INTERNATIONAL HEALTH TERMINOLOGY
STANDARDS DEVELOPMENT ORGANISATION
UK Terminology Centre (UKTC)

NHS
Connecting for Health

THE CLINICAL TERMS VERSION 3
(THE READ CODES)

INFORMATION RETRIEVAL - EXPERIMENTS

APRIL 2008

Purpose of this document

This document is one of a series that, taken together, describe the contents, structure and function of Clinical Terms Version 3 (The Read Codes).

This introduction is intended to provide information on Clinical Terms Version 3. It is also a guide to the other available documents each of which is updated independently. For this reason, different chapters may have different version numbers.

INFORMATION

Distribution	On request
Author	UK Terminology Centre
Further Copies From	UK Terminology Centre Helpdesk Service Support Unit Prospect House, Floor 2 Fishing Line Road Redditch Worcestershire B97 6EW Tel: +44 (0) 1392 206248 Fax: +44 (0) 1392 206945 E-mail: datastandards@nhs.net Internet: http://www.connectingforhealth.nhs.uk/systemsandservices/data/readcodes
Date of Issue	April 2008
Reference Number	176v1.0

Published by the UK Terminology Centre

Table of Contents

1. Introduction

This document sets out the results of the 'Information Retrieval (Analysis and Reporting)' Project for the NHS Connecting for Health.

- The project involved the following elements:
- Querying a generated patient database of over one million events with core concept, core concept & single qualifier and core concept and multiple qualifiers accessed using the Ancestor Table method.
- Querying a generated patient database of over one million events with core concept, core concept & single qualifier and core concept and multiple qualifiers accessed using the Range Number method.
- The core concept & single qualifier and core concept and multiple qualifiers were then tested against a database in which the EVENTS and QUALIFIERS tables were merged, so effectively de-normalising this section of the database.
- A set of queries involving core concept only were performed on a real, anonymised GP database.
- A set of queries simulating a decision support system were conducted against the real GP database.

To obtain consistency across the different database platforms tested, all queries were written in SQL.

Note:
To facilitate the readability of the report, medical terms are used within the document rather than their Read Codes. The Read Codes for all the terms used are contained in Appendix B.

2. Test Environment

The test platform with an IBM compatible PC with the following specification:

Processor:	Intel Pentium running at 133MHz
Motherboard:	Triton II VX
Memory:	32MB
L2 Cache:	256K
Hard disks:	1 x 1.2GB Fujitsu 1 x 2.5 GB Seagate ST52520A Both drives connected using E-IDE interface with Mode 4 standard data transfers.
Operating System:	Window 95 with Service Pack 1 installed

Databases: Primary database used InterBase version 4.2 single user version for Intel platforms. In addition a number of tests were conducted using: Paradox version 5 tables using Delphi. Indexes involving Read Codes were created as case sensitive.

Access version 2 tables using Delphi. All Read Codes were stored as long integers in Access as this database is case insensitive when conducting queries. *Note that all the database tables were stored on the Seagate hard drive, with the Windows operating system and development tools on the Fujitsu drive.*

Two databases were used. These had similar structures indexes. Detailed information on the InterBase database is given in Appendix A.

Database Volumes - Generated Database

EVENTS table	1,224,408 rows
QUALS table	1,601,099 rows
PATIENTS table	100,000 rows
ANCESTOR table	1,445,852 rows
IDRANGE	193,772 rows

Database Volumes - GP Database

EVENTS table	617,584 rows
PATIENTS table	9,861 rows
ANCESTOR table	1,445,852 rows
IDRANGE	193,772 rows

The generated database (available from the NHS Connecting for Health) contains procedures and disorders selected by a random number generator from Clinical Terms Version 3 (CTV3). Qualifiers were added by pointing the random number generator at the qualifiers and atoms (or their children) listed in the templates belonging to each disorder or procedure.

The GP data was real data, initially in 4-byte Read Code format, which was converted using the 4byte to Version 2 mapping files. All 5-byte (Version 2) codes are also present in CTV3. Being a historical database, no qualifiers were available and none were generated randomly. To ensure the data remained anonymous, personal details of patients were deleted and names, addresses and other personal information from the generated patient database were substituted.

2.1 Database Access

To minimise human intervention in timing queries, a simple Delphi application was developed which transferred SQL queries to the appropriate databases (InterBase, Access and Paradox) and automatically recorded start and end times for the queries.

In the case of InterBase and Paradox databases the query program used the native database drivers contained in the BDE, in the case of Access the BDE used the Microsoft ODBC driver supplied with Microsoft Office version 4.2. In the case of Access as a single SQL query was phrased, the overhead was minimal as only the SQL was passed to the database and the

results returned. If table level access had been used a significantly greater overhead would have been present.

As noted earlier in this section the Read Codes were stored as long integers when Access was used, as Access is case-insensitive when conducting queries. As a consequence, the results for Access are not directly comparable with the other two databases, though as Access cannot use standard alphanumeric based Read Codes it is a far reflection of how Read Codes could be implemented using Access. A long integer column in Access uses 4 bytes compared to the 5 bytes used by InterBase and Paradox for storing Read Codes, this may give a slight advantage to Access in retrieving the data but it is not huge difference. It should be noted that in experiments conducted, access to the Read Codes was using indexes; the actual format of the data should not have a significant effect on the locating of the data within the database.

3. Establish Comparative Timings for Ancestor Table Method

The first stage of this process involved developing a program to create the ANCESTOR table from the Hierarchy file (HIER) in the CTV3 release. This program includes all the child codes for each Read Code including the Read Code itself in the ANCESTOR table - i.e. the Read Code H33.. Asthma has H33.. as a child code along with all related terms.

The number of child codes that a given Read Code has varies from as little as 1 to 174691[1].

A set of seven queries retrieving data using a core concept only were run against the generated patient database using the Ancestor Table Method. These queries were conducted against the primary database used for these experiments, InterBase and against Paradox & Access databases. The basic query used was:

```
SELECT COUNT (events.eventid)
FROM ancestor, events
WHERE ancestor.parentcodes = '<i>Read Code</i>'
AND ancestor.childcode = events.readcode
```

The results are given below:

Core Concept Only - Ancestor Method (Count Only)						
Core Concept	Child Codes	Number of Records	Paradox	InterBase	Access	Access*
			Query Time in seconds			
Disorders	43277	430970	41	3728	217	19
Operations & procedures	16802	178919	33	1578	145	9
Gastrointestinal & digestive disorders	2879	32851	30	292	118	2
Tumour	3899	43949	33	383	127	3
Tumour of lung	63	827	8	8	74	1
Asthma	44	535	6	7	75	1
Heart Attack	50	440	4	4	73	1

* *A COUNT (1) was used in this query instead of COUNT (EVENTID)*

[1] 174691 is the number of child codes for the root Read Code '.....'.

