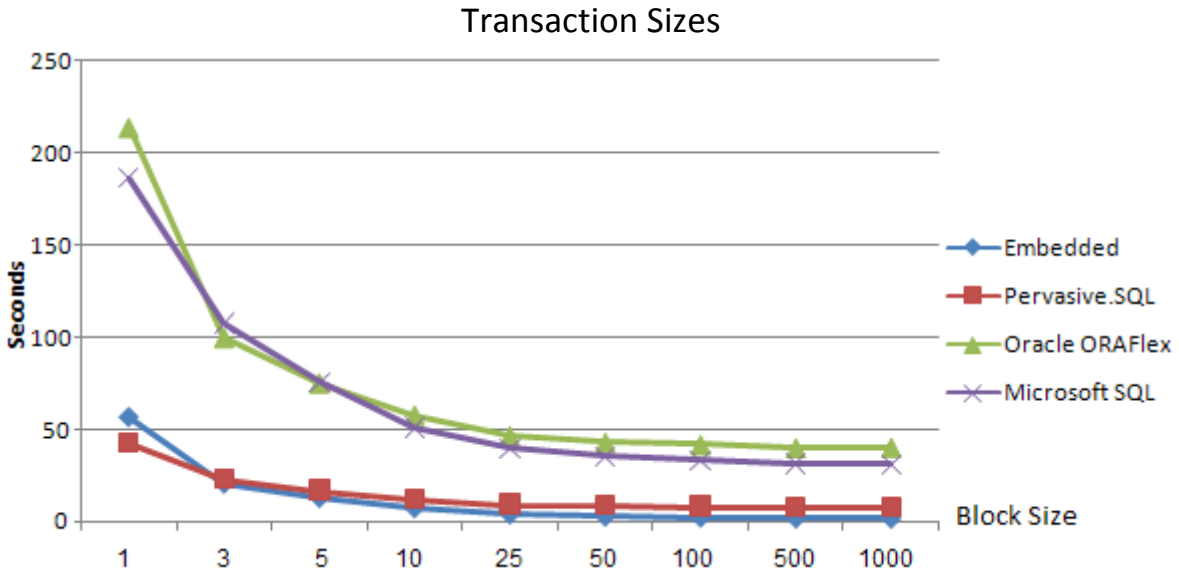


Figure 3.3.1. The diagram shows the performance when using transaction blocks.

All the database managers show the same behaviour. Creating a transaction for each record leads to significantly longer execution time and increasing the transaction size dramatically improves the performance. Using small transaction sizes leads to creating many transactions which gives more overhead compared to using large transaction sizes and the difference is shown in the diagram.

This test also shows the difference between using set based and row based engines when a VDF application is used. Pervasive.SQL and the ED are performing the task faster than Microsoft SQL and Oracle.

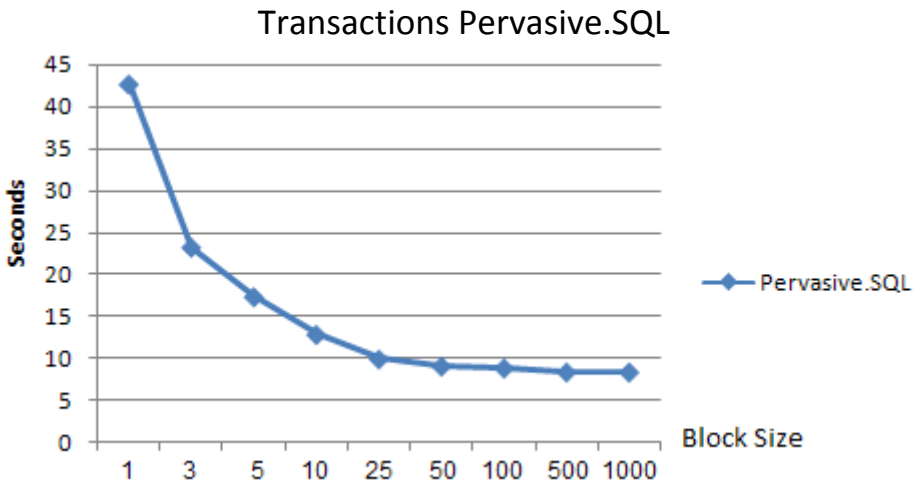


Figure 3.3.2. The diagram shows performance when updating TimesWashed for all records in the table Garments transactions.

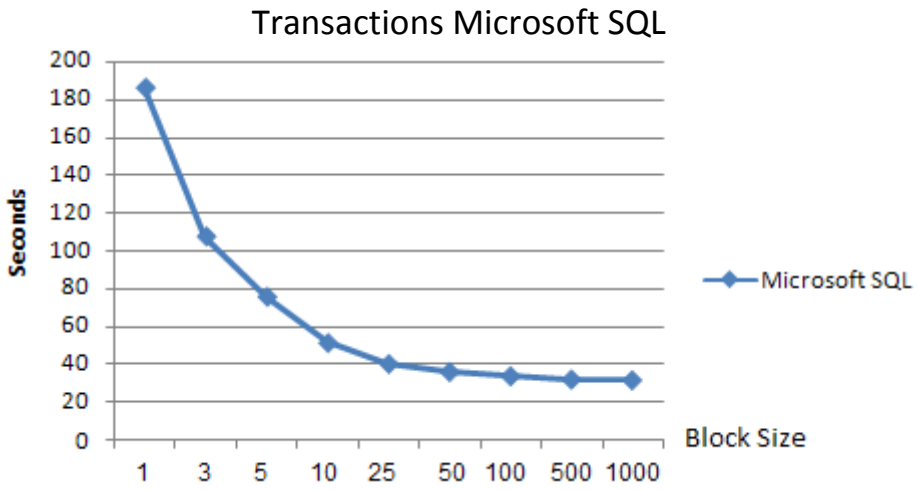


Figure 3.3.3. The diagram shows performance when updating TimesWashed for all records in the table Garments transactions

The improvement in performance is huge and it is easy to see where larger transaction block sizes do not increase the performance any more. Note that the scale of the y-axis is different in the diagrams.

Pervasive.SQL updates all records in about seven seconds when creating transaction blocks in a VDF application and using a CK to connect to the database. When performing the same task from Pervasive Control Center it is done faster but the difference is only about one second. This shows that connecting Pervasive.SQL to a VDF application using a CK does not decrease the performance a lot.

Microsoft SQL updates all records in the table Garments in about 30 seconds when using a large transaction size, a VDF application and a CK to connect to the database. When using the Management Studio the same task is performed in about three seconds. This shows the performance loss when connecting a set based engine to a VDF application through a CK compared to a row based engine.

This is meant to show the difference between a set based and a row based database engine and the difference when using a VDF application and no application at all. It is not meant to compare Microsoft SQL with Pervasive.SQL. A similar result as Microsoft SQL had is expected if the same test is performed with Oracle but that is not necessary in this case.

Pervasive.SQL and the ED are performing the task almost equally fast but this is when using a single client.

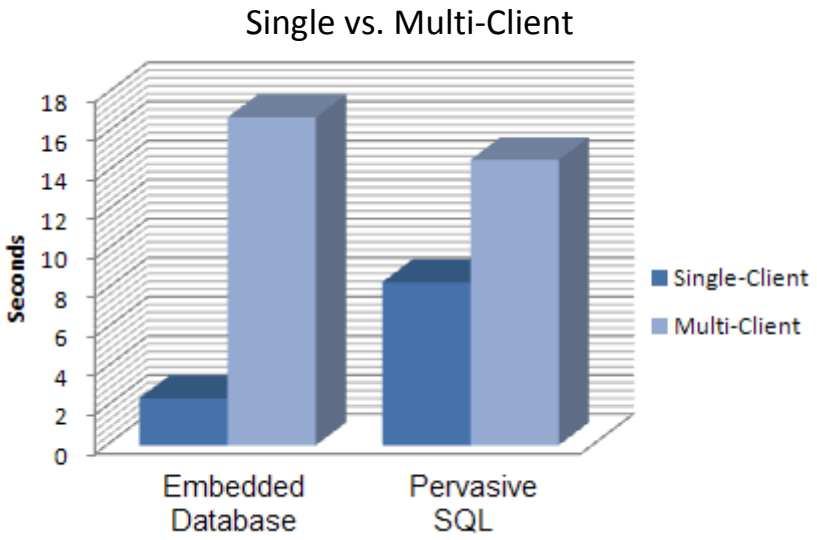


Figure 3.3.4. The diagram shows the performance when updating the table Garments with a transaction size of 1 000 records.