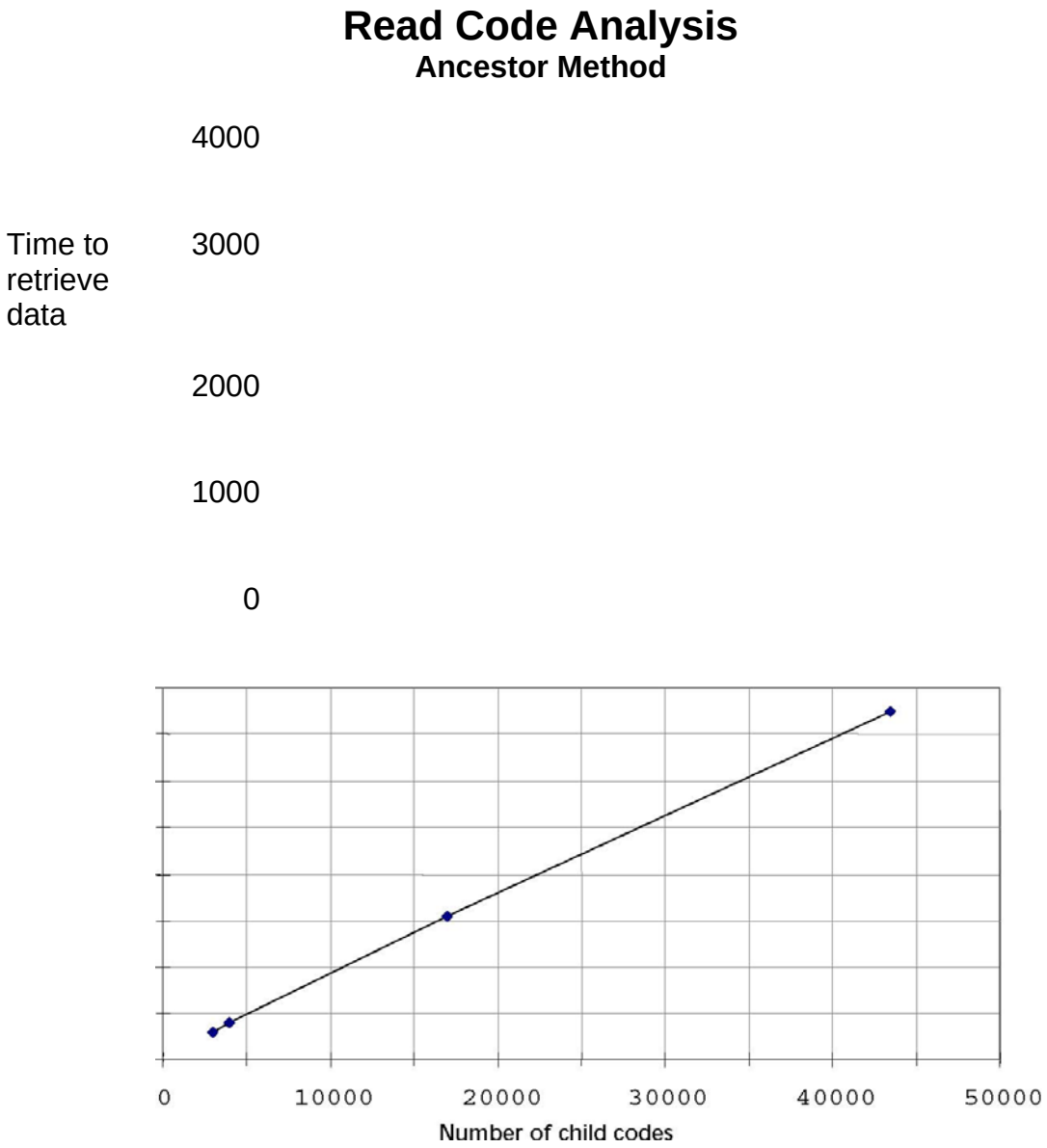

Figure 1. Interbase Query Times

As can be seen from the results, InterBase produced reasonable data retrieval times provided the number of child codes for a core concept was small but performance decreased significantly with an increase in the number of child codes. This is clearly shown in the graph on the next page. However this relationship may be misleading as the random generated database has the property that the number of child codes for a core concept is necessarily directly proportional to the number of events to be found in the events table that are children of the query node.

However, as we will see, optimisations to the SQL code for InterBase queries can significantly improve performance.

Both Paradox and Access show much better performance than InterBase

when the number of child codes reaches the thousands. While Paradox shows the best timings when doing a count of EventID, Access produces very fast times when doing a COUNT (1). This implies that Access can directly use the index on Read Code in the EVENTS table without needing to retrieve data from the EVENTS table. No similar improvement was seen in InterBase or Paradox if a COUNT (1) was used instead of a COUNT (EventID).

To more accurately understand the effect on query time to retrieve actual data rather than just a count of the number of records a further set of queries was run, but instead of a count, the EventIDs were retrieved and stored in a database table.

Core Concept Only - Ancestor Method (Event ID retrieved)					
Core Concept	Child Codes	Number of Records	Paradox	InterBase	Access
			Query time in seconds		
Disorders	43277	430970	44	4861	318
Operations & procedures	16802	178919	36	2053	225
Gastrointestinal & digestive disorders	2879	32851	39	377	142
Tumour	3899	43949	35	507	150
Tumour of lung	63	827	9	13	81
Asthma	44	535	8	12	81
Heart Attack	50	440	5	5	80

As can be seen, the time to perform this operation shows an increase over the figures for the count of EventID. In the case of InterBase the increase is about 30% for these queries involving a large number of rows returned (>10,000), while for Access it ranges, for these queries, from a low of 18% up to 55%, Paradox queries only took about 10% longer than the count queries.

Overall these results were reasonably good for all the databases indicating only a relatively low overhead for retrieving actual data. This indicates in the case of a count on EventId the query performs a direct access to the Events table for all the tested databases.

Due to the strong link between the number of child codes and the time to perform a query, InterBase without optimisation produced very poor timing for most of the queries involving a core concept and a single qualifier.