The difference between using multiple clients simultaneously with the ED compared to Pervasive.SQL is shown in the diagram. The performance loss is significantly larger when using multiple clients and the ED.

When using Pervasive.SQL with a single client, the time to update the entire table Garments is done in around eight seconds. The same task with a single client is performed by the ED in about two seconds. When connecting several clients to perform this task simultaneously, the database managers handle the queries differently from each other.

Pervasive.SQL uses record locking or page locking. Record locking is to lock only the record that are being updated and page locking is locking a certain number of records including the record to be updated to avoid locking the entire table. This leads to more clients being able to write to the same table simultaneously since no client locks the entire table.

The ED uses file locking and files are locked randomly. This means that all clients can ask to lock the same file and the server lets the first client lock the file regardless of the query. This leads to the other clients having to wait while the first client finishes writing to the table.

The results of the differences are that when using three clients and Pervasive.SQL, the clients performs the same task in between 13 to 15 seconds. With the ED one client performs the task in a couple of seconds and another client needs 26 seconds. The result for the ED is more spread since the client locks the file leading to other clients having to wait. See the appendix for more information about each client's execution times.

3.4 Relations and Data Dictionaries

The table DeliveryNotes has both parent and child tables (see section 1.6). When a record is loaded by the data dictionary for the table DeliveryNotes, the data dictionaries for the parent tables loads the related records.

This test will show the difference in loading related records by the different database managers. The single row cursor is used when finding records in DeliveryNotes so the cache timeout and block size property cannot be used to affect the result.

The test will read 1 000 records into the record buffer for the table DeliveryNotes using the find mode *equal*. For each record, all records that are parent tables to DeliveryNotes will be read into each record buffer.

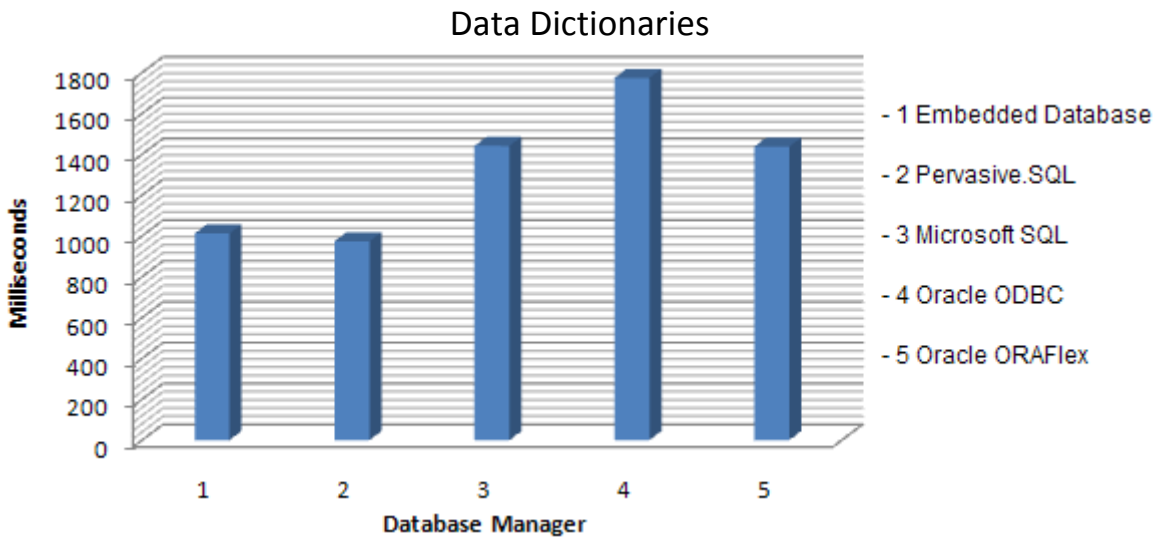


Figure 3.4.1. The diagram shows the performance when finding 1 000 records in the table DeliveryNotes and loading each related record by each data dictionary.

Since the ED and Pervasive.SQL uses the row based engine, they are performing this task faster than the set based database managers. Microsoft SQL and Oracle with the dedicated drivers are about equally fast and Oracle with the ODBC driver is a bit behind them in performance.

3.5 Pervasive.SQL with the ODBC Driver

Since Oracle is being tested with both the ODBC driver and the dedicated driver, another test was performed to show the performance when using Pervasive.SQL and the ODBC driver. This test will show the differences in performance loss when using a set based engine and a row based engine with the ODBC driver.

The ODBC driver is designed to be used with set based engines. Since Pervasive.SQL uses a row based engine, queries needs to be converted several times between a row and a set based environment.

The diagrams below show Pervasive.SQL with the dedicated driver and the ODBC driver. For comparison, Oracle with the dedicated driver and the ODBC driver will be included in the diagrams. That way it is easy to compare what kind of loss each database manager gets when switching from the dedicated to the ODBC driver.

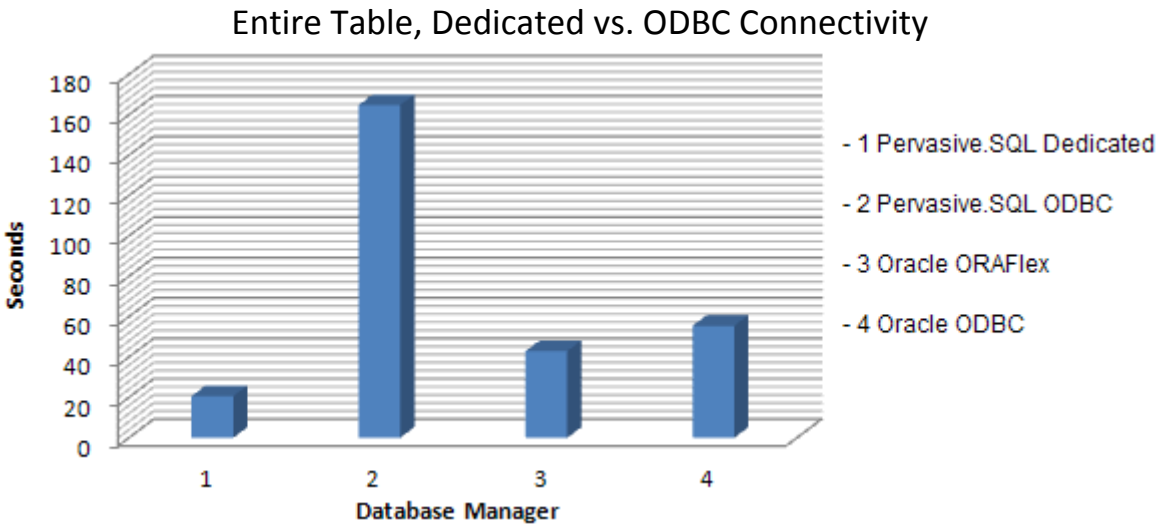


Figure 3.5.1. The diagram shows the performance when reading the entire table DeliveryRows.

There is a difference in running the dedicated driver versus the ODBC driver but in the case of using Pervasive.SQL with the ODBC driver, there is a larger difference than it is when using Oracle.