Core Concept and Single Qualifier - Ancestor Method

Core Concept (no. child codes)	Attribute (no. child codes)	Value (no. child codes)	Number of records	Paradox	InterBase	InterBase Optimised	Access
				Query Time in seconds			
Disorder (43277)	Site (143)	Stomach structure (29)	2934	52	8436	760	900
Gastrointestinal and digestive disorder (2879)	Site (143)	Human body structure (7619)	21500	114	>3600	536	748
Gastrointestinal and digestive disorder (2879)	Site (143)	Stomach structure (29)	2847	50	1535	944	741
Tumour (3899)	Site (143)	Stomach structure (29)	345	53	1680	1018	735
Asthma (44)	Chronicity (1)	Acute (7)	36	30	234	49	667
Operations & procedures (16802)	Priority (1)	Emergency (1)	40526	60	30204	1835	805
Male genital organ procedure (323)	Site (143)	Testis Structure (30)	259	16	1023	205	615
Male genital organ procedure (323)	Priority (1)	Emergency (1)	1000	59	846	122	662

Without optimisation, for all but the simplest queries (those involving least child codes) InterBase produces results which are likely to be unacceptable in practice. But it proved possible to improve performance by using a SORT MERGE type query which could be forced using a RIGHT JOIN statement in the FROM section of the SQL code. This improvement was maximised if the RIGHT JOIN added to the SQL statement involved the smallest number of child codes. In essence the optimisation ensures the parts of the query with the least child codes will be performed first, and will use this subset of the data as input to perform the rest of the query.

Paradox produced the best results with results of between five and ten times the performance of the optimised InterBase results and the standard queries for Access.

```
SELECT COUNT (events.eventid)
FROM quals, events, ancestor anc1,
        ancestor anc2, ancestor anc3
```

```
WHERE events.readcode = anc1.childcode
        AND anc1.parentcodes = 'H33..'

        AND quals.attrc = anc2.childcode
        AND anc2.parentcodes = 'X75WW'

         AND quals.valrc = anc3.childcode
        AND anc3.parentcodes = 'X906m'

        AND events.eventid = quals.eventid

PLAN JOIN (ANC2 INDEX (ANCESTOR_PRIM_IDX),QUALS INDEX
(QUALS_ATTRC,QUALS_ATTVAL),EVENTS INDEX
(EVENTS_EVENTID,EVENTS_ID_RC),
ANC3 INDEX (ANCESTOR_PRIM_IDX), ANC1 INDEX
(ANCESTOR_PRIM_IDX))

         COUNT
===========

         36

Current memory = 2546688
Delta memory = 1559552
Max memory = 2546688
Elapsed time = 234.37 sec
Buffers = 4096
Reads = 26121
Writes 0
Fetches = 652092
```

1 Un-optimised SQL Code. Example query: the number of events in which Asthma (H33..) had Chronicity (X75WW) of Acute (X906m).

4. Establish Comparative Timings for Range Number Method

This part of the project involved two main stages:

- Producing a comparable set of query times as for the Ancestor Method.
- Producing new range number tables for the October 1996 and March 1997 Read Code release, to determine the effect the of updating the IDRANGE table with the new Read Codes without renumbering all the Read Codes (see section 5).

The initial results from the range code method proved a remarkable change on the Ancestor method results:

The query used was:

```
SELECT COUNT (events.eventid)
FROM events, idrange
WHERE events.coreid BETWEEN idrange.idlow AND idrange.idhigh
        AND idrange.id = <Read Code ID Value>
```

Core Concept - Range Number Method					
Core Concept	Number of Ranges	Number of Records	Paradox	InterBase	Access
			Query times in seconds		
Disorders	311[2]	430970	5510	266	>4000
Operations & procedures	4	178919	71	48	438
Gastrointestinal & digestive disorders	39	32851	671	77	3777
Tumour	97	43949	1604	257	>4000
Tumour of lung	2	827	35	6	252
Asthma	5	535	91	6	535
Heart Attack	1	440	19	4	140

InterBase shows far better results than for the Ancestor Method. Previously, counting all disorders in the EVENTS table using the Ancestor method took 3728 seconds, but using range number this dropped to 266, that is about 14 times quicker. The reason for the significant improvement can perhaps be traced to the cause of the earlier poor results. The number of ranges even in the 'worst' case is much lower than the number of child codes i.e. 311 ranges compared with 43277 child codes.

While InterBase performs well using range numbers, both Paradox and Access perform poorly using this method. This seems to be because of the way indexes are used for these queries. It should be noted that using COUNT (1) instead of COUNT (EVENTID) had no effect on the results for Access unlike the query using the ANCESTOR table method.

When a single qualifier was added the initial InterBase results were less

[2] Note that there are significantly more ranges associated with disorders than procedures because of the way that the Read hierarchy was numbered. Disorders were numbered only after the History and observations chapter. Therefore those few, but scattered concepts in history and observations were subsequently added to the disorders' ranges. This anomaly was retained as it provided further observations on a worst case scenario.

encouraging though with optimisation these improved significantly:

Core Concept and Single Qualifier - Range Number Method						
Core Concept (no. of ranges)	Attribute (no. of ranges)	Value (no. of ranges)	Number of records	Paradox	InterBase	InterBase Optimised
				Query times in seconds		
Disorder (311)	Site (1)	Stomach structure (2)	2934	170	40273	340
Gastrointestinal and digestive disorder (39)	Site (1)	Human body structure (13)	21500	405	>3600	1020
Gastrointestinal and digestive disorder (39)	Site (1)	Stomach structure (2)	2847	160	5047	120
Tumour (97)	Site (1)	Stomach structure (2)	345	168	>3600	310
Asthma (5)	Chronicity (1)	Acute (1)	36	140	574	32
Operations & procedures (4)	Priority (1)	Emergency (1)	40526	150	1258	249
Male genital organ procedure (6)	Site (1)	Testis Structure (3)	259	160	57	36
Male genital organ procedure (6)	Priority (1)	Emergency (1)	1000	150	1952	110

As can be seen, InterBase when optimised produced reasonable results better in most cases than InterBase using the Ancestor table method.

Due to the very poor performance of the Access database with the core concepts-only queries, tests using the range number method on core concept & single qualifier were not performed.

In optimising the queries for InterBase similar techniques were used as for the Ancestor method. Therefore the sections of the query that ran quickest were performed first to produce a subset of the data before doing the slow sections on this initial subset of the records.

5. Establish Performance of Range Numbers When Not Reassigned

The next stage of the project involved creating new IDRANGE tables for the October 1996 and March 1997 Read Code releases. While the same method of creating the new range numbers was used, two different incremental values (1 and 100) were used.

Incremental value of 1
In this case the starting point was the original set of range numbers which were generated with an incremental value of 1. All new Read Codes were then issued with numbers after the current values. As a result an increase in the number of ranges was guaranteed. This is the worst case scenario.

Incremental value of 100
In this case all the original range numbers were given a value 100 times that originally assigned allowing 'gaps' in the number scheme. Now, for each new release, numbers would be assigned in the gaps between existing Read Codes were possible, minimising the number of new ranges.

Number of Ranges for Different Read Code Releases

Core Concept	April 1996	October 1996	October 1996	March 1997	March 1997
	Inc. 1	Inc. 1	Inc. 100	Inc. 1	Inc. 100
Disorders	311	676	537	840	595
Operations & procedures	4	153	80	374	159
Gastrointestinal & digestive disorders	39	90	73	128	92
Tumour	97	113	105	158	116
Tumour of lung	2	3	3	4	4
Asthma	5	5	5	6	6
Heart Attack	1	1	1	2	2

In practice, the number of ranges did increase in both cases but was significantly lower when the initial incremental value was 100.

To test the effect of the increase in the number of ranges the core concept queries were re-run using the single incremental value IDRANGE tables for both October 1996 and March 1997. The results achieved using the revised Read Code release on the InterBase database are as follows:

Core Concept - Range Number Method for Different Read Code Releases

Core Concept	April 1996	October 1996	March 1997
	Inc. 1	Inc. 1	Inc. 1
	Query times in seconds		
Disorders	266	1905	2025
Operations & procedures	48	810	903
Gastrointestinal & digestive disorders	77	183	199
Tumour	257	267	270
Tumour of lung	6	7	7
Asthma	6	9	9
Heart Attack	4	4	4

As can be seen even the worst case, that of the March 1997 Read Codes with a single increment, the results of these are still better than the original ancestor table queries performed in section 3. While most of the timings given above may be acceptable, the results for ‘Disorders’ and ‘Operations & Procedures’ are much poorer than the original range number results due to a combination of a large number of ranges coupled with a large number of returned rows.

It would have been possible to achieve better results by using a larger increment between the range numbers, for example 1000. In addition, areas of the Read Codes where there is most development could have been given even greater increments. Using these techniques it would be possible to minimise the effective of Read Code changes on the Range Number method.

6. Determine the Effect of Searches on Multiple Qualifiers

This involved a series of five queries performed using both the Ancestor Table method and the Range Number method but for two and three qualifiers.

The standard SQL used for the Ancestor Table method in queries containing two qualifiers was:

```
SELECT COUNT (events.eventid)
FROM quals quals1, quals quals2, events,
 ancestor anc1, ancestor anc2,
 ancestor anc3, ancestor anc4, ancestor anc5

WHERE events.ReadCode = anc3.childcode
 AND anc3.parentcodes = '7C...'
```

```
 AND events.EventID = quals1.EventID
 AND events.EventID = quals2.EventID

 AND quals1.AttRC = anc1.childcode
 AND anc1.parentcodes = 'X9019'

 AND quals1.ValRC = anc2.childcode
 AND anc2.parentcodes = 'Xa1GA'

 AND quals2.AttRC = anc4.childcode
 AND anc4.parentcodes = 'XM0Rs'

 AND quals2.ValRC= anc5.childcode
 AND anc5.parentcodes = '7NB32'
```

In this example, the query was for all events containing a male genital organ procedure (7C...) that was qualified by Site (X9019) which was Testis structure (Xa1GA) and by Laterality (XMORs) which was Left side (7NB32). Note that no attempt is made to deal with nesting of qualifiers here as this was not the point of the experiment i.e. Laterality: Left side is not specifically attached to Testis structure.

As can be seen the SQL is much more complex than previously but this is not intrinsically a problem provided the database executes the SQL in an optimum fashion. As expected from the previous tests InterBase needed the code order changed to achieve reasonable results together with the query splitting into two parts as below:

```
drop table ids;
create table ids (eventid integer);

insert into ids
SELECT events.eventid
FROM quals RIGHT JOIN ancestor anc3 ON quals.valrc =
anc3.childcode,
 events, ancestor anc1, ancestor anc2
WHERE anc3.parentcodes = '7NB32'

 AND anc1.parentcodes = '7C...
 AND events.readcode = anc1.childcode

 AND quals.attrc = anc2.childcode
 AND anc2.parentcodes = 'XM0Rs'

 AND events.eventid = quals.eventid;

PLAN SORT MERGE (JOIN (ANC1 INDEX (ANCESTOR_PRIM_IDX), ANC2
INDEX
(ANCESTOR_PRIM_IDX), EVENTS INDEX (EVENTS_RC_IDX)),JOIN (ANC3
```



```
INDEX
(ANCESTOR_PRIM_IDX), QUALS INDEX (QUALS_VALRC)))

Records affected: 72
Current memory = 2537472
Delta memory = 1209344
Max memory = 2537472
Elapsed time = 120.73 sec
Buffers = 4096
Reads = 25621
Writes 0
Fetches = 91466

SELECT count (quals.eventid)
FROM quals RIGHT JOIN ancestor anc2 ON quals.valrc =
anc2.childcode,
        ancestor anc1, ids
WHERE anc2.parentcodes = 'Xa1GA'

 AND quals.attrc = anc1.childcode
 AND anc1.parentcodes = 'X9019'

 AND quals.eventid = ids.eventid;

PLAN SORT MERGE (ANC1 INDEX (ANCESTOR_PRIM_IDX), SORT MERGE
(IDS
NATURAL,JOIN (ANC2 INDEX (ANCESTOR_PRIM_IDX), QUALS INDEX
(QUALS_VALRC))))

        COUNT
===========

         27

Records affected: 1
Current memory = 1451008
Delta memory = -1086464
Max memory = 2561024
Elapsed time = 10.93 sec
Buffers = 4096
Reads = 1292
Writes 0
Fetches = 2998
```

In essence, a single qualifier query (here named ‘ids’) is performed and the results of this query then used with subsequent qualifiers to produce the desired result. As can be seen, quite reasonable results can be achieved as the initial query retrieves a relatively small number of rows.

Paradox continued to produce excellent results without any need to change the standard code. The results for Access were more in line with InterBase though on the whole slower than optimised InterBase times. As there was not time to optimise the Access database it is likely that better timings could be achieved with code changes or splitting the query into separate parts.