The diagram shows the performance when using the default settings of the CK. Since Pervasive.SQL uses a row based engine, it might not use the cache timeout and the block size properties efficiently and that leads to performance loss. These properties are meant to increase the efficiency of set based engines so for Pervasive.SQL with ODBC it only gives more overhead. The ODBC driver only allows block sizes of at least size two so when using Pervasive.SQL and the row based engine it leads to more overhead.

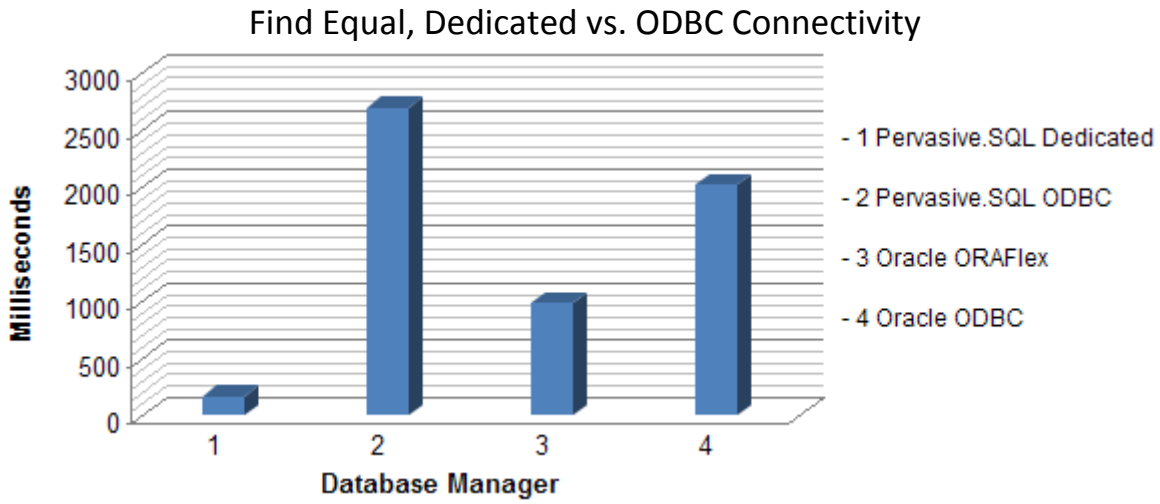


Figure 3.5.2. The diagram shows the performance performing 10 000 *Find equal* operations in the table Garments.

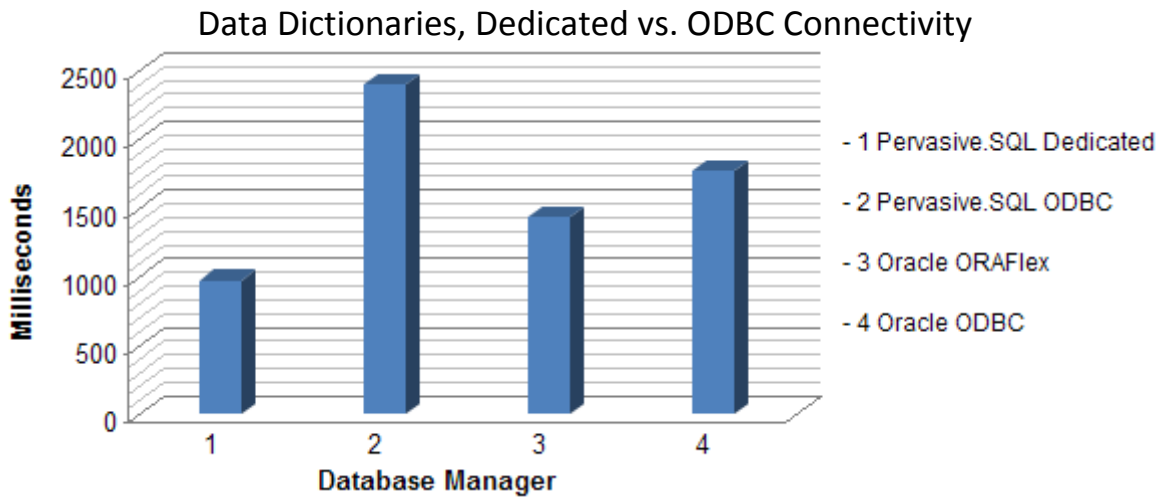


Figure 3.5.3. The diagram shows the performance when reading 1 000 records in the table DeliveryNotes and reading all related records into each tables record buffer.

All three diagrams show the same results. Both database managers loose performance when switching from the dedicated driver to the ODBC driver but that is expected since the ODBC has a more general interface than the dedicated drivers.

Pervasive.SQL with the row based engine looses a lot more performance than the set based engines does.

3.6 Summary

The performance test has shown the advantage of using a row based engine when working with applications created using VDF. The ED and Pervasive.SQL with the dedicated driver did perform significantly better than both Microsoft SQL and Oracle. The ED is very fast when only one client is working against the database. When more than one client tries to access a table at the same time the performance decreases significantly.

Pervasive.SQL with the row based engine is fast with the default settings. It is not possible to optimize Pervasive.SQL in the same way as the other database managers but it is not as necessary to do so either. Pervasive.SQL is faster in all tests compared to the set based engines, and to the ED when using several clients. The row based engine performs each find operation on the database instead of a cached segment in the memory. This guarantees that the data loaded in the record buffer is the most recently added or modified data.

A drawback with Pervasive.SQL is that when using the ODBC driver the performance decreases a lot. Since Pervasive.SQL uses the row based engine, it is not possible to optimize the performance when using the ODBC driver since it is meant to optimize set based engines.

Microsoft SQL is one of the set based managers in the test. With the default settings in the CK the performance compared to the row based managers is lower. Although, depending on the application, it is possible to increase the performance a lot by altering the settings in the CK. Doing this increase the performance but it has consequences. When increasing the performance, the CK fetches multiple records from the database and stores it in the memory for faster access if another find is performed. Depending on the timeout value of the cache, the data in the memory might not be the most recently added or modified data. If a high timeout value is used, the probability to use inaccurate data increases.

Oracle was tested with both the ODBC driver and the dedicated driver. Oracle uses a set based engine as well so the performance is not as good as the row based managers. With the ODBC driver the performance drops even more since the ODBC interface is more general compared to using a driver specifically designed for Oracle.

Since Oracle is set based it should be possible to increase the performance as it was with Microsoft SQL. Since Oracle with the ODBC driver is slower compared to the other drivers, it needs other values of the settings in the CK to be able to show the same performance. When tested, it turns out that unreasonably high values of the cache timeout is not enough to see the increase in performance that the tests showed with the dedicated drivers for the set based database managers.

Another problem with Oracle and the ODBC driver is that when a setting is modified in the CK it takes an unreasonable amount of time to start the application. The problem is that the cache files that are created needs to be recreated with the new settings and creating the cache files when using Oracle takes a lot of time compared to the other database managers. This problem does not occur when using the ODBC driver with Pervasive.SQL.

Oracle was also tested with the dedicated driver which has other properties than the other CK have. It has the same option to fetch multiple records and when increasing the block size, the performance increases too. The problem with the cache is that it does not have a timeout value so this has to be used carefully to avoid using incorrect data. Queries can be set to bypass the cache to guarantee that the latest data is fetched from the database.

Oracle is set based but the CK can be configured to act as a row based engine. It does increase the performance for some queries but only a small amount and it cannot be compared to using a row based engine since it is still Oracles' set based engine that executes the queries. It does however guarantee that the most recent data is used without losing performance. When using the find mode *equal* it increases the performance with about ten percent. The row based or set based option can be switched dynamically so it can be used where it increases the performance.

4 BACKUP AND CLUSTERING

The idea with backups is to avoid loosing data, and all database managers provide or supports software that is used for scheduled and continuous backups. It is used to make sure that a backup session is run daily, weekly or when needed, and helps users to monitor the backups through logs.