The basic query for range numbers was:

```
SELECT COUNT (events.eventid)
FROM events, quals quals1, quals quals2,
 idrange id1, idrange id2, idrange id3, idrange id4, idrange id5

WHERE    id2.id = 173774
 AND     quals1.attid between id2.idlow and id2.idhigh
 AND     id3.id = 140698
 AND     quals1.valid between id3.idlow and id3.idhigh
 AND     quals1.eventid = events.eventid

 AND     id4.id = 173442
 AND     quals2.attid BETWEEN id4.idlow AND id4.idhigh
 AND     id5.id = 155113
 AND     quals2.valid BETWEEN id5.idlow AND id5.idhigh
 AND     quals2.eventid = events.eventid

 AND     id1.id = 75303
 AND     events.coreid BETWEEN id1.idlow AND id1.idhigh
```

The range number results were only performed using InterBase due to the previous relatively poor performance of Access and Paradox using this method, and time constraints. The InterBase results were good with similar timings to the single qualifier queries performed earlier.

The results were broadly as expected on the InterBase platform with the range number method showing the best performance. It is though encouraging that with optimisation the queries with additional qualifiers were not significantly worse than for a single qualifier. This shows that with sensible query ordering reasonable results can be obtained though the potential for very slow queries is shown in the unoptimised results.

Ancestor Table Method							
Concept (No. child codes)	Attribute (No. child codes)	Value (No. child codes)	Number of records	Paradox	InterBase	InterBase Optimised	Access
				Query times in seconds			

Operations and procedures	Site (143)	Testis structure (30)	92	43	>3600	845	1968
(16802)	Priority (1)	Emergency (1)					
Operations and procedures	Site (143)	Testis structure (30)	19	57	>3600	2701	N/A
(16802)	Priority (1)	Emergency (1)					
	Method (81)	Excision (7)					
Male genital organ procedure	Site (143)	Testis structure (30)	92	42	>3600	134	1640
(323)	Priority (1)	Emergency (1)					
Male genital organ procedure	Site (143)	Testis structure (30)	27	41	>7200	132	1650
(323)	Laterality (5)	Left (1)					
Male genital organ procedure	Site (143)	Testis structure (30)	19	57	>3600	990	N/A
(323)	Priority (1)	Emergency (1)					
	Method (81)	Excision (7)					

Range Number Method					
Concept (No. of ranges)	Attribute (No. of ranges)	Value (No. of ranges)	Number of records	InterBase	InterBase Optimised
				Query times in seconds	
Operations and procedures (4)	Site (1) Priority (1)	Testis structure (3) Emergency (1)	92	169	N/A

Operations and procedures (4)	Site (1) Priority (1) Method (1)	Testis structure (3) Emergency (1) Excision (2)	19	2036	200
Male genital organ procedure (6)	Site (1) Priority (1)	Testis structure (3) Emergency (1)	92	176	N/A
Male genital organ procedure (6)	Site (1) Laterality (1)	Testis structure (3) Left (1)	27	131	N/A
Male genital organ procedure (6)	Site (1) Priority (1) Method (1)	Testis structure (3) Emergency (1) Excision (2)	19	2420	120

7. Examine the Effect of a Fixed Field Database

The aim here was to find out what speed advantage might be gained by merging the EVENTS and QUALIFIERS tables. The speed advantage would be gained at the cost of flexibility of adding new types of qualifiers and therefore best suits an environment when only a restricted set of qualifiers is used. Alternatively, some qualifiers which are frequently queried might be stored in fixed fields in an events table, while others are placed in a qualifiers table.

A program to create a merged EVENTS and QUALIFIERS table was developed. The program populated an additional six fields created in a new EVENTS table. The fields were:

- Site
- Method
- Priority
- Laterality
- Chronicity

- Interpretation

After creating the new table, testing was conducted using both single and multiple qualifiers.

As expected these tests achieved significantly faster results than had been achieved earlier. Though the Ancestor table method using InterBase still benefited from optimisation of the queries. The query for any procedure (7....) which has priority = emergency (X78uH) is:

```
SELECT COUNT (eventsmerged.eventid)
FROM eventsmerged, ancestor anc1, ancestor anc2
WHERE anc1.parentcodes = '7C...'
 AND eventsmerged.readcode = anc1.childcode

 AND anc2.parentcodes = 'X78uH'
 AND eventsmerged.priorityrc = anc2.childcode
```

Optimised SQL where the qualifier, in this case 'site' (Stomach structure - Xa1Fn), has far fewer child codes than the core concept (Gastrointestinal disease - J....), that is 29 compared to 2879:

```
SELECT COUNT (eventsmerged.eventid)
FROM eventsmerged RIGHT JOIN ancestor anc2 ON
        eventsmerged.siterc = anc2.childcode, ancestor anc1
WHERE anc2.parentcodes = 'Xa1Fn'

 AND anc1.parentcodes = 'J....'
 AND eventsmerged.readcode = anc1.childcode
```

Optimised SQL where core concept, gastrointestinal & digestive disorders (J....), has fewer child codes than the qualifier, in this case site (human body structure - Xa16m), 2879 compared to 7619:

```
SELECT COUNT (eventsmerged.eventid)
FROM eventsmerged RIGHT JOIN ancestor anc1 ON
 eventsmerged.readcode = anc1.childcode, ancestor anc2
WHERE anc1.parentcodes = 'J....'

 AND anc2.parentcodes = 'Xa16M'
 AND eventsmerged.siterc = anc2.childcode
```

Ancestor Table Method

Core Concept	Attribute	Value	Number of records	InterBase	InterBase Optimised
				Query times in seconds	

Disorder	Site	Stomach structure	2934	4941	50
Gastrointestinal and digestive disorder	Site	Human body structure	21500	10235	362
Gastrointestinal and digestive disorder	Site	Stomach structure	2847	381	32
Tumour	Site	Stomach structure	345	531	31
Asthma	Chronicity	Acute	36	6	N/A
Operations & procedures	Priority	Emergency	40526	1856	120
Male genital organ procedure	Site	Testis Structure	259	5	N/A
Male genital organ procedure	Site	Emergency	1000	46	N/A

Range Number Method

Core Concept	Attribute	Value	Number of records	InterBase	InterBase Optimised
				Query times in seconds	
Disorder	Site	Stomach structure	2934	33	N/A
Gastrointestinal and digestive disorder	Site	Human body structure	21500	107	N/A
Gastrointestinal and digestive disorder	Site	Stomach structure	2847	17	N/A
Tumour	Site	Stomach structure	345	7	N/A
Asthma	Chronicity	Acute	36	6	N/A
Operations & procedures	Priority	Emergency	40526	64	N/A
Male genital organ procedure	Site	Testis Structure	259	3	N/A
Male genital organ procedure	Site	Emergency	1000	5	N/A

Ancestor Table Method

Concept	Attribute	Value	Number of records	InterBase	InterBase Optimised
				Query times in seconds	
Operations and procedures	Site Priority	Testis structure Emergency	92	271	15
Operations and procedures	Site Priority Method	Testis structure Emergency Excision	19	316	15
Male genital organ procedure	Site Priority	Testis structure Emergency	92	11	N/A
Male genital organ procedure	Site Laterality	Testis structure Left	27	17	N/A
Male genital organ procedure	Site Priority Method	Testis structure Emergency Excision	19	17	N/A

Range Number Method

Concept	Attribute	Value	Number of records	InterBase	InterBase Optimised
				Query times in seconds	
Operations and procedures	Site Priority	Testis structure Emergency	92	4	N/A
Operations and procedures	Site Priority Method	Testis structure Emergency Excision	19	12	N/A
Male genital organ procedure	Site Priority	Testis structure Emergency	92	8	N/A

Male genital organ procedure	Site Laterality	Testis structure Left	27	8	N/A
Male genital organ procedure	Site Priority Method	Testis structure Emergency Excision	19	4	N/A

8. Demonstrate Information Retrieval Running on a 'Real' Primary Care Database

The core concept queries used previously were repeated on InterBase and Paradox databases using both the Ancestor and Range Number methods. This time, although the database structure was unchanged, the data was not randomly generated, but drawn from a real database from general practice.

Core Concept GP Database - Ancestor Table Method				
Core Concept	Child Codes	Number of Records	Paradox	InterBase
			Query time in seconds	
Disorders	43277	130447	25	421
Operations & procedures	16802	5035	22	41
Gastrointestinal & digestive disorders	2879	7409	19	39
Tumour	3899	2913	19	19
Tumour of lung	63	164	2	5
Asthma	44	6060	12	18
Heart Attack	50	419	4	4

It is interesting in the Ancestor table method to see the relative improvement in the Ancestor table results using InterBase with the GP Database than with the generated database. While the GP database was about half the size of the generated database (617,584 records compared to 1,224,408), the main reason would seem to be the smaller number of rows returned by each query. In the case of disorders on the generated database 430,970 records were counted taking 3728 seconds compared to 130,447 records counted in 421 seconds in the GP database. This is seen even more clearly for operations where the 178,919 counted on the generated database took 1578 seconds compared to only 41 seconds for the 5035 records on the GP database. While as mentioned in section 3 there is a relationship between query time and the number of child codes, the results in this section indicated a very significant link between query time and the records found.

The results for Paradox using GP data were better than with the generated data, with query time down by between 40 - 50 %, which is quite close to the difference in size between the databases which is 49.6 %.

Core Concept GP Database - Range Number Method				
Core Concept	Number of ranges	Number of Records	Paradox	InterBase
			Query time in seconds	
Disorders	311	130447	1722	62
Operations & procedures	4	5035	38	18
Gastrointestinal & digestive disorders	39	7409	239	26
Tumour	97	2913	518	14
Tumour of lung	2	164	20	3
Asthma	5	6060	48	12
Heart Attack	1	419	10	5

As expected from the earlier results, the InterBase proved to have better query times using the Range Number method while the reverse was true for Paradox. This confirms that Range Number method is a better access method for InterBase while the Ancestor Table method would be the preferred access method when using Paradox.

9. Decision Support System Experiments

In this experiment, the question tested was whether the information retrieval methods support questions such as 'Does this patient have a cardiovascular disease' with rapid performance as is required in even simple decision support systems, such as those automatically checking for drug allergies and contraindications in the patient record when a clinician is prescribing drugs.

A program was developed to simulate a simple decision support system with this program linked to the GP Database. The program used the ancestor table access method when using a Paradox database and the range number method when using the GP Database stored in InterBase. These combinations were chosen as they had generated the best performance times during the previous tests in the project.

The program allowed the selection of a patient within the GP Database, and then allowed a Read Code to be entered; events with this code or its related codes were then displayed. The time taken for the data to be retrieved and displayed was automatically recorded. The following queries were performed:

Drugs (Read Code)	InterBase	Paradox
	Query Time (seconds)	
Penicillin (x009C)	1	1
Beta-blockers (bd...)	1	1
Conditions (Read Code)		
Cardiovascular (G....)	2	1
Gastrointestinal (J...)	1	1
Cancer (B....)	1	1

The results indicate high speed responses in both databases for all tested codes. The times were in almost all cases one second or less.

The longest response was for cardiovascular disorders for patient 35 on the InterBase database which was two seconds; this involved a large number of returned events (8). This was only if this query was the first search done on the system. If another patient involving the same or even a different condition was searched first the search time dropped to about one second.

10. Software Developed for Project

Program Name	Development Tool	Description
Query	Delphi	Used run queries against Paradox tables.
Range number	Delphi	Used to create IDRANGE tables with new releases of Read Codes
Ancestor table creation	Delphi 2 compatible with Delphi 1.	Used to create ANCESTOR table from the HIER table.
Populate Events	Delphi 1	Populate a new EVENTS table including qualifier data from the QUALS table.
Decision Support System	Delphi 1 for Paradox Delphi 2 for InterBase	Simulates queries in a decision support system.

11. Conclusion

The aim of the project was to investigate two methods of implementing Read

Codes in a RDBMS from the point of view of performance and usability. As mentioned at the start of the report three different types of database were used InterBase, Paradox and Access. While InterBase and Paradox stored the Read Codes as alphanumeric values, that is in the way the codes are actually written, in Access the codes were stored as long integers as Access is case insensitive when searches are conducted; please refer to section 2 on the effect this may have had on the performance measurements.

11.1 Ancestor Table

This proved to have the best performance for both Paradox and Access but rather poor performance on the InterBase database. This method has a number of advantages over the range number method specifically:

- It uses actual Read Codes; this means there is no need to translate a Read Code to an
  alternative integer value.

- As the system queries Read Codes directly no extra storage is required in the patient record for holding alternative representations of Read Codes.