As a developer, there is a second aspect to backups. A developer might be interested in taking quick backups to set up test environments or to take backups of specific tables to test and debug new functions. In this case the third party backup software is not suitable to use so if there is an easy way to perform this through a tool provided by the database manager itself, the tool will be evaluated.

- Automatic backups

  For creating automatic backups, as mentioned above, a third party backup software is normally used that guarantees data integrity. These tools will not be evaluated but the support for them for each database manager will be.

- Manual backups

  Taking manual backups is interesting for developers. It can be possible through a tool provided by the database manager or the file system might let a user make copies of single tables.

- Recovery

  Backups are taken to make sure that if anything happens, the data is not lost. If data do get lost, it has to be recovered from a backup and restored to the database.

- Online/offline backups

  Some systems might require that backups can be taken while the database is online. It may not be possible to go offline to perform a backup and therefore the support for online backups will be evaluated.

Backups are meant to protect the data that are in the tables but a system also needs protection from hardware failures. To protect the system from hardware failures, clustering can be used. Clustering solutions can be set up in different ways.

One way is to have two servers that each has their own physical database. One of the servers is the primary server that the users are connected to and all queries are committed on the primary server. The database is periodically synchronized to the secondary server and if the primary server goes offline, the secondary server can go online. The data used will be from the latest synchronization but the system is running again.

A second solution is to have two physical servers that have access to the same database. They create a virtual server so the clustering is transparent to the users. If the primary server goes offline, the secondary server will notice that and go online instead. Uncommitted transactions have to be rolled back but otherwise the system is up and running again.

A third solution is to use a combination of the first two. Having two database locations that are synchronized and to let one of the locations be the primary site. At the primary site two servers can be used to set up a cluster. If the primary site goes offline, the secondary site can go online.

Clustering can also be used to share the client load among several servers to ensure balance in the load on each server. Running several servers in a clustered environment has several advantages at the same time as all servers appears as a single resource to the clients.

4.1 Microsoft SQL

Microsoft SQL Server Management Studio only offers simpler backup support. It is easy to backup the entire database by following a few instructions. Doing this dumps the entire database to a file that can be restored later on.

When a database has a backup created once, it is possible in the future to perform a differential backup. With this option the backup will add all changes that have been made since the last backup to an earlier created backup. This avoids having to make full backups every time since the data that has not been changed since the last backup does not have to be written again.

A Microsoft SQL Server 2005 database is stored in two physical files. One file contains logging information and the other the actual tables and the structure of the database. Since these are the only files available it is only possible to perform manual backups of the entire database.

Backups of single tables are not supported but it is possible to split tables into different file groups. When data is placed in more than one location it is possible to link the data together in the same database using file groups. A user can make a backup of a single file group so this method can be used to backup single tables. An idea is to put all tables that are related to each other into one file group so that backups can be done on groups of tables that are connected to each other.

A solution for single tables is to export the data to a file that can be imported later on. This is a performance intensive task depending on the size of the table, but this can be done when the database is online without disturbing users. This way the data can be saved and a developer can import the data again if a test on the table failed. Exporting data is done without a transaction so if it is done when the database is online, the data may not be consistent since users can modify the data while it is being exported. The data is not locked before the export starts.

SQL Server Enterprise Manager has an interface that allows for connecting third party backup software to perform more advanced backups like encrypted backups, scheduled backups to tape, more efficient backups and recovery sessions.

Microsoft SQL has support for an online backup method that locks the database when the backup session starts. All changes to the data are stored in a buffer during the session and committed when the backup session is finished.

For hardware security, Microsoft SQL supports clustering and has its own software to do this. Microsoft Cluster Server can be used to set up two or more servers in a cluster that can reduce the server load on each server or to have servers monitoring the primary server and go online if the primary server goes offline.

4.2 Pervasive.SQL

Pervasive.SQL does not have any advanced backup utilities to perform backups on its own. The only tool is to export and import data through text files. This can be considered a bit too simple but Pervasive.SQL has a file structure that can be useful for developers. A Pervasive.SQL database has the data for each table in separate files. Each file is named after the table it contains the data for and it can be useful to make copies of only parts of the database or when debugging a new function.

If a developer needs to set up a new test environment, all table files can be copied to where the test environment is to be set up. It is not possible to dump the entire database into a file that can be attached as a new database since the only available feature is to export the data into text files.

To create backups of a Pervasive.SQL database, a program called Pervasive Backup Agent enables usage of third party backup software. Through the backup agent the most popular backup software is supported and it is possible to use the third party software to create online backups. A Pervasive.SQL database can be set to run in continuous mode which locks the database and all changes are stored in a buffer instead of written to the database while the backup session runs. When it is finished, all buffered changes are committed.

Pervasive.SQL does not have clustering software but it is possible to use Microsoft Cluster Server to set up a clustered environment.

4.3 Oracle

Oracle offers many backup features. One is offline backups to tape or to disc, which later on can be restored easily. This feature dumps the entire database to a file that can be loaded later on.

Oracle also supports online backups by switching the database to ARCHIVELOG mode. When this mode is enabled, the database does not overwrite log files. The log files are needed when the database is locked to be able to commit all actions performed during the backup when it is finished.

If the database is not in the archivelog mode and online backups are preferred, Oracle also supports exporting data to files while the database is running.

Oracle stores all data in one physical file which prevents manual backup of single or specific tables. It is possible to export the data in a table to a file and later import it.

Oracles standard interface supports scheduled backups of the entire database.

For more advanced backup options the Recovery Manager (RMAN) in combination with Oracle Secure Backup can be used which is used to schedule backups to tapes. It performs more efficient backups than the standard backups that are supported through Oracle Control Center. Oracle Secure Backup also supports encrypting backups.

RMAN is used to control the underlying procedures before or after a backup or a restoring session. It keeps backup history and it removes dependencies on the operating system or script that is used for the backup or restoring session.

Through these two tools an interface is available for the use of third party backup software if that is preferred.

Oracle has software for setting up and monitoring a clustered environment. The software is called Oracle Data Guard and can be used in combination with other Oracle software to increase availability and security. It is meant as the connection and guarding software between a primary and a secondary site.

4.4 Summary

All database managers show that backup is a very important part when using databases. The support of the tools that are provided with the database managers by default varies but they all have interfaces for connecting with more advanced backup software.

If third party software is not considered, Oracle has better backup support by default compared to Pervasive.SQL and Microsoft SQL. Pervasive.SQL does not offer any built in solutions at all but it has software that is compatible with the most popular third party software for each task.

For developers, the structure of a Pervasive.SQL database is good for extracting specific tables from a database.

Since all database managers either have tools for advanced backups or support for third party software that can perform the same task, the conclusion is that choosing a database manager should not be based on the backup possibilities. Creating backups and keeping the database available to users is a priority of each database manager.

5 CONCLUSIONS

After testing different database managers the conclusion is that the choice is different depending on the environment it is going to be used in.

The Embedded Database is a very easy database to use. It is automatically installed with Visual DataFlex (VDF) and when a new project is created in VDF, a database is created in the embedded format for that project. The structure of the Embedded Database allows for very fast database access when only one client needs access at once (see section 1.2.3).

A problem with the Embedded Database is that it is not suitable to use when the server load increases and it does not have important properties like ACID transactions (see glossary) which database managers should have. This makes it unsuitable for critical systems and other options needs to be considered. Pervasive.SQL, Microsoft SQL and Oracle have all properties of ACID and can guarantee the integrity and security that a critical system needs.

Installing and setting up the database managers was the starting point of the test and neither Pervasive.SQL nor Microsoft SQL showed any problems. Oracle on the other hand had some problems when it came to setting it up for a VDF application. Pervasive.SQL and Microsoft SQL supports features that VDF has and needs no configuring to be compatible. Installing and setting up the database should be as simple as it was for both Microsoft SQL and Pervasive.SQL.

Migrating data into an Oracle database gave several problems compared to Microsoft SQL and Pervasive.SQL which showed no complications at all. The reason is that Oracle does not support auto increment columns and therefore recnum columns do not work in VDF when using the ODBC driver. The solution is to either create recnum support manually or to use the dedicated driver from Mertech which can add recnum support automatically.

Oracle did catch up to the other database managers after showing that the tools to handle databases are more powerful than the tools the other database managers provided. Pervasive.SQL's tools did not handle a few basic tasks compared to the opponents' tools. This is not a serious problem since these tasks should be performed by the Database Builder and Database Explorer. It is possible to use the database manager's tools but they do not update the data dictionaries when modifying a table as the Database Builder does. The Database Builder and Database Explorer also turned out to be easier to use than any of the three database managers' tools.

When measuring the performance there are differences between using a set based and a row based engine (see section 1.3.1). The set based engines are Microsoft SQL and Oracle and the row based engines are the Embedded Database and Pervasive.SQL. The Embedded Database has high performance when using a single client but when the number of clients increases, the performance decreases significantly. It is a very fast database manager for single client usage.

Pervasive.SQL performed better in all tests compared to the set based database managers and they performed equally well compared to each other. Oracles performance was better with the ODBC driver than Pervasive.SQL with the same driver.

The set based managers performance can be increased since a block size cursor is used (see section 1.3). Changing the settings in the connectivity kit (CK) increases the performance for the dedicated drivers but it can also lead to an increasing risk of using incorrect data (see section 3.1).

The performance test clearly shows that when it comes to performance and using VDF the choice is simple. The performance test came out with a clear winner and that is Pervasive.SQL since it uses the row based engine. It not only performs significantly better but it also guarantees that the latest added or modified data is used since it always fetches records from the database. The other CK try to optimize record fetching by using a cache which can lead to a use of incorrect data.

All three managers either have advanced backup software or supports third party software to perform the task on the database. They all have advanced support for protecting the database in software and to ensure high availability and scalability if needed. Oracle is the choice if backup support is expected to be provided when installing the environment but otherwise the choice does not matter.

The final verdict when comparing different database managers in combination with VDF is that a database with a row based engine should be used if performance is a high priority. That was the only thing in all tests that showed a significant difference.

The Embedded Database is the choice in a non-critical system with a small number of users. If the number of users increase or the data needs reliable transactions, another database manager needs to chosen.

If the performance is not an issue, Microsoft SQL is as good as Pervasive.SQL. It is very easy to set up and it supports recnum by default. Microsoft SQL has a better integrated environment than Pervasive.SQL since Microsoft has developed tools to use with their own database manager, while Pervasive.SQL supports third party software for most tasks. Microsoft SQL also has a dedicated driver that allows for more secure optimization than the dedicated driver for Oracle. Microsoft SQL is also easier to set up and Oracle showed problems when altering settings by launching applications slower.

Oracle has a very advanced enterprise environment with lots of tools for maintaining and monitoring databases but they are not as suitable as Microsoft SQL and Pervasive.SQL to be used together with VDF.

If performance is important, Pervasive.SQL is the best choice. If performance is less important the choice depends on the system the database manager are going to be used in. If the priority is to get a database manager that is easy to install and maintain then both Pervasive.SQL and Microsoft SQL are good choices. If there is a strong need for an integrated enterprise environment then Microsoft SQL is the best choice. Both should be used with the dedicated drivers since the ODBC driver significantly decreases the performance.

6 REFERENCES

1 **Visual DataFlex and SQL Database Servers**, Modified: 2007-01-01,
URL: http://www.visualdataflex.com/features.asp?pageid=757, Cited: 2008-10-11.

2 **Microsoft SQL Server 2005**, Modified: N/A,
URL: http://www.microsoft.com/Sqlserver/2005/en/us/default.aspx, Cited: 2008-10-07.

3 **Oracle Database 11g**, Modified: N/A,
URL: http://www.oracle.com/technology/products/database/oracle11g/index.html, Cited: 2008-10-18.

4 **Pervasive Software – PSQL Overview**, Modified: N/A,
URL: http://ww2.pervasive.com/Database/Products/PSQLv10/Pages/PSQLOverview.aspx, Cited: 2008-10-09.

5 **Fourth Generation Language**, Modified: N/A,
URL: http://knowledgerush.com/kr/encyclopedia/4GL/, Cited: 2009-03-06.

6, **SheerPower 4GL** Modified: 2009-01-03, URL: http://www.sp4gl.com/, Cited: 2009-03-02.

7 **PowerBuilder**, Modified: N/A,
URL: http://www.sybase.com/products/modelingdevelopment/powerbuilder, Cited: 2009-03-02.

8 **SQL – Structured Query Language**, Modified: N/A, URL: http://www.sql.org/, Cited: 2009-02-26.

9 **Matlab**, Modified: N/A, URL: http://www.mathworks.com/products/matlab/, Cited: 2009-03-02.

10 **James Martin**, Modified: N/A, URL: http://jamesmartin.com/, Cited: 2009-02-26.

11 **Martin, James** (1981), *Application Development Without Programmers*, ISBN 0-13-038943-9.

12 **Martin, James** (1985), *Fourth-generation languages. Vol I: principles*, ISBN: 0-13-329673-3.

13 **Martin, James** (1986), *Fourth-generation languages. Vol. II: representative 4GLs*,
ISBN: 0-13-329749-7.

14 **Martin, James** (1986), *Fourth-generation languages. Vol. III. 4GLs from IBM*, ISBN: 0-13-329764-0

15 **R J, Casimir** (1989), *Fourth generation problems*, ISSN: 0362-1340.

16 **Santosh K. Misra; Paul J.Jalics** (1988), *Third-Generation Versus Fourth-Generation Software Development*, ISSN: 0740-7459.

17 **Klepper, Robert; Bock, Douglas** (1995), *Third and fourth generation language productivity differences*, ISSN: 0001-0782.

18 **Richard H, Cobb** (1985), *In praise of 4GLS*, ISSN: 0011-6963.

19 **Sumner, Mary; Benson, Robert** (1988), *The impact of fourth generation languages on systems development*, ISSN: 0378-7206.

20 **Dimitris N. Chorafas** (1986), *Fourth and fifth generation programming languages. Vol. 1: integrated software, database languages, and expert systems*, ISBN: 0-07-010864-1.

21 **Windows Server 2003 Active Directory**, N/A,
URL: http://www.microsoft.com/windowsserver2003/technologies/directory/activedirectory/default.mspx,
Cited: 2008-10-16.

22 **Verner, June; Tate, Graham** (1988), *Estimating Size and Effort in Fourth-Generation Development*

23 **Harel, E; McLean** (1982), *The Effects of Using a Nonprocedureal Language on Programmer Productivity*

7 APPENDIX

The appendix contains all test data from the performance test and also parts of the source code and a glossary for technical terms.

7.1 Performance Test

All data below are measures in milliseconds.

7.1.1 Reading an Entire Table

Reading the entire table DeliveryRows with the default settings of the cache timeout and the block size properties.

Embedded Database	Pervasive SQL	Microsoft SQL	Oracle ODBC	Oracle ORAFlex	Pervasive ODBC
8 657	20 609	57 391	55 516	43 266	165 484
8 656	20 610	57 843	55 390	43 062	166 641
8 687	20 703	57 063	55 547	42 859	163 391
8 688	20 593	57 172	55 406	42 704	164 500
8 656	20 594	56 954	55 344	42 531	163 828
8 669	**20 622**	**57 285**	**55 441**	**42 884**	**164 769**

Microsoft SQL, cache timeout 10 ms. The header is the block size property. The task performed was to read the entire table DeliveryRows.