11.2 Range Numbers

This proved to have good performance on the InterBase database though it was necessary to do some optimisation on the SQL to ensure performance when qualifiers in addition to core concepts were used. There are a number of issues with regard to the use of range numbers which are not present for the Ancestor table method.

- Potential data duplication. If range numbers are used either only the integer value assigned to the Read Code is stored for an event or this Read Code must be stored as well leading to data duplication. But as the integer value is ‘transient’ (that is it may be reassigned at some future date) the Read Code itself would also have to be stored.

- There is the problem that the Range Number method achieves its performance from minimising the number of ranges searched. But with each new release the number of ranges increases as can be seen in the tests described in section 5 and reduces performance.

It is likely though that a combination of a high initial incremental value between Read Codes coupled with an optimum initial allocation of integers to the Read Codes would minimise this problem. It is suggested that if Range Numbers are used full reallocation of codes is only done once per year, with new numbers allocated as necessary for additional Read Codes issued in the

release occurring during the year.

11.3 De-normalisation

The experiments conducted on the de-normalised database were interesting in that the single qualifier results were significantly better than on the original database with separate EVENTS and QUALS tables. This is particularly true for the Ancestor Table method in the case of single qualifiers and less evident for multiple qualifiers for range numbers where the results were already quick. This would indicate that though performance gains are possible using this technique, if the best query techniques are used in the relational database structure the advantages are not that large.

It is recommended that this approach should only be used if the database and hardware combination used, cannot cope with using a normalised database structure. The reasons for this are that the benefit is only achieved:

- when the query involves a qualifier,
- and then only in a limited number of cases as with multiple qualifiers good results can often be achieved within a normalised database structure.

It should be noted that the results obtained using Paradox and the Ancestor Table method show good performance with a normalised database in all tests from core concept only to multiple qualifiers. In the case where a database and hardware platform achieve good results it would be much better to retain a normalised database structure rather than gather limited performance benefit with a de-normalised database.

One optimisation technique for both Range Code and Ancestor method is to store in the ANCESTOR or RCID tables the number of child codes or ranges for the Read Code. The system can then prioritise the order in which the query is performed. In general the greater the number of children or ranges linked to a Read Code the slower the query runs. By running the part of the query that small number of child or ranges first, the slower sections of the query are run with only a subset of the records so minimising the effect of the high number of children or ranges. This will only be effective when the efficiency of a query is dominated by the number of children of the query concepts rather than the number of matches in the patient record.

As the number of ranges or child codes does not vary between releases it is sensible to store these in the database rather than calculate them each time a query is processed.

Overall there is not a single answer to the question of which is the best technique for using Read Codes but different answers are applicable for different databases. It does seem clear that it is possible to achieve a good

level of performance on PC level hardware platform provided the appropriate technique, i.e. range number or ancestor table, and database are chosen together with a sensible optimisation scheme.

In the final two experiments, the conclusions drawn from the earlier parts of the project were applied to data from a real GP database. The GP database was about half the size of the generated database used in the previous parts of the project. The results for searches on core concepts proved similar to those done on the generated database with the InterBase database achieving its best times using the Range Number method while using Paradox the Ancestor table was the best access method. The only new conclusion that was drawn was that the InterBase was very sensitive to the number of records found in a query as well as the number of child codes when the Ancestor table method was used, see results in section 8.

The final part of the project, which was to simulate a decision support system using the GP database, indicated that using the Ancestor Table method with Paradox and the Range Number Method with InterBase allowed data to be retrieved on a given patient for a drug or condition almost instantly with retrieval times always under two seconds and normally under one second. This indicates that CTV3 Read Codes are quite capable of being used in an online system and providing a high speed response.