2	10	15	25	50	100	200	500	2 000
107 156	57 828	54 547	50 375	43 406	42 031	42 703	45 750	63 985
109 141	57 937	54 297	50 594	43 000	41 985	42 844	45 735	64 343
107 016	57 453	53 984	51 265	42 859	41 907	42 360	45 734	64 032
108 593	57 672	54 969	50 515	42 922	42 015	42 187	45 719	64 375
107 125	57 610	54 359	50 610	43 000	41 922	42 016	45 687	63 828
107 806	**57 700**	**54 431**	**50 672**	**43 037**	**41 972**	**42 422**	**45 725**	**64 113**

Microsoft SQL, cache timeout 1 000 ms.

2	10	15	25	50	100	200	500	2 000
106 813	56 890	53 437	50 750	43 297	42 235	42 203	39 828	37 610
107 125	57 110	53 188	50 422	43 265	42 375	41 672	38 938	36 782
107 625	57 453	53 203	50 563	43 282	42 250	42 015	40 172	36 407
106 891	57 187	53 953	50 656	43 296	42 422	43 063	39 734	36 890
106 312	57 719	54 297	50 532	43 219	42 109	42 265	39 906	37 594
106 953	**57 272**	**53 616**	**50 585**	**43 272**	**42 278**	**42 244**	**39 716**	**37 057**

Microsoft SQL, cache timeout 30 000 ms

100 000
14 875
14 844
14 813
14 859
14 829
14 844

ODBC driver connected to Oracle, cache timeout 1 000 ms.

2	10	15	25	50	100	200	500	2 000
56 469	51 359	49 171	50 969	50 734	51 828	50 750	53 750	59 735
56 672	51 250	48 657	51 484	50 860	50 906	51 203	53 594	56 843
56 406	52 047	49 360	51 359	50 437	50 735	51 516	54 797	57 625
56 235	52 328	48 703	51 047	50 828	52 359	51 656	56 468	57 438
55 625	52 063	48 422	51 188	50 641	53 375	51 297	53 875	57 140
56 281	**51 809**	**48 863**	**51 209**	**50 700**	**51 841**	**51 284**	**54 497**	**57 756**

ODBC driver connected to Oracle, cache timeout 5 000 ms.

2	10	15	25	50	100	200	500	2 000
58 969	54 578	53 484	53 969	53 719	53 625	54 719	57 328	61 641
58 625	54 313	53 047	54 047	53 859	54 531	54 406	56 281	60 203
59 125	54 578	52 922	54 609	56 188	54 109	58 187	56 016	61 906
59 562	54 328	53 422	54 032	54 359	54 578	54 938	56 421	60 531
59 500	54 469	52 375	53 750	54 031	54 172	55 515	56 062	61 344
59 156	**54 453**	**53 050**	**54 081**	**54 431**	**54 203**	**55 553**	**56 422**	**61 125**

The ORAFlex driver connected to Oracle, no timeout setting.

2	10	15	25	50	100	200	500	2 000
98 734	43 547	52 297	44 000	44 796	38 468	38 562	26 687	21 719
99 546	43 640	49 297	42 031	37 657	40 703	34 672	25 453	22 079
98 938	42 641	50 265	44 782	41 375	34 313	38 969	24 579	21 828
99 094	43 734	51 953	40 718	45 656	34 187	35 156	24 640	22 531
99 109	44 875	48 563	43 625	45 516	36 328	39 250	26 047	22 234
99 084	**43 687**	**50 475**	**43 031**	**43 000**	**36 800**	**37 322**	**25 481**	**22 078**

Reading the entire table DeliveryRows with the ORAFlex driver connected to Oracle and DeliveryRows set to row based mode.

For Row
44 672
43 781
43 297
44 594
43 375
43 944

7.1.2 Find Equal

Performing 10 000 *Find equal* operations on the table DeliveryNotes.

Embedded Database	Pervasive SQL	Microsoft SQL	Oracle ODBC	Oracle ORAFlex	Pervasive ODBC
203	141	1 063	2 047	984	2 672
203	156	1 063	2 015	1 016	2 703
203	156	1 094	2 047	985	2 719
203	156	1 094	2 031	969	2 703
219	141	1 109	2 016	953	2 703
203	140	1 078	2 000	968	2 688
203	141	1 187	2 015	1 016	2 703
219	156	1 141	2 000	953	2 687
203	141	1 109	2 031	985	2 688
218	312	1 125	2 000	953	2 672
208	**164**	**1 106**	**2 020**	**978**	**2 694**

Performing 10 000 *Find equal* operations on the table DeliveryNotes with the ORAFlex driver connected to Oracle and DeliveryNotes set to row based mode.

For Row
891
875
875
875
890
875
891
891
875
875
881

7.1.3 Transactions

Updating a column in a table for each record using the table Garments and transactions. This is performed using the Embedded Database with only one client and the first header of the table is the transaction sizes.

1	3	5	10	25	50	100	500	1 000
56 890	21 437	13 609	8 109	4 735	3 547	2 953	2 463	2 421
56 906	21 078	13 594	8 000	4 640	3 531	2 953	2 474	2 422
56 969	21 094	13 578	7 985	4 625	3 500	2 938	2 482	2 407
57 000	21 109	13 594	7 968	4 609	3 485	2 906	2 469	2 406
57 016	21 078	13 562	7 968	4 610	3 484	2 922	2 449	2 406
56 956	**21 159**	**13 587**	**8 006**	**4 644**	**3 509**	**2 934**	**2 467**	**2 412**

Pervasive.SQL, transactions.

1	3	5	10	25	50	100	500	1 000
42 688	23 484	16 985	12 391	9 875	9 094	8 578	8 625	8 172
42 672	23 594	18 312	12 656	9 953	9 046	9 656	8 313	8 437
42 656	21 906	16 453	12 438	9 890	9 266	8 750	8 406	8 266
42 657	23 422	16 421	14 109	9 985	8 937	8 610	8 282	8 453
42 656	23 750	18 438	12 563	9 828	8 984	8 609	8 187	8 407
42 666	**23 231**	**17 322**	**12 831**	**9 906**	**9 065**	**8 841**	**8 363**	**8 347**

Microsoft SQL, transactions.

1	3	5	10	25	50	100	500	1 000
186 328	108 781	75 985	51 469	40 641	36 453	34 015	32 282	31 843
186 203	107 828	76 297	51 657	40 219	36 265	34 016	32 234	31 860
186 375	107 688	75 656	51 750	40 343	36 735	34 062	32 406	31 937
186 375	107 297	76 047	51 843	40 641	35 984	34 094	32 453	31 875
186 266	107 844	75 875	51 250	40 750	36 031	34 141	32 313	31 875
186 309	**107 888**	**75 972**	**51 594**	**40 519**	**36 294**	**34 066**	**32 338**	**31 878**

Oracle using the ORAFlex driver, transactions.

1	3	5	10	25	50	100	500	1 000
221 828	100 609	76 078	57 657	46 687	43 109	41 110	39 375	41 171
212 781	99 829	75 625	58 968	47 485	43 719	43 235	39 453	40 860
211 469	100 656	75 968	58 547	47 203	43 687	42 594	41 500	40 937
210 812	100 187	75 813	58 406	47 500	44 188	42 406	41 219	40 985
213 156	100 313	75 562	58 657	47 703	44 031	42 469	41 532	40 532
214 009	**100 319**	**75 809**	**58 447**	**47 316**	**43 747**	**42 363**	**40 616**	**40 897**

Oracle using the ODBC driver, transactions. Using transactions with more than one record did not work.

1	3	5	10	25	50	100	500	1 000
822 203	-	-	-	-	-	-	-	-
824 672	-	-	-	-	-	-	-	-
824 063	-	-	-	-	-	-	-	-
823 593	-	-	-	-	-	-	-	-
823 610	-	-	-	-	-	-	-	-
823 628	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**

Embedded Database, Multi-Client. Updating a column for each record in the table Garments with a transaction size of 1 000 records. Clients A, B and C ran simultaneously.