Appendix A

Summary Database Information for InterBase

```
/* Extract Database f:\readcodes\april96\ib0496.gdb */
CREATE DATABASE "f:\readcodes\april96\ib0496.gdb" PAGE_SIZE 1024 ;

/* Table: ANCESTOR, Owner: SYSDBA */
CREATE TABLE ANCESTOR (PARENTCODES VARCHAR(5) NOT
NULL,
 CHILDCODE VARCHAR(5));

/* Table: EVENTIDS, Owner: SYSDBA */
CREATE TABLE EVENTIDS (EVENTID INTEGER);

/* Table: EVENTS, Owner: SYSDBA */
CREATE TABLE EVENTS (EVENTID INTEGER,
        PATIENTID INTEGER,
        READCODE VARCHAR(5),
        CORETID VARCHAR(5),
        COREID INTEGER,
        ENCOUNTERID SMALLINT,
        ARERID SMALLINT,
```

```
        ENCOUNTERDATE DATE,
        ONSETDATE DATE);

/* Table: EVENTS3, Owner: SYSDBA */
CREATE TABLE EVENTS3 (EVENTID INTEGER,
        PATIENTID INTEGER,
        READCODE VARCHAR(5),
        CORETID VARCHAR(5),
        COREID INTEGER,
        SITERC VARCHAR(5),
        SITEID INTEGER,
        PRIORITYRC VARCHAR(5),
        PRIORITYID INTEGER,
        METHODRC VARCHAR(5),
        METHODID INTEGER,
        CHRONICITYRC VARCHAR(5),
        CHRONICITYID INTEGER,
        LATERALITYRC VARCHAR(5),
        LATERALITYID INTEGER,
        INTREPRETATIONRC VARCHAR(5),
        INTREPRETATIONID INTEGER,
        ENCOUNTERID SMALLINT,
        CARERID SMALLINT,
        ENCOUNTERDATE DATE,
        ONSETDATE DATE);

/* Table: ID0397QK, Owner: SYSDBA */
CREATE TABLE ID0397QK (ID INTEGER,
        IDLOW INTEGER,
        IDHIGH INTEGER);

/* Table: ID0797SN, Owner: SYSDBA */
CREATE TABLE ID0797SN (ID INTEGER,

        IDLOW INTEGER,
        IDHIGH INTEGER);

/* Table: ID1096SN, Owner: SYSDBA */
CREATE TABLE ID1096SN (ID INTEGER,
        IDLOW INTEGER,
        IDHIGH INTEGER);

/* Table: IDRANGE, Owner: SYSDBA */
CREATE TABLE IDRANGE (ID INTEGER NOT NULL,
        DLOW INTEGER,
        IDHIGH INTEGER);

/* Table: IDS, Owner: SYSDBA */
CREATE TABLE IDS (EVENTID INTEGER,
```

```
        READCODE CHAR(5));

    /* Table: IDS2, Owner: SYSDBA */
    CREATE TABLE IDS2 (EVENTID INTEGER,
        ATTRC CHAR(5),
        VALRC CHAR(5));

    /* Table: IDS3, Owner: SYSDBA */
    CREATE TABLE IDS3 (EVENTID INTEGER);

    /* Table: IDS4, Owner: SYSDBA */
    CREATE TABLE IDS4 (EVENTID INTEGER);

    /* Table: MALE, Owner: SYSDBA */
    CREATE TABLE MALE (EVENTID INTEGER);

    /* Table: PATIENTS, Owner: SYSDBA */
    CREATE TABLE PATIENTS (PATIENTID INTEGER,
        SURNAME VARCHAR(17),
        FORENAME VARCHAR(13),
        TITLE VARCHAR(10),
        DATEOFBIRTH DATE,
        SEX VARCHAR(1),
        ADDRESS1 VARCHAR(29),
        ADDRESS2 VARCHAR(26),
        ADDRESS3 VARCHAR(7),
        POSTCODE VARCHAR(7));

/* Table: QUALS, Owner: SYSDBA */
CREATE TABLE QUALS (EVENTID INTEGER NOT NULL,
    QUALNO SMALLINT NOT NULL,
    ATTRC VARCHAR(5),
    ATTID INTEGER,
    VALRC VARCHAR(5),
    VALTID VARCHAR(5),
    VALID INTEGER,
    QUALREF SMALLINT);

/* Index definitions for all user tables */
CREATE INDEX ANCESTOR_PRIM_IDX ON ANCESTOR(PARENTCODES);
CREATE INDEX EVENTS_COREID ON EVENTS(COREID);
CREATE INDEX EVENTS_EVENTID ON EVENTS(EVENTID);
CREATE INDEX EVENTS_ID_RC ON EVENTS(EVENTID, READCODE);
CREATE INDEX EVENTS_PATIENTID ON EVENTS(PATIENTID);
CREATE INDEX EVENTS_RC_IDX ON EVENTS(READCODE);
CREATE INDEX EVENTS3_CHRONICITYID ON EVENTS3(CHRONICITYID);
CREATE INDEX EVENTS3_CHRONICITYRC ON EVENTS3(CHRONICITYRC);
CREATE INDEX EVENTS3_COREID ON EVENTS3(COREID);
CREATE INDEX EVENTS3_EVENTID ON EVENTS3(EVENTID);
```

```
CREATE INDEX EVENTS3_INTREPRETATION ON
EVENTS3(INTREPRETATIONRC);
CREATE INDEX EVENTS3_LATERALITYID ON EVENTS3(LATERALITYID);
CREATE INDEX EVENTS3_LATERALITYRC ON
EVENTS3(LATERALITYRC);
CREATE INDEX EVENTS3_METHODID ON EVENTS3(METHODID);
CREATE INDEX EVENTS3_METHODRC ON EVENTS3(METHODRC);
CREATE INDEX EVENTS3_PRIORITYID ON EVENTS3(PRIORITYID);
CREATE INDEX EVENTS3_PRIORITYRC ON EVENTS3(PRIORITYRC);
CREATE INDEX EVENTS3_READCODE ON EVENTS3(READCODE);
CREATE INDEX EVENTS3_SITEID ON EVENTS3(SITEID);
CREATE INDEX EVENTS3_SITERC ON EVENTS3(SITERC);
CREATE INDEX ID0397QK_ID_IDX ON ID0397QK(ID);
CREATE INDEX ID0797SN_ID_IDX ON ID0797SN(ID);
CREATE INDEX ID1096SN_ID_IDX ON ID1096SN(ID);
CREATE INDEX IDRANGE_ID ON IDRANGE(ID);
CREATE INDEX RCIDX ON IDS(READCODE);
CREATE INDEX EVENTID_ATTRC ON IDS2(ATTRC);
CREATE INDEX EVENTID_VALRC ON IDS2(VALRC);
CREATE INDEX EVENTID_IDX3 ON IDS3(EVENTID);
CREATE INDEX MALE_IDX ON MALE(EVENTID);
CREATE INDEX PATIENTS_PATIENTID ON PATIENTS(PATIENTID);
CREATE INDEX QUALS_ATTID ON QUALS(ATTID);
CREATE INDEX QUALS_ATTRC ON QUALS(ATTRC);
CREATE INDEX QUALS_ATTVAL ON QUALS(ATTRC, VALRC);
CREATE INDEX QUALS_EVENTID ON QUALS(EVENTID);
CREATE INDEX QUALS_VALID ON QUALS(VALID);
CREATE INDEX QUALS_VALRC ON QUALS(VALRC);

/* Grant permissions for this database */
```

Appendix B

Terms & Read Codes Used

Please note that the number of child codes and ranges associated with each Read Code applies to the April 1996 release of CTV3 used in this project.

Core Concept

Term	Read Code	Child Codes	Ranges
Disorders	X0003	43277	311
Operations & Procedures	X0001	16802	4
Gastrointestinal & digestive disorders	J....	2879	39

Tumour	B....	3899	97
Tumour of lung	Xa0KF	63	2
Asthma	H33..	44	5
Heart Attack	X200E	50	1
Male genital organ procedure	7C...	323	7

Attribute Codes

Term	Read Code	Child Codes	Ranges
Site	X9019	143	1
Laterality	XM0Rs	5	1
Method	X900S	81	1
Chronicity	X75WW	1	1
Priority	X904E	1	1

Value Codes

Term	Read Code	Child Codes	Ranges
Human body structure	Xa16M	7619	13
Stomach structure	Xa1Fn	29	2
Acute	X906m	7	1
Emergency	X78uH	1	1
Excision	X793K	7	2
Left side	7NB32	1	1
Testis structure	Xa1GA	30	3

Appendix C

Database Level Optimisations (InterBase)

Various optimisations of the database set-up (as opposed to query optimisations) were tried to improve efficiency.

De-fragmentation

- The InterBase database was created on a newly formatted hard disk

with no other data or programs installed.

- After the database was loaded a report on the database structure was generated. It was found that a number of the indexes including the main index for the Ancestor table had a ‘depth’ of four, ideally this should not be greater than two. By exporting and re-importing the data, it proved possible to decrease the depth to three in the worst case.

- Some experimentation with page size was tried, switching from the default level of 1024 bytes to 2048 and 4096. This did not result in any noticeable charge in performance so the default value was used in testing for this project.

The Database Cache

After some experimentation, a value of 4096 pages provided the best level of performance. This was the value used in testing for this project.