Client A	Client B	Client C
14 978	27 384	19 885
13 718	28 090	26 912
13 862	19 054	27 142
14 348	7 936	8 160
14 625	7 917	7 946
14 306	**18 076**	**18 009**

Pervasive.SQL, Multi-Client. The clients A, B and C ran simultaneously.

Client A	Client B	Client C
14 906	14 735	15 922
13 547	14 484	14 609
14 657	13 578	14 688
15 843	13 891	14 875
14 516	14 515	14 406
14 694	**14 241**	**14 900**

7.1.4 Relations and Data Dictionaries

Reading 1 000 records of the table DeliveryNotes and all related tables data dictionaries loads each related record.

Embedded Database	Pervasive SQL	Microsoft SQL	Oracle ODBC	Oracle ORAFlex	Pervasive ODBC
1 004	968	1 453	1 812	1 438	2 609
1 047	985	1 438	1 719	1 468	2 391
1 007	968	1 453	1 797	1 438	2 406
1 016	985	1 484	1 750	1 437	2 515
1 000	969	1 485	1 766	1 438	2 360
1 015	968	1 453	1 797	1 406	2 281
1 011	969	1 422	1 734	1 422	2 375
1 000	969	1 406	1 766	1 453	2 344
1 016	969	1 406	1 781	1 437	2 344
1 000	985	1 406	1 812	1 438	2 375
1 012	**974**	**1 441**	**1 773**	**1 438**	**2 400**

7.2 Glossary

ACID

ACID is a term that stands for (Atomicity, Consistency, Isolation, Durability). If a database manager has all four properties of ACID it ensures that transactions are handled correctly.
Atomicity: A transaction is always either committed or rolled back, never partly committed.
Consistency: Only valid data will be written to the database. If it is consistent before a transaction it will be consistent after, whether the transaction was successfully executed or not.
Isolation: No other operation can see the data that is written by a transaction until it is committed.
Durability: If a user is notified of success, the committed changes will persist and cannot be undone.

Cluster

A cluster contains several computers linked together to form a single resource. Clusters are used to increase performance or to prevent having a single point of failure.

Connectivity Kit

They are used to connect Visual DataFlex applications to a database. It handles the conversion needed for the queries to be correctly sent to the database manager.

Cursor

A cursor is used for processing individual records in a result set returned by a query.

Data Dictionary Object (DDO)

When using a Visual DataFlex application each table has a DDO that is based on the data dictionary file class created for the table. The DDO is a layer between the application logic and the data in a table, and it contains the information and rules about the table.

Descending index

It is an index where one or more of the columns are in descending order.

Find

Find is used to retrieve records from the database using a find mode, a table name and an index.

Find Mode

The find mode is given when performing a find against the database. It decides how the query should fetch the next record.

Found

Found is a global variable that indicates if the latest find performed matched a record in the database or not.

Index

Indices can be created using one or more columns of a table and it enables the database manager to perform queries faster.

Microsoft Cluster Server (MSCS)

MSCS is a Windows service used to allow multiple servers to act as a single cluster.

Open DataBase Connectivity (ODBC)

It is a standard interface aimed to be independent from which programming language, database manager or operating system is used.

Recnum

Recnum is a numeric column added to each table by Visual DataFlex, when working in recnum mode, and is used to guarantee that each record in the table can be uniquely identified.

Record Buffer

It is a global variable that stores one record of the table it represents.

RowId

RowId is what makes each record unique in Visual DataFlex. In recnum mode, recnum is the RowId and in standard mode it can be based on any column that is unique.

Runtime

A runtime is a virtual machine that can run an application. It has a library that enables for all functionality of a programming language to be used in different environments. The Visual DataFlex runtime enables functionality that Windows does not have, for Visual DataFlex applications.

Schema

A database can be divided into several schemas. Tables, triggers etc. can belong to certain schemas instead of the entire database.

Sequence

A sequence is a variable used with databases. A sequence is normally used to generate a unique value for each new record added to a table or to count how many records the table contains. The sequence contains an integer value that can be incremented whenever it is necessary.

Server Message Block (SMB)

SMB is a protocol used by Windows to share access to files, printers or similar over a network.

Thread

Threads are used to split an application into two or more simultaneously running tasks.

Transaction

A transaction is used to perform actions as a unit against a database. This means that either all of the actions are committed or none of them.

Trigger

A trigger is a procedure that is called when a certain event occurs, like before an insert in a table.

7.3 Source Code

```
Class ClcStopWatch is a cObject

   Procedure Construct_Object
      Forward Send Construct_Object

      Property DWord  pdwStartTime  Public 0
      Property DWord  pdwStopTime  Public 0

      Property Date   pdStartDate   Public 0
      Property String psStartHHMMSS Public ""
      Property Date   pdStopDate    Public 0
      Property String psStopHHMMSS  Public ""
   End_Procedure // Construct_Object

   Procedure Start
      Set pdwStartTime  to (WinAPI_GetTickCount())
      Set psStartHHMMSS to (ClxTimeStampHhMmSs())
      Set pdStartDate   to (ClxToday())
   End_Procedure // Start

   Procedure Stop
      Set pdwStopTime  to (WinAPI_GetTickCount())
      Set psStopHHMMSS to (ClxTimeStampHhMmSs())
      Set pdStopDate   to (ClxToday())

      Send ClxAssertClassProcedure (pdwStartTime(Self) > 0) "ClcStopWatch" "Stop" "Start must be
called first"
   End_Procedure // Stop

   Function ElapsedMilliSeconds Returns Integer
      If (pdwStopTime(Self) >= pdwStartTime(Self)) Begin
         Function_Return (pdwStopTime(Self) - pdwStartTime(Self))
      End
      Else Begin
         Function_Return (WinAPI_GetTickCount() - pdwStartTime(Self))
      End
   End_Function // ElapsedMilliSeconds

   Function ElapsedSeconds Returns Number
      Function_Return (ClxRound(ElapsedMilliSeconds(Self) / 1000.0, 0.1))
   End_Function // ElapsedSeconds

End_Class // ClcStopWatch
```

```
Object oTabPagePerformanceTest is a dbTabPage
        Object oStopWatch is a ClcStopWatch
        End_Object

        Procedure StopWatch_s
          Integer iSec

          Send Stop of oStopWatch
          Get ElapsedSeconds of oStopWatch to iSec

          Showln (SFormat("%1", iSec))
        End_Procedure // StopWatch_s

        Procedure StopWatch_ms
          Integer iMillisec

          Send Stop of oStopWatch
          Get ElapsedMilliSeconds of oStopWatch to iMillisec

          Showln (SFormat("%1", iMillisec))
        End_Procedure // StopWatch_ms

        Set Label to "Performance Test"
        Object oGroupQueries is a Group
          Set Size to 201 124
          Set Location to 9 13
          Set Label to "Queries"
          Object oButtonEntireTable is a Button
            Set Label to "Entire table | DeliveryRows"
            Set Size to 14 99
            Set Location to 14 12

            Procedure OnClick
              Integer iCount

              For iCount From 0 to 6
                Send Start to oStopWatch

                Clear DeliveryRows
                Find ge DeliveryRows
                While (Found)
                  Find gt DeliveryRows
                Loop

                Send StopWatch_ms
              Loop

            End_Procedure // OnClick
          End_Object    // oButtonEntireTable
```

```
Object oButtonFindEQ is a Button
        Set Label to "Find eq | DeliveryNotes"
        Set Size to 14 99
        Set Location to 34 12

        Procedure OnClick
          Send Start to oStopWatch

          Clear DeliveryNotes
          Find ge DeliveryNotes
          While (Found)
            Find gt DeliveryNotes
          Loop

          Integer iDeliveryNotesNo iLoop i
          Move 110150 to iDeliveryNotesNo

          For i From 0 to 11
            Send Start of oStopWatch

            For iLoop From 0 to 10000
              Clear DeliveryNotes
              Move iDeliveryNotesNo to DeliveryNotes.DeliveryNotesNo
              Find eq DeliveryNotes by Index.1
              Increment iDeliveryNotesNo
            Loop

            Send StopWatch_ms
          Loop
        End_Procedure // OnClick
      End_Object    // oButtonFindEQ

      Object oButtonChangeValueTransactions1 is a Button
        Set Label to "Change Value | Garments 1"
        Set Size to 14 99
        Set Location to 57 12

        Procedure OnClick
          Integer iLoop iValue

          Move 1 to iValue
          Clear Garments
          Find ge Garments
          While (Found)
            Find gt Garments
          Loop

          For iLoop From 0 to 6
            Send Start to oStopWatch

            Clear Garments
            Find ge Garments
```

```
        While (Found)
          Reread Garments
          Move iValue to Garments.TimesWashed
          Saverecord Garments
          Unlock

          Find gt Garments
        Loop

        Send StopWatch_ms
        Increment iValue
      Loop
    End_Procedure // OnClick
End_Object    // oButtonChangeValueTransactions1

Object oFormTransactions is a Form
    Set Size to 13 91
    Set Location to 92 19
End_Object    // oFormTransactions

Object oButtonChangeValueTransactionsX is a Button
    Set Label to "Change Value | Garments x"
    Set Size to 14 91
    Set Location to 110 19

    Procedure OnClick
      Integer iTransactions iValue iLoop

      Get Value of oFormTransactions to iTransactions
      Decrement iTransactions
      Move 1 to iValue

      Clear Garments
      Find ge Garments
      While (Found)
        Find gt Garments
      Loop

      For iLoop From 0 to 6
        Send Start to oStopWatch

        Clear Garments
        Find ge Garments
        While (Found)
          Begin_Transaction
            Send ProcessRecords iTransactions iValue
          End_Transaction
          Find gt Garments
        Loop

        Send StopWatch_ms
        Increment iValue
```

```
        Loop
      End_Procedure // OnClick

      Procedure ProcessRecords Integer iRecords Integer iValue
        Integer iLoop

        For iLoop From 0 to iRecords
          Reread Garments
          Move iValue to Garments.TimesWashed
          Saverecord Garments
          Unlock

          If (iLoop < iRecords) Begin
            Find gt Garments
          End

          If (not(Found)) Begin
            Procedure_Return
          End
        Loop

      End_Procedure // ProcessRecords
    End_Object    // oButtonChangeValueTransactionsX
  End_Object    // oGroupQueries

  Object oGroupDataDicitonaries is a dbGroup
    Set Size to 201 228
    Set Location to 9 172
    Set peAnchors to anNone
    Set Label to "Data Dictionaries"

    Object oFormDeliveryNotesDeliveryNo is a dbForm
      Entry_Item DeliveryNotes.DeliveryNotesNo
      Set Label to "DeliveryNotes.DeliveryNo"
      Set Size to 13 85
      Set Location to 30 140
      Set peAnchors to anNone
      Set Label_Col_Offset to 120
      Set Prompt_Button_Mode to pb_PromptOff
    End_Object    // oFormDeliveryNotesDeliveryNo

    Object oFormDeliveryAddressDeliveryAddressNo is a dbForm
      Entry_Item DeliveryAddress.DeliveryAddressNo
      Set Label to "DeliveryAddress.DeliveryAddressNo"
      Set Size to 13 85
      Set Location to 45 140
      Set peAnchors to anNone
      Set Label_Col_Offset to 120
    End_Object    // oFormDeliveryAddressDeliveryAddressNo
```

```
Object oFormDeliveryAddressCustomerNo is a dbForm
        Entry_Item DeliveryAddress.CustomerNo
        Set Label to "DeliveryAddress.CustomerNo"
        Set Size to 13 85
        Set Location to 60 140
        Set peAnchors to anNone
        Set Label_Col_Offset to 120
    End_Object    // oFormDeliveryAddressCustomerNo

    Object oFormCustomerName is a dbForm
        Entry_Item Customer.Name
        Set Label to "Customer.Name"
        Set Size to 13 85
        Set Location to 75 140
        Set peAnchors to anNone
        Set Label_Col_Offset to 120
    End_Object    // oFormCustomerName

    Object oFormVATconditions is a dbForm
        Entry_Item VATconditions.VAT
        Set Label to "VATconditions.VAT"
        Set Size to 13 85
        Set Location to 90 140
        Set peAnchors to anNone
        Set Label_Col_Offset to 120
    End_Object    // oFormVATconditions

    Object oFormPaymentTermPaymentTermNo is a dbForm
        Entry_Item PaymentTerm.PaymentTerm
        Set Label to "PaymentTerm.PaymentTerm"
        Set Size to 13 85
        Set Location to 105 140
        Set peAnchors to anNone
        Set Label_Col_Offset to 120
    End_Object    // oFormPaymentTermPaymentTermNo

    Object oFormCustomerStatusDescription is a dbForm
        Entry_Item CustomerStatus.Description
        Set Label to "CustomerStatus.Description"
        Set Size to 13 85
        Set Location to 120 140
        Set peAnchors to anNone
        Set Label_Col_Offset to 120
    End_Object    // oFormCustomerStatusDescription
```

```
Object oFormCountryCodeCountry is a dbForm
   Entry_Item CountryCode.Country
   Set Label to "CountryCode.Country"
   Set Size to 13 85
   Set Location to 135 140
   Set peAnchors to anNone
   Set Label_Col_Offset to 120
End_Object    // oFormCountryCodeCountry

Object oFormCategoryDescription is a dbForm
   Entry_Item Category.Description
   Set Label to "Category.Description"
   Set Size to 13 85
   Set Location to 150 140
   Set peAnchors to anNone
   Set Label_Col_Offset to 120
End_Object    // oFormCategoryDescription

Object oFormCustomerGroupDescription is a dbForm
   Entry_Item CustomerGroup.Description
   Set Label to "CustomerGroup.Description"
   Set Size to 13 85
   Set Location to 165 140
   Set peAnchors to anNone
   Set Label_Col_Offset to 120
End_Object    // oFormCustomerGroupDescription

Object oFormServiceCenterName is a dbForm
   Entry_Item ServiceCenter.Name
   Set Label to "ServiceCenter.Name"
   Set Size to 13 85
   Set Location to 180 140
   Set peAnchors to anNone
   Set Label_Col_Offset to 120
End_Object    // oFormServiceCenterName

Object oButtonDataDictionaries is a Button
   Set Label to "Data Dictionaries"
   Set Size to 14 65
   Set Location to 12 152

   Procedure OnClick
      Integer iDeliveryNotesNo iCount iLoop

      Clear DeliveryNotes
      Find ge DeliveryNotes
      While (Found)
         Find gt DeliveryNotes
      Loop

      Move 100001 to iDeliveryNotesNo
```

```
            For iLoop From 0 to 11
               Move 0 to iCount
               Send Start of oStopWatch

               While (iCount < 1000)
                  Clear DeliveryNotes
                  Move iDeliveryNotesNo to DeliveryNotes.DeliveryNotesNo
                  Increment iDeliveryNotesNo
                  Find eq DeliveryNotes by Index.1

                  If (Found) Begin
                     Relate DeliveryNotes
                     Send Request_Assign of DeliveryNotes_DD
                     Increment iCount
                  End
               Loop

               Send StopWatch_ms
            Loop
         End_Procedure // OnClick
      End_Object    // oButtonDataDictionaries
   End_Object    // oGroupDataDicitonaries
End_Object    // oTabPagePerformanceTest
```